Distributed Learning for Robust Fuzzy Neural Networks

by

Leijie Zhang

Thesis submitted in fulfilment of
the requirements for the degree of

Doctor of Philosophy

under the supervision of
Professor Chin-Teng Lin

Computational Intelligence and Brain Computer Interfaces Lab
Australian Artificial Intelligence Institute
Faculty of Engineering and Information Technology
University of Technology Sydney

February 2023

Certificate of Original Authorship

I, Leijie Zhang, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctoral of Philosophy in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:
Signature removed prior to publication.

05/02/2023

Abstract

Uncertainty processing and privacy preservation are two particularly significant issues machine learning models need to consider when dealing with real-world data. To date, distributed fuzzy neural networks (DFNNs) have addressed these issues by locally processing the uncertainty on different agents in multi-agent scenarios. However, today's DFNNs can neither handle data that is too high in dimension, nor can they process uncertainty once it reaches too high a level. Additionally, DFNNs tend to learn a shared group of fuzzy rules that they apply to all clients, which limits the ability to personalise local models to the data they are working with. This is especially problematic when processing non-independent and identically distributed (non-IID) local data. Worse still, many client devices hold large amounts of easily-obtainable unlabelled samples that current DFNNs simply ignore. Hence, this thesis proposes novel robust fuzzy neural networks (RFNNs) and corresponding new distributed learning architectures as a solution to these problems.

Creatively, this thesis presents a new robust fuzzy neural network (RFNN) that can be applied to local clients to handle data with high levels of uncertainty and high dimensionality, and also improves the ability of local clients to personalise models to suit the given data, desired privacy levels, and so on. Additionally, RFNNs overcome some current issues with building highly generalisable models. The RFNN architecture comprises an adaptive inference engine and neural-structured consequent components. However, unlike traditional fuzzy inference processes, which use AND operations, the proposed inference engine can adaptively learn the firing strength of each fuzzy rule, even when the input data has high dimensionality. Lastly, the engine then assigns an uncertainty tolerance to the membership values. The neural-structured consequent components leverage neural network structures to enhance the reasoning ability of the fuzzy rules when dealing with complex inputs.

As a second innovation, this thesis presents a novel RFNN with interpolation consistency regularisation (ICR) to utilise a large amount of easily-obtainable unlabelled data. ICR, which was recently proposed to regularise semi-supervised problems, can force decision boundaries to pass through sparse data areas, thus increasing a

model's robustness. Then, a corresponding novel distributed learning method called distributed semi-supervised fuzzy regression (DSFR) model is proposed. The framework of the DSFR introduces a novel distributed fuzzy C-means method (DFCM) and distributed variant of interpolation consistency regularisation (DICR). The DICR is solved by the alternating direction method of multipliers (ADMM), which locates the parameters in the antecedent and consequent components of the DSFR model. Notably, the DSFR model converges very quickly since it does not involve any back-propagation procedures. It is also scalable to large-scale datasets because it benefits from both the DFCM and DICR.

The last innovation to be presented is a novel federated fuzzy neural network (FedFNN) with evolutionary rule learning (ERL). FedFNN offers a new distributed learning solution for FNNs, including the RFNNs, to enable them to cope with data heterogeneity, personalised privacy preservation, and high levels of data uncertainty in decentralized scenarios. Technically, FedFNNs maintain a global set of rules in the central server and a personalised subset of these rules for each local client. ERL is inspired by the theory of biological evolution; it encourages rule variations while activating superior rules and deactivating inferior rules for local clients with non-IID data. More specifically, ERL is an iterative procedure consisting of two stages: a rule cooperation stage that updates global rules by aggregating local rules based on the activation status of each, and a rule evolution stage that evolves the global rules and updates the activation status of each local rule. This procedure makes FedFNNs both more generalisable and more personalisable when dealing with non-IID issues and data uncertainty.

In summary, the main contributions of this research include: 1) A new and effective end-to-end robust fuzzy neural network (RFNN) architecture that is robust to data uncertainty and is able to process high-dimensional samples. It includes an adaptive inference engine that can generate representative firing strengths for high levels of uncertainty and an adaptive consequent component that enhances the reasoning ability of the fuzzy rules; 2) A distributed fuzzy C-means method (DFCM) that can be used directly with any labelled or unlabelled training data available over an interconnected network, plus a distributed interpolation consistency regularisation (DICR) mechanism that uses unlabelled data to optimise the parameters in the consequent component; 3) A federated fuzzy neural network framework (FedFNN) with evolutionary rule learning (ERL) that handles data uncertainty and non-IID issues in decentralised scenarios.

Publications

Related Papers

L. Zhang, Y. Shi, Y. C. Chang, and C. T. Lin, "Robust Fuzzy Neural Network With An Adaptive Reference Engine", *IEEE Transactions on Cybernetics*, 2023.

L. Zhang, Y. Shi, Y. C. Chang, and C. T. Lin, "Federated Fuzzy Neural Network with Evolutionary Rule Learning.", *IEEE Transactions on Fuzzy Systems*, 2022.

Y. Shi, **L. Zhang**, Z. Cao, M. Tanveer, and C. T. Lin, "Distributed Semi-supervised Fuzzy Regression with Interpolation Consistency Regularization.", *IEEE Transactions on Fuzzy Systems*, 2021.

L. Zhang, Y. Shi, Y. C. Chang and C. T. Lin, "Hierarchical fuzzy neural networks with privacy preservation on heterogeneous big data", *IEEE Transactions on Fuzzy Systems*, pp. 1-1, 2020.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the remarkable development of data collection devices, the data available for machine learning tasks has become richer not only in quantity but also in dimension [1]. Big data and its benefits have become a ubiquitous part of research in many areas, including social networks, the Internet of Things, commerce, astronomy, biology, medicine, and more [2], [3]. However, with increasing regularity, the gains being made seem to come at the threat of violations to the privacy and security of our personal data [4]. This issue is attracting serious attention in both society and the research community, especially since the Facebook data privacy scandal [5]. In addition to privacy concerns, uncertainty is another problematic characteristic of big data. Uncertainty can arise from a variety of sources, including measurement errors in sensing instruments, the discrete sampling of measurements, feature perturbation, which is generally unavoidable during data collection and processing [6]. Even radiation pollution, device reading errors, and uncontrolled environmental factors can lead to missing or erroneous data. In fact, with the development of data-collecting devices and more diverse data-collection scenarios, uncertainty now commonly exists in big data. Fortunately, distributed fuzzy neural networks (DFNNs) [7]–[9] have been proposed to solve these two major issues. However, as shown in Fig. 1.1, what they cannot handle is complex real-world scenarios where the data are heterogeneous and enormous in volume and dimension. Worse still, big data is usually only partially labelled due to the high costs of labelling, which limits an FNN's ability to learn a more representative fuzzy set. To solve these issues, we need novel and efficient machine-learning techniques.

1.1 Privacy Preservation and Distributed Learning

As one of the most salient problems in data science, privacy-preserving machine learning is attracting attention [10]–[13]. Generally, there are two types of privacy-preserving machine learning methods; one relies on perturbation and randomisation, and the other on segmenting the data. Perturbation/randomisation methods alter the data before releasing or sharing them [11], which offers some limited privacy protection at the sacrifice of some performance. Methods based on segmentation typically distribute the data among multiple agents. Neighbouring agents cooperate with each other to learn global results but without revealing their individual data to others [14]. Distributed machine learning algorithms have emerged out of these multi-agent data sharing scenarios as a solution to preserving privacy [15]–[19]. Distributed algorithms are well suited to big data environments given there are limits on the amount of data a single agent or a centralised algorithm can feasibly handle. Hence, to meet the communications overhead and storage capacity needed to process information at the big data scale, the data needs to be distributed throughout a network of interconnected agents [20].

Additionally, federated learning models [21]–[23] offer decentralised learning architectures that allow local models to optimise their parameters using gradient descent. The federated learning approach learns a shared model by aggregating the updates obtained from local clients without needing to access their data. Notably, differences in the distributions of each client's data is a key challenge in federated learning that many researchers have sought to solve [24]–[28]. Some of these solutions allow clients to build personalised models. However, few existing federated learning methods are able to simultaneously cope with data uncertainty and non-independent and identically distributed (non-IID) data in a distributed learning scenario. Further, although several methods incorporate Bayesian treatments [29] or Gaussian processes [27], [28] as a potential solution, their performance relies heavily on learning good posterior inferences and kernel functions, which is very time-consuming.

1.2 Data Uncertainty and Fuzzy Neural Networks

As a set of data's complexity increases, perturbations in the sample distributions, inaccurate recordings, obsolete features, and other problems inevitably find their way into the data, creating uncertainty in the data collected. Loosely defined as "how well the data speaks to the question of interest with respect to the model" [30], uncertainty

over whether the available data will meet the demands of the task begins at the collection stage. Neither sensors nor humans are immune to measurement errors. Qualitative data, such as social media accounts, are often subjective in nature, and the data collected may not be relevant or might be incomplete.

One approach to ameliorate this issue is to apply a regularisation technique to the data, such as dropout or injecting noise [31], [32]. The idea is to add randomness into the training process to avoid over-training the weights. However, these regularisation methods result in a trade-off between data fitting and model generalisation [33]. For example, with a noise injection technique, reducing the amount of noise injected will also reduce the model's generalisability, while adding more noise will make it more difficult to fit the model to the data distribution. Thus, finding a proper hyperparameter with which to penalise the regularisation terms is crucial with these types of methods. In practice, this is not always easy, as uncertainty levels vary between tasks and scenarios. That means a new hyperparameter needs to be found for every new task and dataset.

Another approach is combining neural networks with statistical algorithms [34], [35] to generate deep statistical models [36]–[38]. These models rest on the theoretical foundations of probabilistic reasoning [39], providing a highly flexible approach to dealing with data uncertainty. A deep statistical model might come in the form of a Bayesian neural network (BNN) [40], a deep Gaussian process (DGP) [36], or one of many others. The strengths of deep statistical approaches are that they can show high levels of resilience to uncertain inputs, and they can also provide probabilistic guarantees over their predictions. However, the weaknesses are that most existing deep stochastic models are based on Bayesian inferences, which requires training a posterior distribution based on the inputs, and this is usually a time-consuming process.

That said, these methods have set drastically higher benchmarks in accuracy for a wide range of machine learning tasks [41], such as image recognition, natural language processing, and speech processing. However, the downside of these approaches is that they lack robustness to high levels of data uncertainty [42], especially when the data carries high levels of uncertainty [42], so good performance in these situations is hard to guarantee. As a result, researchers have struggled to harness the power of DNNs for low signal-to-noise ratio tasks – for example, managing a control system in a noisy scenario. Data uncertainty comes from issues like sample corruptions [43], adversarial attacks [44], and the growing complexity of data sources [45]. In fact, the increasing level of uncertainty in data today is becoming one of the most pressing issues to deal with when designing algorithms for DNNs.

Currently, a powerful and effective solution for dealing with uncertainty is to

build a model through a fuzzy inference system, which relies on fuzzy logic [46]–[48]. As learning machines that find the parameters of fuzzy systems (i.e., fuzzy sets, fuzzy rules), FNNs [49]–[53] comprise an antecedent and a consequent component that offer a specific architecture for tackling data uncertainty. FNNs deal with data uncertainty through 'fuzzification' operations and architectures based on if-then rules. Notably, FNNs do not handle data uncertainty particularly well [52], [53]. In fact, generally, the gradient vanishes at the *fuzzy AND* operation when the input dimensionality is too high. As a result, they tend to suffer from a bottleneck problem where high-dimensional datasets cannot be processed. This has limited the expansion of fuzzy models to wider applications (see Fig. 1.1), and so we find most of the current FNNs are designed for control systems, where the learning features are often low-dimensional [54], [55]. In addition, FNNs cannot make use of unlabelled data, which means much useful information that could help learning is ignored. Last, because the uncertainty they do handle is only taken care of in centralised scenarios, FNNs are not able to handle privacy issues.

1.3 Distributed Fuzzy Neural Networks

As mentioned, FNNs cannot handle data uncertainty in decentralised scenarios. To address this issue, DFNNs [7]–[9], [56], [57] learn a global FNN by combining many local models. However, existing DFNNs are fragile to non-IID data. In addition, as DFNNs tend to learn a shared group of global rules for all clients, their ability to personalise the model to the local client is limited, and their learned global rules are less adaptive. Further, they regard the process of integrating the local models as a convex optimisation problem, solving it with the alternating direction method of multipliers (ADMM) [58]. However, this overlooks the powerful learning ability of feedforward FNNs.

Overall, as depicted in Fig.1.1, DFNNs face four key issues: their inability to deal with high levels of uncertainty in the data; a lack of ability to personalise local models, which is especially important when the data is heterogeneous or non-IID; the bottlenecks created by data of high dimensionality; and the inability to use unlabelled data. These challenges are described in Subsections 1.3.1 through 1.3.4.

1.3.1 The Challenge of Limited Fuzzy Reasoning Ability

Existing DFNNs follow the traditional architectures of FNNs, where the fuzzy consequent layers are single levels of fuzzy connected layers. However, this kind of

linear map is limited in its ability to reason about data with high levels of uncertainty, especially when the uncertainty is introduced via different types of noise. In today's real-world applications, data with high levels of uncertainty is quite common. In addition, the current distributed learning schemes for FNNs formulate the task to be completed as a convex problem to be solved by ADMM. Yet this scheme needs the consequent layers of FNNs to have a closed-form solution. This eliminates the possibility of applying a complex consequent layer, in turn limiting the fuzzy reasoning ability of existing DFNNs.

1.3.2 The Challenge of Data Hetetorgeneity

Data heterogeneity is attracting serious attention in both society and the research community, especially post the Facebook data privacy scandal [5]. Usually, data heterogeneity stems from the nature of the information to be captured and/or the methods used to generate or acquire the data. Further, it can exist at either the sample, the feature level, or both. Traditional data processing methods, such as data cleaning, data integration, dimension reduction, and data normalisation, may need to be applied in combination to ensure they effectively reduce data disparity [59].

Heterogeneous data poses many challenges to DFNNs. The enormous scale of the data required to train a deep model today, category imbalances, and the inherent non-IIDness of data make almost every aspect of working with a DFNN difficult, from providing enough processing power to maintaining model accuracy to processing data uncertainty. Moreover, DFNNs address privacy concerns by combining many local models to learn a global FNN; however, existing DFNNs are fragile to non-IID data. In addition, as DFNNs tend to learn a shared group of global rules for all clients, their ability to personalise the model to local clients is limited, and their learned global rules are not very adaptive.

1.3.3 The Curse of Dimensionality Challenge

In a standard fuzzy system, the number of fuzzy rules increases exponentially with the number of system variables [60]–[62]. This limits the feasibility of applying standard FNNs to heterogeneous big data. In an attempt to overcome the rule explosion problem, Raju, as far back as 1991, proposed the first hierarchical fuzzy system [60]. It consisted of a number of low-dimensional fuzzy systems connected in a hierarchical structure such that the total number of rules increased linearly with the number of input variables. In the years since then, hierarchical fuzzy systems have

been studied in depth [62]–[65] and have been applied to many practical problems, such as price negotiation [66], video de-interlacing [67], linguistic attribute hierarchy [68], weapon target assignment [69], cloud resources demand prediction [51], and more. However, there is very little literature on using hierarchical fuzzy neural networks (HFNNs) to address uncertainty and privacy concerns nor what to do about the high computational overhead associated with processing big data.

1.3.4 The Challenge of Unlabelled Data Utilization

Over the past few decades, semi-supervised learning (SSL) algorithms have been thoroughly investigated for ways to use both labelled and unlabelled data to train better models [70]–[77]. The different categories of SSL algorithms generally include self-training [71]–[73], co-training [74], graph-based methods [75]–[77], and MixUp-based methods [78], [79]. Most of these methods work by adding regularisation terms into the loss functions as a way of dealing with unlabelled samples. Further, most of the methods only work in centralised computing scenarios, where training data is processed at a central node. However, rising concerns over data privacy and security [80]–[83] have made distributed computing a far more popular paradigm. Consequently, in many real-world scenarios [4], [5], [84], the available training data is spread across interconnected networks comprising multiple agents. Typically, these agents are not allowed to share the data they have with others and can only communicate non-sensitive information to their neighbours. As such, the training data must be stored and processed on multiple local nodes instead of in one central place. To date, only a few researchers have investigated distributed semi-supervised learning (DSSL) [85]–[87], and none have considered ambiguity or uncertainty in the training samples [85]–[87]. Both properties are common in real-world datasets, so a framework that considers these factors would invariably increase model performance, especially with complex data.

Notably, fuzzy systems have been used with SSL methods for years and have found their way into many applications [88]–[92]. But actual fuzzy SSL algorithms, on the other hand, are still in their infancy, and the few that do exist usually rely heavily on human knowledge and can only be processed in a centralised way.

1.4 Research Aims

The aim of this research is to develop effective and efficient distributed models using RFNNs. Unlike today's DFNNs, which are limited in processing low-dimensional labelled data, the proposed RFNNs, specifically designed for local clients, will be

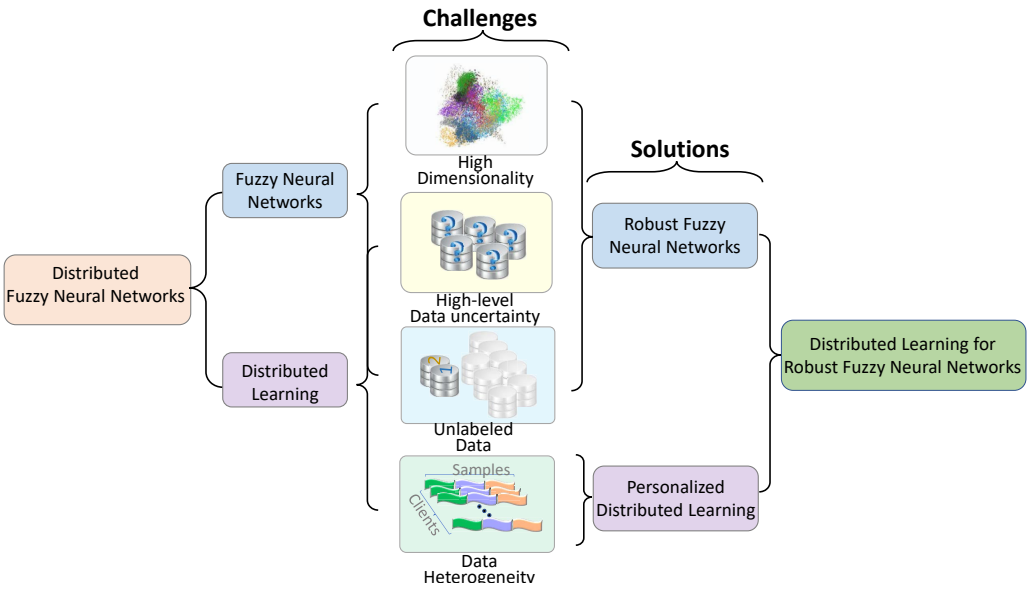


Figure 1.1: Research map of distributed learning for robust fuzzy neural networks.

able to generate more meaningful fuzzy rules because they can learn from both labelled and unlabelled samples. In addition, RFNNs are also capable of delivering robust performance when dealing with data that have high levels of uncertainty, are heterogeneous, or have high dimensionality. In terms of the distributed learning scheme, this research will not just simply merge an existing distributed learning model with an FNN. Rather, we propose a novel distributed learning scheme that solves the current problems with: computing and communication overheads; the lack of ability to deal with non-IID data; and performance problems when handling heterogeneous data among local clients. Additionally, the new RFNN architecture proposed can build highly generalisable models to handle complex real-world tasks in a distributed setting where the data has high uncertainty, has high dimensionality and are heterogeneous. In addition, RFNNs will not only improve the reasoning ability of local models by leveraging useful information from unlabelled data, but the global fuzzy rules will also be applied selectively to support the local data available.

Overall, this study answers five main research questions (RQs):

- **RQ1**: Can we design a new local FNN that solves the curse of dimensionality as well as performs robust to high levels of data uncertainty?
- **RQ2**: Can we design a new FNN that learns from unlabelled data to enhance the fuzzy reasoning ability of a DFNN?

- **RQ3**: Can we design a novel DFNN architecture that protects privacy while also handling data uncertainty and heterogeneity that lowers computation and communication overheads?

To answer these questions, we devised four research objectives (ROs):

- **RO1**: (Aims to answer RQ1) Develop a robust fuzzy framework that is able to cope with datasets of high dimensionality and high uncertainty. The solution should include a novel inference engine that does not suffer from the vanishing gradient issue associated with current fuzzy AND operations. The inference engine should be able to learn adaptively from big data with high levels of uncertainty, and, in describing uncertainty, it should generate robust fuzzy sets that outperform the current benchmarks. Flexible consequent layers should be used to handle more complex tasks. Thus, Research Objective 1 involves developing a new and effective architecture for FNNs, equipped with these components, that is robust to data uncertainty and is able to process high-dimensional samples.

- **RO2**: (Aims to answer RQ2) Design an FNN that investigates and leverages the uncertainty in unlabelled data. Today, there are massive amounts of unlabelled data that current FNNs cannot exploit. Yet making use of these valuable features during both the structure learning stage and the parameter stage would significantly contribute to enhancing the reasoning capabilities of the fuzzy rules. Therefore, Research Objective 2 involves developing an algorithm that can comprehensively consider the distributions in unlabelled data so as to build more representative fuzzy sets. Moreover, these unlabelled data should also be incorporated into the consequent layer learning. Alongside this, a novel consequent learning method needs to be devised that is able to learn a more precise decision boundary for the uncertain data.

- **RO3**: (Aims to answer RQ3) Design a distributed learning model for FNNs that can handle data heterogeneity and non-IID issues on local clients. In real-world applications, the training samples located on local clients are usually heterogeneous, and training features from different clients may vary hugely in terms of distribution. Moreover, the numbers of samples in different categories can be unbalanced. These data heterogeneity issues make it difficult for distributed learning models to generate a global optimal model that produces good performance for all local clients. Furthermore, the growing number of clients also leads to a decrease in computing efficiency and gives rise to privacy concerns. Thus, Research Objective 3 involves designing a novel learning

scheme that encourages local clients to learn their own personalised FNNs instead of a global shared one. Considering that local data are heterogeneous, not all the rules learned from one client will be helpful to all others. Some may positively contribute, but others may be misleading. Hence, the newly-devised local FNN should be an expert in processing its own client data and should only share supportive rules, not misleading rules.

1.5 Solutions

Corresponding to RO1, our solution is an RFNN with an adaptive inference engine (AIE) that is capable of dealing with different levels of uncertainty in high-dimensional data. Inspired by the idea of the relational network [93], which uses DNNs to learn a distance metric so as to compare samples in few-shot learning tasks [94], the AIE is designed as a learnable neural network module that can automatically adapt itself to learn membership function values and generate corresponding firing strengths. Unlike relational networks, which directly concatenate representations of the samples to be compared, our AIE uses non-linear mapping to transform the outputs of the antecedent component into more representative firing strengths. Equipped with this AIE, the RFNN not only avoids vanishing gradients when processing high-dimensional data, it can also process high levels of uncertainty in the membership function values. The RFNN consists of three components: an antecedent component, the AIE, and a consequent component. The AIE connects the antecedent component and the consequent component in sequence to implement the fuzzy if-then rules. Unlike deep statistical models, the RFNN has fuzzy logic built right into its structure instead of needing to learn a posterior distribution of the model weights. This eliminates much of the need to compute distributions, which reduces overall overheads. Moreover, unlike existing FNNs, the RFNN includes an FNN-based inference engine to further process any uncertainty that might exist in the membership function values. In this way, RFNNs generate more suitable firing strengths. Neural networks are used as consequent layers, which work as a non-linear estimator of the input samples. This enhances the reasoning ability of the fuzzy rules. The full structure of the RFNN is trained via backpropagation without extra hyperparameters.

In terms of solving RO2, we propose a fuzzy SSL method that can better utilise unlabelled samples, which was motivated by a recently developed technique called interpolation consistency regularisation (ICR) [79]. Unlike other SSL methods, which essentially use unlabelled data to supplement the available training data, ICR expands the sample space to capture more extrinsic information. ICR not only helps to train a better estimator based on augmented training samples but also en-

courages consistency between predictions based on both augmented samples, i.e., $f(\lambda x_i + (1-\lambda)x_j)$, and the interpolated predictions based on those samples, i.e., $\lambda f(x_i) + (1-\lambda)f(x_j)$. Further, ICR can push the decision boundaries toward low-density areas, which increases the model's robustness and generalisation performance. Currently, ICR methods have been widely used in semi-supervised classification tasks with backpropagation training schemes [79], [95] but seldom with semi-supervised regression tasks.

To meet RO3, we propose a FedFNN with ERL. This implementation handles data uncertainty and non-IID issues. The theory of biological evolution [96] states that variants of the same species can evolve to adapt to their different living environments by selectively activating and expressing their genes. Inspired by this, we designed an architecture that uses RFNNs as local models. We see these as compositions of fuzzy rules that capture valuable local data information, such as distributions, from multiple views. Similar to a specie's genes, each direction of the FedFNN is an essential functional component that can be activated or deactivated for clients according to their performance on local data. Thus, FedFNN aims to generate a group of global fuzzy rules that can be selectively activated for each local client. The result is an architecture that outperforms competing approaches with non-IID data.

1.6 Structure of this Dissertation

Chapter 2 starts with a comprehensive literature review of the problems with existing distributed fuzzy models. Chapter 3 describes the typical structure of an FNN along with the ADMM distributed algorithm. Chapters 4, 5, and 6 present the three models proposed in this research, which solve several significant challenges faced by existing distributed fuzzy models. These three chapters offer detailed problem modelling and experiments to verify efficacy. The thesis concludes in Chapter 7 with a summary of the material covered and our intentions for future work.

Chapter 2

Literature Reviews

This chapter provides a review of the existing literature that describes the primary challenges posed by big data and the corresponding machine-learning solutions. The literature reviews can be categorized into four main areas: 1) Privacy preservation and its solutions, which encompass distributed learning and federated learning; 2) Data heterogeneity and its solutions involve robust deep neural networks; 3) Unlabeled data and semi-supervised learning methods; and 4) Data uncertainty and its solutions, which include fuzzy neural networks and deep probabilistic models. These literature reviews offer valuable insights into these specific issues of big data.

2.1 Privacy Preserving Algorithms

The existing literature on privacy-preserving mostly targets non-fuzzy machine learning and deep learning algorithms. In non-fuzzy machine learning techniques, computing with encrypted data [81], privacy-preserving identification [82], and privacy-preserving probabilistic inference [83] have been used to protect privacy. In addition, schemes like differential privacy [93] are becoming increasingly popular for protecting privacy with machine learning models. Methods to incorporate DP have been developed for principal component analysis [94], support vector machines [95], and risk minimization [96]. In terms of conventional deep learning methods, [97] proposed a deep learning system that allows participants to independently train and share a small model's key parameters, thereby preserving the privacy of data for participants, while still benefiting from the contributions of others. In addition, [37] applying differential privacy and secure multi-party computation to independently trained neural networks before aggregation, helps to protect sensitive information when it is passed between networks.

Unfortunately, none of these algorithms are suitable for fuzzy neural networks. Anonymizing sensitive data to preserve privacy suffers from information loss and attributes and links can still be disclosed. Langari et al. [98] looked to address these problems with a hybrid anonymizing algorithm based on K-member fuzzy clustering and the Firefly algorithm (KFCFA). The approach considering K-anonymity (KA)[99] and its extensions, L-diversity (LD)[100] and T-closeness (TC) [101] as its three constraints.

2.2 Heterogeneous data

As a way to resolve conflicts in heterogeneous data merged from a variety of sources, Li et al. proposed an optimization framework to minimize the overall deviation between ground truths and the observations provided [102]. Zhang et al.'s solution combined principal component analysis (PCA) to reduce the number of dimensions with correlation analysis to investigate the interactive relations between features[103]. However, important information can be lost with this strategy, as PCA and correlation analysis do not consider nonlinear representative and interactive relations between features. A method based on support vector machine (SVM) was developed by Lewis et al. to infer functional gene annotations from heterogeneous data comprising protein sequences and structures [104]. Chen et al. developed a stacked denoising autoencoder to learn feature representations from hierarchical human mobility data. With this approach, they were able to predict the risk of traffic accidents [105]. Zuo et al. [106] proposed a fuzzy heterogeneous method to address domain adaptation with heterogeneous data spaces. However, none of the above methods [102]–[106] are a suitable solution to the uncertainty, scale issues and privacy concerns associated with big data environments.

2.3 Distributed Learning

Distributed learning algorithms [15]–[19] have been successfully applied to many real-world applications, including wireless sensor networks [16] and privacy-preserving [107]. Qin et al. [16] combined graph theory with the distributed consensus theory of multi-agent systems to design two decentralized algorithms for dealing with distributed problems in wireless sensor networks. One algorithm is based on K-means, the other on FCM. Targeting privacy preservation, Liu et al. [107] devised a novel shadow coding schema to save and recover privacy information over distributed networks. Two interesting works in **ye2020distributed1** and **ye2020distributed2**

provided novel learning algorithms for distributed games. Deep learning solutions involving consensus models over distributed deep architectures have also been proposed with the aim of improving performance [108], [109]. For example, Forero et al. [15] proposed a fully distributed method based on support vector machine algorithms by applying an ADMM strategy to obtain optimal global parameters without the need to exchange and process information through a central communications unit. Ye et al. [18] later presented a decentralized ELM algorithm that combines the Jacobian and Gauss-Seidel Proximal ADMM methods. Obviously, all these distributed learning algorithms do well in alleviating privacy concerns and dealing with decentralized scenarios. However, none address uncertainty in the training data.

There are also several examples of fuzzy algorithms for distributed learning [7], [8], [53], [110]. For example, Fierimonte et al. [7] developed a decentralized FNN with random weights, where parameters in the fuzzy membership functions are chosen randomly as opposed to being trained. In subsequent work, they introduced an online implementation of the same FNN structure [8]. Notably, a random method of identifying parameters can result in very large deviations in accuracy during the learning process. Additionally, the algorithms are only applied in the consequent layers of the FNN [7], [8], which means that, strictly speaking, this decentralized FNN model is only partially distributed. A more recent proposition by Shi et al. [53] involves a distributed FNN with a consensus learning strategy. A novel method of distributed clustering optimizes the parameters in the antecedent layer, while a similar method of distributed parameter learning does the same for the consequent layer. This solution successfully manages data uncertainty in a distributed setting, but it does not consider unlabeled samples. Dang et al. [110] proposed a transfer fuzzy clustering method to enable neighbouring agents to learn from each other collaboratively. Similar to distributed learning, multi-view learning converges to an optimal estimator by collaboratively learning from multiple datasets. By applying a large margin learning mechanism, [111] proposed a two-view fuzzy model that collaboratively learns from each other. A multi-view fuzzy logic system is introduced in [112] by adopting nonnegative matrix factorization to build a hidden space in order to extract useful information shared among different datasets.

2.4 Semi-supervised learning

SSL algorithms [71], [72], [74] were developed to improve model performance with additional training data in the form of unlabeled samples, while still leveraging the accuracy afforded by labelled samples. The first tools to solve SSL problems were self-training [71], [72] methods, which uses unlabeled samples in an iterative two-

part procedure. First, the model is trained on a set of only labelled samples. The trained model is then used to predict the labels of unlabeled data. High-confidence predictions are then added to the training set, and the model is retrained. An alternative strategy is co-training [74], which takes a similar iterative approach to self-training. The difference is that co-training algorithms maintain two separate estimators which work together on different training subsets and teach each other during the learning process. Subsequently, generative models [70], [113], [114] were proposed. These algorithms assume that the model can represent hybrid distributions as identified in a set of unlabeled samples.

More recently, an increasing number of researchers are incorporating regularization terms into the loss function of SSL algorithms as a way to extract more useful information from unlabeled data. Generally, these regularization terms fall into one of three categories. These are: traditional regularization [115], which summarizes and transfers traditional SSL models into regularization terms; consistency regularization [116], which forces the predictions generated to be low-entropy so the decision boundary does not land in a dense sample area; and entropy minimization [117], [118], which guarantees that the distribution of the augmented dataset will be the same as the original.

To implement SSL in deep architecture, an interpolation consistency training procedure, Verma et al. [119] proposed a solution inspired by MixUp [78] that learns a decision boundary which ensures consistency between predictions based on the interpolated samples and interpolations based on the resulting predictions. Meanwhile, Berthelot et al. [120] was developing MixMatch – the current state-of-the-art in model performance. MixMatch basically combines many of the recent dominant SSL mechanisms, including entropy minimization and MixUp tools. However, all these algorithms are designed for centralized scenarios and cannot be applied directly to decentralized problem-solving.

Only a few DSSL methods have been proposed [85]–[87] Chang et al. [85] uses unlabeled data to reduce distribution errors and applied time-consuming kernel ridge regression on distributed nodes, after which the weighted average of those nodes' outputs is calculated to obtain a final estimator. Taking advantage of a wavelet neural network, Xie et al. [86] devised a new DSSL scheme that incorporates a graph-based regularization term into a distributed loss function. However, the process involves constructing a relationship graph of all the sample points, which is quite time-consuming, especially with large-scale datasets. In pursuit of better efficiency, the researchers later modified their information-sharing strategy to use an event-triggered communication scheme [87], and in a later work still, they updated the objective function to reduce the complexity of loss function by avoiding the twice

continuously differential in their previous two models [121]. While there are many ways to reduce computing time, high computational overhead is an inherent problem with graph-based regularization that cannot be solved completely. Ultimately, the choice of DSSL strategy comes down to one of either less robustness or more time consumption.

2.5 Fuzzy Neural Networks

Fuzzy systems and sets [122], [123] were originally introduced to operate complex control systems. Over past decades, they have diverged into several types [124]–[127]. Of these different streams, T–S fuzzy models [127] have achieved great success at dealing with data uncertainty. By adopting a group of fuzzy rules and membership functions, T-S fuzzy models merge a number of local models together to create a nonlinear model with the ability to process uncertainty. FNNs [46], [128] were built upon this property by integrating fuzzy logic into a neural network structure. Using fuzzy if-then rules, FNNs not only excel at handling data uncertainty, they also make neural network structures explainable. In recent years, FNNs have been applied to a range of scenarios, such as distributed learning [7], [53], robot–environment interaction [129] and mimic habitual sequential tasks [130]. However, current implementations struggle with two severe problems. First, they have a deep reliance on the quality of the fuzzy rules that are generated. Second, they suffer from the curse of dimensionality.

2.6 Robust Deep Neural Networks

One method of easing the issues arising from data uncertainty for DNNs is to use a regularization-based deep neural network [131]–[136]. Adopting ℓ_1 and ℓ_2 regularization on a loss function with soft weight-sharing, Nowlan and Hinton [131] observed good performance with ambiguous data by penalizing the model complexity. This was the first attempt to enhance a deep model's robustness by regularizing its parameters. Since then, most DNNs have adopted dropout [137] strategies against data uncertainty [132], [133]. By randomly dropping neurons during the training process, dropout alleviates over-reliance on the model's parameters, which prevents deep models from adapting to the inputs too much. Unfortunately, dropout only works for low levels of uncertainty, and it cannot cope with special situations like problems with adversary attacks.

In parallel, injecting deep model parameters with noise, especially Gaussian noise,

has also proven to be an effective method of processing uncertain data [134], [135]. Based on this idea, Poole et al. [136] introduced a novel Gaussian noise injection (GNI) method based on an autoencoder that adds Gaussian noise to all layers and activation functions. The strategy yielded robust results. However, simply injecting noise into the activation function of a DNN delivers equal gains, which was proven statistically and empirically by Camuto et al. [32]. Afterwards, Lu et al. [138] proposed an additional regularization technique for convolutional neural networks (CNNs) so as to yield models with greater generalizability. Here, a learnable noise module adaptively generates noises in the hidden layer to increase the robustness of CNNs. However, similar to dropout, noise injection performance relies on the structure of neural networks, and extra parameters are needed during the training process. They are less able to handle samples with high levels of uncertainty.

2.7 Distributed fuzzy neural networks

DFNNs [7]–[9], [56], [57] have been proposed to handle the uncertainties encountered in distributed applications. The authors in [7] proposed a DFNN model that randomly sets the parameters in antecedents and only updates the parameters in consequent layers. Later, they extended this work to an online DFNN model [8]. Their models assume that all clients share the information in antecedent layers, making this technically not a seriously distributed method. To avoid this problem, a fully DFNN [9] model was proposed by adopting consensus learning in both the antecedent and consequent layers. As its subsequence variant, a semisupervised DFNN model [57] was presented to enable the DFNN to leverage unlabeled samples by using the fuzzy C-means method and distributed interpolation-based consistency regularization. However, the existing DFNNs cannot handle situations in which the data distribution varies across clients. The authors in [56] proposed a DFNN with hierarchical structures to process the heterogeneity existing in the variables of training samples. However, instead of processing the data heterogeneity across distributed clients, they focused on variable composition heterogeneity, which meant that data variables were collected from different sources. Generally, by employing the well-known Takagi-Sugeno (T-S) fuzzy if-then rules [127], the existing DFNN models build the antecedent layers of their local models in traditional ways (e.g., K-means) and calculate the corresponding consequent layers with closed-form solutions. Then, the original DFNNs are transformed into convex optimization problems. While efficient and effective, they are not able to learn local models with personalized rule sets. Worse, they fail to utilize the strong learning abilities of neural networks that enable local FNNs to investigate more adaptive rules.

2.8 Deep Probabilistic Models

The existing probabilistically-directed deep generative models, which are composed of statistic inferences and deep architectures, are flexible tools for modelling uncertainty even with large-scale datasets. Using Bayesian theory to learn a posterior distribution for DNNs parameters, Bayesian neural networks (BNNs) [37] are one of the most promising deep statistic models with which to learn from complex samples. Due to the potential advantages of dealing with uncertainty in this way, most recent variants of BNNs have also been designed to handle uncertainty [139] and to detect adversarial examples [140]–[142]. The robustness of BNNs is discussed and statistically guaranteed in [40], [143]. However, most existing BNNs heavily rely on Monte Carlo approximation as a posterior inference [144], which is very time-consuming.

Beyond BNNs, Gaussian processes (GPs) [34] have found traction in a wide variety of tasks due to their distinguished performance in dealing with uncertainty [145], [146]. The downside of these methods, however, is that they were designed for specific tasks, and they are hard to generalize to other tasks, especially when the noise levels are heavy. Also, GP models are intractable and rather ineffectual with large-scale datasets. In past years, deep Gaussian processes (DGPs) have greatly alleviated some of the above issues [36], [38], [147]. The most concerning problem with the current iterations of DGPs is that they focus on regression tasks and are seldom used for classification problems. Further, similar to BNNs, the training process of DGPs relies on Bayesian inference and is less time efficient.

2.9 Federated Learning

FL [21] is an emerging distributed paradigm in which multiple clients cooperatively train a neural network without revealing their local data. Recently, many solutions[21], [148], [149] have been presented to solve FL problems, among which the most known and basic solution is federated averaging (FedAvg) [21], which aggregates local models by calculating the weighted average of their updated parameters.

However, FL has encountered various challenges [150], among which the non-IID issue is the core problem that makes the local model aggregation process harder and leads to performance degradation. Numerous FL algorithms have been presented to solve the non-IID problem, e.g., stochastic controlled averaging for FL (SCAFFOLD) [24]; FedProx [25]; model-contrasted FL (MOON) [151], which attempts to increase the effect of local training on heterogeneous data by minimizing the dissimilarity between the global model and local models; FedMA [29] and FedNova [152], which improve the aggregation stage by utilizing Bayesian nonparametric methods

and local update normalization, respectively; CCVR [153], which calibrates the constitutive classifiers using virtual representations to eliminate the global model bias caused by local distribution variance; and FedEM [154], which introduces expectation maximization to make the learned model robust to data heterogeneity. Though these methods have been proposed based on FedAvg by trying to learn a more robust global model, they focus on learning a shared global model, which degrades their performance when the data distributions heavily vary across clients.

Recently, personalized FL (PFL) [26], [27], [155], [156] has been proposed; this approach aims to process heterogeneous local data with personalized models. Many of the existing PFL methods were proposed to solve the distributed meta-learning problem [156]–[159]. Among the methods that target normal PL tasks, multitask learning [160] is applied to learn personalized local models by treating each client as a learning task; model mixing [161], [162] achieves the same goal by allowing clients to learn a mixture of the global model and local models. Notably, the authors in [163] presented a new local model structure that comprises a global feature encoder and a personalized output layer. By contrast, LG-FedAvg provides clients with a local feature encoder and a global output layer.

However, very few of the mentioned FL methods are able to handle data uncertainties, except for that of [29], [152], which adopts Bayesian treatment, and that of [27], which adopts a Gaussian process. In addition, building Bayesian posteriors and Gaussian kernels is time-consuming. In contrast, our study uses FNNs as local models, which are viewed as assemblies of fuzzy rules. Thus, taking rules as basic functional units, we break down the task of learning a global model into learning global fuzzy rules, each of which can independently investigate its local sample space and contribute to the local training process.

2.10 Summary

In this chapter, we have discussed the significant challenges posed by big data and examined the current solutions available. While some solutions have shown reasonable success in addressing specific issues, such as deep learning (DL) in handling data privacy and feedforward neural networks (FNNs) in processing data uncertainty, none of them have been able to simultaneously address and achieve state-of-the-art performance in all areas, including data uncertainty, data heterogeneity, privacy preservation, and effective utilization of unlabeled data. This highlights the importance of the contribution made by this work, as it aims to address these challenges comprehensively and advance the current state-of-the-art.

Chapter 3

Preliminary

In this chapter, three key base models are described to lay a foundation for DFNNs. These models are the centralized FNNs, the centralized HFNNs, and the distributed HFNNs, respectively. The centralized FNNs are the basis of all FNNs-related models, including DFNNs. The centralized HFNNs can be seen as more general structures of FNNs, which are able to process uncertain data with features collected from multiple sources. Based on the centralized HFNNs, the distributed HFNNs are more general in processing data uncertainty for decentralized tasks.

3.1 Centralized Fuzzy Neural Networks

Here, the architecture of FNN that is applied with the first order of the Takagi-Sugeno (T-S) method of fuzzy inference system is described. Taking $x = [x_1, x_2, \cdots, x_D]$ as the input data with a dimensionality of D and its corresponding target output as $y \in \mathbb{R}$, then the k-th fuzzy rule of the T-S system is

$$\begin{gathered}\text{Rule } k\text{: IF } x_1 \text{ is } A_{k1} \text{ and } \cdots \text{ and } x_D \text{ is } A_{kd} \\ \text{Then } y = w_{k0} + \textstyle\sum_{j=1}^{D} w_{kj} x_j\end{gathered}$$

where A_{kj} is a Gaussian membership fuzzy set and its membership function is calculated via,

$$\varphi_{kj}(x_j) = \exp\left[-\left(\frac{x_j - m_{kj}}{\sigma_{kj}}\right)^2\right] \tag{3.1}$$

where m_{kj} and σ_{kj} are the mean and standard variance of the Gaussian membership function, respectively. Usually, the FNN consists of four feed-forward layers, whose structure is provided in Fig.3.1.

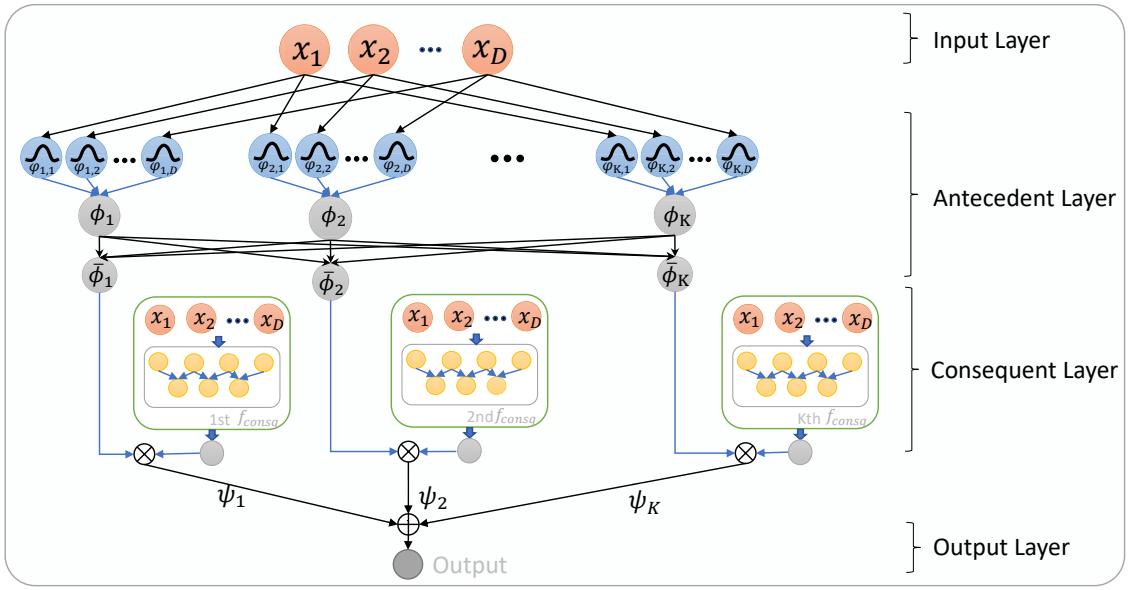


Figure 3.1: The structure of FNN

The first layer is the input layer, where each node stands for a single input feature, i.e. x_i. This layer is responsible for receiving the input features and then transmitting them to the next layer.

The second layer is the antecedent layer, where each node corresponds to one fuzzy set and outputs a membership value according to (6.1). Afterwards, firing strengths corresponding to these fuzzy sets can be calculated by applying fuzzy AND operation:

$$\phi_k(x) = \prod_{j=1}^{D} \varphi_{kj}(x_j), \tag{3.2}$$

where $\phi_k(x)$ is the firing strength of fuzzy rule k. The obtained firing strength is then normalized by

$$\bar{\phi}_k(x) = \frac{\phi_k(x)}{\sum_{k=1}^{K} \phi_k(x)}. \tag{3.3}$$

where K is the total number of fuzzy rules.

The third layer is the consequent layer, where each node performs a defuzzification process for each fuzzy rule k using a weighted average operation as follows:

$$\psi_k(x) = \bar{\phi}_k(x)(w_{k0} + \sum_{j=1}^{D} w_{kj} x_j), \tag{3.4}$$

The fourth layer is called the output layer, which calculates the overall output by summing the outputs of all fuzzy rules in the third layer as follows,

$$\hat{y} = \sum_{k=1}^{K} \psi_k(x), \tag{3.5}$$

Generally, FNN involves the identification of structure and parameters. The structure learning is to identify the parameter of the Gaussian membership function in the antecedent layer for each fuzzy rule k, i.e. m_{kj} and σ_{kj}, $j \in \{1, \cdots, D\}$; the parameter learning is to identify the output weights $w_{k0}, \cdots, w_{kD}$ in the consequent layer. Both structure learning and parameter learning can be addressed via gradient back propagation[164].

3.2 Centralized Hierarchical fuzzy neural networks

Hierarchical fuzzy neural networks consist of a two-level hierarchical structure. The lower level contains multiple FNNs, and the outputs of these FNNs are in the higher level. During the data processing stage, heterogeneous big data is segmented at both the feature level and the sample level. In feature-level segmentation, the heterogeneous features are assigned to multiple subsets by a feature splitter to ensure that each subset only contains homogeneous features. Further, each low-level FNN is responsible for only one subset. The samples in the sample-level segmentation for each low-level FNN are further randomly assigned to multiple agents. A distributed computation framework is then applied to these agents in each low-level FNN to preserve data privacy and to deal with the enormous amounts of data samples. The above data processing and HFNN structure are shown in Fig.3.2.

Suppose $\mathcal{D} := \{(X_i, Y_i) |\ X_i \in \mathbb{R}^{|\mathcal{F}|}, Y_i \in \mathbb{R}, i \in \mathcal{N}\}$ denote the set of training data, where $\mathcal{F}$ and $\mathcal{N}$ represents the set of features and samples, respectively, and $|\cdot|$ is the cardinality operator. As shown in Fig.3.2, the data is first processed blackby a feature splitter where the full set of heterogeneous features is transformed into multiple independent subsets of homogeneous features and assigned to a low-level FNN. Let $\mathcal{F}_b, b \in \{1, 2, ..., B\}$ denote the subset of homogeneous features associated with the b-th low-level FNN. Then in each low-level FNN, the training data is further segmented by a sample distributor. $\mathcal{N}_{b,l}$ denotes the subset of training data assigned to the l-th agent ($l \in \{1, 2, ..., L\}$) of the b-th low-level FNN. Accordingly, the sample vector is collected in $X_i^{b,l}$, $\forall i \in \mathcal{N}_{b,l}$ with $x_{i,j}^{b,l}$ denoting its j-th component.

Next, let us briefly recall the structure of the low-level FNNs, which follows the first-order of the Takagi-Sugeno method for fuzzy inference systems and has a layer-

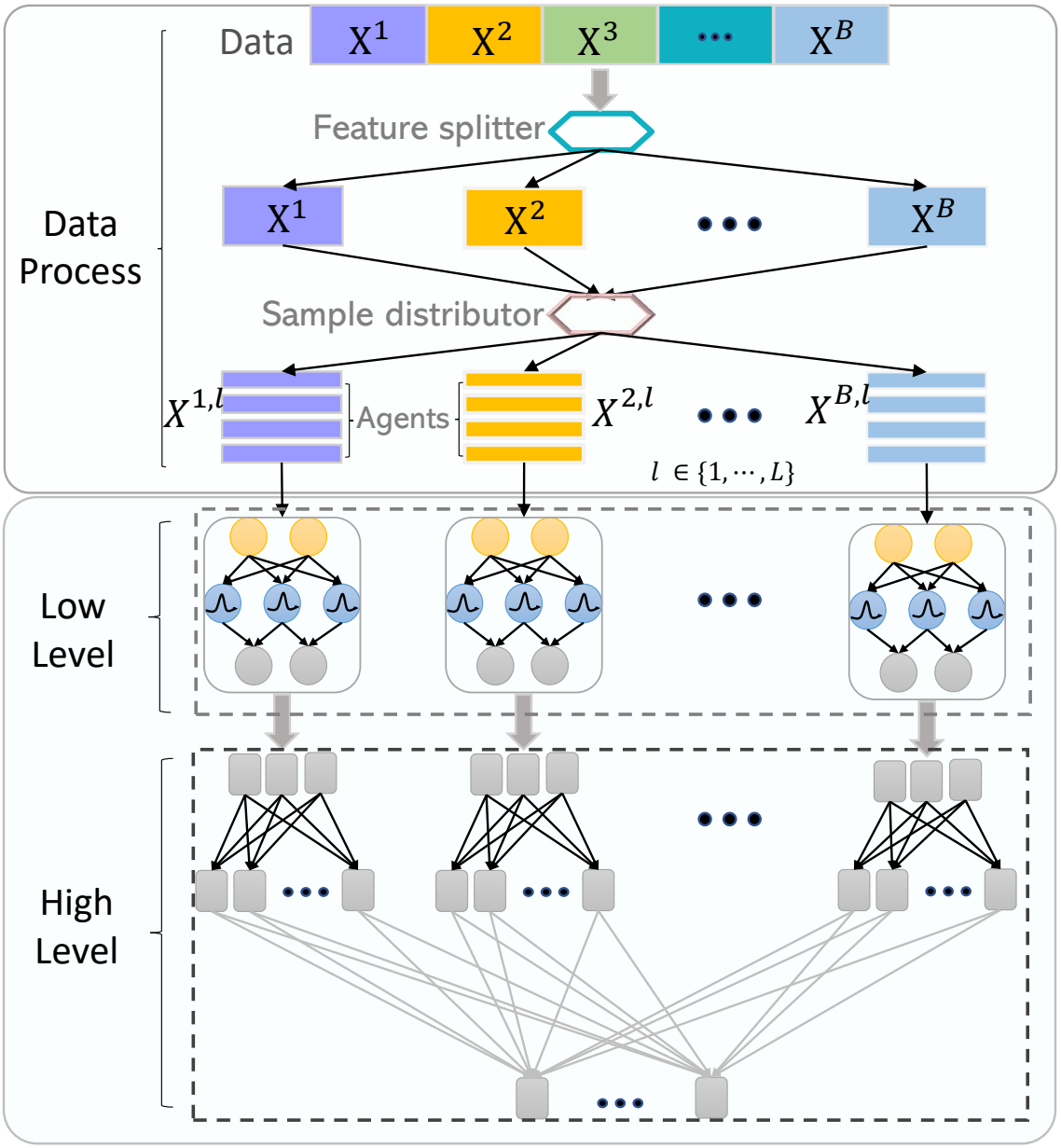


Figure 3.2: Data processing and PP-HFNN structure.

by-layer network structure [128]. Now, suppose the output of the l-th agent of the b-th low-level FNN is z_b^l, then the k_b^l-th fuzzy rule can be expressed as

$$\text{Rule } k_b^l\text{: IF } x_{i,1}^{b,l} \text{ is } A_{r,1}^{b,l}, \cdots \text{ and } x_{i,|\mathcal{F}_b|}^{b,l} \text{ is } A_{r,|\mathcal{F}_b|}^{b,l}$$
$$\text{Then } z_{r_b^l} = w_{k,0}^{b,l} + \sum_{j=1}^{|\mathcal{F}_b|} w_{k,j}^{b,l} x_{j,1}^{b,l},$$

where $A_{k,j}^{b,l}$ is a Gaussian fuzzy set of the j-th input of rule k_b^l, and $w_{k,j}^{b,l}$ is the corresponding weight of the consequence. The membership function of $A_{k,j}^{b,l}$ can be written as

$$\varphi_{k,j}(x_{i,j}^{b,l}) = \exp\left[-\left(\frac{x_{i,j}^{b,l} - m_{k,j}^{b,l}}{\sigma_{k,j}^{b,l}}\right)^2\right] \tag{3.6}$$

where $m_{k,j}^{b,l}$ and $\sigma_{k,j}^{b,l}$ are respectively the center and width of the corresponding fuzzy set. Generally, each low-level FNN is composed of four feed-forward layers, as shown on the right-hand side of Fig.3.2.

Layer 1 is the antecedent layer: its inputs are $x_{i,j}^{b,l}$, and its outputs are derived through the fuzzification process outlined in (6.1).

Layer 2 is the rule layer: each node in this layer represents a fuzzy rule, which uses the AND operation to match the outputs of the antecedent layer as follows:

$$\phi_k(X_i^{b,l}) = \prod_{j=1}^{|\mathcal{F}_b|} \varphi_{k,j}(x_{i,j}^{b,l}), \tag{3.7}$$

where $\phi_k(X_i^{b,l})$ is the firing strength of fuzzy rule k_b^l, and then normalized by

$$\bar{\phi}_k(X_i^{b,l}) = \frac{\phi_k(X_i^{b,l})}{\sum_{k=1}^{K_l^b} \phi_r(X_i^{b,l})}. \tag{3.8}$$

where K_l^b is the total number of fuzzy rules in the l-th agent of the b-th low-level FNN.

Layer 3 is the consequent layer: each node here performs a defuzzification process for each fuzzy rule r_b^l using a weighted average operation as follows:

$$\psi_k(X_i^{b,l}) = \bar{\phi}_k^{b,l}(X_i^{b,l})(w_{k,0}^{b,l} + \sum_{j=1}^{|\mathcal{F}_b|} w_{k,j}^{b,l} x_{j,1}^{b,l}), \tag{3.9}$$

Layer 4 is the output layer: the overall output of the l-th agent of the b-th low-level FNN is derived by summing the outputs of the fuzzy rules in Layer 3 as follows:

$$z_i^{b,l} = \sum_{k=1}^{K_l^b} \psi_k^{b,l}(X_i^{b,l}). \tag{3.10}$$

For each l-th agent of the b-th low-level FNN, a matrix is defined as $H^{b,l} := [H_1^{b,l}, \cdots, H_K^{b,l}]$, where $H_k^{b,l} \in \mathbb{R}^{|\mathcal{N}| \times (|\mathcal{F}_b|+1)}$, and

$$H_k^{b,l} = \begin{bmatrix} \bar{\phi}_k(X_1^{b,l}) & \bar{\phi}_k(X_1^{b,l}) x_{1,1}^{b,l} & \cdots & \bar{\phi}_k(X_1^{b,l}) x_{1,|\mathcal{F}_b|}^{b,l} \\ \bar{\phi}_k(X_2^{b,l}) & \bar{\phi}_k(X_2^{b,l}) x_{2,1}^{b,l} & \cdots & \bar{\phi}_k(X_2^{b,l}) x_{2,|\mathcal{F}_b|}^{b,l} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\phi}_k(X_{|\mathcal{N}|}^{b,l}) & \bar{\phi}_k(X_1^{b,l}) x_{|\mathcal{N}|,1}^{b,l} & \cdots & \bar{\phi}_k(X_1^{b,l}) x_{|\mathcal{N}|,|\mathcal{F}_b|}^{b,l} \end{bmatrix}$$

The output vector is

$$Z^{b,l} := [z_1^{b,l}, \cdots, z_{|\mathcal{N}|}^{b,l}]^T \in \mathbb{R}^{|\mathcal{N}|}.$$

And the weight vector is

$$\mathbf{w}^{b,l} := [w_{1,0}^{b,l}, \cdots, w_{1,|\mathcal{F}_b|}^{b,l}, \cdots, w_{K_l^b,0}^{b,l}, \cdots, w_{K_l^b,|\mathcal{F}_b|}^{b,l}]^T.$$

Eq. (3.10) is equivalent to the following matrix form:

$$Z^{b,l} = H^{b,l} \mathbf{w}^{b,l}. \tag{3.11}$$

It is worth noting that $H^{b,l}$ is dependent on the centers and widths of each rule's fuzzy sets, i.e., $m_k^{b,l}$ and $\sigma_k^{b,l}$ as well as the data input $X_i^{b,l}$.

The outputs of the low-level FNNs are coordinated by a fully-connected layer in the high level of the hierarchy. For each agent l,

$$Y^l = \sum_{b=1}^{B} Z^{b,l} v^{b,l}, \tag{3.12}$$

where $Y^l := [Y_1^l, \cdots, Y_{\mathcal{N}}^l]$, $v^{b,l}$ is the coordination weight of the hierarchy, and B is the number of subsets in terms of homogeneous features. Let Z^l represent the collection of outputs from each low-level FNN of the l-th agent, i.e.,

$$Z^l := [Z^{1,l}, \cdots, Z^{B,l}] \in \mathbb{R}^{|\mathcal{N}| \times B}.$$

Eq. (3.13) is equivalent to the following matrix form:

$$Y^l = Z^l \mathbf{v}^l, \tag{3.13}$$

where

$$\mathbf{v}^l := [v^{1,l}, \cdots, v^{B,l}]^T \in \mathbb{R}^B.$$

Define

$$\mathbf{m}^l := [\mathbf{m}^{1,l}, \cdots, \mathbf{m}^{B,l}],$$

$$\sigma^l := [\sigma^{1,l}, \cdots, \sigma^{B,l}],$$
$$H^l := [H^{1,l}, \cdots, H^{B,l}] \in \mathbb{R}^{|\mathcal{N}| \times n_H},$$

and

$$\mathbf{w}^l := \text{diag}(w^{1,l}, \cdots, w^{B,l}) \in \mathbb{R}^{n_H \times B},$$

where $n_H = \sum_{b=1}^{B} K_l^b(|\mathcal{F}_b| + 1)$, and diag$(\cdot)$ represents a diagonal operation on the matrix.

3.3 Distributed Hierarchical fuzzy neural networks

The training procedure for D-HFNN is to solve the following optimization problem:

$$\min_{\mathbf{m}^l, \sigma^l, \mathbf{w}^l, \mathbf{v}^l} \frac{1}{2} \sum_{l=1}^{L} (||Y^l - H^l \mathbf{w}^l \mathbf{v}^l||^2 + \frac{\lambda}{2} ||\mathbf{w}^l||^2 + \frac{\mu}{2} ||\mathbf{v}^l||^2), \tag{3.14}$$

where L is the number of agents, λ and $\mu > 0$ are factors to trade-off the training error and regularization. Selecting an appropriate value for λ and $\mu > 0$ can make the solution much more stable and generalizable[165]. The difficulty of the optimization problem (3.14) stem from three issues. One is the nonconvex nature of the Gaussian membership function in H^l. To address this first difficulty, the key issue is to identify the parameters of the centers $\mathbf{m}^l$ and the widths σ^l of each rule's fuzzy sets in the antecedent layer of each FNN. The second difficulty is the distributed computing scheme for the low-level FNNs, which is needed to manage the scale of the data and preserve privacy. The last difficulty is the coordination in the high levels of the hierarchy, where the matrix product between $\mathbf{w}^l$ and $\mathbf{v}^l$ is nonconvex. Fortunately, it is convex after fixing either $\mathbf{w}^l$ or $\mathbf{v}^l$, i.e., it becomes bi-convex.

An intuitive way to solve the nonconvex optimization (3.14) is by using a back-propagation method[164]. However, back-propagation methods often suffer from slow training speeds and gradient vanishing problems. In addition, distributed computation is not easily implemented with back-propagation. Hence, it is not an optimal choice for heterogeneous big data environments. A more suitable option is an algorithm that is fast to converge with a massive number of samples and is easy to roll out across a distributed computation framework.

As discussed above, the first difficulty with the optimization problem (3.14) is to identify the parameters of the antecedent layer of each FNN. A simple idea is to group the input data into multiple clusters and use one rule for each cluster [166]. Here, we use the popular K-means algorithm[167] to identify the parameters of the antecedent layer. The K-means algorithm is one of the most efficient clustering

algorithms. To tackle the second difficulty of the optimization problem (3.14), we developed a distributed K-means method, inspired by [168]. As for the coordination in high-level coordination of the hierarchy, the AO method is used since it is well-suited to the bi-convex optimization problems and converges very quickly in practice.

Distributed K-means method for the low-level of HFNN

To introduce the distributed K-means algorithm, let us first recall the centralized K-mean algorithm. The goal of this algorithm is to assemble the input data into multiple clusters $\mathcal{C}_k^b$, each of which has its own center. The centralized K-means algorithm for the b-th low-level FNN of the hierarchy can be defined as

$$\mathbf{m}_k^b = \arg\min_{\mathbf{m}_k^b} \frac{1}{2} \sum_{k=1}^{K_b} \sum_{X_i^b \in \mathcal{C}_k^b} ||X_i^b - \mathbf{m}_k^b||^2 \tag{3.15}$$

where $\mathbf{m}_k^b$ is the k-th center in the antecedent layer of the b-th low-level FNN, and K_b represents the number of corresponding clusters, which is equal to the number of rules. Starting from an initial set of K centers, i.e. $\{\mathbf{m}_1^b(0), \cdots, \mathbf{m}_K^b(0)\}$, the K-means algorithm alternates between an assignment step and an update step as follows.

- Assign each observation X_i^b to the cluster $\mathcal{C}_k^b(t)$, whose center is closest to X_i^b.
- Update the center of each cluster by

$$\mathbf{m}_k^b(t) = \frac{1}{|\mathcal{C}_k^b(t)|} \sum_{X_i^b \in \mathcal{C}_k^b(t)} X_i^b.$$

Once the center of each fuzzy set is obtained by the above procedure, the corresponding standard variance $\sigma_{k,j}^b$ can be calculated as follows:

$$\sigma_{k,j}^b = \sqrt{\frac{1}{|\mathcal{C}_k^b(t)|} \sum_{i=1}^{|\mathcal{C}_k^b(t)|} (x_{i,j}^b - m_{k,j}^b)^2}. \tag{3.16}$$

It is clear that such a centralized K-means algorithm does not preserve privacy at all, but the distributed K-means algorithm does. As shown in Fig. 3.3, the data is processed locally by multiple agents. While some limited information is exchanged between agents, none of it is data. The distributed K-means is built on ADMM, which is an efficient solution for distributed computation. ADMM is an

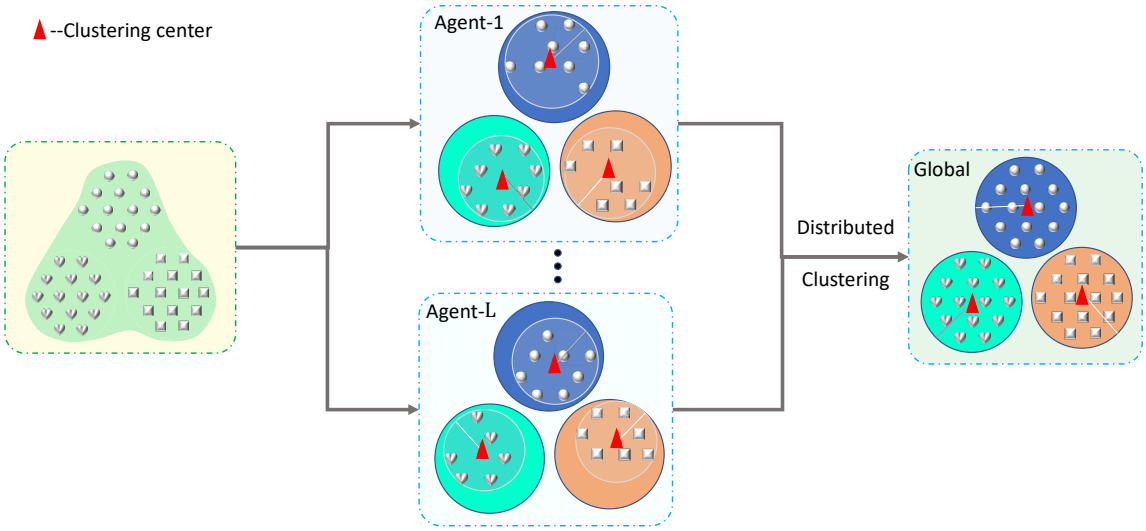


Figure 3.3: Distributed Clustering.

optimization algorithm that solves problems with multiple variables and constraints. It decomposes the problem into subproblems, updates variables independently, and enforces consensus between them. It iteratively optimizes the subproblems while maintaining consistency among the variables. ADMM is effective for large-scale and distributed optimization problems. A recent study proves the convergence of ADMM for a variety of nonconvex and possibly nonsmooth functions given some sufficient conditions[169]. Following our previous work[53], the distributed K-means algorithm solves the following optimization problem:

$$\min_{\mathbf{m}_K^l} \frac{1}{2} \sum_{l=1}^{L} \sum_{b=1}^{B} \sum_{X_i^b \in \mathcal{C}_k^b} ||X_i^{b,l} - \mathbf{m}_k^{b,l}||^2 \tag{3.17a}$$

$$\text{s.t.} \quad \mathbf{m}_k^{b,l} = \mathbf{r}_k^b, \quad l = 1, 2, \cdots, L \tag{3.17b}$$

where $\mathbf{m}_k^{b,l}$ represents the k-th local center of the l-th agent on the b-th low-level FNN, and $\mathbf{r}_k^b$ is the corresponding global center. The global standard variance of the antecedent layer of the low-level FNNs is expressed as

$$\bar{\sigma}_k^b = \sqrt{\frac{1}{|\mathcal{N}|} \sum_{l=1}^{L} |\mathcal{C}_k^{b,l}| (\sigma_k^{b,l})^2} \tag{3.18}$$

where $\bar{\sigma}_k^b$ denotes the k-th global standard variance of the k-th low-level layer, and $\sigma_k^{b,l}$ is the local standard variance of the l-th agent.

The following augmented Lagrangian is constructed for (3.17):

$$\begin{aligned}\mathcal{L}(\mathbf{m}_k^{b,l}, \mathbf{r}_k^b, \boldsymbol{\beta}_k^{b,l}) = \quad & \frac{1}{2}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{X_i^b \in \mathcal{C}_k^{b,l}} ||X_i^{b,l} - \mathbf{m}_k^{b,l}||^2 \\ & + \sum_{l=1}^{L}\sum_{k=1}^{K} \boldsymbol{\beta}_k^{b,l^T}(\mathbf{m}_k^{b,l} - \mathbf{r}_k^{b,l}) \\ & + \frac{1}{2}\rho \sum_{l=1}^{L}\sum_{k=1}^{K} ||\mathbf{m}_k^{b,l} - \mathbf{r}_k^{b,l}||^2 \end{aligned} \tag{3.19}$$

where $\boldsymbol{\beta}_k^l$ is the Lagrange multiplier, and ρ is a positive penalty parameter. The variables are then updated iteratively with the following procedure based on the ADMM:

$$\mathbf{m}_k^{b,l}(t+1) = \arg\min_{\mathbf{m}} \mathcal{L}(\mathbf{m}_k^{b,l}, \mathbf{r}_k^b(t), \boldsymbol{\beta}_k^{b,l}(t)), \tag{3.20}$$

$$\mathbf{r}_k^b(t+1) = \arg\min_{\mathbf{r}_k^{b,l}} \mathcal{L}(\mathbf{m}_k^{b,l}(t+1), \mathbf{r}_k^b, \boldsymbol{\beta}_k^{b,l}(t)), \tag{3.21}$$

$$\begin{aligned}\boldsymbol{\beta}_k^{b,l}(t+1) = \boldsymbol{\beta}_k^{b,l}(t) + \rho \sum_{l=1}^{L}\sum_{k=1}^{K} (\mathbf{m}_k^{b,l}(t+1) - \\ \mathbf{r}_k^b(t+1)),\end{aligned} \tag{3.22}$$

where t is the number of iterations. It is worth noting that the distributed K-means algorithm is applied to each low-level FNN and the centers $\mathbf{m}_k^{b,l}$ can be updated in parallel by different agents. Additionally, there is a closed-form solution for (5.24) as follows:

$$\mathbf{r}_k^{b,l}(t+1) = \frac{1}{L\rho}\sum_{l=1}^{L}(\boldsymbol{\beta}_k^{b,l}(t) + \rho \mathbf{m}_k^{b,l}(t+1)) \tag{3.23}$$

Convergence depends on the following two criteria:

$$||\mathbf{m}_k^{b,l}(t) - \mathbf{r}_k^b(t)||^2 \leq \epsilon_1, \tag{3.24}$$

$$||\boldsymbol{\beta}_k^{b,l}(t+1) - \boldsymbol{\beta}_k^{b,l}(t)||^2 \leq \epsilon_2. \tag{3.25}$$

Alternating optimization for the high-level of HFNN

In the second stage, the AO method is used to obtain the parameters $\mathbf{w}$ and $\mathbf{v}$ in high levels of the hierarchy. Since the raw data is only processed in the first stage, it

is not necessary to use distributed computation for the high-level coordination from a privacy-preserving perspective. Therefore, the optimization problem (3.14) can be simplified to:

$$\min_{\mathbf{w},\mathbf{v}} \frac{1}{2}||Y - H\mathbf{w}\mathbf{v}||^2 + \frac{\lambda}{2}||\mathbf{w}||^2 + \frac{\mu}{2}||\mathbf{v}||^2, \tag{3.26}$$

which does not incorporate the agents. Obviously, (3.26) is a bi-convex optimization problem. After fixing either $\mathbf{w}$ or $\mathbf{v}$, it becomes convex. As mentioned, the AO method is a good choice for this bi-convex optimization problem. The process is as follows:

Update $\mathbf{w}$ with fixed $\mathbf{v}(t)$: Let $\hat{\mathbf{w}}$ be the collection of $\{\mathbf{w}^1, \cdots, \mathbf{w}^B\}$ in vector form. By introducing a matrix $H_w(t) := [v^1(t)H^1, \cdots, v^B(t)H^B]$, the weight $\hat{\mathbf{w}}$ can be calculated by solving the following optimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\hat{\mathbf{w}}} \frac{1}{2}||Y - H_w(t)\hat{\mathbf{w}}||^2 + \frac{\lambda}{2}||\hat{\mathbf{w}}||^2, \tag{3.27}$$

which is a standard least-squares optimization problem. Its closed-form solution can be by setting its partial derivative to 0, i.e.,

$$\hat{\mathbf{w}}(t) = (H_w^T H_w + \lambda I)^{-1} H_w^T Y. \tag{3.28}$$

Update $\mathbf{v}$ with fix $\mathbf{w}(t)$: The hidden output vector $Z(t)$ of the low-level FNNs is generated after $\mathbf{w}(t)$ is updated. Then the weight $\mathbf{v}(t)$ can be updated by

$$\mathbf{v} = \arg\min_{\mathbf{v}} \frac{1}{2}||Y - Z(t)\mathbf{v}||^2 + \frac{\mu}{2}||\mathbf{v}||^2, \tag{3.29}$$

Similarly, (3.29) can be solved through:

$$\mathbf{v}(t) = (Z(t)^T Z(t) + \mu I)^{-1} Z(t)^T Y, \tag{3.30}$$

A summary of the two-stage PP-HFNN optimization algorithm is provided in Algorithm 2.

Algorithm 1 Two-stage optimization algorithm for PP-HFNN

Stage-1: ADMM-based distributed clustering (3.17)

Initialization: Set $t = 0$ and randomly initialize the global cluster center $\mathbf{r}_k^b(t)$ and Lagrange multipliers $\boldsymbol{\beta}_k^{b,l}(t)$ for each agent in the low-level layers, and randomly initiate the local cluster centers $\mathbf{m}_k^{b,l}$ for all agents in all low-level layers.

for $t = 0, 1, 2, \cdots$, **do**

Update the local center $\mathbf{m}_k^{b,l}(t+1)$:

Assignment step: Each agent of each low-level layer assigns its data $X_i^{b,l}$ to the cluster $\mathcal{C}_k^{b,l}(t-1)$, derived in the previous iteration.

Update step: Each agent b_l updates the center of each cluster $\mathcal{C}_k^{b,l}(t)$ by

$$\mathbf{m}_k^{b,l}(t+1) = \frac{1}{|\mathcal{C}_k^{b,l}(t)|} \sum_{X_i^{b,l} \in \mathcal{C}_k^{b,l}(t)} X_i^{b,l} \tag{3.31}$$

Update the global variables $\mathbf{r}_k^b(t+1)$ by (5.27) and broadcast it to each agent l.

Update the dual variables $\boldsymbol{\beta}_k^{b,l}(t+1)$ by (5.25) and broadcast it to each agent l

end for

Stage-2: AO method for high-level layer (3.14)

Initialization: Set $t = 0$, and randomly initialize the output weight $\mathbf{v}$ and the Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$. Transform $\mathbf{w}$ into $\hat{\mathbf{w}}$

for $t = 0, 1, 2, \cdots$, **do**

Fix $\mathbf{v}$ and update $\hat{\mathbf{w}}$ by (3.28).

Fix $\hat{\mathbf{w}}$ and update $\mathbf{v}$ by (3.30).

end for

Chapter 4

Robust Fuzzy Neural Network with An Adaptive Inference Engine

To accomplish the research objective RO1, which is focused on addressing the curse of dimensionality, as well as effectively handling high levels of data uncertainty prevalent in existing FNNs, this chapter presents a novel solution in the form of a robust fuzzy neural network (RFNN) augmented with an adaptive inference engine (AIE). The proposed RFNN with AIE demonstrates the capability to efficiently tackle diverse levels of uncertainty encountered in high-dimensional data.Inspired by the idea of the relation network (RN) [170], which uses DNNs to learn a distance metric so as to compare samples in few-shot learning tasks [171], the AIE is designed as a learnable neural network module that can automatically adapt itself to learn membership function values and generate corresponding firing strengths. Unlike RNs, which directly concatenate the representations of the samples to be compared, the AIE learns a non-linear mapping to transform the outputs of the antecedent component into more representative firing strengths. Equipped with an AIE, our RFNN not only avoids vanishing gradients when processing high-dimensional data, it also further processes the uncertainty that exists in the membership function values when handling high-level data uncertainty. The RFNN consists of three components: an antecedent component, an AIE, and a consequent component. The AIE connects the antecedent component and the consequent component in sequence to implement the fuzzy if-then rules. Different from deep statistical models, the RFNN has fuzzy logic built right into its structure instead of needing to learn a posterior distribution of the model weights. This eliminates much of the need to compute distributions,

reducing overall overheads. Moreover, unlike existing FNNs, the RFNN includes an FNN-based inference engine to further process the uncertainties that exist in the membership function values. In this way, they generate more suitable firing strengths. Further, neural networks are used as consequent layers, which work as a non-linear estimator of the input samples. This enhances the reasoning ability of fuzzy rules. The full structure of the RFNN is trained via backpropagation without extra hyperparameters.

Hence, the contributions of this section include:

- A new and effective architecture that is robust to data uncertainty and able to process high-dimensional samples in the form of an end-to-end RFNN.
- An AIE that can generate representative firing strengths for high levels of uncertainty. Specifically, TSK-FNN is used as the inference engine, which learns a non-linear function that is able to further process uncertainty in the membership function values of FNNs.
- An adaptive consequent component that enhances the reasoning ability of fuzzy rules. Neural network structures with more powerful learning abilities are used as consequent layers to produce more meaningful outputs.

4.1 Formulation

The architecture of the RFNN consists of three components: 1) the antecedent component, where the adaptive fuzzy sets and their corresponding membership functions are generated; 2) the AIE, where the outputs of membership functions are converted into firing strengths; and 3) the consequent component, where qualified consequents of the rules are generated. The structure is illustrated in Fig. 5.1.

Each rule in the RFNN is comprised of a unit in each of the antecedent components and the consequent components connected by the inference unit. In Fig. 5.1, components from a single rule share the same color. Sitting between the two components is the inference unit, which is shared by all rules. Formally, suppose we have an input set of N labeled samples $S = \{(x_1, y_1), \cdots, (x_N, y_N)\}$, where the label $y_i \in \mathbb{R}^C$ of the i-th sample $x_i \in \mathbb{R}^D$ is an one-hot vector. The membership function values, the firing strength and the output of the K-th rule are denoted as φ_K, ϕ_K and ψ_K, respectively. Taking the antecedent unit's centers, denoted as C_k, and the consequent unit of the k-th rule, denoted as $g(\omega; \cdot)$, we can generally treat the RFNN as a fuzzy process, understood as follows:

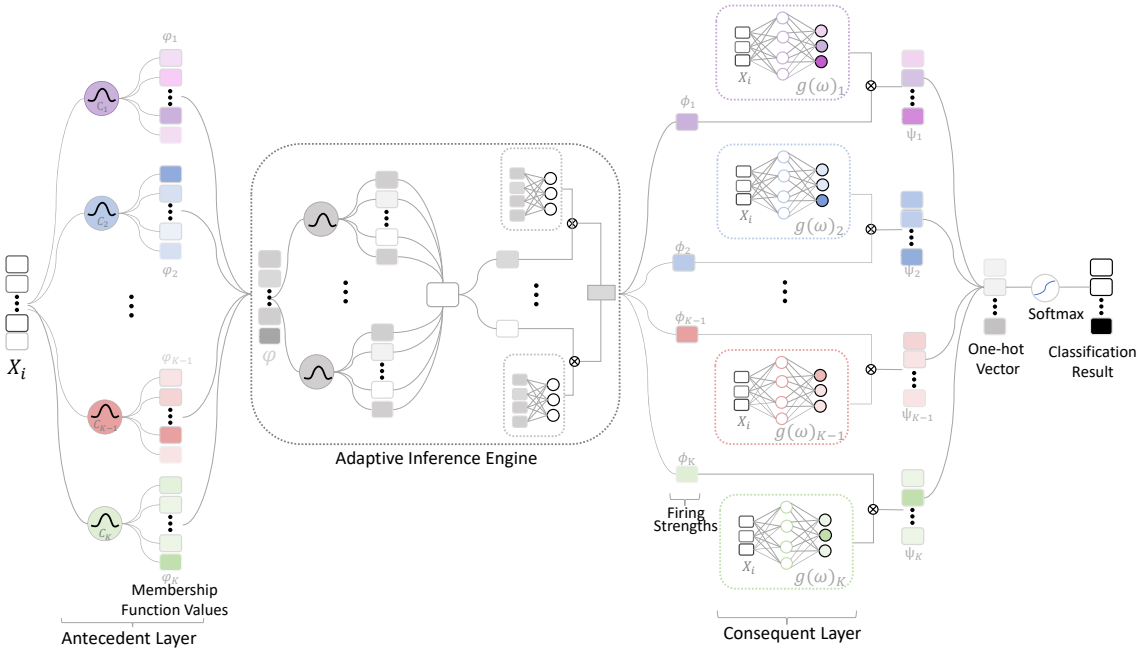


Figure 4.1: Architecture of the RFNN. Each color represents a different data processing rule.

Rule k: IF x_i is close to C_k, then $y_i = g(\omega; x_i)$,

where ω is the parameter of the consequent unit.

Our task is then to optimize the weights in these components so as to learn optimized fuzzy rules that work together to make the model robust against uncertainty.

4.1.1 Antecedent Component

The antecedent component is composed of a group of network units that fuzzify the inputs. As previously mentioned, each unit in the antecedent component can be thought of as the antecedent part of a fuzzy rule. D fuzzy sets need to be generated for each antecedent unit, each of which describes an input feature with a Gaussian distribution. Afterwards, the same number of membership functions are applied to measure the similarities between these input features and their corresponding fuzzy sets.

We set the features of a single rule centre as the D cluster centres of fuzzy sets generated in the antecedent unit. Thus, the data uncertainty can be described by evaluating the similarity of the samples to these centres. All antecedent units work together to describe the uncertainty from multiple views, giving rise to the RFNN. First, the centres $c_k \in \mathbb{R}^D$ are collected in a set $C = \{c_1, c_2, \cdots, c_K\}$, where K is the number of centers. From these, K fuzzy rules are constructed. Each relies on a dissimilarity vector $\ell(x_i, c_k)$ to denote the distances between features of the sample x_i and that of the rule centres c_k. $\ell(x_i, c_k)$ can be calculated via:

$$\ell(x_i, c_k) = ((x_{i,1} - c_{k,1})^2 / \ \sigma_{k,1}, \cdots, (x_{i,D} - c_{k,D})^2 / \ \sigma_{k,D})^T, \tag{4.1}$$

where σ_k is the covariance vector that standardizes the variance of dissimilarity between the features. This measure can be treated as an element-wise Mahalanobis distance.
Here, it can be calculated with:

$$\varphi_k(x_{i,j}) = \exp(-[\ell(x_i, p_k)]_j). \tag{4.2}$$

The rule centres are initialized with the optimized input centres, obtained via a fuzzy c-means (FCM) [172] clustering algorithm, where K is fixed as the desired number of clusters. This initialization step is crucial since it can fix the model architecture; it also helps to facilitate the training process as well. Notably, all weights related to the antecedent component are tuned by backpropagation to instil them with robust fuzzification abilities.

4.1.2 Adaptive Inference Engine

The inference engine converts the outputs of the membership functions in the antecedent component units into firing strengths. These strengths reflect the extent to which the inputs match the corresponding rule. Normally, FNNs apply a fuzzy AND operation to calculate firing strengths — for example, by

$$\phi_k(x_i) = \prod_{j=1}^{D} \varphi_k(x_{i,j}), \tag{4.3}$$

where φ_k denotes the firing strength of x_i to the k-th rule. However, this operation is limited to processing high-dimensional samples, especially within backpropagation schemes, since it can directly lead to the vanishing gradient problem.

Our AIE $f(\theta;\cdot) : \mathbb{R}^D \to \mathbb{R}$ instead learns TSK-FNN [127], [128] to further process the uncertainties in the membership function values and generate corresponding firing strengths. As shown in Fig. 5.1, the AIE's construction is based on a TSK-FNN shared by all rules. By adopting proper rule numbers, the inference unit is able to generate more robust firing strengths in complex scenarios with high levels of uncertainty. Notably, to avoid the limitation of fuzzy AND operations when processing the high-dimensional membership function values, ℓ_2-2norm is used to calculate the firing strengths in the AIE. Hence, the firing strength corresponding to the k-th antecedent unit is calculated by:

$$\phi_k(x_i) = f(\theta; (\varphi_k(x_{i,1}), \cdots, \varphi_k(x_{i,D}))^T), \tag{4.4}$$

where θ denotes the weights of the inference unit. The obtained firing strength is then normalized by

$$\bar{\phi}_k(x_i) = \frac{\phi_k(x_i)}{\sum_{k=1}^{K} \phi_k(x_i)}. \tag{4.5}$$

4.1.3 Consequent Component

With the firing strengths obtained, the next step is the defuzzification process to generate crisp outputs for the fuzzy rules. Traditionally, the rules are defuzzified through a weighted linear combination of the input features. But this approach is not suitable for complex datasets. In our architecture, the defuzzification units in the consequent component $g(\omega;\cdot) : \mathbb{R}^D \to \mathbb{R}^C$ in Fig. 5.1 serve this purpose. These units can be groups of parameterized network structures of any type. In our RFNN, we use 3-layer MLPs as consequent layers. Their outputs multiply the results of the

defuzzification units with their corresponding firing strengths. Thus, the outputs of each rule are derived as follows

$$\psi_k(x_i) = \bar{\phi}_k(x_i) g_k(\omega; x_i), \tag{4.6}$$

where $g_k(\omega; \cdot)$ is the defuzzification unit of the k-th rule. Again, the framework can be tailored to different tasks simply by changing up the network structures in the defuzzification unit.

Before feeding the results to the next level of processing, they are summarized into one raw output that considers all the rules put together. This is calculated by

$$\gamma(x_i) = \sum_{k=1}^{K} \psi_k(x_i), \tag{4.7}$$

The classification head then follows as a last step at the tail of the consequent component. This ensures the architecture is suitable for classification tasks. Here, $\gamma(x_i)$ is processed with a softmax function. Thus, the final prediction is generated from

$$\hat{y}(x_i) = Softmax(\gamma(x_i)), \tag{4.8}$$

4.2 Experiments

Table 4.1: Dataset Information

Dataset	Sample	Feature	Category	ς
GSAD [173]	14,061	128	6	0.0917
SDD [174]	58,590	48	11	0.0
FM [175]	180	43	4	0.1431
WD [176]	4,898	11	7	0.4367
MGT [177]	19,020	10	2	0.2098
SC [178]	58,000	9	7	0.7083
WIL [179]	2,000	7	4	0.0
WFRN [180]	5,456	24	4	0.2960

To evaluate the effectiveness of the RFNN architecture, we conducted extensive experiments with 8 datasets of different types in a variety of settings. The datasets are real-world datasets containing different kinds of sensor data, including driving

monitor signals, shuttle control signals, etc. We selected the datasets for their diverse scenarios, number of features, sizes, number of categories, and category imbalances. Hence, each proves the effectiveness and generalizability of the RFNN from a different view. Note, that we also introduced a category balance factor $\varsigma \in (0, 1)$ to measure sample imbalances across the different categories. This category imbalance factor is calculated by:

$$\varsigma = \sqrt{\sum_{i=1}^{L} (\frac{|\mathcal{D}_i|}{|\mathcal{D}|} - \frac{1}{L})^2}, \tag{4.9}$$

where $|\mathcal{D}|$ is the size of the dataset, $|\mathcal{D}_i|$ is the size of the i-th category, and L is the category number. Intuitively, the bigger the ς is, the more unbalanced the categories are.

A brief description of each dataset follows, with the descriptive statistics listed in Table 6.1.

- The Gas Sensor Array Drift (GSAD) dataset [173] consists of 13,910 instances of gas measurements, each with 128 variables obtained from 16 chemical sensors. The classification task is to detect 6 kinds of gasses at different concentration levels.

- The Sensorless Drive Diagnosis (SDD) dataset [174] is composed of 58,509 samples extracted from electric current drive signals. There are 48 features in each sample, and all samples are classified into 11 categories according to different driving conditions.

- The Flow Meter (FM) dataset [175] is an ultrasonic flow meter diagnostic dataset. Divided into 4 categories, the dataset contains 180 instances of diagnostic parameters extracted from a 4-path liquid ultrasonic flow meter. Each instance is composed of 44 attributes.

- The Wine Quality (WQ) dataset [176] is composed of 4898 physicochemical samples for evaluating 7 different wine qualities. Each testing sample has 12 variants.

- The MAGIC Gamma Telescope (MGT) dataset [177], containing 10,920 samples, was generated by a Monte Carlo program to simulate the registration of high-energy gamma particles. All instances can be classified into two classes, and each instance consists of 10 attributes of different physical parameters.

- The Shuttle Control (SC) dataset [178] is a stat log dataset containing 9 attributes and 58,000 instances across 7 different categories.

- The Wireless Indoor Localization (WIL) dataset [179] is a collection of 2000 instances of the observed signal strengths of 7 WiFi signals visible on a smartphone. The task is to recognize 4 classes of inside locations.

- The Wall-Following Robot Navigation (WFRN) dataset [180] consists of 5456 samples collected by 24 ultrasound sensors. All samples are used to detect movement decisions by the robot.

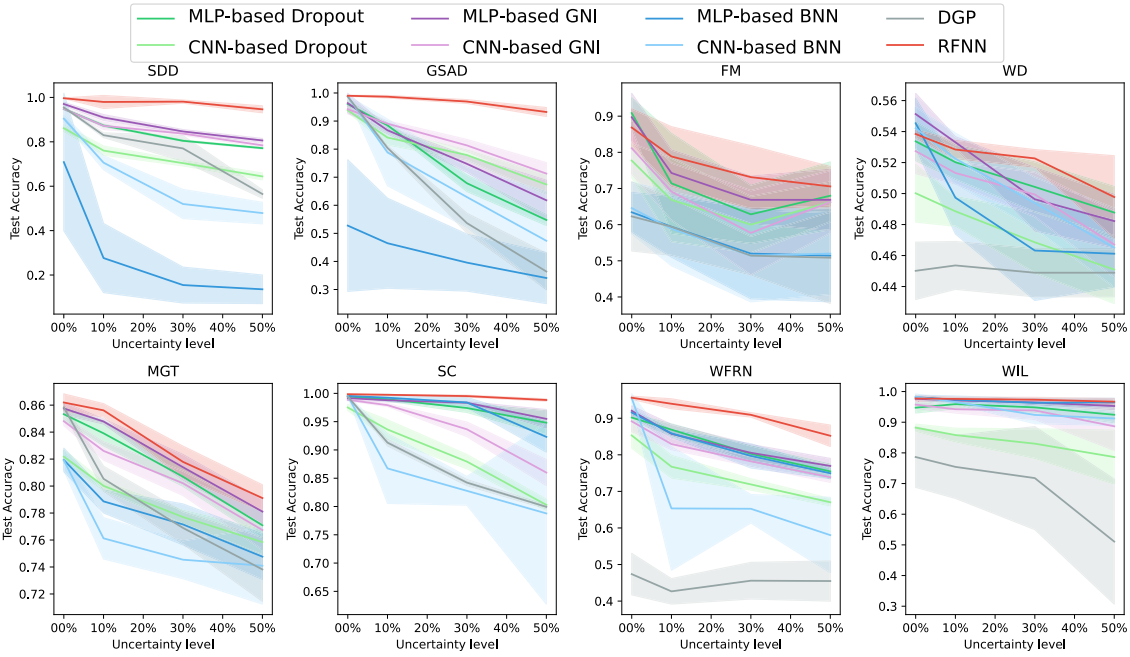


Figure 4.2: Test accuracy of the RFNN and other comparators on all eight datasets. We simulate four levels of data uncertainty on each dataset, where the RFNN performs better than all comparison methods.

All features in all experiments were normalized between -1 and 1. To simulate uncertainty in the different datasets, we randomly sampled a certain proportion of features and perturbed them with noise generated from a normal Gaussian distribution. The proportion of polluted features represents the uncertainty level.

As comparators, we chose an FNN, a stochastic model (GP), several deep stochastic algorithms (BNN, GP, and DGP), and two robust deep models (GNI and Dropout). For fairness, we tested each method with different parameter settings. In addition,

Table 4.2: Average classification accuracy (%) / its standard deviation and mean Averaged F1 Score / its standard deviation of all models on the 8 datasets at 50% level of uncertainty.

Altorithm	Evaluation Metric	Dataset							
		GSAD	SDD	SC	MGT	WFRN	FM	WD	WIL
MLP_dropout	mAP	55.65/2.39	80.35/0.76	94.81/2.14	77.03/0.63	75.47/2.01	68.00/9.35	48.77/1.69	77.03/0.63
	mF1	0.697/0.029	0.802/0.007	0.981/0.012	0.771/0.009	0.759/0.022	0.674/0.179	0.488/0.016	0.924/0.02
CNN dropout	mAP	67.41/3.52	64.42/1.55	80.29/1.02	75.85/0.93	67.03/1.01	66.29/5.50	45.10/2.20	78.65/8.83
	mF1	0.673/0.033	0.648/0.013	0.882/0.011	0.759/0.008	0.672/0.017	0.600/0.067	0.453/0.016	0.819/0.055
MLP GNI	mAP	43.18/6.21	80.61/0.89	95.49/1.64	78.11/0.92	76.95/2.06	67.43/10.80	48.92/1.47	95.29/1.19
	mF1	0.666/0.023	0.796/0.023	0.952/0.014	0.780/0.009	0.771/0.022	0.669/0.063	0.487/0.016	0.954/0.009
CNN GNI	mAP	75.52/5.49	81.19/1.90	89.81/3.87	77.30/1.72	74.35/1.87	69.29/4.33	46.38/2.41	89.22/7.86
	mF1	0.756/0.053	0.812/0.011	0.898/0.039	0.776/0.015	0.742/0.021	0.640/0.033	0.468/0.034	0.892/0.080
MLP BNN	mAP	53.72/8.93	42.61/5.02	92.31/2.71	74.77/1.71	74.95/1.49	51.43/12.78	46.12/2.17	96.49/1.34
	mF1	0.567/0.093	0.456/0.002	0.913/0.017	0.737/0.071	0.729/0.049	0.524/0.012	0.441/0.021	0.919/0.034
CNN BNN	mAP	47.35/17.52	47.90/4.84	82.77/2.53	74.09/2.84	65.25/3.88	52.00/11.32	46.51/2.55	91.23/2.33
	mF1	0.443/0.072	0.469/0.084	0.837/0.013	0.740/0.084	0.612/0.088	0.510/0.011	0.455/0.025	0.913/0.033
GP	mAP	-.—/-.—	-.—/-.—	-.—/-.—	-.—/-.—	-.—/-.—	50.51/9.59	44.24/1.38	87.44/4.26
	mF1	-.—/-.—	-.—/-.—	-.—/-.—	-.—/-.—	-.—/-.—	0.515/0.090	0.452/0.018	0.844/0.026
DGP	mAP	36.41/6.53	56.48/1.89	36.41/6.53	73.82/2.37	45.49/5.49	49.14/13.61	44.89/1.55	51.08/20.38
	mF1	0.344/0.053	0.594/0.089	0.314/0.023	0.758/0.037	0.454/0.049	0.429/0.013	0.428/0.015	0.518/0.038
FNN	mAP	31.80/2.10	12.04/0.86	62.30/2.17	77.31/0.73	54.02/2.01	38.86/8.23	44.81/1.79	64.71/6.17
	mF1	0.348/0.021	0.130/0.086	0.633/0.012	0.713/0.073	0.510/0.001	0.338/0.023	0.418/0.019	0.671/0.017
RFNN	mAP	**93.13/0.87**	**92.28/7.15**	**98.60/0.22**	**78.93/1.49**	**87.01/2.16**	**74.93/6.17**	**50.86/2.95**	**96.69/1.02**
	mF1	**0.932/0.012**	**0.945/0.015**	**0.992/0.003**	**0.783/0.013**	**0.866/0.012**	**0.709/0.106**	**0.490/0.013**	**0.930/0.009**

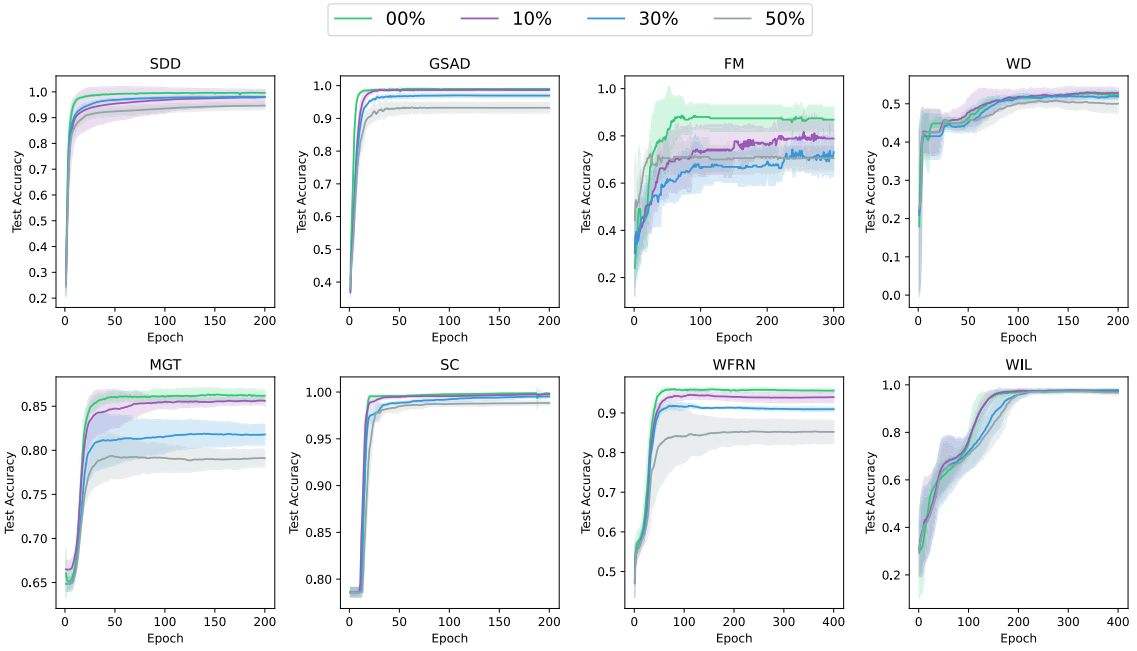


Figure 4.3: The convergence trend of the RFNN on all eight datasets with four levels of uncertainty. A higher level of uncertainty leads to slower convergence speed and worse performance.

Table 4.3: Average classification accuracy (%) and standard deviation of RFNN with MLP-based AIE and RFNN with FNN-based AIE on the 8 datasets at 50% level of uncertainty.

Algorithm	Dataset							
	GSAD	SDD	SC	MGT	WFRN	FM	WD	WIL
MLP	85.58/2.4	88.30/0.86	99.05/3.14	79.57/0.50	80.37/1.14	72.86/6.39	49.78/2.67	96.49/0.64
FNN	**93.13/0.87**	**92.28/7.15**	**98.60/0.22**	**78.93/1.49**	**87.01/2.16**	**74.93/6.17**	**50.86/2.95**	**96.69/1.02**

we tested BNN, GNI, and Dropout with 6 different MLP architectures and 4 different CNN architectures. We then selected the best results from all settings and architectures to include in the main article. The supplement contains the results from many of the variations. Details of each of the models and their corresponding structures and settings follow:

- Dropout — a training strategy that prevents DNNs from overfitting. This method randomly drops units during training to decrease covariance. We tested both the MLP and CNN variants, denoting the models as MLP-based Dropout and CNN-based Dropout. The dropout rate was chosen from an interval of $\{0.05, 0.1, 0.2, 0.3\}$. The best results were regarded as the final performance.

- GNI — a regularization method that randomly adds Gaussian noise to DNNs to improve robustness. We again tested both MLP and CNN architectures, injecting Gaussian noise into the activation layers in the range of $\{0.001, 0.005, 0.01, 0.05, 0.1, 0.3\}$. The best performance was chosen as the final result.

- BNN — a combination of Bayesian and DNN methods. Instead of training specific weights to handle noise attacks, this model optimizes the distribution of the DNN weights. With both the MLP and CNN architectures, we estimated the posterior distribution using the no-Uturn sampler [181], which is a self-tuning variant of a Hamiltonian Monte Carlo algorithm [182]. The number of samples was set to 100 for all datasets.

- GP — a single-layer stochastic process that generates the Gaussian distribution of finite input data.

- DGP — a deep belief network based on a GP algorithm. We tested different numbers of network layers (from 2 to 5), showing the best performance and the final result.

- FNN — a fuzzy neural network where the firing strengths are calculated by certain exact algorithms (fuzzy AND operations). We varied the number of rules from 2 to 50 and report the best performance.

- RFNN — our architecture. The variant tested is the basic form of the model. The number of rules K was determined by searching for the best number of FCM clusters from a range of $[5:5:50]$. We constructed the inference engine with 2 rules. For the consequent component, we also used a 3-layer MLP to build the defuzzification units.

All experiments were conducted with 5-fold cross-validation, and each experiment was repeated 10 times. The final results reported are the mean Average Precision (mAP) and mean F1 Score (mF1) [183] on test data from these runs. Average Precision (AP) and F1-Score of multi-classification task were calculated as:

$$AP = \frac{\sum_{l=1}^{L} \frac{TP_l}{TP_l+FP_l}}{L}, \tag{4.10}$$

$$F1 = 2 \times (\frac{AP \times AR}{AP^{-1} + AR^{-1}}), \tag{4.11}$$

where $AR = (\sum_{l=1}^{L} \frac{TP_l}{TP_l+FN_l})/L$ is the Averaged Recall and TP_l, FP_l, TN_l and $\mathrm{F}N_l$ respectively denote the True Positive, False Positive, True Negative and False Negative of the l-th category.

Table 4.4: mean Averaged F1 Score and standard deviation of all models on the 8 datasets with a hybrid of uncertainty.

Algorithm	Dataset							
	GSAD	SDD	SC	MGT	WFRN	FM	WD	WIL
MLP_dropout	0.662/0.040	0.815/0.006	0.960/0.022	0.789/0.008	0.758/0.012	0.577/0.106	0.489/0.012	0.906/0.031
CNN_dropout	0.706/0.046	0.659/0.004	0.833/0.013	0.767/0.009	0.686/0.012	0.611/0.052	0.449/0.018	0.816/0.057
MLP_GNIs	0.666/0.038	0.807/0.003	0.976/0.009	0.791/0.009	0.775/0.018	0.606/0.145	0.497/0.014	0.927/0.011
CNN_GNI	0.786/0.017	0.812/0.019	0.921/0.024	0.792/0.016	0.762/0.021	0.651/0.084	0.460/0.020	0.918/0.045
MLP_BNN	0.756/0.053	0.792/0.005	0.892/0.027	0.794/0.041	0.749/0.014	0.591/0.078	0.461/0.021	0.914/0.013
CNN_BNN	0.747/0.072	0.817/0.008	0.883/0.023	0.790/0.028	0.752/0.018	0.610/0.032	0.465/0.055	0.912/0.033
GP	-.—/-.—	-.—/-.—	-.—/-.—	-.—/—	-.—/-.—	0.595/0.019	0.454/0.038	0.897/0.026
DGP	0.734/0.053	0.796/0.009	0.836/0.053	0.778/0.037	0.754/0.049	0.589/0.061	0.448/0.055	0.910/0.023
FNN	0.641/0.021	0.732/0.006	0.832/0.017	0.773/0.043	0.740/0.021	0.588/0.029	0.414/0.017	0.871/0.017
RFNN	**0.942/0.008**	**0.960/0.005**	**0.994/0.002**	**0.796/0.008**	**0.855/0.022**	**0.749/0.106**	**0.491/0.008**	**0.929/0.001**

4.2.1 General Performance

To test the RFNN's robustness to uncertainty, we added noise to each of the datasets in the interval of $\{0\%, 10\%, 30\%, 50\%\}$ and tested each of the methods. The

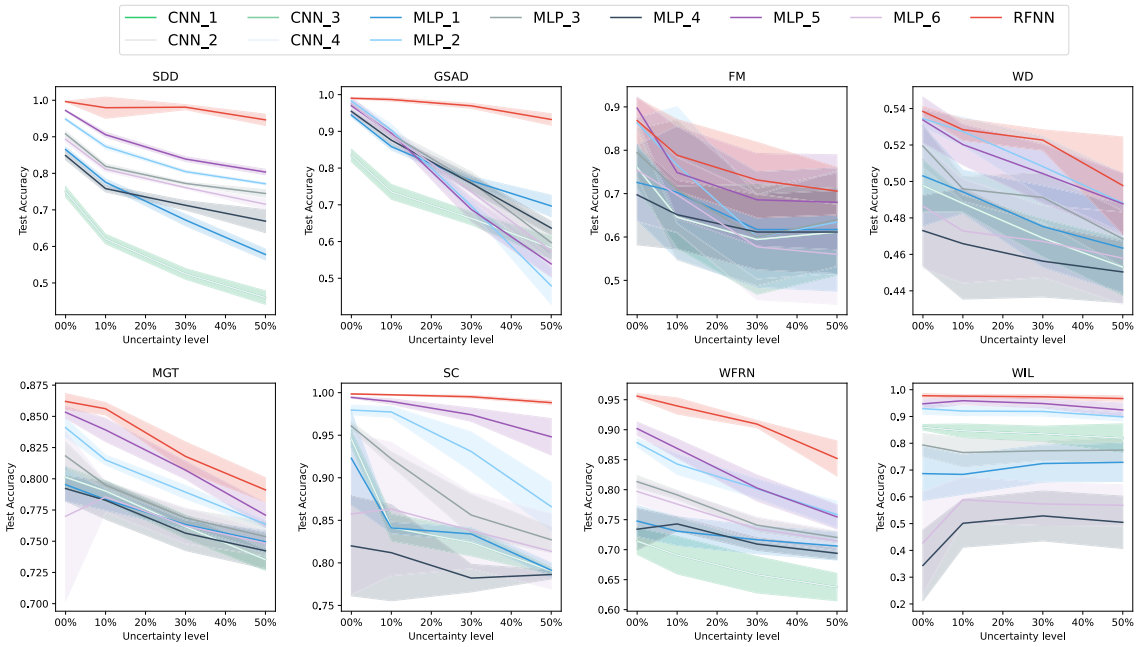


Figure 4.4: Comparison of test performance between the RFNN and different DNN architectures applied with Dropout. Due to page limitation, we choose to show the results of DNNs with a dropout rate of 0.05 which perform better on most datasets..

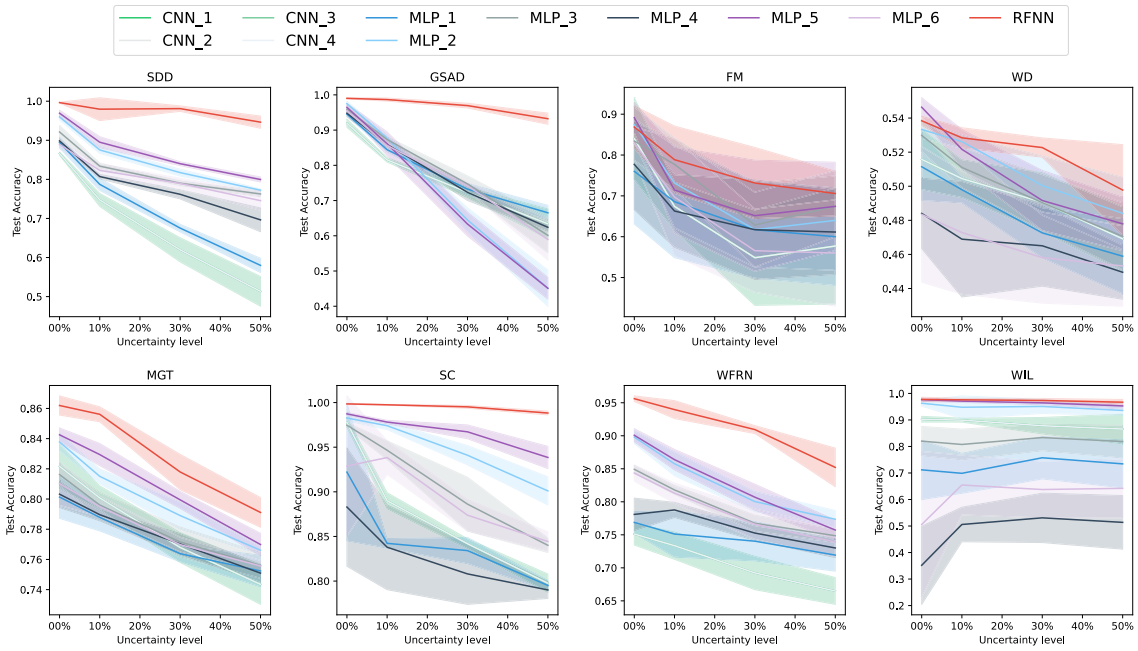


Figure 4.5: Comparison of test performance between the RFNN and different DNN architectures applied with GNI. The noise variance of GNI is selected as 0.01 for the best performance.

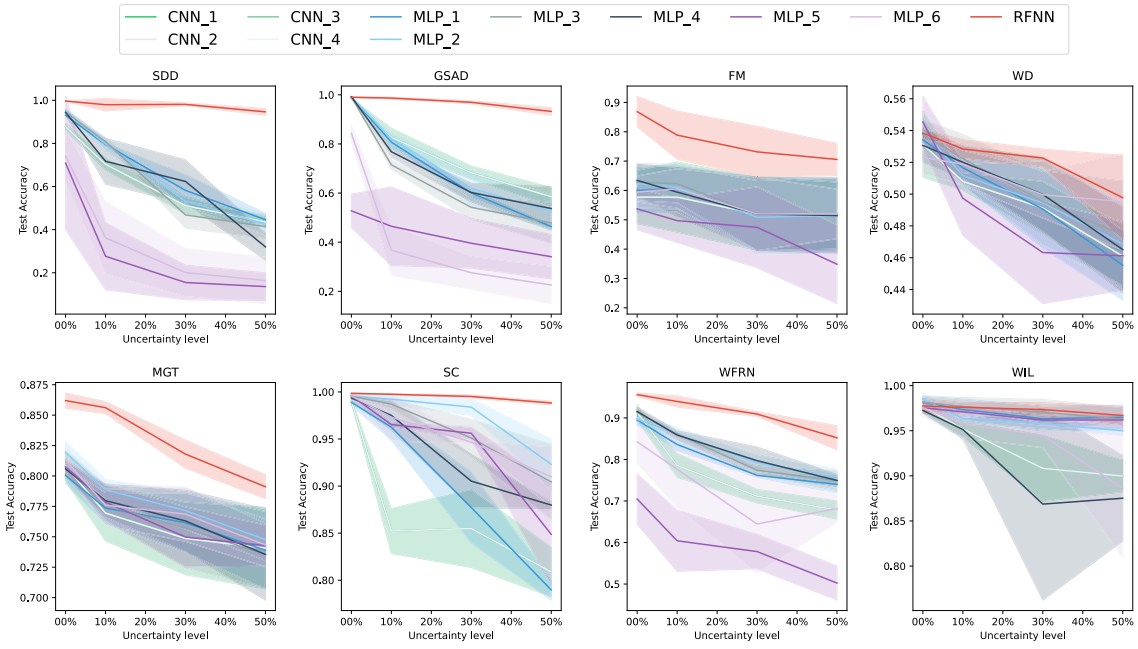


Figure 4.6: Comparison of test performance between the RFNN and different BNN architectures.

results appear in Fig. 4.2. With relatively clean data, our RFNN performs equally well with the state-of-the-art methods. However, the superiority of our model was revealed when the uncertainty increased.

Although each comparison method offers a solution to improve robustness, it is clear from the results that uncertainty is still a great threat to learning. As Fig. 4.2 shows, the performance of all algorithms degraded as uncertainty levels rose. The RFNN appeared the most robust one to the increasing uncertainty, while the BNN and DGP methods tended to fare the worst. For example, both CNN-based BNN and DGP suffered a more than 50% drop in accuracy on the GSAD dataset from clean data to data with 50% uncertainty, whereas the RFNN's accuracy only dropped by about 6%.

To further test the RFNN's robustness when dealing with a high level of uncertainty, we analyzed all the methods when trained on the datasets with 50% noise added. The results are given in Table II. Note that we did not include the GP algorithm in this experiment since it is extremely time-consuming with large-scale datasets. According to the results, the RFNN improved on the previous state-of-the-art's result by a large margin. On average, the RFNN scored 12.01% more in terms of mAP than Dropout, 8.71% more than GNI, and 17.50% more than BNN. In addition, we tested the robustness of our RFNN when processing high-level hybrid uncertainties. We polluted all eight datasets with 3 different levels of noise as follows: 1) 25% of the features were polluted by noise generated from the distribution N(0,1); 2) 5% of the features were randomly set to 2 to serve as outliers; 3) 20% of the features were polluted with the distribution N(0.5,1.2). As the results listed in Table 4.4 show, the RFNN scored 0.165 more in terms of mF1 than Dropout, 0.128 more than GNI, and 0.166 more than BNN.

4.2.2 Ablation Study

To prove the effectiveness of the AIE in processing high-level data uncertainty, we compared our AIE-based RFNN with the RFNN that used other neural networks as their inference engines – over 10 different types in total including MLPs and CNNs. The best performances are shown in Table 4.3. As the results show, our proposed AIE outperformed the other networks for processing high levels of uncertainty on six datasets. It achieved an advantage over comparison methods at an average of 3.52% test accuracy.

4.2.3 Generalization Analysis

Our RFNN has looser constraints and stronger generalizability for processing different levels of uncertainties and dealing with different learning tasks. In addition, the only hyperparameter in our model is the number of rules, which the FCM algorithm selects automatically. Thus, RFNNs can automatically modify their own structures when processing datasets with large differences. By contrast, most of the comparison algorithms need to tune extra hyper-parameters during the training process, and their hyper-parameter settings need to be re-adjusted by hand when the level of data uncertainty or the learning scenario changes. Yet as presented in Figs. 4.4 - 4.6, the RFNNs show greater generalizability than the comparison methods, and they can obviously handle different levels of uncertainty well. In addition, Tables 4.2 and 4.4 show that the RFNN performs very well when processing real-world datasets collected from different scenarios and that RFNNs are more generalizable to different tasks and scenarios.

4.2.4 Convergence analysis

Fig. 4.3 shows test accuracy by epoch. With all eight datasets at different noise levels, the RFNN was able to converge to its optimized accuracy, although we did find that the more uncertain the samples, the more epochs were needed to reach convergence. Interestingly, the volume and dimensionality of the datasets tended to play an important role in the smoothness of the training process. Datasets with larger volumes and smaller dimensionality trained more smoothly than the others because of the greater numbers of samples. Fewer features meant the RFNN found it easier to learn meaningful rules so as to cover key features and eliminate the uncertainty. When coping with different uncertainty levels, the comparators needed to adjust/tune their hyperparameters, whereas RFNN did not need to change its structure to complete its assigned task – even at very high levels of uncertainty. From the perspective of model stability, the RFNN tended to behave more stably with less noisy data as evidenced by the variance of curves drawn in Fig. 4.3.

4.3 Summary

In this chapter, we introduced a novel form of the fuzzy neural network, called the robust fuzzy neural network or RFNN. The RFNN contains an adaptive inference engine (AIE) that is trained by an end-to-end backpropagation learning algorithm to handle uncertainty in data. The AIE provides a nonlinear mapping that further processes the uncertainties within membership function values and generates

representative firing strengths. As a result, our RFNN is not only able to handle high levels of data uncertainty, but it can also directly process data with very high dimensionality. In addition, we presented an architecture where the consequent components are constructed from neural networks. These enhance the reasoning ability of the learned rules. The RFNN is a robust and scalable solution for handling data uncertainty that shows a great deal of promise for further investigation. The AIE and consequent component can be constructed with specific neural network structures that adaptively work for specific applications. Experiments on eight datasets show that our RFNN delivers state-of-the-art accuracy at very high levels of uncertainty. In addition, an ablation study between FNN-based and MLP-based AIEs demonstrates the superiority of our FNN-based AIE in improving the tolerance of uncertainty. Our future work will extend the RFNN with different network structures of inference engines to fit different scenarios.

Chapter 5

Distributed Semi-supervised Fuzzy Neural Networks with Interpolation Consistency Regularization

In this chapter, our aim is to achieve the RO2 by leveraging unlabelled data to enhance the fuzzy reasoning ability of a FNN. To accomplish this, we propose a distributed semi-supervised fuzzy regression (DSFR) model incorporating fuzzy if-then rules and ICR. This model not only enhances the handling of data uncertainty by utilizing clustering information from unlabelled data, but also effectively reduces computation and communication overheads in interconnected networks. A distributed Fuzzy C-means (DFCM) algorithm locates the parameters in an antecedent component and a distributed interpolation consistency regularization (DICR) algorithm obtains the parameters in a consequent component. Both the DFCM algorithm and the DICR algorithm are implemented following the well-known alternating direction method of multipliers (ADMM) [58] to guarantee consensus among all local agents. Data is only ever processed locally by the agent that owns the data, and very little information is passed between neighbours, all of which is non-sensitive. Notably, the DSFR model converges very quickly since it does not involve a backpropagation procedure. Benefiting from DFCM and DICR, it also scales well to large datasets. Thus, the main contributions of this chapter include:

- A novel DSFR model with ICR that handles data uncertainty and has lower computation and communication overheads in interconnected networks than existing DSSL algorithms.

- A DFCM algorithm to locate parameters in the antecedent component of the DSFR model. The DFCM can be used directly with both labelled and unlabeled training data available over interconnected networks.
- A DICR algorithm to obtain parameters in the consequent component of the DSFR model. This is the first implementation to extend ICR to a distributed and semi-supervised scenario. In contrast to existing DDSL algorithms, such as graph-based DDSL [86], [87], DICR results in smaller loss values and enjoys much greater scalability.

5.1 Centralized semi-supervised fuzzy regression

This section sets out the formulation for the centralized semi-supervised fuzzy regression (CSFR) model with ICR. Notably, CSFR is sequentially trained by unsupervised structure learning and semi-supervised parameter learning.

5.1.1 Fuzzy inference system

Let us briefly describe the fuzzy inference system based on a first-order Takagi-Sugeno (T-S) method. Consider the estimation of a scalar output $y \in \mathbb{R}$ from a D-dimensional input $x = [x_1, x_2, \cdots, x_D]$. The k-th fuzzy rule can be represented as

$$\begin{gathered}\text{Rule } k\text{: IF } x_1 \text{ is } A_{k1} \text{ and } \cdots \text{ and } x_D \text{ is } A_{kd} \\ \text{Then } y = w_{k0} + \textstyle\sum_{j=1}^{D} w_{kj} x_j\end{gathered}$$

where A_{kj} is a Gaussian fuzzy set with the following membership function:

$$\varphi_{kj}(x_j) = \exp\left[-\left(\frac{x_j - m_{kj}}{\sigma_{kj}}\right)^2\right] \tag{5.1}$$

where m_{kj} and σ_{kj} are respectively the mean value and standard deviation.

The firing strength for each fuzzy rule is

$$\bar{\phi}_k(x) = \frac{\prod_{j=1}^{D} \varphi_{kj}(x_j)}{\sum_{k=1}^{K} \prod_{j=1}^{D} \varphi_{kj}(x_j)}. \tag{5.2}$$

where K denotes the number of applied fuzzy rules.

The overall output is obtained by summing the outputs of all fuzzy rules multiplied with a weighted vector:

$$\hat{y} = \sum_{k=1}^{K} \bar{\phi}_k(x)(w_{k0} + \sum_{j=1}^{D} w_{kj}x_j), \tag{5.3}$$

The structure learning process, therefore, aims to optimize the parameters of the Gaussian membership functions in (6.1) for each fuzzy rule, i.e., m_{kj} and σ_{kj}, $j \in \{1, \cdots, D\}$, which is done through the FCM clustering method[166]. In turn, the goal of the parameter learning process is to identify the output weights $w_{k0}, \cdots, w_{kD}$ in (5.3), done with a least-squares algorithm[184]. A more detailed description of the structure learning procedure follows next.

5.1.2 Fuzzy C-Means for the structure learning

As mentioned, the structure learning process is governed by the FCM algorithm. The fuzzy rules are generated by clustering training data into several groups where each group corresponds to a fuzzy rule. This procedure follows.

Let $\mathcal{D} := \{(X_i, Y_i)|X_i \in \mathbb{R}^D, Y_i \in \mathbb{R},\ i \in \{1, \cdots, N\}\}$ denote the training data set and x_{ij} denote the j-th feature of the i-th sample X_i. The aim of FCM algorithm is to partition N samples into K groups $\mathcal{C} := \{\mathcal{C}_1, \cdots, \mathcal{C}_K\}$, where K denotes the total number of fuzzy rules and is often assumed to be known as a priori. The most important task for structure learning is to identify the total number K of clusters, where the $k-$th center can be identified follows:

$$\mathbf{m}_k = \arg\min_{\mathbf{m}_k} \frac{1}{2} \sum_{k=1}^{K} \sum_{i=1}^{|\mathcal{C}_k|} u_{ik}^{\alpha} ||X_i - \mathbf{m}_k||^2 \tag{5.4}$$

where $|\mathcal{C}_k|$ denotes the $k-$th cluster, and $\alpha \geq 1$ determines the level of cluster fuzziness. (α is commonly set to 2.) With an iterative technique, the FCM algorithm then assembles and refines the clusters. All the K centers at iteration $t = 0$ are initialized randomly, i.e., $\{\mathbf{m}_1(0), \cdots, \mathbf{m}_K(0)\}$, the procedures from $t+1$ iterates as follows:

- Update the membership value:

$$u_{ik}^{\alpha}(t+1) = \frac{1}{\sum_{c=1}^{K} \left(\frac{||X_i - \mathbf{m}_k(t)||}{||X_i - \mathbf{m}_c(t)||}\right)^{\frac{2}{\alpha-1}}} \tag{5.5}$$

- Update the cluster center:

$$\mathbf{m}_k(t+1) = \frac{\sum_{i=1}^{N} u_{ik}^{\alpha}(t+1) X_i}{\sum_{i=1}^{N} u_{ij}^{\alpha}(t+1)} \tag{5.6}$$

The algorithm stops optimizing when the values of centers stay unchanged during the iteration, and the obtained centers become the centers of each fuzzy set. Meanwhile, the standard variance σ_{kj} of $k-$th fuzzy set can be written as

$$\sigma_{kj} = \sqrt{\sum_{i=1}^{N} u_{ik}^{\alpha}(X_{ij} - m_{kj})^2 / \sum_{i=1}^{N} u_{ik}^{\alpha}} \tag{5.7}$$

And voilà, we have all the parameters in the antecedent layer.

To calculate the output weights in the consequent layer, parameter learning is described next with a closed-form solution.

5.1.3 Closed-form solution for the parameter learning

Parameter learning begins with a hidden matrix defined as $H(X) := [H_1, \cdots, H_K] \in \mathbb{R}^{N \times K(D+1)}$, where

$$H_k(X) = \begin{bmatrix} \bar{\phi}_k(X_1) & \bar{\phi}_k(X_1)x_{11} & \cdots & \bar{\phi}_k(X_1)x_{1D} \\ \bar{\phi}_k(X_2) & \bar{\phi}_k(X_2)x_{21} & \cdots & \bar{\phi}_k(X_2)x_{2D} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\phi}_k(X_N) & \bar{\phi}_k(X_N)x_{N1} & \cdots & \bar{\phi}_k(X_N)x_{ND}, \end{bmatrix} \tag{5.8}$$

and the output vector $Y := [Y_1, \cdots, Y_N]$.

The output weight matrix $\mathbf{w} \in \mathbb{R}^{K(D+1)}$ can be written as:

$$\mathbf{w} = [w_{10}, \cdots, w_{1d}, \cdots, w_{K0}, \cdots, w_{KD}]^T, \tag{5.9}$$

it can be identified by calculating the optimization problem follows,

$$\min \mathcal{F}_s(\mathbf{w}; X) = \min_{\mathbf{w}} \frac{1}{2}||Y - f(\mathbf{w}; X)||^2 + \frac{\mu}{2}||\mathbf{w}||^2, \tag{5.10}$$

where $f(\mathbf{w}; X) = H(X)\mathbf{w}$, $\mu > 0$ trades off the model performance between training error and model generalization. (5.10) is a standard least-squares optimization problem, which can be obtained with a closed-form solution follows:

$$\mathbf{w} = (H^T H + \mu I)^{-1} H^T Y, \tag{5.11}$$

where I is the identity matrix with a dimension of $K(D+1)$.

5.1.4 Semi-supervised fuzzy regression with ICR

As mentioned before, using ICR to solve semi-supervised fuzzy regression problems can push decision boundaries into low-density areas, leading to better generalization performance in semi-supervised scenarios [78], [119]. We followed the data augmentation techniques in [78], [119] to generate interpolation consistency loss and involved it into the objective function (5.10).

The training set consists of a labelled data set $X \in \mathcal{C}_s$, and an unlabeled data set $U \in \mathcal{C}_u$, denoted as $\mathcal{C} = \mathcal{C}_s \cup \mathcal{C}_u$. Accordingly, the number of training samples can be expressed as $N = N_s + N_u$. ICR augments this training set with virtual samples constructed from the unlabeled data set $\mathcal{C}_u$ as follows:

$$\begin{aligned} \tilde{U} &= \lambda U_1 + (1-\lambda)U_2,\ U_1, U_2 \in \mathcal{C}_u, && (5.12) \\ f(\mathbf{w}; \tilde{U}) &= \lambda f(\mathbf{w}; U_1) + (1-\lambda) f(\mathbf{w}; U_2) && (5.13) \end{aligned}$$

where λ is randomly sampled from a beta distribution.

The objective function of semi-supervised fuzzy regression (SFR) with ICR is

$$\min \mathcal{F}(\mathbf{w}; X, U) = \min_{\mathbf{w}} \mathcal{F}_s(\mathbf{w}; X) + \gamma \mathcal{F}_u(\mathbf{w}; U), \tag{5.14}$$

where

$$\begin{aligned} \mathcal{F}_u(\mathbf{w}; U) &= ||f(\mathbf{w}; \tilde{U}) - (\lambda f(\mathbf{w}; U_1) + (1-\lambda) f(\mathbf{w}; U_2))||^2 \\ &= ||H(\tilde{U})\mathbf{w} - (\lambda H(U_1)\mathbf{w} + (1-\lambda) H(U_2)\mathbf{w})||^2, \\ &\triangleq ||B(U)\mathbf{w}||^2, && (5.15) \end{aligned}$$

where $B(U) = H(\tilde{U}) - \lambda H(U_1) - (1-\lambda) H(U_2)$. The closed-form solution of (5.14) is:

$$\mathbf{w} = (\gamma B^T(U) B(U) + \mu I + H^T(X) H(X))^{-1} H^T(X) Y. \tag{5.16}$$

5.2 Distributed semi-supervised fuzzy regression with ICR

In this section, we extend the CSFR model to its distributed version and design distributed training algorithms for it. Since the CSFR model is sequentially trained by structure learning and parameter learning, we will develop distributed structure learning and distributed parameter learning sequentially.

Our distributed computing scenario is conceived as an undirected graph $\mathcal{G} = \{\mathcal{L}, \xi\}$, of L agents (nodes) connected by E edges, where $\mathcal{L}$ and ξ denote the nodes set and the edges set, respectively. Agent l is the target agent, and $\mathcal{N}_l$ is the set of agents

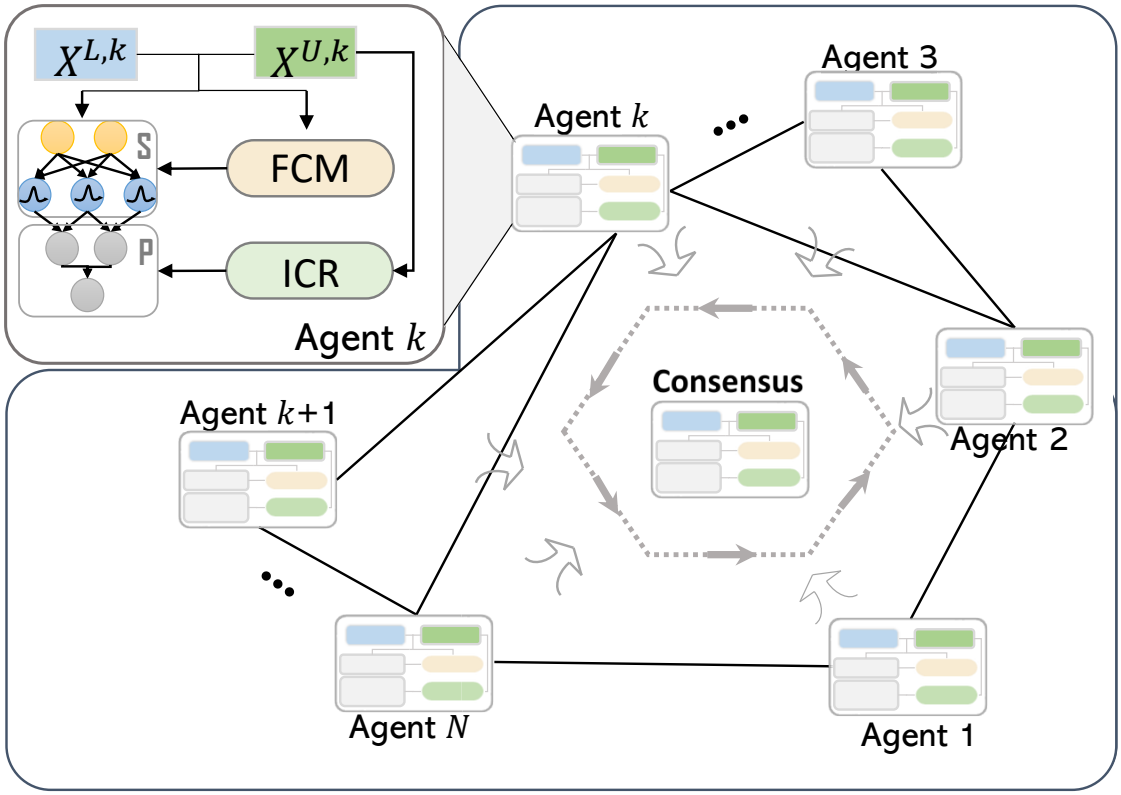


Figure 5.1: Architecture of the DSFR model. The upper-left part depicts detailed structures of each local model, which is presented in the bottom-right part. Different colours are used to distinguish different types of data and methods.

The corresponding structure learning problem is solved through a consensus strat-

egy, formulated as

$$\min_{\mathbf{m}_k^l} \frac{1}{2} \sum_{l=1}^{L} \sum_{k=1}^{K} \sum_{X_i^l \in \mathcal{C}_k^l} (u_{ik}^q)^\alpha ||X_i^l - \mathbf{m}_k^l||^2 \tag{5.17a}$$

$$\text{s.t.} \quad \mathbf{m}_k^l = \mathbf{r}_k, \ l \in \mathcal{L}, \ k \in \mathcal{K}, \tag{5.17b}$$

where $\mathbf{m}_k^l$ represents the local center of the $k-h$ fuzzy set on agent l, $\mathbf{r}_k$ denotes the global center, which integrates all the local centers, and $|\mathcal{C}_k^l|$ is the cardinality operation for a local subset. Moreover, (5.17b) is the constraint that assures all local centers coincide at one global center. Notably, all the local variables among different agents can be parallelly computed, thus improving the computing speed.

After Identifying all the global centers, the global standard variance can be calculated via

$$\bar{\sigma}_{kj} = \sqrt{\frac{1}{N} \sum_{l=1}^{L} |\mathcal{C}^l| (\sigma_{kj}^l)^2} \tag{5.18}$$

where $|\mathcal{C}^l|$ is the cardinality of a local subset $\mathcal{C}^l$ on agent l, σ_{kj}^l denotes the j-th element of fuzzy rule k's standard variance on subset $\mathcal{C}^l$, and $\bar{\sigma}_{kj}$ is corresponding to global standard variance for all agents. Note that the standard variance σ_{kj}^l corresponding to the k-th rule and j-th dimension is calculated in the following element-wise method:

$$\sigma_{kj}^l = \sqrt{\sum_{X_i^l \in \mathcal{C}_k^l} (u_{ik}^l)^\alpha (X_{ij}^l - \mathbf{m}_{kj}^l)^2 / \sum_{X_i^l \in \mathcal{C}_k^l} (u_{ik}^l)^\alpha} \tag{5.19}$$

where X_{ij}^l and $\mathbf{m}_{kj}^l$ are the j-th components of X_i^l and $\mathbf{m}_k^l$, respectively.

The parameter learning process can be similarly modelled as follows:

$$\min_{\mathbf{w}^l} \quad \tfrac{1}{2} \textstyle\sum_{l=1}^{L} (||Y^l - H(X^l)\mathbf{w}^l||^2$$

$$+\gamma ||B(U^l)\mathbf{w}^l||^2) + \tfrac{\mu}{2} ||\mathbf{z}||^2, \tag{5.20a}$$

$$\text{s.t.} \quad \mathbf{w}^l = \mathbf{z}, \ l \in \mathcal{L}, \tag{5.20b}$$

where $\mathbf{w}^l$ represents $l-$th agent's local output weight, and $\mathbf{z}$ is the global weight that integrates local weights for all agents.

5.2.1 Distributed FCM

The optimization problem (5.17) is nonconvex, so using an exhaustive search method to solve the problem would not be efficient. Further, centralized clustering methods, such as the FCM algorithm used for structural learning, have several

shortcomings including high communications overheads, no attention to privacy, and poor scalability when coping with large-scale data. Hence, a distributed variant of the centralized FCM algorithm is needed to address these issues. Accordingly, our formulation of a DFCM algorithm follows.

The first step is to construct the following augmented Lagrangian for (5.17):

$$\begin{aligned}\mathcal{L}_s(\mathbf{m},\mathbf{r},\boldsymbol{\lambda}_s) = \quad & \frac{1}{2}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{X_i^l\in\mathcal{C}_k^l}(u_{ik}^q)^\alpha||X_i^l-\mathbf{m}_k^l||^2 \\ & +\sum_{l=1}^{L}\sum_{k=1}^{K}\boldsymbol{\lambda}_{s,kl}^T(\mathbf{m}_k^l-\mathbf{r}_k) \\ & +\frac{1}{2}\rho_s\sum_{l=1}^{L}\sum_{k=1}^{K}||\mathbf{m}_k^l-\mathbf{r}_k||^2 \end{aligned} \tag{5.21}$$

where $\boldsymbol{\lambda}_{s,kl}$ denotes the Lagrange multiplier and ρ_s is a positive penalty factor. Denoting these parameters at iteration t as $\mathbf{r}(t)$ and $\boldsymbol{\lambda}_s(t)$, the variables are obtained iteratively using the following ADMM-based updating steps:

$$(u_{ik}^l)^\alpha(t+1) = \frac{1}{\sum_{c=1}^{K}(\frac{||X_i^l-\mathbf{m}_k^l(t)||}{||X_i^l-\mathbf{m}_c^l(t)||})^{\frac{2}{\alpha-1}}}, \tag{5.22}$$

$$\mathbf{m}^l(t+1) = \arg\min_{\mathbf{m}}\mathcal{L}(\mathbf{m}^l,\mathbf{r}(t),\boldsymbol{\lambda}_s(t)), \tag{5.23}$$

$$\mathbf{r}(t+1) = \arg\min_{\mathbf{r}}\mathcal{L}(\mathbf{m}^l(t+1),\mathbf{r},\boldsymbol{\lambda}_s(t)), \tag{5.24}$$

$$\boldsymbol{\lambda}_{s,kl}(t+1) = \boldsymbol{\lambda}_{s,kl}(t)+\rho_s(\mathbf{m}_k^l(t+1)-\mathbf{r}_k(t+1)). \tag{5.25}$$

It is worth noting that (5.23) can be solved in parallel among different agents.

The cluster centers $\mathbf{m}^l(t+1)$ for each agent l are updated through the same assignment and update steps from the centralized FCM algorithm. But, it's easy to know that the solutions (5.23) and (5.24) can be calculated by setting the partial derivative of the corresponding variable to zero. Thus, the closed-form solution of (5.23) and (5.24) are as follows:

$$\mathbf{m}_k^l(t+1) = \frac{\sum_{X_i^l\in\mathcal{C}_k^l}(u_{ik}^l(t))^\alpha X_i-\boldsymbol{\lambda}_{s,kl}(t)+\rho_s\mathbf{r}_k(t)}{\sum_{X_i^l\in\mathcal{C}_k^l}(u_{ik}^l(t))^\alpha+\rho_s} \tag{5.26}$$

$$\mathbf{r}_k(t+1) = \frac{1}{\rho_s}\bar{\boldsymbol{\lambda}}_{s,kl}(t)+\bar{\mathbf{m}}_k^l(t+1), \tag{5.27}$$

where

$$\bar{\mathbf{m}}_k^l(t+1) = \frac{1}{L}\sum_{l=1}^{L}\mathbf{m}_k^l(t+1), \tag{5.28}$$

$$\bar{\boldsymbol{\lambda}}_{s,kl}(t) = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{\lambda}_{s,kl}(t). \tag{5.29}$$

Algorithm 2 summarizes the above approach. The convergence behavior is examined by checking the norms of the following two residuals:

$$||\mathbf{m}_k^l(t) - \mathbf{m}_k^q(t)||^2 \leq \epsilon_1, \tag{5.30}$$

$$||\boldsymbol{\lambda}_{s,k}^l(t) - \boldsymbol{\lambda}_{s,k}^l(t-1)||^2 \leq \epsilon_2. \tag{5.31}$$

Algorithm 2 Distributed FCM (5.17)

Initialization: Set $t = 0$ and the Lagrange multipliers $\boldsymbol{\lambda}_{s,kl}(t) = \mathbf{0}$. Initialize cluster centers and assign them to each agent l.
for $t = 0, 1, 2, \cdots$, **do**
  Update the membership value $(u_{ik}^l)^\alpha$ by (5.22) for each agent l.
  Update the local variables $\mathbf{m}^l(t+1)$ by (5.26) and broadcast it to each agent l.
  Update the global variables $\mathbf{r}(t+1)$ by (5.27) and broadcast them to each agent l.
  Update the dual variables $\boldsymbol{\lambda}_s(t+1)$ by (5.25) and broadcast them to each agent l
end for

The computation complexity of DFCM relies on the local variables update and global variables update. The computation complexity for DFCM is $\mathcal{O}(NDK^2T_1)$, where T_1 is the required iteration of DFCM. Note that the DFCM is built on the well-known ADMM algorithm, which generally converges to modest accuracy (such as $\epsilon < 10^{-3}$) within a few tens of iterations. Therefore, the total computational complexity of DFCM is quite limited.

5.2.2 Distributed ICR

Likewise, a distributed variant of ICR is needed to tackle the regression problem. In contrast to existing DDSL algorithms [86], [87], which are typically based on

graph operations, DICR is more faster and more accurate with dramatically reduced computing times, especially with large-scale datasets. The formulation follows.

The augmented Lagrangian for (5.20) is

$$\begin{aligned}\mathcal{L}(\mathbf{w}^l, \mathbf{z}, \boldsymbol{\beta_q}) = \quad & \frac{1}{2}\sum_{l=1}^{L}(||Y^l - H(X^l)\mathbf{w}^l||^2 + \gamma||B(U^l)\mathbf{w}^l||^2) \\ & + \frac{\mu}{2}||\mathbf{z}||^2 + \sum_{l=1}^{L}\boldsymbol{\beta}_l^T(\mathbf{w}^l - z) \\ & + \frac{1}{2}\rho_p\sum_{l=1}^{L}||\mathbf{w}^l - z||^2 \end{aligned} \tag{5.32}$$

where $\boldsymbol{\beta}_l$ is the Lagrange multiplier, and ρ_p is a small positive penalty parameter. This augment Lagrangian is then solved using ADMM method by

$$\mathbf{w}^l(t+1) = \arg\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}, \mathbf{z}(t), \boldsymbol{\beta}(t)), \tag{5.33}$$

$$\mathbf{z}^l(t+1) = \arg\min_{\mathbf{z}} \mathcal{L}(\mathbf{w}(t+1), \mathbf{z}, \boldsymbol{\beta}(t)), \tag{5.34}$$

$$\boldsymbol{\beta}_l(t+1) = \boldsymbol{\beta}_l(t) + \rho_p(\mathbf{w}^l(t+1) - \mathbf{z}(t+1)). \tag{5.35}$$

The closed-form solution of (5.33) is clear; w on the l-th agent is updated by

$$\mathbf{w}^l(t+1) = Q((H^l(X))^T Y^l + \rho_p \mathbf{z}(t) - \boldsymbol{\beta_l}(t)), \tag{5.36}$$

where I is the identity matrix with dimension $K(D+1)$ and Q can be obtained by

$$Q = ((H^l(X))^T H^l(X) + \gamma B^T(U)B(U) + \mu I)^{-1}. \tag{5.37}$$

The solution for (5.34) is similarly found by

$$\mathbf{z}(t+1) = \frac{\sum_{l=1}^{L}(\beta_l + \rho_p \mathbf{w}^l(t+1))}{\mu + \rho_p L}, \tag{5.38}$$

The computation complexity of DICR is $\mathcal{O}(KDLT_2)$ if we ignore the cost of data augmentation, where T_2 is the iteration number required by ADMM procedure. As we analyzed before, T_2 is generally about a few tens. Therefore, the total computation complexity of DICR is small.

Algorithm 3 Distributed ICR

Initialization: set $t = 0$ and initialize the global weight $\mathbf{z}(t)$ and Lagrange multipliers $\beta(t)$ for each agent l.

for $t = 0, 1, 2, \cdots$, **do**

Augmented data generation:

Randomly select M samples from the unlabeled data set $\mathcal{C}_u$ as U_1 and another M samples as U_2. Then generate M number of λ using a beta distribution, and generate M augmented samples using the interpolation technique in (5.12).

Update the local variables $w^l(t+1)$ via (5.36) for each agent l.

Update the global variable $z(t+1)$ via (5.38) and broadcast it to the other agents.

Update the dual variables $\boldsymbol{\beta}(t+1)$ by (5.35) and broadcast them to the other agents.

end for

5.3 Experiments

To verify the effectiveness of the DSFR framework, we conducted extensive experiments on six different types of datasets. Descriptive statistics are provided in Table 6.1; brief descriptions follow:

- The Airfoil dataset [185]. Assembled by NASA, this dataset contains readings of noise tests with different-sized NACA 0012 airfoils.

- The WarCraft Master Skill Level (WMSL) dataset [186]. A set of playing data drawn from the WarCraft real-time strategy game, this set contains the playing data of 3360 gamers. After removing broken records, the final dataset numbered data on 3338 players.

- The Combined Cycle Power Plant (CCPP) dataset [187]. This set contains 9568 samples of gas and steam turbine loads. We chose the full electrical power load as the target label and used the remaining data to train the model.

- The California Housing (CH) dataset [188]. A data set of information about real estate in California, we discarded all non-numeric features and treated the median house value of a district as the target.

- The King County Housing (KCH) dataset [189]. This is a collection of houses sold in King County, USA. After removing time-related and restriction-related

variables, we used house price as the target and the rest of the features for training.

- The artificial dataset. We built this dataset to investigate the performance of the DSFR framework. 5,000 samples $X \in \mathbb{R}^8$ were generated from a Gaussian distribution with a mean of $[0, 0.5, \cdots, 3.5]$ and a standard deviation of $[0.2, 0.4, \cdots, 1.6]$. The corresponding label Y was given by: $Y = 0.3\sum_{i=1}^{8}(X_i)^2 + 0.7\sum_{i=1}^{8}(\cos(X_i))$. We denoted the range of labels as $R = max(X) - min(X)$ and generated two types of noise from Gaussian distributions $\mathcal{N}(0.1R, (0.1R)^2)$ and $\mathcal{N}(R, (0.1R)^2)$, respectively. Then, 5% level of type-1 noise and 10% type-2 noise were randomly added to labels.

It should be noted that all feature values of these datasets are normalized between -1 and 1. As listed in Table 6.1, a 5-fold cross-validation is used in our experiment to randomly generate 5 test sets and 5 training sets, among which we randomly choose 50 samples as labelled data and the rest as unlabeled data. Then we re-conduct the experiment 10 times and get the average results and its standard variance. For fairness, we arrange the training sets on an interconnected network with five fully connected agents, and we set the rule number as 5 in all agents for all these datasets. All the experiments are conducted on computing nodes with 26 cores of 2.11 GHz and 95 GB memory.

Table 5.1: Dataset Information

Dataset	Features	Samples	Labeled	Unlabeled	Test
Airfoil	5	1503	50	1153	300
WMSL	18	3338	50	2621	667
CCPP	4	9568	50	7605	1913
CH	8	20640	50	16462	4128
KCH	15	21613	50	17191	4322

All feature values in all datasets were normalized between -1 and 1. As noted in Table 6.1, we randomly generated 5 test sets and 5 training sets using 5-fold cross-validation, from which we randomly chose 50 samples to serve as the labelled data; the remaining samples were unlabeled data. All experiments were conducted on a computing node comprising 26 CPU cores of 2.11 GHz and a memory of 95 GB. The evaluation metric was normalized root mean square error (NRMSE), calculated by

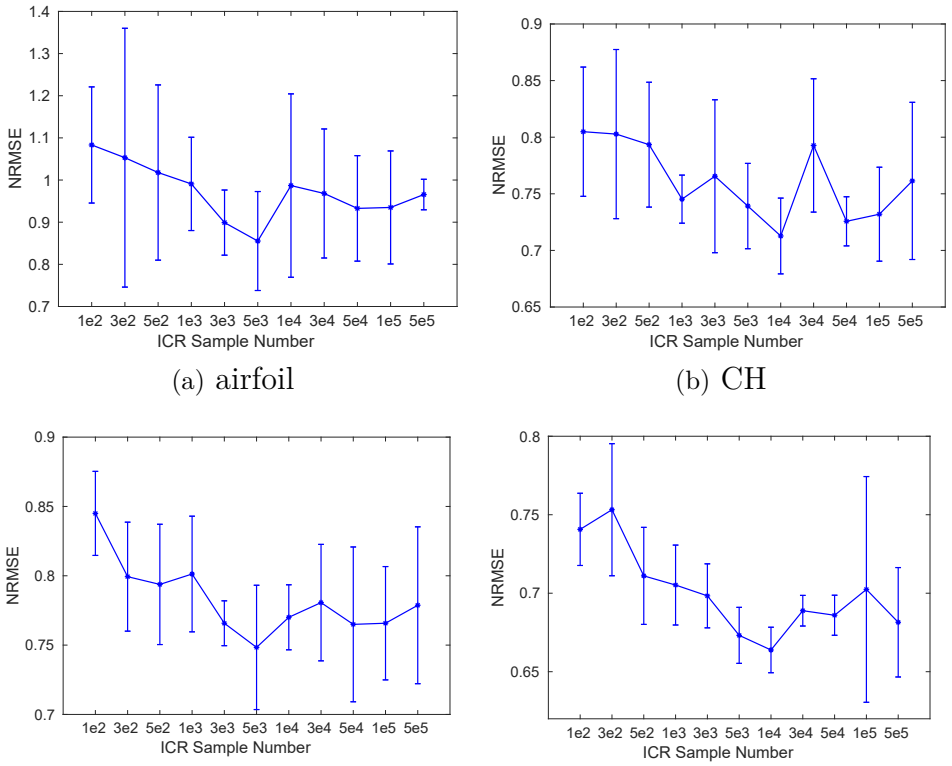
$$\text{NRMSE} = \sqrt{\frac{1}{N\hat{\sigma}_Y^2}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2}, \tag{5.39}$$

Table 5.2: Performance comparison on each dataset

Datasets	Algorithms	Performance					
		Centralized Methods			Distributed Methods		
		Test NRMSE	T-test	Time(s)	Test NRMSE	T-test	Time(s)
Airfoil	FR	0.6657/0.0188	0.0011	0.0581	0.6664/0.0334	0.0052	0.3283
	LapWNN	1.3967/0.2478	0.0002	0.2151	1.4428/0.3934	0.0000	7.1624
	s-SFR	2.4290/1.5583	0.0419	0.1183	1.6619/0.6212	0.0124	0.3123
	SFR-G	1.3124/0.3607	0.0078	0.2066	1.3578/0.1901	0.0043	0.4671
	SFR-ICR	**0.7423/0.0288**	-	0.1363	**0.7506/0.0087**	-	0.8115
WMSL	FR	0.6702/0.0176	0.0047	0.2811	0.6770/0.0237	0.0017	0.6251
	LapWNN	**0.7475/0.0865**	0.0323	27.741	0.8734/0.0576	0.0110	10.678
	s-SFR	0.9871/0.0788	0.0044	0.2953	0.8634/0.0458	0.0013	0.6639
	SFR-G	0.9199/0.0474	0.0468	1.0250	0.8846/0.0412	0.0015	1.0134
	SFR-ICR	0.7648/0.0311	-	0.5166	**0.7588/0.0312**	-	1.4836
CCPP	FR	0.2472/0.0075	0.0003	0.1173	0.2472/0.0075	0.0003	0.3059
	LapWNN	0.7475/0.0865	0.0001	27.741	0.8734/0.0576	0.0001	10.678
	s-SFR	0.3342/0.0528	0.0571	0.0878	0.3263/0.0227	0.0022	0.2690
	SFR-G	0.3187/0.0412	0.0811	7.6291	0.3557/0.0820	0.0091	1.1656
	SFR-ICR	**0.2809/0.0100**	-	0.1227	**0.2856/0.0130**	-	0.5945
KCH	FR	0.5631/0.0080	0.0019	0.2485	0.5657/0.0038	0.0001	0.8836
	LapWNN	0.7262/0.0266	0.0969	93.464	0.9348/0.0291	0.0256	24.184
	s-SFR	0.8063/0.0433	0.0041	0.1947	0.7704/0.0788	0.0103	0.9245
	SFR-G	0.7728/0.0432	0.0123	89.677	0.7915/0.0922	0.0011	9.9487
	SFR-ICR	**0.6952/0.0270**	-	0.5664	**0.6673/0.0211**	-	1.9435
CH	FR	0.5426/0.0180	0.0000	0.8325	0.5394/0.0219	0.0000	3.6808
	LapWNN	1.0023/0.0650	0.0223	94.025	1.0382/0.0294	0.0000	24.801
	s-SFR	0.9387/0.1030	0.0012	0.6695	0.9498/0.1210	0.0457	2.8925
	SFR-G	0.9886/0.0113	0.0093	53.773	0.9313/0.0584	0.0324	9.7301
	SFR-ICR	**0.6875/0.0406**	-	1.1965	**0.7703/0.1217**	-	5.2945
Artificial	FR	1.0067/0.0022	0.0435	0.3371	1.0069/0.0039	0.0017	0.7828
	LapWNN	1.2384/0.0824	0.0052	2.5700	1.0891/0.0462	0.0034	8.2684
	s-SFR	1.1590/0.1202	0.0027	0.2177	1.2021/0.1627	0.0015	0.8537
	SFR-G	1.0715/0.0478	0.0261	1.9241	1.0653/0.0756	0.0007	1.1028
	SFR-ICR	**1.0409/0.0339**	-	0.5266	**1.0121/0.0094**	-	1.4357

To analyze the convergence of our proposed model and the influence of the rule number and the sample number that is chosen for ICR regularization, we fix the Laplacian parameter ρ_p and ρ_s at 0.1 and 0.1, and the parameters μ, γ, and η, which is corresponding to the L_2norm regularizer, the mix-up regularizer, and the graph-based regularizer, at 0.1, 0.1, and 0.0001, respectively.

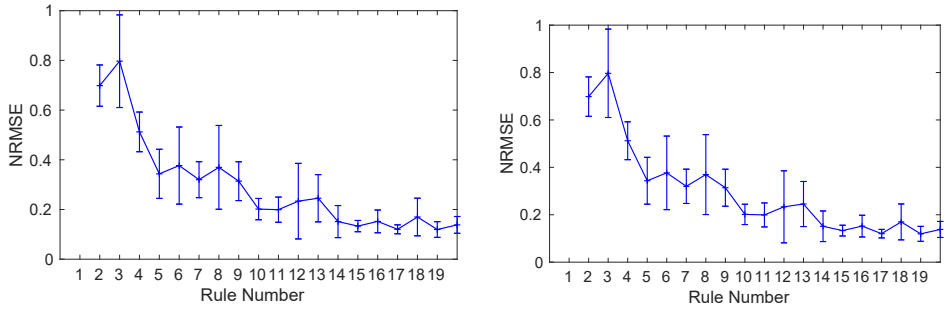
To obtain the best results of our model, all parameters are chosen from an interval of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Besides, we set the FCM parameter α as 1.1 to



(a) airfoil (b) CH

(c) WMSL (d) KCH

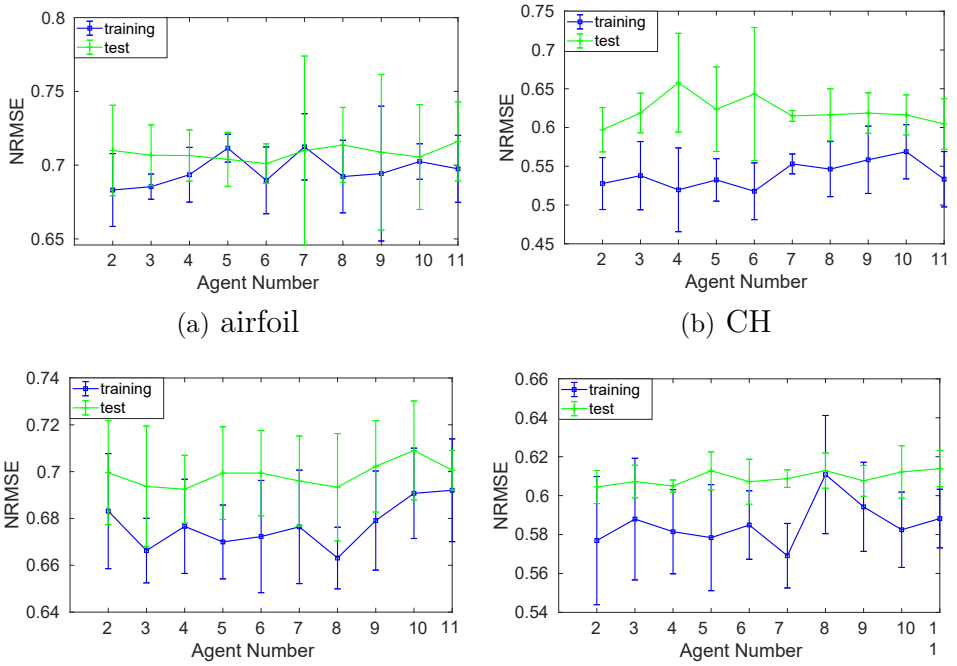
Figure 5.2: NRSME and its standard deviation when varying the number of ICR samples.



(a) CH (b) KCH

Figure 5.3: NRSME and its standard deviation when varying the number of rules.

increase the diversity of fuzzy rules in all experiments. We set the stopping threshold for DFCM and DICR both as 10^{-4}.



(a) airfoil (b) CH

(c) WMSL (d) KCH

Figure 5.4: NRSME and its standard deviation when varying the number of agents

5.3.1 Performance on Different Datasets

All parameters in this section were chosen from an interval of $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. Further, the FCM parameter α was set to 1.1 for all experiments to increase the diversity of fuzzy rules. We repeated each experiment 10 times, reporting the average NRMSE and its standard variance as the results. For a fair comparison, we arranged the training sets on interconnected networks with five fully connected agents, setting the number of rules to 5 for all agents with all datasets. We used 50 samples in the centralized scenario and 10 samples for each agent in the distributed setting.

We compared several alternative methods within the semi-supervised fuzzy regression (SFR) framework that using different regularization, fully-supervised regression (FR) model, and one SSL model. All methods are compared in both centralized and distributed ways and described as follows:

- s-SFR: A variant of SFR, this method applies FCM method in the structure learning process to leverage both labelled and unlabeled samples. A ℓ_2 regularization is used in the parameter learning process to enhance model generalization performance.

- **SFR-ICR**: Our proposed SFR model, based on s-SFR, this fuzzy regression model adds an ICR term in the parameter learning process to further utilize unlabeled data.

- SFR-G: Another variant of SFR, different from SFR-ICR, this model involves a graph-based regularization term. We designed this method to show the efficiency and effectiveness of SFR-ICR under the same settings.

- FR: This is a fully-supervised fuzzy regression model. FR can provide a lower bound of the loss for the SFR-ICR.

- LapWNN: This method involves a graph-based SSL based on a wavelet neural network (WNN) [86]. Since LapWNN was developed for distributed semi-supervised scenarios, thus it is a quite relevant baseline method.

The results shown in Table 5.2, place SFR-ICR as the clear outstanding performer in both the centralized and distributed settings, providing a strong endorsement of the effectiveness of semi-supervised fuzzy regression. T-tests were implemented to show statistical significance between SFR-ICR and other methods. As shown in Table 5.2, all the p-values are smaller than 0.05. Thus the difference between our SFR-ICR and other methods is significant.

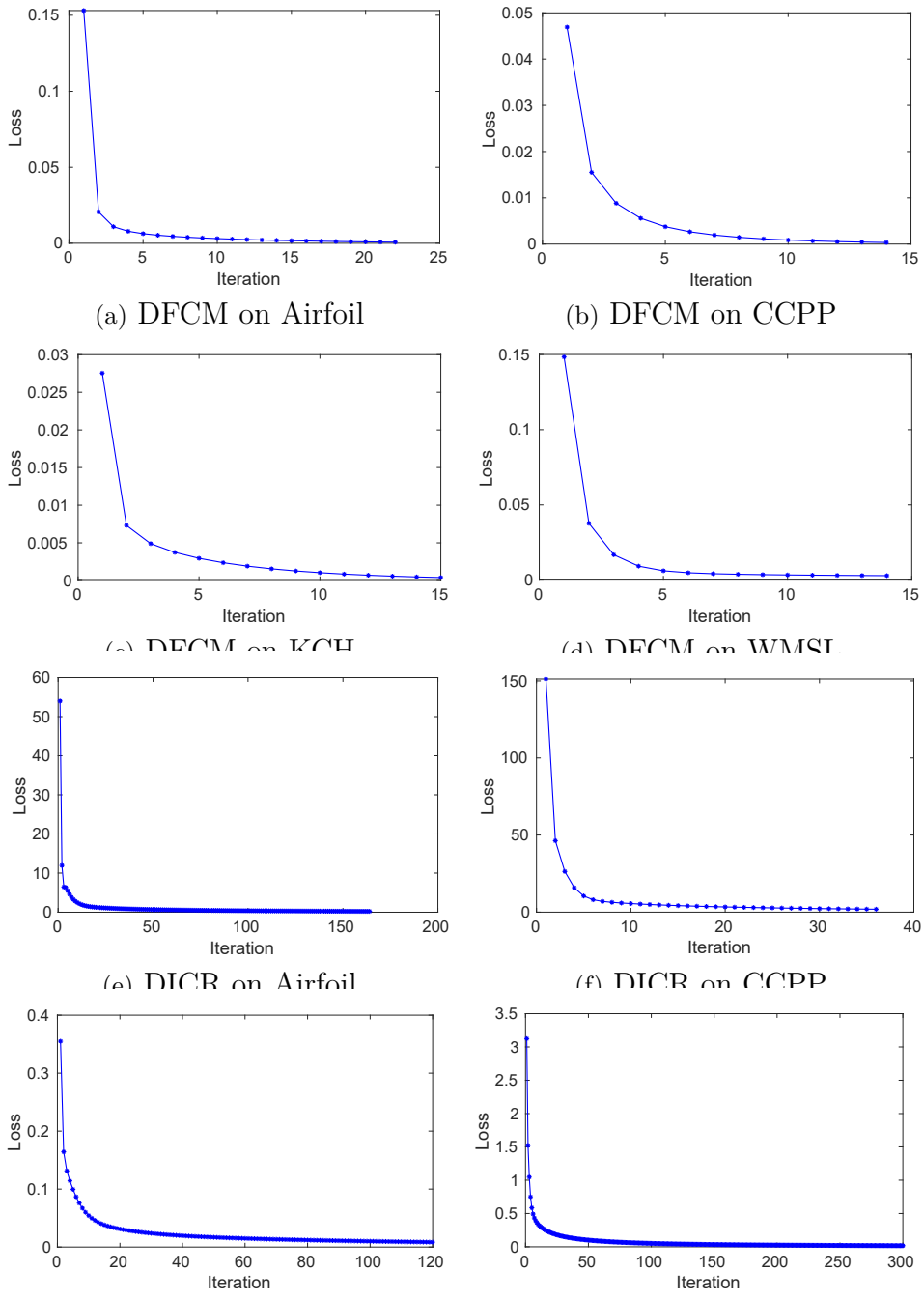
Notably, SFR-G outperformed LapWNN on all datasets in the distributed scenarios, indicating that FNN-based SSL outperforms WNN-based SSL methods. Particularly, SFR-ICR reduced 31.96% NRMSE values compared with the distributed results by LapWNN on average of all datasets. Within the SFR scheme, the NRMSE is also smaller by SFR-ICR than by s-SFR and SFR-G, closely outpacing FR, which indicates that ICR has a much greater ability to extract useful information from unlabeled samples than other regularization methods. Particularly, SFR-ICR respectively reduced 23.19% and 19.44% NRMSE values compared with the distributed results by s-SFR and SFR-G on average of all datasets.

The SFR-ICR is also superior in efficiency. SFR-ICR costs far less time than other methods in centralized ways. For distributed methods, the reason the computing speeds of some small-scale datasets is slower is that communication overhead among agents offsets the computing superiority of SFR-ICR.

We also assessed distributed SFR-ICR (DSFR-ICR) against its centralized counterpart SFR-ICR (CSFR-ICR). DSFR-ICR performed better on all datasets except for California Housing, demonstrating that a distributed approach can further increase the model performance by spreading computing resources across different agents.

5.3.2 Convergence Analysis

To investigate the convergence behaviours of the proposed algorithms, we plotted the loss values of DFCM and DICR with each iteration of the optimization procedure, as shown in Fig. 5.5. In this and the following subsections, we initialized the Laplacian parameter ρ_p and ρ_s at 0.1 and 0.1, and the parameters μ, γ, and η, which is corresponding to the ℓ_2 regularizer, the mix-up regularizer, and the graph-based regularizer, at 0.1, 0.1, and 0.0001, respectively. Due to space limitations, we randomly choose four datasets to show the results. DFCM converges within 25 iterations while DICR converges within 300 iterations for all the datasets. We can find that the required iteration number for DICR is quite larger than that for DFCM. The reason is that the training process of DICR is more complex than DFCM. Note that DICR needs to randomly generate virtual samples in a pair-wise manner to augment the training set. In this way, DICR can better exploit unlabeled samples but sacrifice the convergence speed.



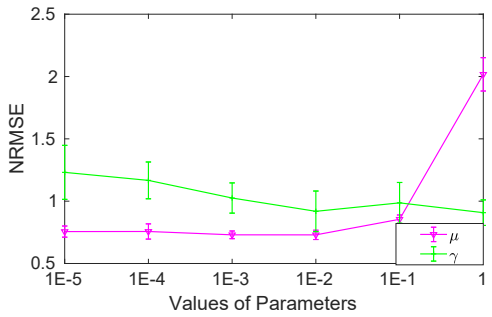
(a) DFCM on Airfoil (b) DFCM on CCPP

(c) DFCM on KCH (d) DFCM on WMSL

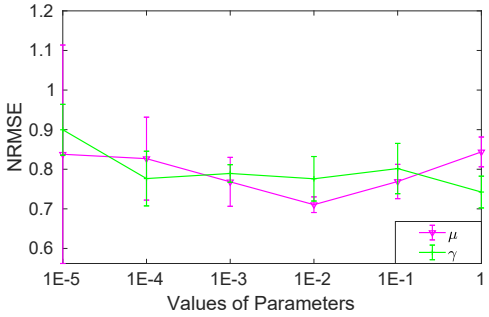
(e) DICR on Airfoil (f) DICR on CCPP

(g) DICR on KCH (h) DICR on WMSL

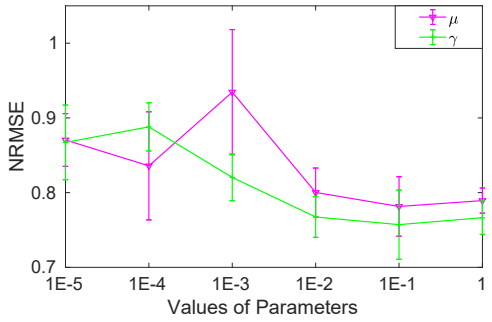
Figure 5.5: Convergence analysis of the DFCM method (top) and DICR method (bottom).



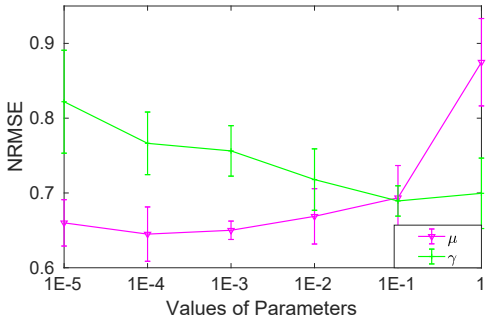
(a) Model parameters on Airfoil



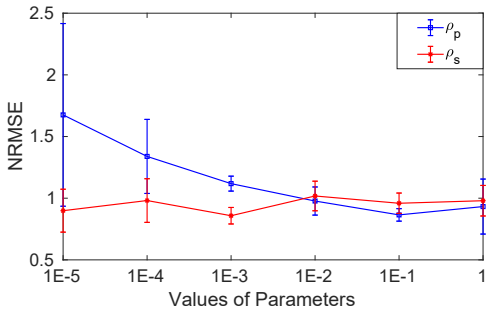
(b) Model parameters on CH



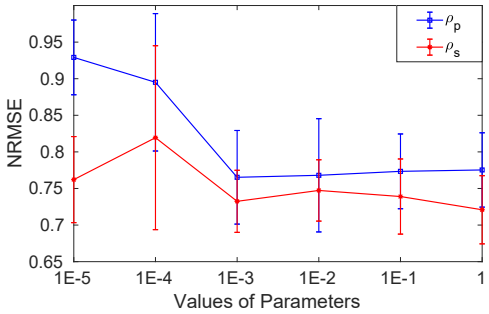
(c) Model parameters on WMSL



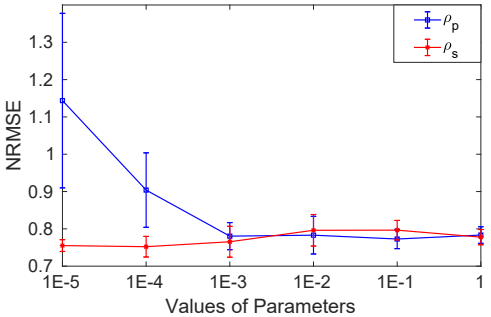
(d) Model parameters on KCH



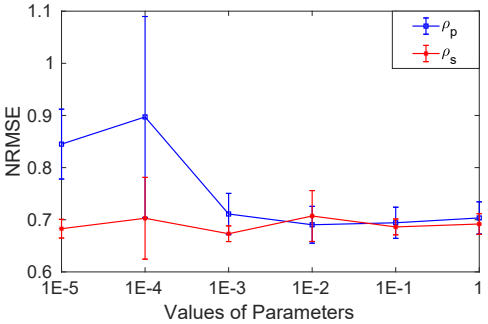
(e) Laplacian parameters on Airfoil



(f) Laplacian parameters on CH



(g) Laplacian parameters on WMSL



(h) Laplacian parameters on KCH

Figure 5.6: NRMSE and its standard deviation with different model parameters (top) and Laplacian parameters (bottom).

5.3.3 Effects of regularization and ADMM parameters

As is convention, we assessed the influence of each parameter by fixing all other parameters using the initialization mentioned in Section V.B and varying the parameter in appropriate intervals: $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. The results of the parameter analysis are presented in Fig. 5.6.

In terms of the regularization parameters γ and μ, the NRMSE drops as parameter values increase, reaching its nadir at $\mu = 1$ with all datasets. This suggests that ICR plays a critical role in the model. The best result for μ, however, differs for every dataset. On Airfoil, CH and KCH datasets, μ decreases before reaching its bottom, then starts to climb again, which indicates that the proportion of the ℓ_2 term needs to be within a reasonable range.

The variance in the ADMM parameter ρ_s at different values was very small. ρ_p performs better with values greater than 10^{-3}. Thus, we can conclude that ρ_s is robust at a wide range of values, but ρ_s performs better when choosing a relatively large value.

5.3.4 Effects of the number of interpolated unlabeled samples

We also investigate how the number of interpolated unlabeled samples affects the performance of DICR. After fixing the other parameters, we set the interpolated sample number within the interval $\{100, 300, 500, 1K, 3K, 5K, 10K, 30K, 50K, 100K, 500K\}$. The results are plotted in Fig. 5.2. All curves show a trend that declines in the beginning and then climbs after reaching a minimum NRMSE. The best performance with the Airfoils and WMSL datasets was with 5K ICR MixUp samples. With CH and KCH, the best performance was at 10K samples. Hence, it is best to set the number of MixUp ICR samples near the training sample numbers for each dataset.

5.3.5 Effects of the rule and agent number

The number of rules was varied in the interval of $\{2, 3, \cdots, 20\}$. As shown in Fig. 5.3, the more rules, the better the performance. Here, we randomly selected 500 labelled training samples and distributed them relatively evenly over different numbers of agents. As shown in Fig. 5.4, DSFR was able to stabilize with a range of agent numbers.

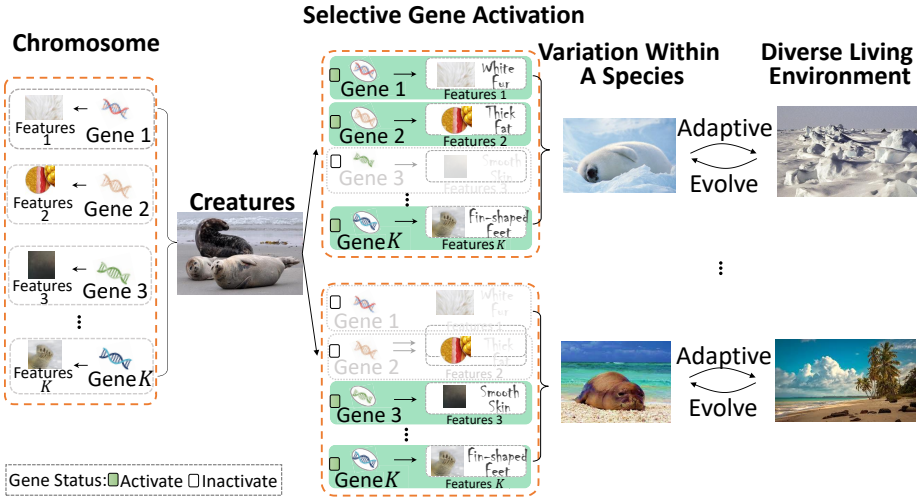
5.4 Summary

In this chapter, we proposed a novel DSFR model with fuzzy if-then rules and ICR. The result is a framework that handles uncertainty and also dramatically reduces computation and communication overheads in interconnected networks with multiple agents. Adopting the ADMM strategy, DSFR involves a DFCM for structure learning and a DICR for parameter learning. Notably, both algorithms can extract useful information from not only labelled samples but also unlabeled ones. The DICR algorithm increases model performance and robustness by restricting the decision boundaries to low-density regions. Comprehensive experiments on both artificial and real-world datasets show that DSFR excels in efficiency and effectiveness compared to state-of-the-art algorithms. A deep variant of DSFR will be considered in future work to handle high-dimensional unlabeled data. This method will modify fuzzy logic to make them more suitable to high-dimensional applications. In addition, we will extend the current distributed learning methods to suit more complex distributed scenarios.

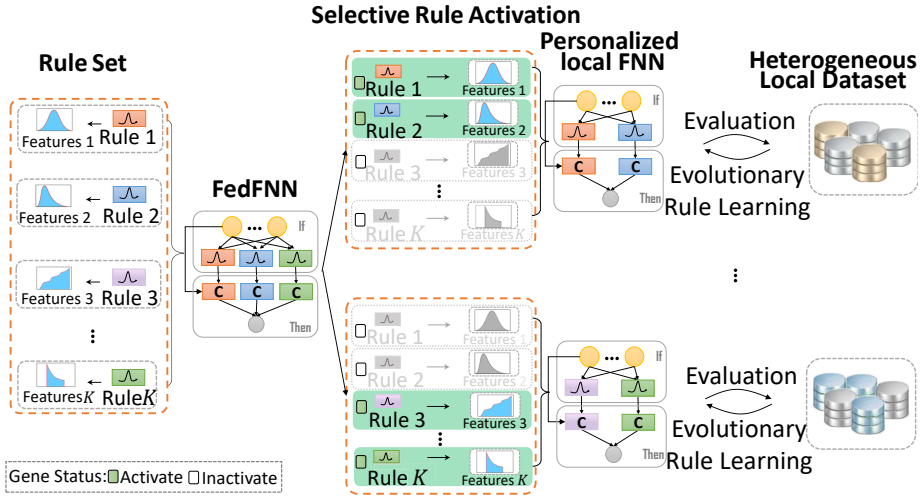
Chapter 6

Federated Fuzzy Neural Network with Evolutionary Rule Learning

To address the challenges of dimensionality, data heterogeneity, and privacy preservation outlined in RO3, this chapter introduces a novel approach called the federated fuzzy neural network (FedFNN) with evolutionary rule learning (ERL). The FedFNN with ERL is designed to effectively handle data uncertainties and non-IID (non-independent and identically distributed) issues commonly encountered in federated learning settings. By employing this approach, we aim to mitigate the impact of dimensionality, accommodate diverse data characteristics, and ensure privacy preservation during the learning process. As shown in Fig. 6.1 (a), the theory of biological evolution [190] states that variants of the same species can evolve to adapt to their different living environments by selectively activating and expressing their genes. Inspired by this, we use FNNs as our local models and consider them as compositions of fuzzy rules, which capture valuable local data information from multiple views, such as distributions. Similar to the genes of a species, each rule of the FedFNN is a basic functional component that can be activated or deactivated for clients according to their performance on local data. Thus, our FedFNN aims to obtain a group of global fuzzy rules that can be selectively activated for local clients to enable them to outperform competing approaches on non-IID data. It is worth noting that our ERL is a novel algorithm different from genetic algorithms [191]. These two algorithms are inspired by the same theory but designed in different ways. The genetic algorithm treats the whole population as the evolution subject. It keeps selecting the fittest individuals in each generation to form more competitive populations until the performance gets stable. However, it needs to be re-adjusted whenever the environment is modified. Instead, the ERL considers the internal di-



(a) The evolvement of variants within a species via the selective activation of genes.



(b) The rule selective activation for FedFNN to generate personalized local FNNs.

Figure 6.1: (a) A brief demonstration of how a species evolve variants to survive in diverse living environments based on genes selective activation and expressions. (b) A brief demonstration of how FedFNN selectively activate a personalized subset of contributive rules for clients to effectively deal with their local non-IID data.

versity of species, which means that a species normally includes several types of populations (subspecies) living in diverse environments. It regards subspecies as its evolution subjects and selectively optimises and shares gene groups among subspecies until they perform well in different environments. In our FedFNN, each agent that handles non-IID data corresponds to a sub-species. The ERL enables agents to co-operate with each other during their evolutions but preserve their personalities to adapt to diverse environments. In general, our FedFNN aims at learning 1) a group of global rules that capture valuable information among local clients and 2) a rule activation strategy for each local client to ensure the personalization and superior performance of the FedFNN.

The ERL is an iterative learning approach with two stages: a rule cooperation stage, where the global rules are updated cooperatively by clients based on their rule activation statuses, and a rule evolution stage, where the activation statuses of local rules are adjusted according to their contributions to the performance of the FedFNN on non-IID local data. The former stage enhances the cooperation among clients for learning more representative global rules, which increases the generalizability of the FedFNN. In contrast, the latter stage fulfils the selective activation of rules that enable the local FNNs to be more adaptive and perform better on non-IID data, which improves the personalization of the FedFNN. For an explanation of the FedFNN model, refer to the diagram shown in Fig. 6.1 (b).

The contributions of this chapter are as follows,

- We are the first to propose FedFNN that integrates fuzzy neural networks into a federated learning framework to handle data non-IID issues as well as data uncertainties in distributed scenarios. FedFNN is able to learn personalized fuzzy if-then rules for local clients according to their non-IID data.
- Inspired by the theory of biological evolution, we design an ERL algorithm for the learning of FedFNN. ERL encourages the global rules to evolve while selectively activating superior rules and eliminating inferior ones during the training procedure. This procedure ensures the ability of generalization and personalization of FedFNN.

6.1 Federated Fuzzy Neural Network

In this section, we describe the general structure of our FedFNN. As depicted in Fig. 6.2, the FedFNN includes one server and several local clients. The server is responsible for communication with local clients and maintaining a group of global

rules by aggregating the uploaded local rules. Local clients download the global rules as local rules for constructing their FNNs, which are then updated via training on their own data. Due to the concerns of data privacy, each local agent learns from its own data without accessing the data of other agents and communicates, while the server communicates with all local agents and aggregates the locally learned rules according to their activation status. An overview of a local FNN is shown in Fig. 6.3.

To mimic the selective activation of genes, the rules in the local clients are activated selectively to make the FedFNN personalized and properly adapted to local non-IID data. Thus, we introduce an activation vector containing the rule status s_k^q of each client. Accordingly, the global server can help local clients activate useful rules and deactivate useless or harmful rules based on their own local data. For example, if $s_k^q = 0$, the server will deactivate the k-th rule for the FNN on the q-th client; otherwise, the corresponding rule will be activated and involved in the operations of the q-th client.

In the FedFNN, we adopt the fuzzy logic presented in the first-order T-S fuzzy system [127]. Suppose that our FedFNN holds Q clients; then, the dataset owned by the q-th client can be denoted as $\mathcal{D}^q := \{x_i^q, y_i^q\}_{i=1}^{N^q}$, where $x_i^q = [x_{i1}^q, x_{i2}^q, \cdots, x_{iD}^q]^T$ and $y_i^q \in \mathbb{R}^C$ are the i-th sample and its one-hot vector label, respectively, and C and N^q denote the category number and the local dataset size. Then, the k-th fuzzy rule of the local FNN on the q-th client can be described as

$$\begin{gathered}\text{Rule } k\text{: If } x_{i1}^q \text{ is } A_{k1}^q \text{ and } \cdots \text{ and } x_{iD}^q \text{ is } A_{kD}^q \\ \text{Then, } y_i^q = g^q(x_i; \theta_k)\end{gathered}$$

where A_{kj}^q is the j-th fuzzy set of rule k on P^q and $g^q(x_i; \theta_k)$ is the corresponding consequent rule built by a fully connected layer parameterized with θ_k. Many types of membership functions can be employed to describe the fuzzy set A_{kj}^q, such as singleton, triangular, trapezoidal, and Gaussian ones [192]. Here we choose the Gaussian membership function for three reasons: 1) Gaussian membership function is differentiable, which is more suitable for end-to-end learning models; 2) the Gaussian membership function is proved to be effective in approximating nonlinear functions on a compact set [164]; 3) Gaussian membership functions are able to represent data features using different Gaussian distributions, which can capture data heterogeneity and handle data uncertainty using several distributions.

Generally, the Gaussian membership function $\varphi^q(x_{ij}^q; m_{kj}^q, \sigma_{kj}^q)$ of a fuzzy set A_{kj}^q has the form

$$\varphi_k^q(x_{ij}^q; m_{kj}^q, \sigma_{kj}^q) = \exp\left[-\left(\frac{x_{ij}^q - m_{kj}^q}{\sigma_{kj}^q}\right)^2\right], \tag{6.1}$$

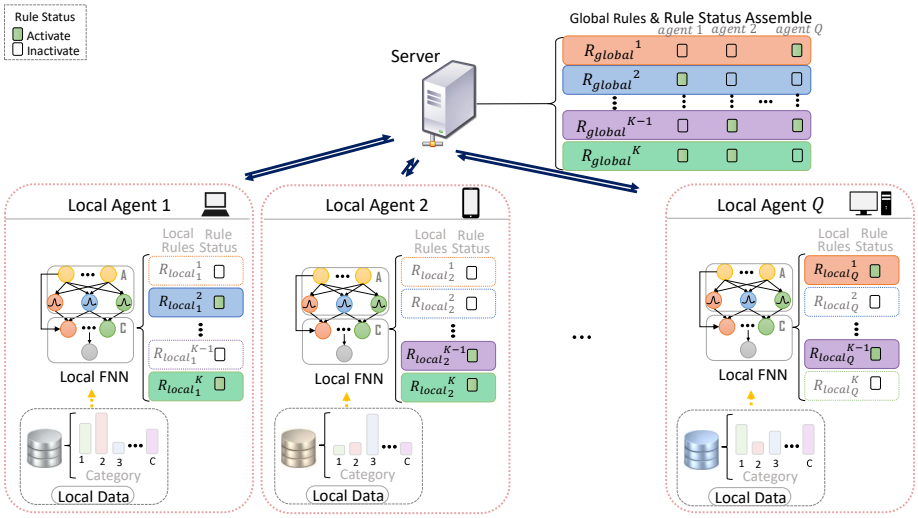


Figure 6.2: Overview of the FedFNN.

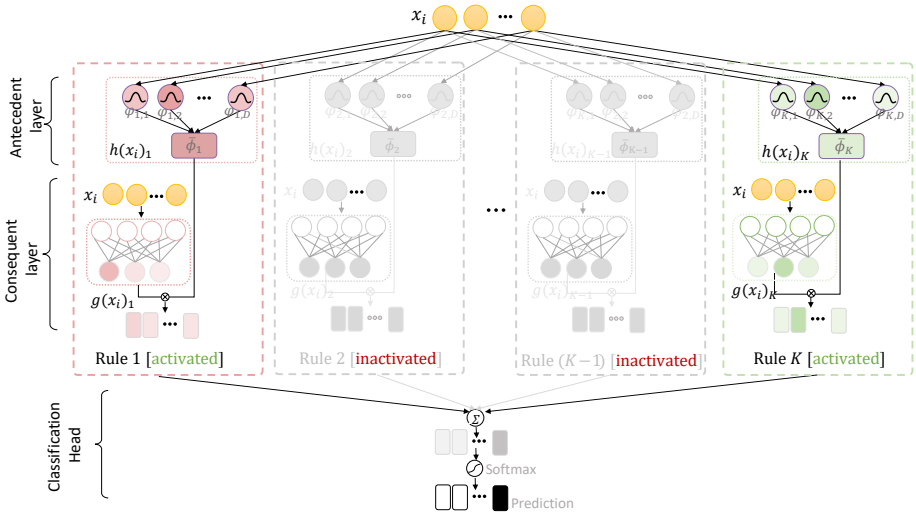


Figure 6.3: Overview of a local FNN.

where m_{kj} and σ_{kj} are the mean and standard deviation of the Gaussian membership function, respectively. Let $\varphi_k^q(x_i^q; m_k^q, \sigma_k^q)$ collect the membership value of the k-th rule on client P^q in vector form. The output of the antecedent layer (also known as the firing strength) considering the rule activation status s_k^q can be written as:

$$h_k^q(x_i^q; m_k^q, \sigma_k^q, s_k^q) = \frac{s_k^q \exp[||\varphi_k^q(x_i^q; m_k^q, \sigma_k^q)||_2]}{\sum_{k=1}^K s_k^q \exp[||\varphi_k^q(x_i^q; m_k^q, \sigma_k^q)||_2]}, \tag{6.2}$$

The consequent output of the k-th rule is denoted as $g^q(x_i^q; \theta_k^q)$ and can be calculated by:

$$g_k^q(x_i^q; \theta_k) = [1; x_i^q]^T \theta_k^q, \tag{6.3}$$

where $\theta_k^q \in \mathbb{R}^{(D+1)\times C}$ denotes the consequent parameters. Thus, by considering rule statuses, the FedFNN is able to eliminate the interruption of deactivated rules for local clients. The antecedent layer $h^q(x_i^q; m^q, \sigma^q)$ and the consequent layer $g^q(x_i^q; \theta^q)$ of the q-th client in our model are

$$h^q(x_i^q; m^q, \sigma^q) = (h_1^q(x_i^q; m_1, \sigma_1, s_1^q), \cdots, h_K^q(x_i; m_k^q, \sigma_k^q, s_K^q)), \tag{6.4}$$

and

$$g^q(x_i^q; \theta^q) = (g_1^q(x_i^q; \theta_1), \cdots, g_K^q(x_i^q; \theta_k^q)), \tag{6.5}$$

respectively. A classification head is further added to the tail to generate the final predictions by considering the outputs of all K rules. The classification head connects all rules, which guarantees that the gradient of the loss function can successfully propagate backward to every component of the local FNNs. Thus, the predictions of the q-th local FNN $f^q(x_i^q; m^q, \sigma^q, \theta^q, s^q)$ can be represented as:

$$f^q(x_i^q; m^q, \sigma^q, \theta^q, s^q) = \text{softmax}\,(\tau)\,, \tag{6.6}$$

where $\tau = \sum_{k=1}^K h_k^q(x_i^q; m_k^q, \sigma_k^q, s_k^q) g_k^q(x_i^q; \theta_k^q)$. Suppose that $w^q = (m^q, \sigma^q, \theta^q)$ collect the parameters of all local rules on client q; the loss of the q-th local FNN can then be defined as

$$\ell^q(x_i^q; w^q, s^q) = -\sum_{c=1}^{C} y_{ic} \log(\hat{y}_{ic}), \tag{6.7}$$

where $\hat{y}_{ic}$ is the c-th output of $f^q(x_i^q; w^q, s^q)$, and its goal is to optimize

$$\min_{w^q, s^q} \mathbb{E}_{(x,y)\sim\mathcal{D}^q}[\ell^q((x, y); w^q, s^q)]. \tag{6.8}$$

Thus, the training objective of our proposed FedFNN can be given by

$$\begin{aligned}\arg\min_{\Theta} \frac{1}{Q}\sum_{q=1}^{Q}\ell^q(x_i^q; w^q, s^q) \\ = \arg\min_{\Theta} \frac{1}{Q}\sum_{q=1}^{Q}\frac{1}{N^q}\sum_{i=1}^{N^q}\ell^q(x_i^q; w^q, s^q),\end{aligned} \tag{6.9}$$

where Θ denotes the set of personal parameters $\{w^q, s^q\}_{q=1}^{Q}$.

6.2 Evolutionary Rule Learning

In this section, we present an in-depth introduction to the ERL method, which enables the FedFNN to be personalized and achieve superior performance on non-IID data. As shown in Fig. 6.4, ERL includes a rule cooperation stage that improves the generalization of global rules and a rule evolution stage that enhances the personalization of the FedFNN. The above two stages are presented in subsection IV.A and subsection IV.B, respectively.

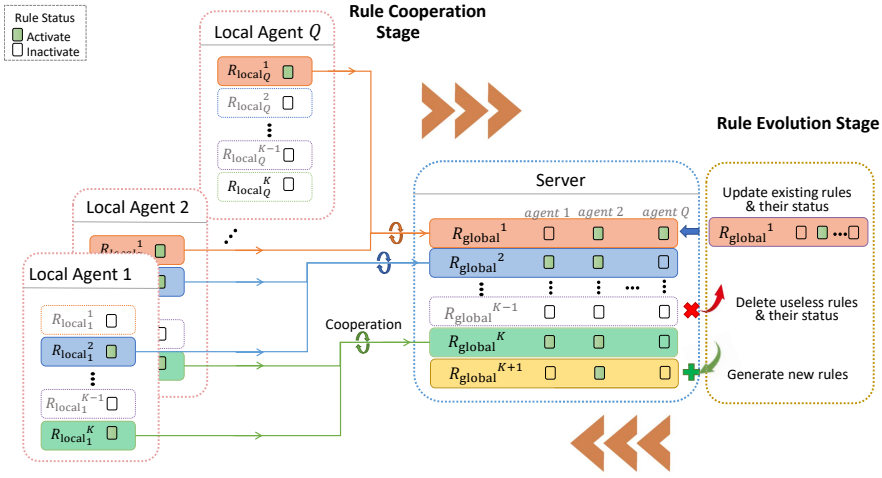


Figure 6.4: Overview of the evolutionary rule learning.

6.2.1 Rule Cooperation Stage

In this stage, we focus on the learning of more general global rules. Generally, the global rules activated by multiple clients are able to capture informative rep-

Algorithm 4 Co-Evolutionary Stage of the ERL Method

Input: Number of clients Q, number of global rules K, number of local epochs E, learning rate η, global ruleset parameter $w(t)$.
Output: The global ruleset parameter $w(t+1)$ for the next round.

Server executes:
for $q = 1, 2, \cdots, Q$ in parallel, do **do**
  send global ruleset parameter $w(t)$ to P^q
  send rule activation status $s^q(t)$ to P^q
  receive $w^q(t)$ from **Local Training**
end for
for $k = 1, 2, \cdots, K$ **do**
  update the k**-th rule** $w_k(t+1)$ using (6.10)
end for

Local Agent Training:
$w^q(t) \leftarrow w(t)$
for epoch $i = 0, 1, 2, \cdots, E$ **do**
  for each batch $\mathbf{b} = \{x, y\}$ of $\mathcal{D}^q$ **do**
    calculate ℓ via (6.7)
    update $w^q(t)$ via $w^q(t) \leftarrow w^q(t) - \eta \nabla \ell$
  end for
end for
return $w^q(t)$ to server

Algorithm 5 Adaptive Evolutionary Stage of the ERL Method

Input: Number of clients Q, number of global rules K, global rule parameter set $w(t)$, hyperparameter β.

Output: The global ruleset parameter $w(t+1)$, the rule status $s(t+1)$, and the rule number K for the next round.

Server executes:

for $q = 1, 2, \cdots, Q$ in parallel, do **do**
  for $k = 1, 2, \cdots, K$ in parallel, do **do**
    update $s_k^q(t+1)$ via (6.12)).
  end for
  if (6.13) and (6.14) both hold or $\sum s^q(t+1) = 0$ **then**
    generate a new rule for client P^q and randomly initialize its parameter w_{K+1}.

    expand global parameter set $w(t+1)$ via $w(t+1) \leftarrow [w(t), w_{K+1}]$.
    expand rule status $s(t+1)$ via $s(t+1) \leftarrow [s(t), \mathbf{0}]$, $s_{K+1}^q(t) = 1$.
    update K via $K = K + 1$
  end if
end for
for $k = 1, 2, \cdots, K$ **do**
  if $\sum_1^Q s_k^q = 0$ **then**
    remove rule k from global rule list and w_k from global parameter set w.
    update K via $K = K - 1$
  end if
end for

resentations across these clients. Thus, updating the general global rules requires cooperation among the local clients.

Technically, the rule cooperation stage highly relies on the rule activation status vectors of the local clients. These vectors are randomly initialized at the beginning and updated in the rule evolution stage. During each communication round, the local clients download the global rules from the server as their local rules. Then, the global server selects the activated rules according to the corresponding rule activation status vectors to build the local FNNs. As described in the Local Training of the algorithm 4, the constructed personalized local FNNs are then trained on their associated non-IID local data to update the activated local rules.

Afterwards, each global rule is updated by calculating the weighted average of the activated local rules. Thus, the parameters of the k-th global rule w_k are updated by activation-status-driven weight averaging:

$$w_k = \sum_{q=1}^{Q} \frac{N^q s_k^q}{\gamma_k}(w_q^k), \tag{6.10}$$

where $\gamma_k = \sum_{q=1}^{Q} N^q s_k^q$ is the factor that measures the contribution of a global rule. This stage is conducted for L rounds before stepping to the rule evolution stage to ensure that the global rules are effectively learned under cooperation among the local clients.

It is worth noting that each global rule is updated by aggregating its corresponding activated local rules instead of aggregating all of them. This is different from existing FL methods that aggregate local models without checking whether their updates are helpful or not. Our ERL approach only shares the useful pieces of information that contribute to each other while avoiding misleading information. This procedure enables the FedFNN to be more general.

6.2.2 Rule Evolution Stage

After L rounds of coevolutionary learning, the FedFNN optimization process falls into a bottleneck since the capability of the current model structure has been fully explored. In this stage, we allow the server to inspect the performance of the learned model on local samples and then implement the FedFNN to increase its performance and personalization for handling non-IID data. In general, as described in algorithm 5, the rule evolution stage includes 3 major mutations: evolving new global rules, activating superior rules, and deactivating local inferior rules.

We adopt a contribution factor π_k^q to measure the importance of the k-th rule on the q-th client, which can be calculated as

$$\pi_k^q = \sum_{i=1}^{|\mathcal{D}^q|} \frac{h_k^q(x_i^q; m_k^q, \sigma_k^q, s_k^q)}{|\mathcal{D}^q|}, \tag{6.11}$$

where $|\mathcal{D}^q|$ is the size of $\mathcal{D}^q$. Here, we introduce a threshold $\bar{\pi}^q$ to trigger the rule activation procedure. $\bar{\pi}^q$ is calculated based on the average contribution of the activated rules as $\bar{\pi}^q = \beta \sum_{k=1}^{K} s_k^q \pi_k^q / K^q$, where β is a hyperparameter and K^q is the number of activated rules in the q-th client. A rule is meaningless to client k when its contribution level is smaller than $\bar{\pi}$. Consequently, the server deactivates this rule for the corresponding clients and eliminates the involvement of those clients in the rule aggregation process. To assure that this step is completed, the rule status $s_k^q(t)$ in the t-th round is updated by a status adjusting operation:

$$s_k^q(t) = \begin{cases} 1, & \pi_k^q > \bar{\pi}_k^q \\ 0, & \text{otherwise} \end{cases} \tag{6.12}$$

It is worth noting that each client should inspect its activated rules to check if the combination of these rules is sufficient for evaluating its local dataset. If not, it means that some of the unique features are not captured by the current activated rule settings. To solve this problem, we introduce two conditions to evaluate the capabilities of local models from different views:

$$\sum_{l=1}^{L} \frac{\ell^q(t-l+1) - \ell^q(t-l)}{L} > 0, \tag{6.13}$$

$$\left(\ell^q(t) - \sum_{q=1}^{Q} \frac{\ell^q(t)}{Q} \right) > 0, \tag{6.14}$$

where t is the current training round. The first condition (6.13) serves as a self-evaluation of the learning performance for each client by monitoring the loss value trends. If condition (6.13) holds, then the current local rules cannot be further improved. The second condition (6.14) serves as a peer evaluation by comparing the loss of the q-th client with the average loss across all clients. If condition (6.14) holds, then the current local rules cannot handle non-IID data well. If both of these conditions hold, then the current architecture in the q-th client is unable to attain high performance. In this case, a new rule should be generated to improve the local model. To avoid the extreme situation when certain agents have no rules to use,

the server will also create new rules for agents once their activated rule number is detected as zero.

This stage imitates the selective activation process of genes by updating the activation status of each rule based on its contribution to the local clients. Consequently, ERL selects useful rules for the local clients from the well-learned global rules, increasing the personalization of the FedFNN. In addition, the updated rule activation schemes benefit the optimization process in the next round of the rule cooperation stage in turn because of the better rule selection effect.

Table 6.1: Dataset Information

Dataset	Sample	Feature	Category
GSAD [173]	14,061	128	6
FM [175]	180	43	4
WD [176]	4,898	11	7
MGT [177]	19,020	10	2
SC [178]	58,000	9	7
WIL [179]	2,000	7	4
WFRN [180]	5,456	24	4

In this section, we conduct extensive experiments on 7 datasets of different types in various settings to evaluate the effectiveness of the proposed FedFNN. Collected from multiple scenes, the selected datasets are representative and widely used. The descriptive statistics of each dataset are listed in Table 6.1. In addition, to verify the superior performance of our model, state-of-the-art DFNNs and FL methods for constructing deep models are adopted as comparison methods. The details of all algorithms adopted in this section and their settings are introduced below.

- DFNN: This is the fully DFNN algorithm proposed in [9], which adopts consensus learning in both the parameter learning and structure learning procedures and achieves state-of-the-art performance among distributed fuzzy models. As mentioned before, this model learns a consensus FNN for all clients, which limits its applications in non-IID scenarios. We use DFNN+ and DFNN* to denote homogeneous and heterogeneous federated setups, respectively.

- FedAvg [21]: This is the most basic and popular algorithm for developing FL models. Similarly, we use FedAvg+ and FedAvg* to refer to the FedAvg models that cope with homogeneous and heterogeneous scenarios, respectively. Considering that the preferable deep model structures for different datasets vary from each other, to make the comparison convincing, we design more

than 20 types of deep models for the comparison and report the best one for each dataset in the following tables and figures.

- MOON [151]: This is the state-of-the-art FL approach for deep models. MOON adapts individual clients based on their dissimilarity with the server, which bestows the outperforming learning ability on the network model in solving heterogeneous distributed scenarios. Here, we utilize MOON for heterogeneous situations, namely, MOON*, for the comparison. Similar to FedAvg, we choose 20 deep models with different structures and list their best performance, as shown below.

- FedFNN: This is the model proposed in our paper, which adopts a rule evolution strategy to make full use of each fuzzy rule and acquire better estimation results for each individual dataset. We set the global number of rules K as 15, the number of ERL iterations as 15, and the number of coevolutionary rounds L as 10. For the hyperparameter setting, $\bar{\beta}$ is set as 0.7.

In our experiment, all the distributed models are assigned with 5 local clients, each of which can only access their own dataset. To simulate heterogeneous local datasets for these clients, we generate five non-IID local data partitions using the Dirichlet distribution $Dir(\alpha)$, where $\alpha \in (0, 100)$ is the concentration parameter and can be seen as the indicator of the non-IID level of the data. Technically, the sample proportions of all categories for all clients are sampled from $Dir(\alpha)$. The local datasets are thus generated based on random sampling from the original dataset based on the obtained category proportions.

It is worth noting that for the fairness of the experiments, the features in all datasets are first normalized between -1 and 1 using the well-known mapminmax normalization method. Then, a certain proportion of the features is randomly polluted by noise generated from a normal Gaussian distribution to simulate uncertainty in the different datasets. This perturbed sample proportion is considered the uncertainty level and is used to verify the uncertainty processing abilities of the comparison algorithms. In this section, all experiments are conducted with 5-fold cross-validation, and each experiment is repeated 10 times. The final reported results are the mean average precision (mAP) values obtained on the test data during these runs.

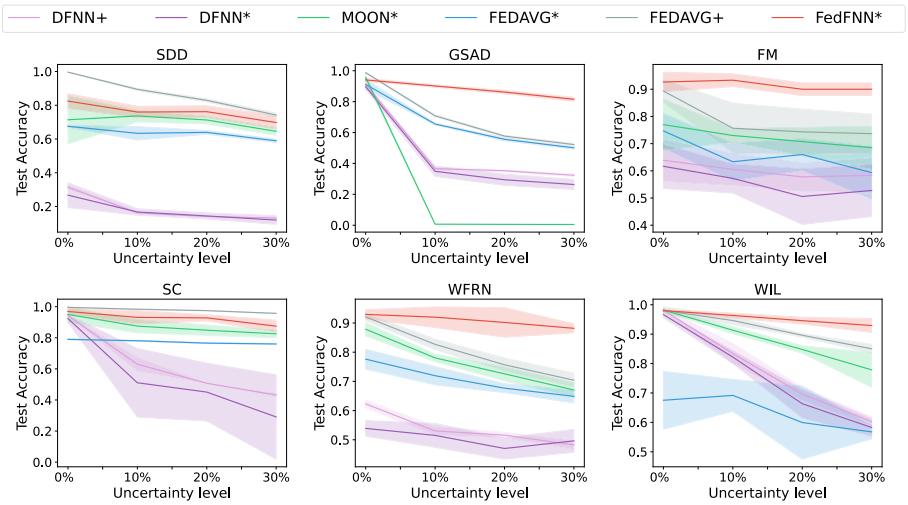


Figure 6.5: Performance of different algorithms when addressing datasets with different uncertainty levels.

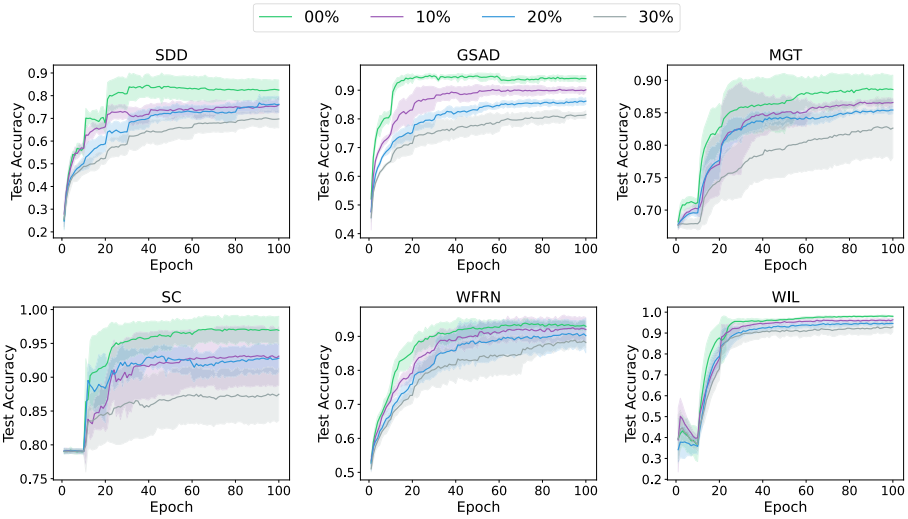


Figure 6.6: The convergence of the FedFNN optimization process under different uncertainty levels on 6 datasets.

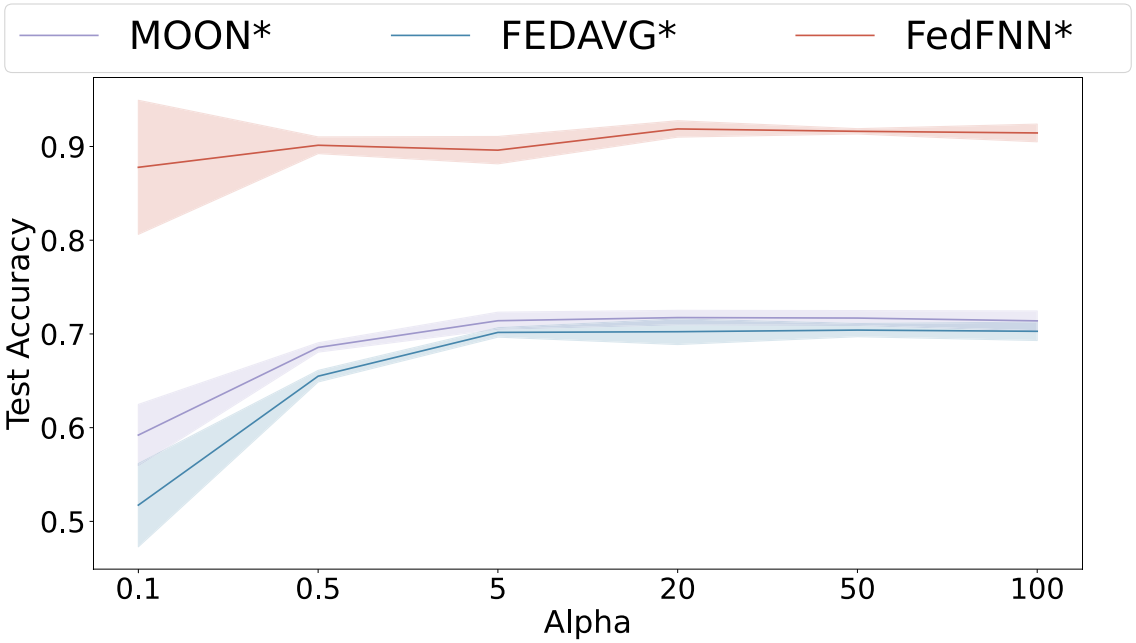


Figure 6.7: Performance of the FedFNN on GSAD at different non-IID levels.

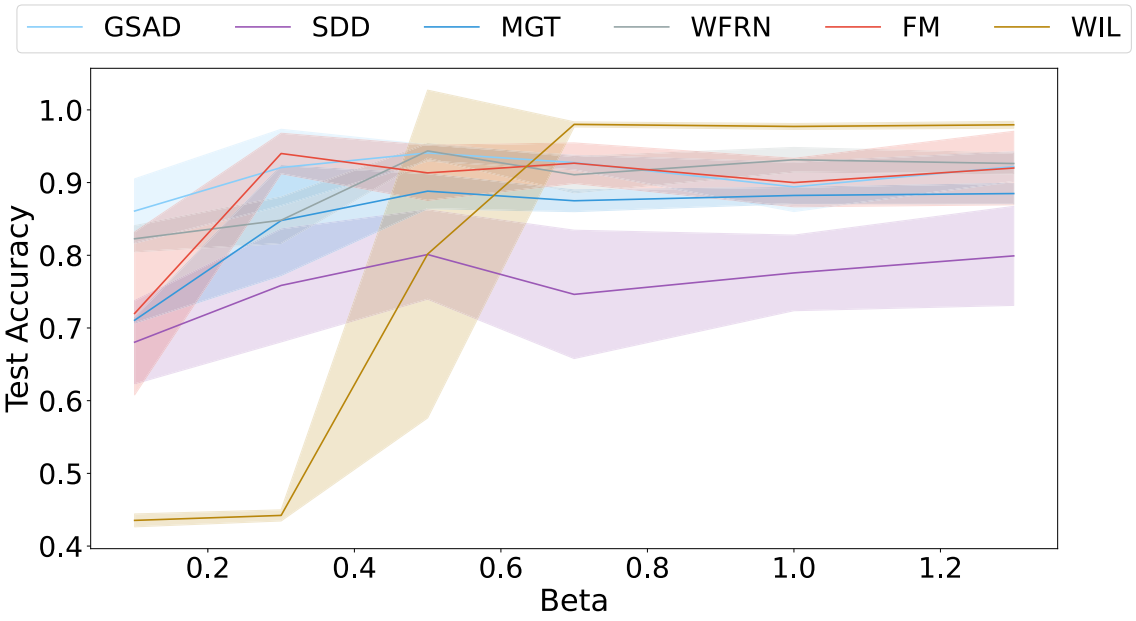


Figure 6.8: Performance of the FedFNN when using different β.

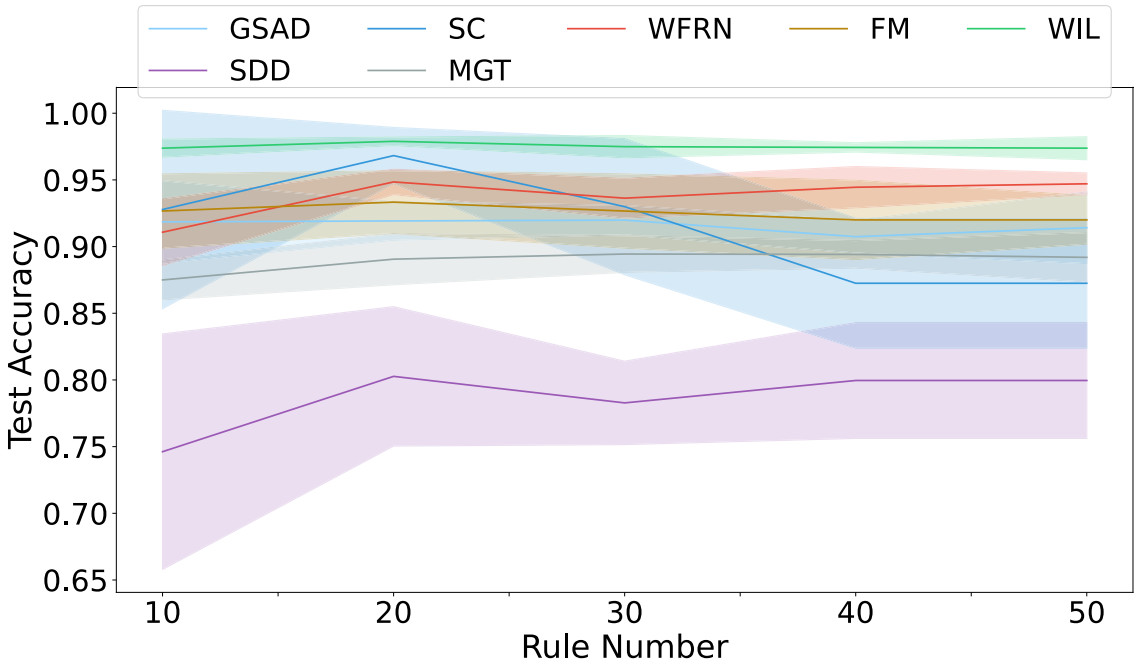


Figure 6.9: Performance of the FedFNN using different initial global rule numbers.

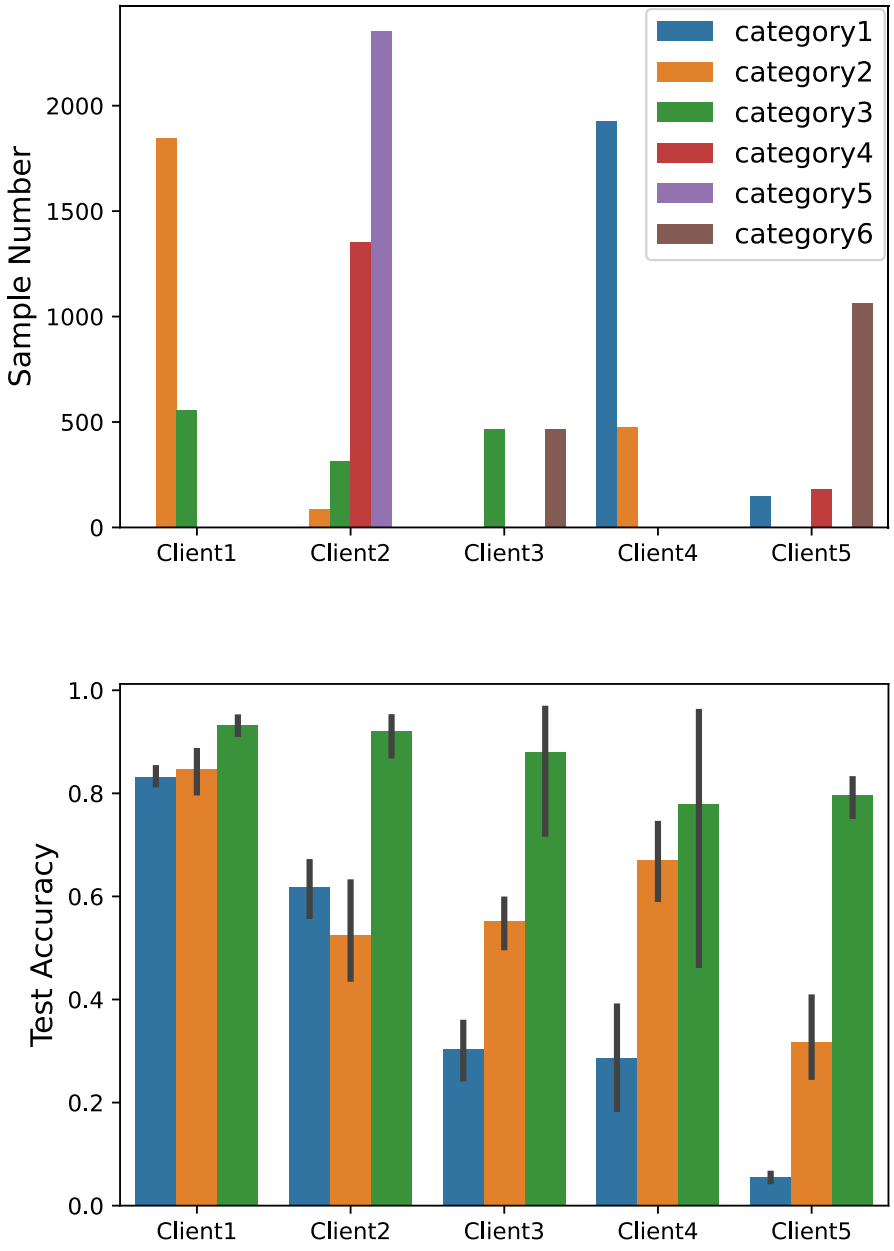


Figure 6.10: Performance of the FedFNN on all clients when training on the GSAD dataset.

Table 6.2: Average classification accuracies (%) and standard deviations achieved by each algorithm on the 7 datasets at a 10% level of uncertainty

Algorithm	Dataset						
	GSAD	SDD	SC	MGT	WFRN	FM	WIL
DFNN+	36.64/1.08	16.26/0.67	63.02/4.66	83.22/0.49	53.04/1.59	60.56/6.02	83.40/3.48
DFNN*	34.91/3.38	16.74/1.96	51.08/7.12	79.02/3.66	51.50/4.12	57.22/5.41	82.20/2.20
FedFNN*	**90.13/0.89** (↑ 21.57)	**75.98/3.49** (↑ 2.3)	**93.12/4.34** (↑ 5.6)	**86.58/0.56** (↑ 11.28)	**91.99/3.56** (↑ 13.94)	**93.33/2.36** (↑ 20.03)	**96.35/0.86** (↑ 5.01)
MOON*	68.56/0.50	73.68/3.58	87.52/4.38	75.30/3.45	78.05/1.30	73.00/2.59	91.34/1.19
FedAvg*	65.49/0.62	63.38/4.02	79.08/0.35	64.25/0.56	71.94/3.16	63.33/7.07	69.17/5.45
FedAvg+	70.80/0.72	89.43/0.71	98.35/0.20	82.14/0.69	82.69/1.82	75.67/9.25	94.63/1.41

Table 6.3: Number of parameters required by each state-of-the-art algorithm

Algorithm	Dataset						
	GSAD	SDD	SC	MGT	WFRN	FM	WIL
FedDNN	2,003,462	1,591,051	1,973,255	32,070	539,396	549,124	1,578,756
FedFNN	15,450	9,525	1,320	630	2,220	3,930	690

Table 6.4: Comparison of computing time (s) between FedFNN and DFNN

Algorithm	Dataset						
	GSAD	SDD	SC	MGT	WFRN	FM	WIL
FedFNN	1805	7259	7050	2348	691	111	260
DFNN	683	1642	1534	545	178	38	85

6.2.3 The Performance of the FedFNN on Non-IID Datasets

6.3 Experiments

We conduct extensive experiments with the aforementioned algorithms on all seven datasets with a non-IID level of 0.5 and an uncertainty level of 10% to verify the effectiveness of our FedFNN. The obtained results are summarized in Table 6.2, in which the green-coloured values are those for which our FedFNN is superior to MOON* on different datasets. According to the table, our model outperforms the existing state-of-the-art FL method MOON* in terms of test accuracy by an average of 11.43%, which proves the extraordinary heterogeneity handling ability of our FedFNN.

To verify the robustness of our FedFNN when dealing with different levels of non-IID data, we choose the concentration parameter α from the set of $\{0.1, 0.5, 5, 20, 50, 100\}$ and compare our FedFNN with FedAvg and MOON on the GSAD dataset. The results are shown in Fig. 6.7, in which the test accuracy curve of the FedFNN for

different non-IID levels is flatter and higher than those of the compared FL algorithms. Thus, we can conclude that our FedFNN achieves superior performance when dealing with a wide range of non-IID data. In addition, when the given data are uncertain, our FedFNN* that processes non-IID local data can even beat the FedAvg+ version that processes local IID data on most datasets, which further proves the robustness of our FedFNN.

It is worth noting that our model is far more lightweight than existing FL algorithms. As listed in Table 6.3, our model has an average of approximately 100 times fewer parameters than existing FL algorithms and achieves a better heterogeneity processing capability. Besides, we compared the computation overhead of our FedFNN and the DFNN on the Quadro P5000 GPU device with a running storage of 26384 MiB. Our results are listed in Table 6.4. Though the ERL consumes extra time compared with DFNN, our FedFNN increases more than 40% on average test accuracy DFNN.

In addition, the personalization abilities of the algorithms are discussed in this section. To explicitly demonstrate the high level of the heterogeneous federated scenario, we plot the number of samples in each category for each local client of the GSAD dataset and the performance achieved by each local model when $\alpha = 0.5$ in Fig. 6.10. According to Fig. 6.10, our local models outperform MOON* and FedAvg* on all clients and achieve much higher test accuracy on the 5-th client. MOON* and FedAvg* fail to learn the features of all categories because of the severe data heterogeneity.

Clearly, the global models of MOON and FedAvg tend to deal with the samples in the first five categories and overlook the information learned for the 6th category after several rounds of aggregation; this is because the samples in the 6th category are mostly allocated to client 5, whose learned information is easily disturbed by other clients. Instead, our personalized local FNN for the 5-th client achieves relatively high performance on all clients by automatically generating new rules to estimate the unique samples in the 6-th category. The newly generated rules are unique to this client and cannot be disturbed by other clients; thus, our proposed personalized local FNN exhibits superiority in dealing with heterogeneous local clients based on its flexible structure.

6.3.1 The Performance of the FedFNN on Datasets with Uncertainty

To investigate the effectiveness of the FedFNN when dealing with data uncertainty, we conduct experiments under multiple uncertainty level settings $\{0\%, 10\%, 20\%, 30\%\}$.

The performances of all algorithms on all datasets, except the FM dataset, under these uncertainty levels, are depicted in Fig. 6.5. Intuitively, our methods achieve state-of-the-art performance for all uncertainty levels and all datasets. In addition, we list the performance attained by all algorithms when processing datasets with 10% uncertainty in Table 6.2. From the table, our model outperforms the DFNN* in terms of test accuracy by an average of 36.40%. Thus, the extraordinary ability of our FedFNN to deal with data uncertainties can be confirmed.

6.3.2 Convergence Analysis of the FedFNN with ERL

To verify the convergence ability of our proposed FedFNN, the test accuracies it achieves throughout the optimization process are depicted in Fig. 6.6. As the results in this figure show, the FedFNN gradually converges with the increase in the number of communication rounds on all datasets. In addition, the functions and contributions of the rule cooperation stage and the rule evolution stage are clearly proven in Fig. 6.6. During the L-th round of the rule cooperation stage in each ERL iteration, the local clients cooperate with each other to learn more general global rules, and consequently, as shown in Fig. 6.6, the network performance stably improves on all datasets. However, the performance of the local FNNs gradually falls into a bottleneck along with the execution of the rule cooperation stage, as shown in Fig. 6.6, where the test accuracy hardly increases at each ERL iteration. The subsequent rule evolution stage solves this issue by updating the local model architecture. As shown in Fig. 6.6, the performance of the local clients dramatically increases because of the mutation of local rules. Eventually, these rapid performance improvements induced by the latter learning stage vanish after each local client learns all their required activated local rules.

6.3.3 Analysis of Key Parameter Robustness

We conducted extensive experiments to verify the robustness of our key parameters, including α, β and the initialized global rule number K. Their corresponding results are depicted in Fig. 6.7, Fig. 6.8 and Fig. 6.9, respectively. As shown in Fig. 6.7 and Fig. 6.9, our FedFNN can still achieve good performances when setting a wide range of parameters. In addition, we can achieve high performance when setting $\beta >= 0.7$ from the results drawn in Fig. 6.8. Intuitively, our FedFNN is proved to be robust to different parameter settings.

6.4 Summary

This chapter proposes a FedFNN with ERL to handle non-IID issues and data uncertainties in distributed scenarios. The proposed FedFNN integrates fuzzy if-then rules into an FL framework. By considering these fuzzy rules as the basic optimization units, our FedFNN is able to learn a group of general global rules and selectively activate an effective subset of these rules for each local client. This flexible composition approach for fuzzy rules increases the personalization of the local models with respect to handling non-IID data. Inspired by the theory of biological evolution, the proposed ERL method not only encourages the cooperation of local clients at the rule level to improve the generalization of the global rules but also updates the rule activation statuses for all clients to make their local models more personalized. Unlike most existing FL methods that implement aggregation among all local models, the FedFNN with ERL only aggregates the activated rules for their corresponding local clients. This enables the server to selectively aggregate only beneficial local updates, preventing the disturbances brought by harmful updates. Consequently, the FedFNN with ERL provides an effective learning framework for dealing with non-IID issues as well as data uncertainties. Comprehensive experiments verify the superiority and effectiveness of the proposed FedFNN over state-of-the-art methods.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

This thesis presented three robust fuzzy models designed for distributed computing scenarios to solve several key issues with high dimensionality, data heterogeneity, and unlabelled data optimisation that current distributed fuzzy models face. First, we proposed a new network architecture, RFNN, which comprises an AIE that performs non-linear mapping to overcome any uncertainty in membership function values. As such, it generates highly representative firing strengths. Notably, RFNN is an architecture where the consequent components are constructed from neural networks. This enhances the reasoning ability of the learned rules. As a result, our RFNN is not only able to handle high levels of data uncertainty, but it can also directly process data with very high dimensionality.

Adopting the ADMM strategy, the DSFR model involves a DFCM for structure learning and a DICR for parameter learning. Notably, both algorithms can extract not only useful information from labelled samples but also unlabelled ones. The DICR algorithm increases model performance and robustness by restricting the decision boundaries to low-density regions. Finally, a FedFNN with ERL handles non-IID issues and data uncertainty in distributed scenarios. The proposed FedFNN integrates fuzzy if-then rules into a federated learning framework. Inspired by the theory of biological evolution, the proposed ERL method not only encourages the cooperation of local clients at the rule level to help generalise the global rules, it also updates the rule activation statuses for all clients to make the local models more personalised. These three models effectively address the existing significant concerns with distributed fuzzy models.

7.2 Future Work

In the future, we will have a comprehensive plan to improve and expand our research in multiple directions. Firstly, we aim to enhance the RFNN's ability to detect adversarial noise, which can pose significant challenges for deep neural networks (DNNs). Simultaneously, we will focus on enhancing the interpretability of fuzzy models, making them more explainable and trustworthy in various learning tasks.

Additionally, our goal is to extend the applicability of the FedFNN and explore its potential in different domains. One such area of exploration is the application of indoor Wi-Fi localization. By leveraging the capabilities of our models, we aim to develop more accurate and robust methods for estimating device locations within indoor environments using Wi-Fi signals. This has practical implications for indoor navigation, asset tracking, and location-based services, offering improved user experiences and enabling new applications in smart buildings and Internet of Things (IoT) systems.

In terms of Evolutionary Reinforcement Learning (ERL), our focus will revolve around increasing the robustness and communication efficiency of the FedFNN in scenarios involving a large number of agents. We will also investigate privacy preservation techniques, exploring encryption methods to enhance the safety and privacy of communication information. Additionally, we will explore new schemes to further boost communication efficiency and quality.

Moreover, our research will encompass the proposal of new types of fuzzy rules specifically designed for federated fuzzy neural networks to handle uncertainty in electroencephalography (EEG) signals. As EEG signals inherently exhibit non-IID features, the FedFNN model holds great potential for effectively analyzing EEGs and solving tasks related to EEG signal processing.

By pursuing these research directions, we aim to advance the capabilities and performance of our models, expand their applications to different domains, such as indoor Wi-Fi localization, and contribute to the development of more robust, interpretable, and efficient machine learning techniques.

Bibliography

[1] J. Wang, C. Xu, J. Zhang, and R. Zhong, "Big data analytics for intelligent manufacturing systems: A review," *Journal of Manufacturing Systems*, vol. 62, pp. 738–752, 2022.

[2] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[3] H.-Y. Tran and J. Hu, "Privacy-preserving big data analytics a comprehensive survey," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 207–218, 2019.

[4] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2017, pp. 19–38.

[5] H. Tuttle, "Facebook scandal raises data privacy concerns," *Risk Management*, vol. 65, no. 5, pp. 6–9, 2018.

[6] S. Prabhakar and R. Cheng, "Data uncertainty management in sensor networks," in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 647–651, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_115. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_115.

[7] R. Fierimonte, M. Barbato, A. Rosato, and M. Panella, "Distributed learning of random weights fuzzy neural networks," in *2016 IEEE International Conference on Fuzzy Systems*, IEEE, 2016, pp. 2287–2294.

[8] R. Fierimonte, R. Altilio, and M. Panella, "Distributed on-line learning for random-weight fuzzy neural networks," in *2017 IEEE International Conference on Fuzzy Systems*, IEEE, 2017, pp. 1–6.

[9] Y. Shi, C.-T. Lin, Y.-C. Chang, W. Ding, Y. Shi, and X. Yao, "Consensus learning for distributed fuzzy neural network in big data environment," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2020.

[10] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 439–450.

[11] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in neural information processing systems*, 2009, pp. 289–296.

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

[13] N. Papernot, P. McDaniel, A. Sinha, and M. P. Wellman, "Sok: Security and privacy in machine learning," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2018, pp. 399–414.

[14] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," *ACM Sigkdd Explorations Newsletter*, vol. 4, no. 2, pp. 28–34, 2002.

[15] P. A. Forero, A. Cano, and G. B. Giannakis, "Consensus-based distributed support vector machines," *Journal of Machine Learning Research*, vol. 11, no. May, pp. 1663–1707, 2010.

[16] J. Qin, W. Fu, H. Gao, and W. X. Zheng, "Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory," *IEEE transactions on cybernetics*, vol. 47, no. 3, pp. 772–783, 2016.

[17] X. Bi, X. Zhao, G. Wang, P. Zhang, and C. Wang, "Distributed extreme learning machine with kernels based on mapreduce," *Neurocomputing*, vol. 149, pp. 456–463, 2015.

[18] Y. Ye, M. Xiao, and M. Skoglund, "Decentralized multi-task learning based on extreme learning machines," *arXiv preprint arXiv:1904.11366*, pp. 1–11, 2019.

[19] S. Scardapane, D. Wang, and M. Panella, "A decentralized training algorithm for echo state networks in distributed big data applications," *Neural Networks*, vol. 78, pp. 65–74, 2016.

[20] L. Georgopoulos and M. Hasler, "Distributed machine learning in networks by consensus," *Neurocomputing*, vol. 124, pp. 2–12, 2014.

[21] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[22] K. Bonawitz, H. Eichner, W. Grieskamp, *et al.*, "Towards federated learning at scale: System design," *Proceedings of Machine Learning and Systems*, vol. 1, pp. 374–388, 2019.

[23] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[24] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 5132–5143.

[25] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, 2020.

[26] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *International Conference on Machine Learning*, PMLR, 2021, pp. 9489–9502.

[27] I. Achituve, A. Shamsian, A. Navon, G. Chechik, and E. Fetaya, "Personalized federated learning with gaussian processes," *Advances in Neural Information Processing Systems*, vol. 34, pp. 8392–8406, 2021.

[28] F. Linsner, L. Adilova, S. Däubener, M. Kamp, and A. Fischer, "Approaches to uncertainty quantification in federated deep learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2021, pp. 128–145.

[29] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," in *International Conference on Learning Representations*, 2020, pp. 170–178. [Online]. Available: `https://openreview.net/forum?id=BkluqlSFDS`.

[30] D. J. Stracuzzi, M. G. Chen, M. C. Darling, *et al.*, *Uncertainty in data analytics*. `http://engineering.purdue.edu/~mark/puthesis`, United States, 2016.

[31] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.

[32] A. Camuto, M. Willetts, U. Simsekli, S. J. Roberts, and C. C. Holmes, "Explicit regularisation in gaussian noise injections," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 16 603–16 614. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/c16a5320fa475530d9583c34fd356ef5-Paper.pdf.

[33] H. Noh, T. You, J. Mun, and B. Han, "Regularizing deep neural networks by noise: Its interpretation and optimization," *arXiv preprint arXiv:1710.05179*, 2017.

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

[35] J. M. Bernardo and A. F. Smith, *Bayesian theory*. John Wiley & Sons, 2009, vol. 405.

[36] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Artificial intelligence and statistics*, PMLR, 2013, pp. 207–215.

[37] T. Charnock, L. Perreault-Levasseur, and F. Lanusse, "Bayesian Neural Networks," *arXiv*, pp. 1–16, 2020, ISSN: 23318422. DOI: 10.1590/s0104-65001997000200006.

[38] N. Li, W. Li, J. Sun, Y. Gao, Y. Jiang, and S.-T. Xia, "Stochastic Deep Gaussian Processes over Graphs," *NeurIPS 2020 - Advances in Neural Information Processing Systems 33 pre-proceedings*, no. NeurIPS, pp. 1–12, 2020.

[39] M. Pinsky and S. Karlin, *An introduction to stochastic modeling*. Academic press, 2010.

[40] G. Carbone, M. Wicker, L. Laurenti, A. Patane, L. Bortolussi, and G. Sanguinetti, "Robustness of bayesian neural networks to gradient-based attacks," *arXiv*, 2020, ISSN: 23318422. arXiv: 2002.04359.

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

[42] L. Zhao, T. Liu, X. Peng, and D. Metaxas, "Maximum-entropy adversarial data augmentation for improved generalization and robustness," *arXiv preprint arXiv:2010.08001*, 2020.

[43] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *arXiv preprint arXiv:1903.12261*, 2019.

[44] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[45] T. Kim and C. H. Park, "Anomaly pattern detection for streaming data," *Expert Systems with Applications*, vol. 149, p. 113 252, 2020.

[46] J.-S. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[47] C. El Hatri and J. Boumhidi, "Fuzzy deep learning based urban traffic incident detection," *Cognitive Systems Research*, vol. 50, pp. 206–213, 2018.

[48] Z. Deng, Y. Jiang, F.-L. Chung, H. Ishibuchi, K.-S. Choi, and S. Wang, "Transfer prototype-based fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 5, pp. 1210–1232, 2015.

[49] I. Couso, C. Borgelt, E. Hullermeier, and R. Kruse, "Fuzzy sets in data analysis: From statistical foundations to machine learning," *IEEE Computational Intelligence Magazine*, vol. 14, no. 1, pp. 31–44, 2019.

[50] G. Fu and Z. Kapelan, "Fuzzy probabilistic design of water distribution networks," *Water Resources Research*, vol. 47, no. 5, pp. 1–12, 2011.

[51] D. Chen, X. Zhang, L. L. Wang, and Z. Han, "Prediction of cloud resources demand based on hierarchical pythagorean fuzzy deep neural network," *IEEE Transactions on Services Computing*, 2019.

[52] K. V. Shihabudheen and G. N. Pillai, "Recent advances in neuro-fuzzy system: A survey," *Knowledge-Based Systems*, vol. 152, pp. 136–162, Jul. 2018, ISSN: 09507051. DOI: 10.1016/j.knosys.2018.04.014.

[53] Y. Shi, C.-T. Lin, Y.-C. Chang, W. Ding, Y. Shi, and X. Yao, "Consensus learning for distributed fuzzy neural network in big data environment," *IEEE Transactions on Emerging Topics in Computational Intelligence*, to appear, 2020.

[54] H. Liang, J. Zou, K. Zuo, and M. J. Khan, "An improved genetic algorithm optimization fuzzy controller applied to the wellhead back pressure control system," *Mechanical Systems and Signal Processing*, vol. 142, p. 106 708, 2020.

[55] Y. Pan, P. Du, H. Xue, and H.-K. Lam, "Singularity-free fixed-time fuzzy control for robotic systems with user-defined performance," *IEEE Transactions on Fuzzy Systems*, 2020.

[56] L. Zhang, Y. Shi, Y.-C. Chang, and C.-T. Lin, "Hierarchical fuzzy neural networks with privacy preservation for heterogeneous big data," *IEEE Transactions on Fuzzy Systems*, vol. 29, no. 1, pp. 46–58, 2020.

[57] Y. Shi, L. Zhang, Z. Cao, M. Tanveer, and C.-T. Lin, "Distributed semi-supervised fuzzy regression with interpolation consistency regularization," *IEEE Transactions on Fuzzy Systems*, 2021.

[58] S. P. Boyd, "Convex optimization: From embedded real-time to large-scale distributed.," in *KDD*, 2011, p. 1.

[59] L. Wang, "Heterogeneous data and big data analytics," *Automatic Control and Information Sciences*, vol. 3, no. 1, pp. 8–15, 2017.

[60] G. Raju, J. Zhou, and R. A. Kisner, "Hierarchical fuzzy control," *International journal of control*, vol. 54, no. 5, pp. 1201–1216, 1991.

[61] X.-J. Zeng and M. G. Singh, "Approximation theory of fuzzy systems-mimo case," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 2, pp. 219–235, 1995.

[62] L.-X. Wang, "Analysis and design of hierarchical fuzzy systems," *IEEE Transactions on Fuzzy systems*, vol. 7, no. 5, pp. 617–624, 1999.

[63] R. J. Campello and W. C. do Amaral, "Hierarchical fuzzy relational models: Linguistic interpretation and universal approximation," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 3, pp. 446–453, 2006.

[64] Y. Chen, B. Yang, A. Abraham, and L. Peng, "Automatic design of hierarchical takagi–sugeno type fuzzy systems using evolutionary algorithms," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 3, pp. 385–397, 2007.

[65] L.-X. Wang, "Fast training algorithms for deep convolutional fuzzy systems with application to stock index prediction," *IEEE Transactions on Fuzzy Systems*, 2019.

[66] X. Fu, X.-J. Zeng, D. Wang, D. Xu, and L. Yang, "Fuzzy system approaches to negotiation pricing decision support," *Journal of Intelligent & Fuzzy Systems*, vol. 29, no. 2, pp. 685–699, 2015.

[67] P. Brox, I. Baturone, and S. Sánchez-Solano, "Tuning of a hierarchical fuzzy system for video de-interlacing," in *International Conference on Fuzzy Systems*, IEEE, 2010, pp. 1–6.

[68] H. He and J. Lawry, "The linguistic attribute hierarchy and its optimisation for classification," *Soft computing*, vol. 18, no. 10, pp. 1967–1984, 2014.

[69] M. A. ŞAHIN and K. Leblebicioğlu, "A hierarchical fuzzy decision maker for the weapon target assignment," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 8993–8998, 2011.

[70] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[71] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd annual meeting of the association for computational linguistics*, 1995, pp. 189–196.

[72] E. Riloff, J. Wiebe, and T. Wilson, "Learning subjective nouns using extraction pattern bootstrapping," in *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, 2003, pp. 25–32.

[73] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models.," *WACV/MOTION*, vol. 2, 2005.

[74] A. Blum and T. Mitchell, "Combining labeled and unlabeled data with co-training," in *Proceedings of the eleventh annual conference on Computational learning theory*, 1998, pp. 92–100.

[75] A. Anis, A. El Gamal, A. S. Avestimehr, and A. Ortega, "A sampling theory perspective of graph-based semi-supervised learning," *IEEE Transactions on Information Theory*, vol. 65, no. 4, pp. 2322–2342, 2018.

[76] Y. Xu, H. Chen, J. Luo, Q. Zhang, S. Jiao, and X. Zhang, "Enhanced moth-flame optimizer with mutation strategy for global optimization," *Information Sciences*, vol. 492, pp. 181–203, 2019.

[77] J.-W. Jin and C. P. Chen, "Regularized robust broad learning system for uncertain data modeling," *Neurocomputing*, vol. 322, pp. 58–69, 2018.

[78] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[79] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," *arXiv preprint arXiv:1903.03825*, 2019.

[80] J. B. Predd, S. B. Kulkarni, and H. V. Poor, "Distributed learning in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 56–69, 2006.

[81] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ecg classification with branching programs and neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp. 452–468, 2011.

[82] M. A. Pathak and B. Raj, "Privacy-preserving speaker verification and identification using gaussian mixture models," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 2, pp. 397–406, 2012.

[83] M. Pathak, S. Rane, W. Sun, and B. Raj, "Privacy preserving probabilistic inference with hidden markov models," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 5868–5871.

[84] F. Yan, S. Sundaram, S. Vishwanathan, and Y. Qi, "Distributed autonomous online learning: Regrets and intrinsic privacy-preserving properties," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2483–2493, 2012.

[85] X. Chang, S.-B. Lin, and D.-X. Zhou, "Distributed semi-supervised learning with kernel ridge regression," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1493–1514, 2017.

[86] J. Xie, S. Liu, and H. Dai, "A distributed semi-supervised learning algorithm based on manifold regularization using wavelet neural network," *Neural Networks*, vol. 118, pp. 300–309, 2019.

[87] J. Xie, S. Liu, and H. Dai, "Distributed semi-supervised learning algorithm based on extreme learning machine over networks using event-triggered communication scheme," *Neural Networks*, vol. 119, pp. 261–272, 2019.

[88] Y. Yan, L. Chen, and W.-C. Tjhi, "Fuzzy semi-supervised co-clustering for text documents," *Fuzzy Sets and Systems*, vol. 215, pp. 74–89, 2013.

[89] S. Poria, A. Gelbukh, D. Das, and S. Bandyopadhyay, "Fuzzy clustering for semi-supervised learning–case study: Construction of an emotion lexicon," in *Mexican International Conference on Artificial Intelligence*, Springer, 2012, pp. 73–86.

[90] S. Zhou, Q. Chen, and X. Wang, "Fuzzy deep belief networks for semi-supervised sentiment classification," *Neurocomputing*, vol. 131, pp. 312–322, 2014.

[91] T. Zhang, Z. Deng, H. Ishibuchi, and L. M. Pang, "Robust tsk fuzzy system based on semi-supervised learning for label noise data," *IEEE Transactions on Fuzzy Systems*, 2020.

[92] Y. Jiang, D. Wu, Z. Deng, *et al.*, "Seizure classification from eeg signals using transfer learning, semi-supervised learning and tsk fuzzy system," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 12, pp. 2270–2284, 2017.

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

[94] K. Chaudhuri, A. D. Sarwate, and K. Sinha, "A near-optimal algorithm for differentially-private principal components," *The Journal of Machine Learning Research*, vol. 14, no. 1, pp. 2905–2943, 2013.

[95] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft, "Learning in a large function space: Privacy-preserving mechanisms for svm learning," *arXiv preprint arXiv:0911.5708*, 2009.

[96] J. C. Duchi, M. I. Jordan, and M. J. Wainwright, "Privacy aware learning," *Journal of the ACM (JACM)*, vol. 61, no. 6, pp. 1–57, 2014.

[97] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, 2015, pp. 1310–1321.

[98] R. K. Langari, S. Sardar, S. A. A. Mousavi, and R. Radfar, "Combined fuzzy clustering and firefly algorithm for privacy preserving in social networks," *Expert Systems with Applications*, vol. 141, p. 112 968, 2020.

[99] L. Sweeney, "K-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, 2002.

[100] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam, "L-diversity: Privacy beyond k-anonymity," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 1, no. 1, 3–es, 2007.

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

[102] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1187–1198.

[103] J. Zhang and G. Huang, "Research on distributed heterogeneous data pca algorithm based on cloud platform," in *AIP Conference Proceedings*, AIP Publishing LLC, vol. 1967, 2018, p. 020 016.

[104] D. P. Lewis, T. Jebara, and W. S. Noble, "Support vector machine learning from heterogeneous data: An empirical analysis using protein sequence and structure," *Bioinformatics*, vol. 22, no. 22, pp. 2753–2760, 2006.

[105] Q. Chen, X. Song, H. Yamada, and R. Shibasaki, "Learning deep representation from big and heterogeneous data for traffic accident inference," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[106] H. Zuo, J. Lu, G. Zhang, and W. Pedrycz, "Fuzzy rule-based domain adaptation in homogeneous and heterogeneous spaces," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 2, pp. 348–361, 2018.

[107] S. Liu, Q. Qu, L. Chen, and L. M. Ni, "Smc: A practical schema for privacy-preserved data sharing over distributed data streams," *IEEE Transactions on Big Data*, vol. 1, no. 2, pp. 68–81, 2015.

[108] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable ADMM approach," in *International conference on machine learning*, 2016, pp. 2722–2731.

[109] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 3466–3473.

[110] B. Dang, Y. Wang, J. Zhou, *et al.*, "Transfer collaborative fuzzy clustering in distributed peer-to-peer networks," *IEEE Transactions on Fuzzy Systems*, 2020.

[111] Y. Jiang, Z. Deng, F.-L. Chung, and S. Wang, "Realizing two-view tsk fuzzy classification system by using collaborative learning," *IEEE transactions on systems, man, and cybernetics: systems*, vol. 47, no. 1, pp. 145–160, 2016.

[112] T. Zhang, Z. Deng, D. Wu, and S. Wang, "Multiview fuzzy logic system with the cooperation between visible and hidden views," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 6, pp. 1162–1173, 2018.

[113] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell, "Text classification from labeled and unlabeled documents using em," *Machine learning*, vol. 39, no. 2-3, pp. 103–134, 2000.

[114] S. Baluja, "Using labeled and unlabeled data for probabilistic modeling of face orientation," *International journal of pattern recognition and artificial intelligence*, vol. 14, no. 08, pp. 1097–1107, 2000.

[115] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," *arXiv preprint arXiv:1810.12281*, 2018.

[116] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, big, simple neural nets for handwritten digit recognition," *Neural computation*, vol. 22, no. 12, pp. 3207–3220, 2010.

[117] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Advances in neural information processing systems*, 2005, pp. 529–536.

[118] D.-H. Lee, "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," in *Workshop on challenges in representation learning, ICML*, vol. 3, 2013.

[119] V. Verma, A. Lamb, J. Kannala, Y. Bengio, and D. Lopez-Paz, "Interpolation consistency training for semi-supervised learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3635–3641.

[120] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 5049–5059.

[121] J. Xie, S. Liu, H. Dai, and Y. Rong, "Distributed semi-supervised learning algorithms for random vector functional-link networks with distributed data splitting across samples and features," *Knowledge-Based Systems*, p. 105 577, 2020.

[122] L. A. Zadeh, "Fuzzy sets," in *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, World Scientific, 1996, pp. 394–432.

[123] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995, vol. 4.

[124] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," in *Proceedings of the institution of electrical engineers*, IET, vol. 121, 1974, pp. 1585–1588.

[125] M. Sugeno, "On stability of fuzzy systems expressed by fuzzy rules with singleton consequents," *IEEE Transactions on Fuzzy systems*, vol. 7, no. 2, pp. 201–224, 1999.

[126] J. M. Mendel and R. B. John, "Type-2 fuzzy sets made simple," *IEEE Transactions on fuzzy systems*, vol. 10, no. 2, pp. 117–127, 2002.

[127] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[128] C.-T. Lin and C. S. G. Lee, "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on Computers*, no. 12, pp. 1320–1336, 1991.

[129] H. Huang, C. Yang, and C. P. Chen, "Optimal robot–environment interaction under broad fuzzy neural adaptive control," *IEEE Transactions on Cybernetics*, vol. 51, no. 7, pp. 3824–3835, 2020.

[130] A. Salimi-Badr and M. M. Ebadzadeh, "A novel self-organizing fuzzy neural network to learn and mimic habitual sequential tasks," *IEEE transactions on cybernetics*, 2020.

[131] S. J. Nowlan and G. E. Hinton, "Simplifying neural networks by soft weight-sharing," *Neural computation*, vol. 4, no. 4, pp. 473–493, 1992.

[132] B. Mele and G. Altarelli, "Lepton spectra as a measure of b quark polarization at LEP," *Physics Letters B*, vol. 299, no. 3-4, pp. 345–350, 1993, ISSN: 03702693. DOI: 10.1016/0370-2693(93)90272-J.

[133] C. Wei, S. Kakade, and T. Ma, "The implicit and explicit regularization effects of dropout," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 13–18 Jul 2020, pp. 10 181–10 192. [Online]. Available: http://proceedings.mlr.press/v119/wei20d.html.

[134] C. M. Bishop, "Training with noise is equivalent to tikhonov regularization," *Neural computation*, vol. 7, no. 1, pp. 108–116, 1995.

[135] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural computation*, vol. 8, no. 3, pp. 643–674, 1996.

[136] B. Poole, J. Sohl-Dickstein, and S. Ganguli, "Analyzing noise in autoencoders and deep networks," *arXiv preprint arXiv:1406.1831*, 2014.

[137] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[138] Y. Lu, Z. Zhang, G. Lu, Y. Zhou, J. Li, and D. Zhang, "Addi-reg: A better generalization-optimization tradeoff regularization method for convolutional neural networks," *IEEE Transactions on Cybernetics*, 2021.

[139] Q. Yin, B. Xu, K. Zhou, and P. Guo, "Bayesian pseudoinverse learners: From uncertainty to deterministic learning," *IEEE Transactions on Cybernetics*, vol. 52, no. 11, pp. 12 205–12 216, 2022. DOI: 10.1109/TCYB.2021.3079906.

[140] A. Bekasov and I. Murray, "Bayesian adversarial spheres: Bayesian inference and adversarial examples in a noiseless setting," *arXiv preprint arXiv:1811.12335*, 2018.

[141] Y. Gal and L. Smith, "Sufficient conditions for idealised models to have no adversarial examples: A theoretical and empirical study with bayesian neural networks," *arXiv preprint arXiv:1806.00667*, 2018.

[142] Y. Li and Y. Gal, "Dropout inference in bayesian neural networks with alpha-divergences," in *International conference on machine learning*, PMLR, 2017, pp. 2052–2061.

[143] L. Cardelli, M. Kwiatkowska, L. Laurenti, N. Paoletti, A. Patane, and M. Wicker, "Statistical guarantees for the robustness of Bayesian neural networks," *arXiv*, pp. 5693–5700, 2019, ISSN: 23318422.

[144] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017, pp. 3–14.

[145] H. Liu, Y.-S. Ong, Z. Yu, J. Cai, and X. Shen, "Scalable gaussian process classification with additive noise for non-gaussian likelihoods," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 5842–5854, 2022. DOI: 10.1109/TCYB.2020.3043355.

[146] H. Liu, Y.-S. Ong, and J. Cai, "A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design," *Structural and Multidisciplinary Optimization*, vol. 57, no. 1, pp. 393–416, 2018.

[147] Z. Dai, A. Damianou, J. González, and N. Lawrence, "Variational auto-encoded deep gaussian processes," *arXiv preprint arXiv:1511.06455*, 2015.

[148] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*, PMLR, 2019, pp. 4615–4625.

[149] S. P. Singh and M. Jaggi, "Model fusion via optimal transport," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 045–22 055, 2020.

[150] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106 775, 2021.

[151] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 713–10 722.

[152] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[153] M. Luo, F. Chen, D. Hu, Y. Zhang, J. Liang, and J. Feng, "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5972–5984, 2021.

[154] A. Dieuleveut, G. Fort, E. Moulines, and G. Robin, "Federated-em with heterogeneity mitigation and variance reduction," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 553–29 566, 2021.

[155] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, IEEE, 2020, pp. 794–797.

[156] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[157] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of gradient-based model-agnostic meta-learning algorithms," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 1082–1092.

[158] Y. Huang, L. Chu, Z. Zhou, *et al.*, "Personalized cross-silo federated learning on non-iid data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 7865–7873.

[159] J. Tang, Y. Duan, Y. Zhou, and J. Jin, "Distributed slice selection-based computation offloading for intelligent vehicular networks," *IEEE Open Journal of Vehicular Technology*, vol. 2, pp. 261–271, 2021.

[160] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4427–4437.

[161] X. Ma, J. Zhang, S. Guo, and W. Xu, "Layer-wised model aggregation for personalized federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 092–10 101.

[162] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International Conference on Machine Learning*, PMLR, 2021, pp. 2089–2099.

[163] K. Singhal, H. Sidahmed, Z. Garrett, S. Wu, J. Rush, and S. Prakash, "Federated reconstruction: Partially local federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 11 220–11 232, 2021.

[164] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE transactions on Fuzzy Systems*, vol. 6, no. 1, pp. 12–32, 1998.

[165] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 513–529, 2011.

[166] S. L. Chiu, "Fuzzy model identification based on cluster estimation," *Journal of Intelligent & fuzzy systems*, vol. 2, no. 3, pp. 267–278, 1994.

[167] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, Oakland, CA, USA, vol. 1, 1967, pp. 281–297.

[168] P. A. Forero, A. Cano, and G. B. Giannakis, "Distributed clustering using wireless sensor networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, 2011.

[169] Y. Wang, W. Yin, and J. Zeng, "Global convergence of ADMM in nonconvex nonsmooth optimization," *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[170] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 1199–1208.

[171] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[172] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[173] A. Vergara, S. Vembu, T. Ayhan, M. A. Ryan, M. L. Homer, and R. Huerta, "Chemical gas sensor drift compensation using classifier ensembles," *Sensors and Actuators B: Chemical*, vol. 166, pp. 320–329, 2012.

[174] F. Paschke, C. Bayer, M. Bator, *et al.*, "Sensorlose zustandsüberwachung an synchronmotoren," in *ProcEEDings 23. Workshop comPutational intElligEncE*, 2013, p. 211.

[175] K. S. Gyamfi, J. Brusey, A. Hunt, and E. Gaura, "Linear dimensionality reduction for classification via a sequential bayes error minimisation with an application to flow meter diagnostics," *Expert Systems with Applications*, vol. 91, pp. 252–262, 2018.

[176] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision support systems*, vol. 47, no. 4, pp. 547–553, 2009.

[177] J. Dvořák and P. Savickỳ, "Softening splits in decision trees using simulated annealing," in *International Conference on Adaptive and Natural Computing Algorithms*, Springer, 2007, pp. 721–729.

[178] I. Cohen, F. Cozman, N. Sebe, M. Cirelo, and T. Huang, "Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 12, pp. 1553–1566, 2004. DOI: 10.1109/TPAMI.2004.127.

[179] J. G. Rohra, B. Perumal, S. J. Narayanan, P. Thakur, and R. B. Bhatt, "User localization in an indoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks," in *Proceedings of Sixth International Conference on Soft Computing for Problem Solving*, Springer, 2017, pp. 286–295.

[180] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela, "Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study," in *2009 6th Latin American Robotics Symposium (LARS 2009)*, IEEE, 2009, pp. 1–6.

[181] M. D. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo.," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1593–1623, 2014.

[182] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," *arXiv preprint arXiv:1701.02434*, 2017.

[183] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *arXiv preprint arXiv:2008.05756*, 2020.

[184] J. de Jesús Rubio, "SOFMLS: Online self-organizing fuzzy modified least-squares network," *IEEE Transactions on Fuzzy Systems*, vol. 17, no. 6, pp. 1296–1309, 2009.

[185] R. Lopez, E. Balsa-Canto, and E. Oñate, "Neural networks for variational problems in engineering," *International Journal for Numerical Methods in Engineering*, vol. 75, no. 11, pp. 1341–1360, 2008.

[186] J. J. Thompson, M. R. Blair, L. Chen, and A. J. Henrey, "Video game telemetry as a critical tool in the study of complex skill learning," *PloS one*, vol. 8, no. 9, e75129, 2013.

[187] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014.

[188] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997.

[189] Y. Hu, Y. Zhao, Y. Cai, Z. Fu, and S. Ding, "Comparison of statistical learning and predictive models on breast cancer data and king county housing data," 2017.

[190] M. J. Wade and S. Kalisz, "The causes of natural selection," *Evolution*, vol. 44, no. 8, pp. 1947–1955, 1990.

[191] S. Mirjalili, "Genetic algorithm," in *Evolutionary algorithms and neural networks*, Springer, 2019, pp. 43–55.

[192] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, "Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [book review]," *IEEE Transactions on automatic control*, vol. 42, no. 10, pp. 1482–1484, 1997.