Deep Neuro-Fuzzy Systems Based on Complex Fuzzy Theory & Complex-Valued Structures with Applications to Regression Modeling

---

Chuan Xue

18th October 2023
Final Version

Deep Neuro-Fuzzy Systems Based on Complex Fuzzy Theory & Complex-Valued Structures with Applications to Regression Modeling

BY

CHUAN XUE

A dissertation submitted to

The University of Sheffield

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

18th October 2023

Chuan Xue

Deep Neuro-Fuzzy Systems Based on Complex Fuzzy Theory & Complex-Valued Structures with Applications to Regression Modeling

PhD Dissertation, 18th October 2023

Supervisors: Prof Mahdi Mahfouf
Dr Hua-Liang Wei

The University of Sheffield

Department of Automatic Control and Systems Engineering

Amy Johnson Building

Portobello Street

Sheffield, S1 3JD

The bounds of human possibility are not as confining as we think they are; they are made to seem to be tight by our weaknesses, our vices, our prejudices that confine them.

Jean-Jacques Rousseau

Acknowledgement

I am humbled to express my most sincere gratitude to Professor Mahdi Mahfouf, who accepted me as his Ph.D. student during a hard time in the middle of the COVID-19 pandemic, for the invaluable position he offered, as well as his advice, guidance, patience, and efforts, which helped me overcome all those difficulties to make it thus far.

Also, I wish to thank my parents for their solid financial support that allowed me to finish the degree single-mindedly without financial worries, a rare luxury in both the pandemic and post-pandemic era and a privilege that most of my age could barely imagine.

Declaration

I, Chuan Xue, declare that the work presented in this thesis is my own. All material in this thesis which is not of my own work, has been properly accredited and referenced.

Sheffield, 18th October 2023

Chuan Xue

Abstract

Deep neuro-fuzzy systems are a category of machine learning structures that integrate the self-learning potential of neural networks as well as the transparency of fuzzy systems, but not without flaws. Firstly, the size of the rule-base of some fuzzy systems applying complete rule partitions often increases exponentially as the input dimension grows. Secondly, the classic fuzzy logic commonly applied by conventional neuro-fuzzy models can only express relatively uncomplicated semantics, which leads to limited generalization capability and risk of overfitting of the model. Thirdly, such models are sensitive to outliers and noises in the data. This thesis mainly focuses on constructing novel deep neuro-fuzzy models based on complex-valued structure and the complex fuzzy theory to enhance the algorithm performance against the above three aspects.

Given the first deficiency above, the complex-valued structure is introduced into the classic Wang-Mendel (WM) algorithm to form a new complex-valued Wang-Mendel (CVWM) method. Subsequently, a deep complex-valued single-iteration fuzzy system (DCVSF) is created for high-dimensional datasets by further integrating CVWM into hierarchical structures. DCVSF only requires one iteration to train, and its rule-base is merely the square root scale of a counterpart architecture using the original WM method. The second proposed model is a rapid adaptive complex neuro-fuzzy inference system (RACFIS) that adopts complex fuzzy theory as reasoning logic. Complex fuzzy sets (CFSs) can hold more information and represent more complex semantics than type-1 fuzzy sets, leading to better performance for regression tasks. A closed-form solution also exists for the first-order derivatives of complex fuzzy membership functions, which enables gradient optimization policy, making RACFIS models more efficient than type-2 models without such solutions. The third model is a robust learning method called an exclusionary neural complex fuzzy inference system (ENCFIS). Robust machine learning is an emerging topic for advanced artificial intelligence. As the first robust learning attempt in the territory of fuzzy systems, ENCFIS is highly adaptable to various real-world datasets and realizes strong noise immunity at a very low cost of accuracy. The above models are tested on synthetic and real-world datasets, including a Sunspot time series dataset and two metallurgical datasets. The experimental results indicate that all models reached the expected performance.

Contents

1

Intro

> *In all things of nature there is something of the marvelous.*
>
> **– Aristotle –**

1.1 Introduction and Background

The seminal idea of fuzzy sets & logic came from Professor Zadeh [1]–[3]. Such profound theoretical innovation did not immediately raise large-scale attention because of its unorthodox interpretation regarding crucial principles such as certainty compared with the prevailing crisp or probability modeling methods at the time. With the advancing of computational intelligence, the capacity of this idea to perform imprecise reasoning as well as to provide a modeling method that is analogous to human intuition gradually attracted awareness [4]. For a long time, expert systems, fuzzy systems, and artificial neural networks have been considered the three main topics of intelligent systems [5]. In recent years, as the traditional methodology of manual modeling in fuzzy systems has long been unable to tackle the increasingly sophisticated application scenarios in this new wave of machine learning fever, attention to conventional fuzzy methods has shown a significant downward trend. Especially after Google DeepMind successfully developed the AlphaGo [6], which impressed the world by defeating all human players in the game of Go that once had been considered a field where human intuition could easily outsmart the machines.

For a moment, deep learning almost becomes synonymous with AI by showing momentum to surpass human beings in many specific areas. Despite this, it is far from perfect for the challenges posed by the complexity of the real world. A common approach in the deep learning area is simply piling up network scale and data scale to squeeze more performance, but this strategy does not lead to endless improvements. Excessively increasing the depth of the network may lead to degeneration problems, which even degrades the network performance [7]. In addition, most deep networks are black-box models with little transparency, which hinders applications that demand an understanding of the model internals. Consequently, some new trains of thought are gaining attention, and one of those is to combine deep networks with fuzzy systems. Even though the primitive neuro-fuzzy system [8] emerged as early as the late 1980s, the topic of deep neuro-fuzzy systems has only become popular in recent years [9].

Generalized neuro-fuzzy systems can be roughly divided into two factions according to the prime objective, i.e., prioritizing interpretability or the maximum algorithm performance, although varying widely in design. At the theoretical research level, the interpretability and the mechanisms behind fuzzy representations have always been a primary concern of researchers in fuzzy systems since the transparency of the model and a good understanding of the knowledge representation are crucial when it comes to the manual construction of fuzzy rule-bases. This property is not always necessary for many application scenarios of neuro-fuzzy systems. Perhaps, for purposes such as data mining and knowledge abstraction, the models employed need to be as interpretable as possible to facilitate the extraction of expert knowledge. However, most of the problems to be solved by neuro-fuzzy systems are no different from that of ordinary neural networks, for which users are not concerned with whether knowledge can be obtained from the data in a way that humans understand but acquire the best performance. Most likely, the application of the interpretability of neuro-fuzzy systems is only meaningful to simple tasks such as model debugging, determination of hyperparameters, or adjustment of network structure, because specific knowledge interpretations are not part of the mandate. The dimensionality of today's datasets is often high, and the explosion of the rule-base caused by the curse of dimensionality can also make the model interpretation impracticable, even if the system has impeccable interpretability. From the perspective of system designing, it is hard to achieve both strong interpretability and the best algorithm accuracy for the same model, and it is often the case to sacrifice one for the good of the other. Considering that transparency is often of less importance for neuro-fuzzy systems, many researchers choose to put the performance and the accuracy of the model as the priority.

The deep neuro-fuzzy system is a product following this train of thought because the increase in network depth also dramatically reduces the transparency of the neuro-fuzzy system, which makes neuro-fuzzy systems share more common properties with general deep learning models. For this type of algorithm, strategies used on ordinary deep learning algorithms are also conducive to deep neuro-fuzzy systems, including adjusting the network structure and optimization policy, etc. What is exclusive about

the deep neuro-fuzzy system is that the model performance is closely related to the logic employed for fuzzy reasoning. Most of the existing neuro-fuzzy systems rely on type-1 logic for inference, but the simple semantic representation of type-1 fuzzy membership means it may require multiple rules to fully express an intricate logic statement, which often makes the final rule-base incredibly large. Such a massive rule-base subsequently brings two problems. One is that the model is prone to overfitting, which may reduce the generalization performance of the algorithm. The other is that if the number of input variables is immense, due to the exponential relationship between the size of the rule-base and the number of input variables, the curse of dimensionality is more likely to occur under a large base of exponent. In addition, the shallow semantics of type-1 logic also leads to relatively inaccurate representations for some sophisticated objects, which further affects the model performance [10]. Hence, new kinds of fuzzy logic are applied as a basis for fuzzy reasoning, of which the most popular is type-2 fuzzy logic [11]. Type-2 fuzzy logic has richer semantic expressions, with the ability to represent both intra-individual uncertainty and inter-individual uncertainty, increasing the information contained in a single rule, thereby leading to a reduced size of the rule-base and an increase in the generalization capability of the model [10]. Type-2 logic also allows for more accurate descriptions of some problems, which is advantageous to the model precision. Many deep neuro-fuzzy models based on type-2 fuzzy logic have emerged in the last decade and have shown a significant performance boost compared to their counterparts using type-1 logic [9]. Notably, the type-2 deep fuzzy models are also approaching the ceiling.

To further improve the accuracy and efficiency of the deep fuzzy algorithm, it is necessary to investigate new theories. In addition to type-1 and type-N logic, complex fuzzy set theory [12], [13] has also been successfully employed to build deep neuro-fuzzy models. A complex fuzzy set is a category of fuzzy sets in which the membership grades are represented via complex numbers, giving the fuzzy membership a two-dimensional attribute. The idea of complex fuzzy logic mainly originates from the wave-particle duality in quantum mechanics. It usually defines the fuzzy membership within the unit circle in the complex plane, of which the modulus can be considered the embodiment

of the particle, whereas its phase characteristic can correspond to the wave. In addition, the interaction between different complex fuzzy sets is carried out in the complex plane, which inevitably endows the operation rules with vector properties. The work of complex fuzzy theory is revolutionary, as it overturns the traditional understanding of logic in the field of theoretical mathematics, indicating inference logic does not have to be scalar but can also be vectorial [14]. This theory subverts the traditional research perspective of fuzzy theory and makes abstract algebra represented by the group theory gain importance in the study. A group is a class of algebraically closed structures consisting of co-domains and operational relations, for instance, the group of integers can be seen as a group composed of addition operations and the integer field. Similarly, all types of fuzzy logic, including type-1 logic, type-2 logic, and complex fuzzy logic, can be considered a unique group in abstract algebra. The introduction of group theory unifies the study of fuzzy mathematics with other fields of mathematics, and it allows research of fuzzy theory to conduct in a purely theoretical and abstract way that is away from human intuition. The study of complex fuzzy theory is highly dependent on group theory, and many of its characteristics are successfully proved based on the properties of the circle group and the Abelian group [15]. From a mathematical point of view, the complex fuzzy theory has sufficient theoretical plausibility.

Complex fuzzy theory is also meaningful from the angle of applied value. Firstly, the complex fuzzy set is defined in the complex plane, for which the magnitude-phase attribute of complex numbers gives the complex fuzzy set a natural advantage in dealing with quasi-periodic problems, making it ideal for applications such as time series forecasting and streaming data processing. Secondly, the rule-base of the complex fuzzy set has two-dimensional degrees of freedom, which brings two superiorities. One is that a single rule can contain richer information than one-dimensional type-1 fuzzy rules, which leads to better generalization performance [16]. The other benefit is that the two-dimensional complex plane diffuses the effect of outliers on the training process, making the model more robust to some data types [16]. Thirdly, complex fuzzy sets can express sophisticated semantics beyond the reach of previous fuzzy theories. Some research suggests that complex fuzzy logic can express meanings such as “counter" and “non" by

a single rule, which requires multiple rules for previous fuzzy logic to describe [17]. Unfortunately, even with substantial mathematical proof, due to the counter-intuitive nature of its formality, the semantic representation of the complex fuzzy membership has yet to be fully deciphered. Therefore, relying on expert knowledge to construct complex fuzzy inference systems is temporarily infeasible, while current research focuses on complex neuro-fuzzy systems. Despite this, its properties can still exceedingly narrow the size of the rule-base of a deep neuro-fuzzy model while bringing new possibilities for function approximation and the construction of data prediction algorithms.

It is worth noting that a special kind of deep neural network structure has also appeared in recent years, i.e., complex-valued neural networks (CVNNs) [18], and some researchers even added fuzzy elements to this foundation to establish complex-valued neuro-fuzzy systems (CVNFSs) [19], [20]. Although the words “complex" and “fuzzy" exist in both definitions, the complex neuro-fuzzy system and the complex-valued neuro-fuzzy system are two distinct concepts. The former emphasizes the application of complex fuzzy theory as inference logic for the algorithm, while the latter refers only to the involvement of complex values at the signal processing level [16]. CVNNs have quite limited application scenarios, and the most common one is the presence of complex-valued numeric representations in the data, to which signal processing methods for complex numbers are inevitable [18]. There is also a situation where some unique associations between several real-valued terms in the data and combining them into complex-valued forms for processing may achieve better results [18]. Besides, it has also been the case that the complex-valued structure can be used as a dimensionality reduction approach when the data dimensionality is high, and the model does not require high output precision [20]. In contrast, there is no restriction over application scenarios for most complex fuzzy neural systems. Although the complex-valued structure is essential in this type of algorithm, it only relates to the operation of the rule base, and the complex values are fuzzy memberships, which are converted into real-valued weights once the defuzzification of the complex fuzzy set finishes, and thus the input and output values of the network can be completely free of complex numbers. The complex neuro-fuzzy system is considered the most promising attempt after the type-2

model, but the current research in this field is still in its infancy, and only a handful of models have been proposed so far [21].

In this thesis, deep learning architectures based on both complex-valued structures and complex fuzzy theory are developed. In total, three models are proposed, the first with excellent capability to handle the curse of dimensionality [22], the second focusing on computational efficiency when dealing with real-world datasets, and the third being a robust learning model specifically for noisy data. All models are tested on specially designed synthetic datasets and three real-world datasets to simulate the performance under different application scenarios. The first one is a periodic time series that records the sunspot event over the last two centuries. The other two are high-dimensional metallurgical datasets, namely Charpy impact data and Ultimate Tensile Strength (UTS) data. Each dataset has its own uniqueness and poses different challenges to models. Note that the work in this thesis is limited to the study of numerical regression scenarios, and models for pattern recognition and classification purposes are not within the scope of discussion.

1.2 Research Aim

This study explores the potential of complex-valued structures and complex number fuzzy theory in building novel deep neuro-fuzzy systems. The focus is on mitigating several core problems, including the curse of dimensionality, model efficiency and generalization performance, and noise resistance of the algorithm. Experimental deep neuro-fuzzy architectures are developed to serve these purposes and are validated on various datasets to achieve convincing conclusions. The ultimate goal of the research is to obtain deep neuro-fuzzy systems with higher efficiency and better accuracy in sophisticated and tricky real-world scenarios.

1.3 Research Objectives

The objectives of this research are partitioned into the following list of stages:

1. Stage 1. The investigation and validation of the existing algorithms.

a) Investigate the existing algorithms based on complex fuzzy sets with respect to the architecture and the selection of fuzzy membership functions. Explore and summarize how those algorithms and membership functions work regarding different data types.

b) Investigate the existing neuro-fuzzy models based on complex-valued architectures. Analyze their rationale and determine whether there is potential to continue to research in this direction.

c) Investigate the possibility of improving the existing algorithms regarding the perspectives such as the choice of membership function, optimization method, and network structure, for the purpose of improving the model accuracy as well as facilitating their application on regression learning problems.

2. Stage 2. Explore the feasibility of complex-valued structures and complex fuzzy theory in mitigating the curse of dimensionality

a) Investigate the drawbacks of the existing neuro-fuzzy algorithms in dealing with the higher dimensional datasets (within 20 dimensions), exploring the possibility of utilizing the properties of complex-valued structures in this area.

b) Try to develop a new algorithm combined with the concepts of complex-valued structures and complex fuzzy theory to better model higher dimensional datasets and offset the curse of dimensionality.

c) Investigate the comprehensive performance of this newly proposed method, investigate its potential from the perspectives of accuracy, computational complexity, and transparency.

3. Stage 3. Investigate the potential of complex-valued structures and complex fuzzy

theory in dealing with real-world application scenarios.

a) According to the existing knowledge and experience, introduce the concept of complex-valued structures and complex fuzzy theory into the recursive deep neuro-fuzzy architectures to develop a new algorithm for real-world scenarios.

b) Apply this algorithm to applications including time series prediction and function approximation. Compare it with other state-of-the-art algorithms to verify whether it has advantages in terms of algorithm performance and efficiency.

4. Stage 4. Create robust learning methods using complex-valued structures and complex fuzzy theory.

a) Study current robust learning solutions for regression problems and summarize their strengths and weaknesses. Discuss the possibility of improving such solutions by introducing complex-valued structures and complex fuzzy theory.

b) Develop novel deep neuro-fuzzy models utilizing complex-valued structures and complex fuzzy theory for robust regression scenarios with significant noise and many outliers.

c) Compare the proposed algorithm with other similar methods to demonstrate its characteristics.

1.4 Research Contributions

The contributions of this research are as follows:

a) A new deep complex-valued single-iteration fuzzy system (DCVSF) is proposed, which shows that complex-valued structures can be used to offset the effects of the curse of dimensionality. The research has been published in Proceedings of 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), the title of the paper is “A New Deep Complex-Valued Single-Iteration Fuzzy System for Predictive Modelling."

b) A new rapid adaptive complex fuzzy inference system (RACFIS) which is designed for real-world applications is also proposed. A manuscript named “RACFIS: A New Rapid Adaptive Complex Fuzzy Inference System for Regression Modelling" has been submitted to a prestigious international journal with a view to publication.

c) A robust learning method called exclusionary neural complex fuzzy inference system (ENCFIS) is also developed, especially for regression scenarios where statistically significant noises are presented. The manuscript “ENCFIS: An Exclusionary Neural Complex Fuzzy Inference System for Robust Regression Learning" has also been submitted to a prestigious international journal with a view to publication.

1.5 Thesis Outline

The rest of the thesis is organized as follows:

- Chapter 2: The main task of this chapter is to perform a literature review of the development trajectories from crisp classical logic to fuzzy logic and from classic fuzzy logic to complex fuzzy logic. Besides, a brief introduction is included regarding fuzzy inference systems and defuzzification methods. The motivation for the transition from artificial neural networks to the neuro-fuzzy system is explained, and several new neuro-fuzzy architectures based on complex fuzzy theory are elaborated and evaluated. Considering the purely complex-valued structure, i.e., in the absence of complex fuzzy rules, also has unique values for ameliorating deep neuro-fuzzy systems, the potentiality of employing such designs in deep neuro-fuzzy systems to counter the curse of dimensionality is analyzed at the end of the chapter.

- Chapter 3: This chapter provides a detailed illustration of three real-world datasets applied in this thesis. The first dataset is the Sunspot time series data which contains the record of sunspot events since 1874. The rest two datasets

are function approximation data collected from two metallurgical processes, i.e., Charpy impact and Ultimate Tensile Strength (UTS). The performance indices adopted for benchmark tests are also mentioned as the background knowledge to allow the work to be better understood by non-specialist readers.

- Chapter 4: In this chapter, the concept of the complex-valued neural network and the Wang-Mendel (WM) fuzzy algorithm are fused to form a novel complex-valued Wang-Mendel (CVWM) method. This method reduces the fuzzy rule-base of the model to its square root level compared to the original WM method. By constituting a hierarchical architecture, a deep complex-valued single-iteration fuzzy system (DCVSF) is further elicited for higher dimensional situations. Such architecture can effectively mitigate the curse of dimensionality and can be trained using only one iteration. Considering that sparsity is often observed among high-dimensional datasets, which significantly weakens the performance of many similar algorithms, the t-distributed stochastic neighbor embedding (t-SNE) dimensionality reduction algorithm is therefore employed to increase data density for the model. Simulation results indicate that CVWM and DCVSF both exhibit competitive nonlinear approximation performance.

- Chapter 5: In this chapter, a new rapid adaptive complex fuzzy inference system (RACFIS) is developed for real-world datasets. The optimization policy includes a novel three-parameter quasi-hyperbolic momentum (QHM) optimization method and unsupervised learning is introduced, for the first time, to estimate the antecedent parameters for the complex neuro-fuzzy model. RACFIS reveals outstanding performance on all experimental datasets, obtaining excellent accuracies with an average of 10 times lower iteration counts (as compared with all benchmark models) and a reduction in fuzzy rule-bases by nearly 20%$\sim$30% (as compared with non-complex fuzzy models). RACFIS also presents a superior generalization performance that transcends all benchmark models, which is consistent with the previous analysis of the advantages of complex fuzzy logic. Furthermore, a mean impact value (MIV) algorithm

based on a radial basis function (RBF) network is designed to select variables of higher relevance.

- Chapter 6: This chapter connects the notion of robust learning with complex fuzzy theory for the first time, proposing an exclusionary neural complex fuzzy inference system (ENCFIS) for regression problems with the presence of heavy noises. The core of the algorithm includes pre-training the antecedent parameters using the convex clustering method, using the M-estimator to estimate the linear consequent parameters, and using the Huber Loss to replace the L2 loss function as the optimization reference. Experimental results reveal that this novel architecture has extraordinary performance on a dataset with massive (45%) label noises. Similar performance is also observed when testing on a distorted time series dataset (25% corrupted). Simulation results on metallurgy datasets also indicate that the approximation performance of ENCFIS is not compromised for the increase in robustness, making it an ideal candidate for common industrial scenarios with weak noise but tricky data characteristics.

- Chapter 7: This chapter discusses the conclusions of the thesis as well as the outlook over deep neuro-fuzzy architectures utilizing complex-valued structures and complex fuzzy logic.

2

Literature Review

> *There are three kinds of intelligence: one kind understands things for itself, the other appreciates what others can understand, the third understands neither for itself nor through others. This first kind is excellent, the second good, and the third kind useless.*
>
> – **Niccolo Machiavelli** –

This literature review begins with mathematical preliminaries of fuzzy theory, followed by a presentation of a relatively complete development trajectory of fuzzy theory. An overview of fuzzy inference engines and the transition track from artificial neural networks (ANNs) to neuro-fuzzy systems is then provided. The introductions to different types of complex fuzzy sets are also included, as well as an evaluation of currently existing complex fuzzy systems, as theoretical foundations for the models developed in chapters 5 and 6. The possibility of applying purely complex-valued structures to deep neuro-fuzzy systems to ease the curse of dimensionality is also discussed at the end of this chapter, as a theoretical preparation for the model in Chapter 4.

2.1 Mathematical Preliminaries for Fuzzy Set Theory

Fuzzy set theory, as a category of generalized set theory, is like any other mathematical theory, which does not come into being out of thin air but is the achievement from the intersection of classical set theory, classical algebra theory, and classical logic. The study of the mathematical premises of fuzzy theory helps to build a deeper understanding of the theory itself as well as provides better guidance for future research in numerous perspectives.

2.1.1 Algebraic Structures

In general algebra, the algebraic structure refers to a situation where one or more sets are algebraically closed under an assured series of operations. According to the explanation by Cohn [23], an algebraic structure should contain sets as well as operations or relations that conform to certain axioms. It is now accepted that if given a set of elements U and several operations Γ_i ruled over the set U, in which if all the operational objects for any arbitrary Γ_i belong to U and the operational outcomes of that Γ_i are covered by U as well, then this structure can be referred to as a closure group, i.e., an algebraic structure.

A vast number of different algebraic structures have emerged so far, and each pos-

sesses a varying set of attributes that are the result of the axioms that define the structure. Three representatives of such definition are the group composed of integers under the algebraic operation of addition; the domain that consists of rational numbers under rational operations such as addition and multiplication; and the territory which covers real and complex values under additions, multiplications, and non-integer exponentiations.

As specified by Gratzer [24], like many other mathematical definitions, relations are necessary for algebraic structures, among which the equivalence relation is the most prevalent. The general illustration of this case is that two structures are considered equivalent if an isomorphism holds between two of them. Given two sets χ and γ as well as two binary operations $'' + ''$ and " $\otimes''$, two individual groups $X = \{\chi, +\}$ and $\gamma = \{\gamma, \otimes\}$ can be defined. In compliance with the theorem, group X and group Y are isomorphisms if and only if a bijective mapping $\Lambda : \chi \to \gamma$ in which $\Lambda(a + b) = \Lambda(a) \otimes \Lambda(b)$ holds for all $a, b \in \chi$. Isomorphism is a property with several attributes including reflexivity, symmetry, and transitivity, which makes it identical to the relation of equivalence. The system behaviors of two isomorphic algebraic structures follow uniform principles and cast the equivalent formation on their individual domains, even if those respective domains might comprise of entirely different types of entities.

2.1.2 Partially Ordered Sets and Lattices

In order theory, a partial order [25] refers to a homogeneous relation " $\leq$ " which applies to a set of elements U. Regarding the relation " $\leq$ ", note that it is a one-way relation which does not hold the other way around, i.e., if $\alpha \leq \beta$, it can be described as " α is related to β ", but this does not deservedly signify " β is related to α " on account of the asymmetry of the relation itself. Additionally, a non-strict partial order is also an antisymmetric preorder which indicates that it must obey the following three axioms:

a) Reflexivity: $\alpha \leq \alpha$, i.e., each object relates to its own.

b) Antisymmetry: if $\alpha \leq \beta$ and $\beta \leq \alpha$, then $\alpha = \beta$. This implies two disparate objects

cannot relate to one another at the same time.

c) Transitivity: if $\alpha \leq \beta$ and $\beta \leq \gamma$, then $\alpha \leq \gamma$. In other words, the relation is transitive in the same direction.

Note that all elements α, β and γ belong to set U. A set with a partial order is referred to as a partially ordered set or a poset. Within a partially ordered set, relations such as " $\leq$ ", " $\geq$, and " $=$ " are available for only some of the elements from U. Otherwise, if such a conclusion holds for all elements of U, then U is said to be totally ordered by " $\leq$ ", " $\geq$ ", and " $=$ ".

A lattice is an algebraic structure composed of a set of elements U and a one-direction relation " $\leq$ " which partially orders U. For the purpose of constructing a lattice, this partial ordering is supposed to allow that within it all partially ordered subsets own a unique least upper bound (join) and a unique greatest lower bound (meet), i.e., supremum and infimum. If the supremum and infimum of any finite non-empty subset of U can be determined, then this structure is referred to as a complete lattice. There are several important properties for complete lattices, such as idempotency, mutativity, and associativity. Furthermore, a lattice is said to be distributive if and only if $(a \wedge b) \vee (a \wedge c) = a \wedge (b \vee c)$ and $(a \vee b) \wedge (a \vee c) = a \vee (b \wedge c)$ stands for $\forall a, b, c \in U$, where $\wedge$ denotes lattice meet while $\vee$ represents the lattice join, respectively.

Pykacz [26] provides a concise analysis of the relations between different properties of a lattice. According to his interpretation, when given a lattice $(U, \leq)$, if both higher bound and lower bound can be determined for a set of elements U, then this lattice is defined as bounded. If any element of U from a bounded lattice has a complementary counterpart in U, then this lattice is delimited as complemented. If the complemented set happens to be distributive, it has been proved that De Morgan's Law applies to the lattice in this case. Moreover, some lattices that are not distributive might still possess a weaker condition of modularity, for instance, $(a \wedge b) \vee (a \wedge c) = a \wedge (b \vee (a \wedge c))$ or if $a \leq b$, then $b = a \vee \left(a \vee b^{\perp}\right)^{\perp}$, where a, b, c belong to a lattice V, " $\perp$ " denotes the orthocomplementation map. This property is also referred to as orthomodularity which supersedes distributivity in quantum logic.

2.1.3 Classical Logic, Tautologies, and Modus Ponens

Logic is a path for bits of intelligence to quantitatively trace a reasoning process that enables replications as well as manipulations with mathematical principles. The focus on all kinds of logic study has something in common, i.e., the study of facticity in logical propositions. Among the logics proposed so far, classical logic [27] is the logic understood by most people in which the truth is binary, i.e., a proposition is either true or false. As the most widely applied deductive logic, classical logic is therefore the basis of numerous mathematical theories. In classical logic, simple propositions refer to linguistic statements or sentences that judge “true" or “false" in the universe of discourse, i.e., an element or set can be judged to be completely true or utterly false in this domain. Traditionally, we use the binary value $(0,1)$ to express the true or false of the element, where the false proposition is assigned the value 0, and the true proposition is designated the value 1. Additionally, there are five commonly used connectives that apply to classical logic, which are disjunction “$\vee$", conjunction “$\wedge$", negation ”$-$", implication ”$\rightarrow$" and equivalence ”$\leftrightarrow$", respectively.

In classical logic, there exists a category of compound propositions that always hold, disregarding the truth values of the separate simple propositions, and such a combination is said to be a tautology [28]. As a result of this unique feature, a mass of proofs and reasoning processes rely heavily on the use of tautologies. Modus ponens (MP) is considered the most classic tautology, that is a deduction of which the effect is to determine the truth value of a consequent when the truth value of the antecedent and the production rule are both provided. Designate two simple propositions X, Y and a rule $X \rightarrow Y$, according to the definition of MP, if both the proposition X and the rule $X \rightarrow Y$ stand, then the proposition Y must be true. Similarly, modus tollens (MT) which is defined as the opposite of MP, indicates a logic deduction where the premise is necessarily to be false if the consequent and the rule are both false. It is worth mentioning that MP and MT are not logically perfect. The most direct manifestation of this imperfection is that some inferences conflict with human intuition. Nevertheless, this method is still widely employed in many theories including fuzzy theory for deductive reasoning.

2.1.4 Multi-Valued Logic and Fuzzy Logic

The earliest systematic logic theory was developed by the ancient Greek philosopher Aristotle, who is also considered the father of classical logic. Even as the founder of classical logic, Aristotle did not believe that binary logic and the law of excluded middle are absolute conditions for the formation of logic and implied the possibility of multi-valued logic [29]. The law of excluded middle stipulates that the intersection of a set and its complement must be an empty set. Since then, the discussion about multi-valued logic stagnated for a long time, and it was not in the field of vision until reproposed by Polish mathematician Lukasiewicz in the 1920s [30]. Unlike binary logic, which uses authenticity as a criterion, multi-valued logic usually applies justification as a standard to construct a logic system. The concept and application of multi-valued logic have spawned entirely new fields, such as Rose logic [31] and assuredly fuzzy logic.

In fact, fuzzy logic is a further expansion of multi-valued logic theory. From this perspective, akin to probabilistic logic, fuzzy logic is a logic system with infinitely many values within the interval $[0, 1]$. Fuzziness is a phenomenon in that predicates can be applied to objects nonabsolutely, but to a certain degree with boundary conditions. The invention of fuzzy logic is revolutionary. As a tool for imprecise reasoning, this logic system realizes the ability to model human semantics for the first time [32]. And as a powerful tool for describing uncertainty in addition to probability, it has also played an important role in mathematical theory.

2.2 Zadeh's Fuzzy Sets & Logic Theory

As the founder of the fuzzy theory, Professor Zadeh made an immeasurable contribution to the development of this territory. Even though it has been over half a century since he first proposed the concept of fuzzy logic, his legacies are still instructive to current research on fuzzy systems. This section lists his profound theoretical contributions separately to express respect for this great researcher.

2.2.1 Type-1 Fuzzy Sets & Logic

Classic type-1 fuzzy set theory was first proposed by Professor Zadeh [1] in 1965, followed by fuzzy algorithms [2], fuzzy reasoning [3], and other supporting theories. Dating from the end of the 1980s, other researchers successively joined in the study of type-1 fuzzy theory, such as Yager [33], Kandel [34], Kosko [35], etc. In the early 1990s, the type-1 fuzzy theory matured and began to be applied in practice as an approach to intelligent control.

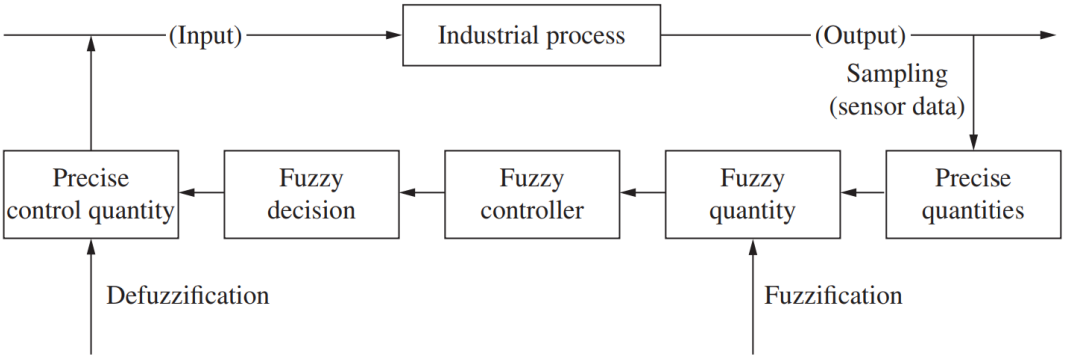

Figure 2.1: Typical closed-loop fuzzy control industrial control system [36]

The principle of type-1 fuzzy sets is very concise and easy to understand. By extending two truth values $\{0,1\}$ in binary logic into an infinite number of values between the interval $[0,1]$, a notion which is called membership grade is created. A typical type-1 fuzzy set consists of a real membership function and its supporting set, where the support can contain any kind of element, including numbers and language variables. Assume that X is a set consists of a collection of elements denoted by x in the universe of discourse W and $\mu_A(x)$ represents its corresponding membership value. Hence, the type-1 fuzzy set A can be defined in the form of an ordered pair as follows:

$$A = \{(x, \mu_A(x)) \mid x \in W\}. \tag{2.1}$$

Type-1 fuzzy set theory makes it possible to model vague human language expressions and perform imprecise inferences on the mathematical level through a series of logical operations. Analogous to crisp set theory, basic logic operations such as intersection,

union, complement, aggregation, and implication also apply to fuzzy sets. Fuzzy sets satisfy De Morgan's law, but the way to implement complement operation differs from that of crisp sets because the law of excluded middle does not hold for them. The Cartesian product is usually deployed to establish a relation between two different type-1 fuzzy sets, and fuzzy reasoning can be conducted by the composition of a series of relations. Fuzzy reasoning is guided by fuzzy rules, for which a single rule is often in the form of an "If... then..." proposition. A fuzzy system must have a rule-base that defines the properties and functions of this system according to a series of fuzzy rules. This form makes the model organized by type-1 fuzzy logic highly interpretable. It is worth noting that if the rule base of a system is too large, the interpretability will be compromised as a result.

In theory, if the size of the rule base is not limited, as a universal approximator, type-1 fuzzy logic can model almost all types of systems [37]. Intelligent systems based on type-1 fuzzy logic have many advantages, such as not necessarily requiring a very precise description of the object, relatively simple system design, and low computational cost, etc. This feature is used to design intelligent controllers, for instance, fuzzy PID controllers [38], which are widely applied to control various industrial processes and plenty of household appliances. However, the shortcomings of type-1 fuzzy logic are also significant. In terms of modeling, the accuracy of the type-1 fuzzy system is relatively low compared to other crisp modeling methods, and once humans have a deeper understanding of a certain object to implement accurate system modeling, this method will be of less necessity. Another frequently criticized problem of type-1 fuzzy logic is that its rulebase is not compact enough when describing sophisticated semantics, which often leads to the overfitting of the model and a massive rule-base with poor interpretability. The above factors limit the application of type-1 fuzzy logic, which implies room to improve and expand in the field of fuzzy logic.

2.2.2 Type-N Fuzzy Sets and Interval Type-2 Fuzzy Sets & Logic

The generalized type- N fuzzy set theory is also the theoretical achievement of Professor Zadeh [3] as an expansion of type-1 fuzzy theory in 1975. According to his description, the fuzzy membership of a type-N fuzzy set $A^{(n)}$ should comply with the measurable mapping $\mu_{A^{(n)}} : X \to F\left([0,1]^{n-1}\right)$, i.e., the primary membership mapping, where $\mu_{A^{(n)}}(x)$ is considered the membership mapping of a type-$(\mathrm{N}-1)$ fuzzy set on the interval $[0,1]$, also known as the secondary membership. Therefore, the type-N fuzzy set can be denoted as $F\left(X \times [0,1]^{n-1}\right)$.

Nevertheless, human understanding of type-1 fuzzy theory was in its infancy at the time, and the research on type-N fuzzy theory was in a state of no one for a long time. Until the late 1990s, as a direct expansion of the type-1 fuzzy sets & logic, type-2 fuzzy sets & logic have finally received attention. In the original description of a type-2 fuzzy set, the membership function is three-dimensional, where the third dimension is the value of the membership grade at each point on its two-dimensional domain, which is said to be the "footprint of uncertainty" (FOU) [39]. Considering that the general definition of the type-2 fuzzy set is complex for many application scenarios, its simplified version, i.e., interval type-2 fuzzy set in which the third dimension is a constant, is more popular.

The definition of an interval type-2 fuzzy set is not complicated. Given a universe of discourse X in which a type-2 fuzzy set $\tilde{A}$ is defined as follows:

$$\tilde{A} = \left\{ \lceil (x,u), \mu_\lambda(x,u) \rceil \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \right\}, \tag{2.2}$$

where J_x is the primary membership grade, $\mu_\lambda(x,u) \in [0,1]$ is the membership of J_x which is also known as secondary membership grade, and $\mu_\lambda(x,u)$ represents a type-1 fuzzy set. In fact, a type2 fuzzy set can be regarded as a collection of type-1 fuzzy sets. Akin to any other fuzzy theory, an interval type-2 fuzzy logic also includes the corresponding fuzzy logic system, referred to as type-2 FLS [11].

Compared with type-1 fuzzy systems, interval type-2 fuzzy systems have several superiorities [10]. a) A type-2 set can model both intra-individual uncertainty and inter-individual uncertainty at the same time, while a type-1 fuzzy set can only model the intra-individual. b) For many problems, the number of rules required for the type-2 fuzzy system is usually less than that of a type-1 fuzzy system. c) A type-2 fuzzy model usually has a smoother approximation process to the target, which leads to increased robustness. d) Given the same number of rules, a type-2 fuzzy system can realize the input and output mapping that the type-1 fuzzy system is not capable of. All characteristics mentioned above allow the interval type-2 fuzzy logic to have a wider range of applications than its traditional type-1 counterpart. In recent years, researchers have also attempted to utilize generalized type-2 logic to build machine-learning models and to study more advanced fuzzy logic, such as type-3 and type-4. Unfortunately, barely appreciable breakthroughs have been achieved due to the abstract nature of the semantic explanations of high-order fuzzy sets. Consequently, researchers may need to switch to a new train of thought, such as abandoning the traditional way of relying on human semantics to investigate fuzzy sets and seeking advancements from the perspective of abstract mathematical theory alternatively.

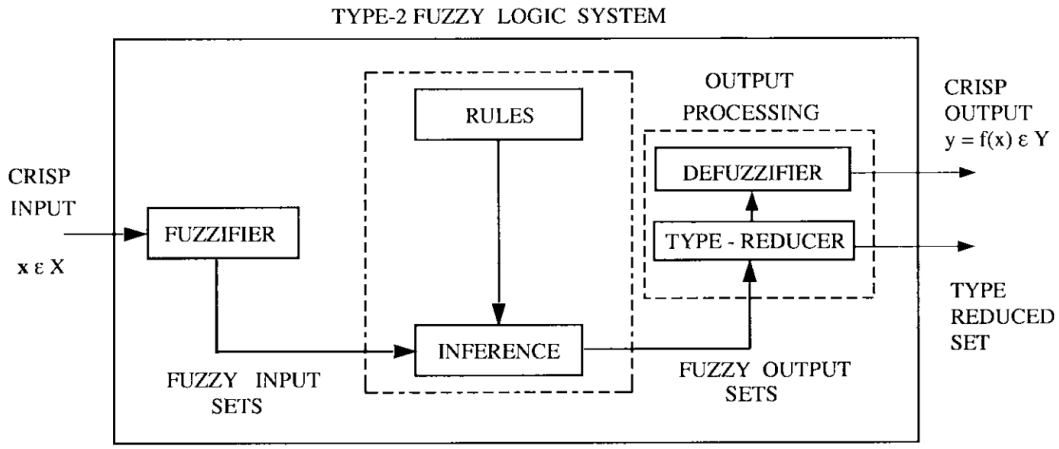

Figure 2.2: A simple demonstration of a Type-2 FLS [11]

2.2.3 Fuzzy Z-Number Theory

Z-number theory [40] is the last theoretical achievement of Professor Zadeh. Z-number is a new fuzzy set framework that combines constraints and reliability. The mathematical definition of Z-number is very simple that it can be represented in the form of an ordered pair of fuzzy numbers, for instance, $\Xi = (A, B)$, where Ξ denotes a Z-number, A represents the fuzzy restriction and B refers to the dependability of the first component. In general, the position of A is often held by a fuzzy set like $R(x)$, and B is a natural language qualifier that is used to describe the reliability of $R(x)$.

The proposal of the Z number provides a new direction for describing the reliability of a fuzzy proposition itself, which makes up for the deficiency of fuzzy theory in modeling uncertainty to some extent. In real-life scenarios, z-number is everywhere, such as (the expense is about one million dollars, very sure) or (the effectiveness of a Covid-19 vaccine is about 95%, sure), etc. However, natural language quantifiers are inaccurate, which means a widely accepted quantitative uncertainty measure for Z-number is required to enable applications such as decision-making. Given that the research is in the early stage, and the nature of the Z number has yet to be fully understood, suggesting a convincing and comprehensive theory is currently unavailable to support the application. Nevertheless, the application prospect of this idea is still worth expecting.

2.3 Fuzzy Inference Systems

Fuzzy reasoning, as a realization of approximate reasoning, is a crucial foundation of fuzzy theory. In actual implementations, this method does not focus on deduction based on axiomatic forms as classical logic or semantic operations employing assignments. Instead, fuzzy reasoning relies on numerical calculations to get the answer from premises. In 1973, Zadeh [41] first gave the most basic rule for the fuzzy inference system, i.e., fuzzy modus ponens (FMP). Later, through the efforts of Mamdani [42] this theory was successfully algorithmized and formed as today's compositional rule of inference (CRI). Given that the method of numerical reasoning is easier to implement with a computer, it becomes a vital branch in the field of computational intelligence.

2.3.1 Mamdani Fuzzy inference Systems

Early fuzzy inference systems are called pure fuzzy logic systems, and as the name suggests, the input and output of such systems are all fuzzy sets, which seriously restricts their practical value since most real-world actuators can only generate accurate outputs. In 1975, Mamdani [43] proposed the Mamdani fuzzy inference system by incorporating Zadeh's ideas, aiming to find a new way to control the steam engine. This framework consists of three main modules, i.e., fuzzification, fuzzy reasoning, and defuzzification. Assume that there is a fuzzy system with i noninteractive inputs $x_1, x_2, \ldots, x_i$ as antecedents and an output y as the consequent, which can be signified by a series of n IF-THEN rules in Mamdani form as:

$$IF\ x_1\ is\ A_1^k\ and\ \ldots\ and\ x_i\ is\ A_i^k\ Then\ y = C(x),\ for\ k = 1, 2, \ldots, n,$$

where $A_1^k, A_2^k, \ldots, A_i^k$ are the fuzzy sets of the k th antecedent groups, and $C(x) = \sum_{m=0}^{i} c_m x_m\ (x_0 = 0)$. If $c_m = 0$ stands when $m = 1, 2, \ldots, i$, then the system above is said to be a type-1 Mamdani fuzzy system. If no special restriction for c_m, then it is called a type-2 Mamdani fuzzy system. According to the definition, the antecedent and subsequent of a fuzzy rule are both fuzzy linguistic values, which are essentially adding fuzzy generators and fuzzy eliminators to the input and output components of the pure fuzzy logic system. Considering that both the input and output of a Mamdani system are crisp, they can be easily applied to realistic engineering actuators [44].

The Mamdani fuzzy inference system possesses several outstanding features of great research value. Firstly, the fuzzy inferencing of the Mamdani fuzzy system is carried out independently for each component, leading to concise parameter settings that can directly correspond to the parameters in algorithms like neural networks and evolutionary algorithms, which is beneficial for parameter optimization. Secondly, the Mamdani fuzzy inference system is proven to be a universal function approximator on any compact space, i.e., it can approximate any continuous function in a compact space with arbitrary precision, and such factor makes the application study of the Mamdani system theoretically sound. Thirdly, the fuzzy rules of a Mamdani system have a transpar-

ent and intuitive form that enables its linguistic knowledge-carrying capacity, which brings about a powerful and interpretable tool for expressing expert knowledge in multitudinous application domains. However, the inevitable deficiency of such a design of fuzzy inference framework is that the operation for defuzzification is often cumbersome, which results in the poor accuracy of the final product if an inappropriate defuzzification operator is selected.

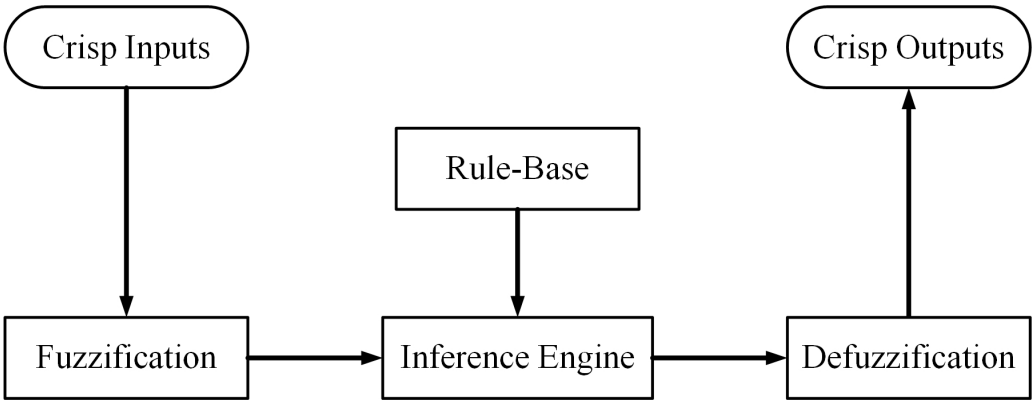

Figure 2.3: The structure demonstration of a Mamdani fuzzy inference system.

2.3.2 Takagi-Sugero-Kang (TSK) Fuzzy inference Systems

Takagi-Sugeno-Kang fuzzy (TSK) inference system [45], [46], also said to be Sugeno fuzzy inference system, adopts fuzzy singleton as the output membership function that is both constant and linear mapping of input values, which is an attempt for a systematic method to generate fuzzy rules from a series of input-output data. The defuzzification operation in Sugeno inferencing is of better computational efficiency in comparison to that of a Mamdani one, which benefits from employing a weighted average or sum of a spot of data points instead of deriving a centroid of a geometric plane shape. The expression of the i th fuzzy rule of a first-order Sugeno fuzzy system is as follows:

$$IF\ x_1\ is\ A_1^{(i)}\ and\ x_2\ is\ A_2^{(i)}\ and\ ...\ and\ x_n\ is\ A_n^{(i)}\ then\ y\ is\ s^{(i)T}x + s_0^i,$$

where $A_j^{(i)}$ is a fuzzy set, $s^{(i)} = \left(s_1^{(i)}, s_2^{(i)}, s_3^{(i)}, \ldots, s_n^{(i)}\right)^T$ are the parameters of linear functions, $i = 1, 2, \ldots, m$ is the number of fuzzy rules, and $j = 1, 2, \ldots, n$ is said to be the dimension of the input vector. Therefore, the crisp output of the system is given as

follows:

$$f(x) = \frac{\sum_{i=1}^{m} \left(s^{(i)T} x + s_0^i \right) \mu_{A^{(i)}}(x)}{\sum_{i=1}^{m} \mu_{A^{(i)}}(x)}, \tag{2.3}$$

where $x = (x_1, x_2, \ldots, x_n)^T$ is the input vector of this Sugeno system. It is also proved that a Sugeno model is equivalent to a linear regression system in fuzzy space, which leads to good approximation performance of this model to be widely adopted in system identification, pattern recognition, image processing, and data mining.

The Sugeno fuzzy logic system has a wide range of application prospects in the field of state prediction thanks to the above characteristics. Firstly, previous prediction methods are subject to fuzzy semantic representations that are not friendly to computation, leading to a lack of adaptiveness and requiring additional manual intervention. Dissimilar to earlier attempts, the Sugeno fuzzy logic system is rule-based in which data and information, such as error and expert experience and knowledge, can all be included. This property enables the flexibility of designing appropriate correction sub-systems in a prediction model. Secondly, the fuzzy attribute of a Sugeno system means it does not need to establish a precise mathematical model of the object, which is considered a challenging task in prediction. In addition, the Sugeno fuzzy inference system is a nonlinear architecture, which makes it very suitable for feedback error correction of nonlinear targets. Note that the advantages of the Sugeno inference system over the Mamdani inference system are achieved at the expense of interpretability, which limits its application to some system design tasks that mainly rely on expert knowledge.

2.3.3 Tsukamoto Fuzzy Inference Systems

Tsukamoto [47] proposed a fuzzy inference approach in which the monotonic membership function, sometimes named a shoulder function, is deployed for the consequent fuzzy set of every fuzzy rule. For a shoulder membership function, the output is a crisp value determined by the membership value resulting from fuzzy antecedents. Akin to the Sugeno system, the outcome of the entire system is generated by taking the weighted average of the inference result for each rule. Given a fuzzy model with i non-interactive variables $x_1, x_2, \ldots, x_i$ as inputs and y as the output, its Tsukamoto form n

IF-THEN rules are given as follows:

$$IF\ x_1\ is\ A_1^p\ and\ ...\ and\ x_i\ is\ A_i^p\ Then\ y^p = B\left(x^p\right),\ for\ p = 1,2,\ldots,n,$$

where $A_1^p, A_2^p, \ldots, A_i^p$ are the fuzzy sets of the p th antecedent groups, and $B(x)$ is a monotonic function whose successive values are increasing, decreasing, or constant. Consequently, the overall output will be calculated by the weighted average of each output y^p:

$$f(x) = \frac{\sum_{i=1}^{p} w^p B\left(x^p\right)}{\sum_{i=1}^{p} w^p}, \tag{2.4}$$

where w^p is the weight of the p th rule that corresponds to its monotonic function.

Tsukamoto fuzzy inference systems have very distinct characteristics. Since the firing strength of each rule corresponds to a crisp output, the aggregation of the final result for a Tsukamoto model also avoids the computation-consuming defuzzification process. However, the special attributes of the output membership function of this method also lead to limitations such as low transparency, which hinders its application as a general approach. In addition, the monotonic property of the membership functions decides it would only be employed in specific situations, which is not a universal solution. In fact, an ordinary Sugeno system is perfect enough to realize all the functions of the Tsukamoto system and has better interpretability and applicability. Therefore, Tsukamoto inference systems that based on monotonic functions are rarely seen in practice.

2.4 Neuro-Fuzzy Inference Systems

The artificial neural network (ANN) is an implementation method of artificial intelligence generated by imitating the working mechanism of biological neurons. In 1958, Rosenblatt [48] first proposed the concept of primitive perceptron, which is considered the beginning of modern ANNs. Neural network algorithms usually consume plenty of computation resources, but the performance of early computers was far from sufficient, leading to the difficulty in providing support for it. This approach to realizing artificial intelligence was therefore marginalized until the explosive growth of computer performance in the 21st century. Since then, ANN and subsequent deep learning technology have finally become the research focus. In contrast, fuzzy logic methods were once prevalent in the 1990s because their low computational requirements and intelligent attributes seemed very fashionable at the time, but the craze for fuzzy systems faded after the rise of deep learning algorithms. Subsequently, the topic of fuzzy systems continues to be in a downturn, forcing researchers to pay close attention to neural networks and deep learning research. Under the collision of two trains of thought, some researchers introduced the idea of feedforward networks and radial basis models into fuzzy systems design, constructing a type of hybrid system, i.e., neuro-fuzzy systems. Objectively, introducing the data-driven characteristic of deep networks into fuzzy systems is a promising idea, injecting new impetus into this territory. The unique knowledge representation of fuzzy systems enables the interpretability of neuro-fuzzy systems, which also brings new possibilities to the deep learning area dominated by inscrutable black box models.

2.4.1 Feedforward Neural Networks and Backpropagation

Feedforward neural network [49] is the most applied ANN architecture, in which each neuron of the network accepts the input from the previous layer and outputs it to the following layer without any feedback during the process. In terms of architecture, feedforward networks consist of an input layer, several hidden layers, and an output layer. The input layer is usually not included in the total number of layers of the net-

work, and the number of hidden layers does not have specific confines, for which the number can be zero or many. Traditionally, neural networks with two or more hidden layers are considered deep neural networks. From the aspect of the systems theory, the feedforward network is a static nonlinear mapping, which obtains nonlinear processing capability through the map composition of a series of simple nonlinear outputs from internal units. Most feedforward architectures are learning networks with remarkable classification and pattern recognition capabilities, but the lack of feedback keeps them away from the good dynamic performance. Typical feedforward networks include perceptron networks [50], [51], BP networks [52], radial basis function networks [53], convolutional networks [54], etc.

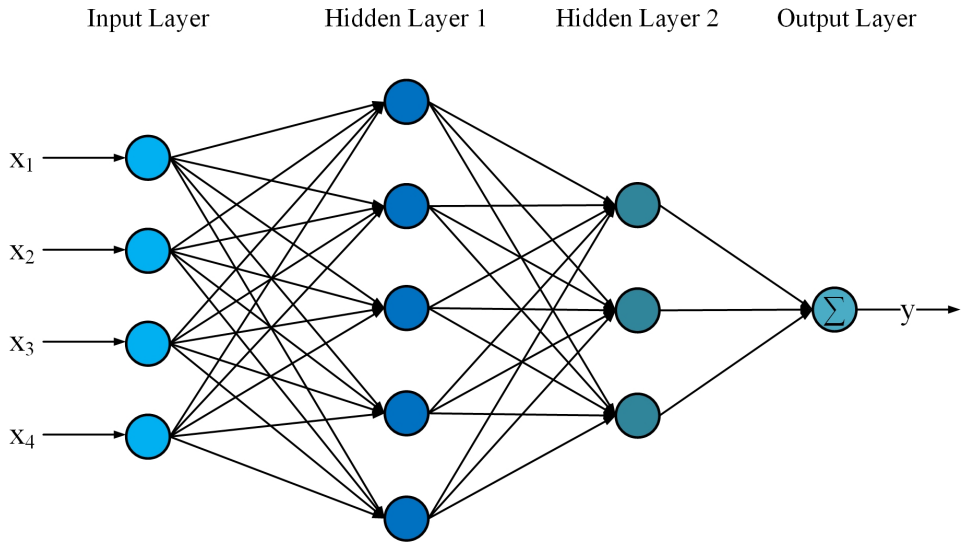

Figure 2.4: An example of the feedforward neural network.

Backpropagation [55], also known as error backpropagation, is an algorithm to realize parameter optimization for multi-layer ANNs by executing gradient descent. The backpropagation algorithm includes two phases, namely signal propagation and weight updating, where the signal propagation process consists of forward propagation and backpropagation two steps. In the forward propagation step, the signal enters the network from the input layer and gets a preliminary result at the output layer, while in the backpropagation stage, this result is compared with the label corresponding to the input to obtain the training error which is usually measured by the loss function. For weight updating, the chain rule is employed to calculate the gradient of the loss function as

well as the gradient of each layer, which is used as prior knowledge of the optimization surface to guide the gradient descent, updating the network parameters of each neuron to minimize the loss function. The weight updating process usually requires a learning rate term as a hyperparameter to control the training speed in order to obtain a solution closer to the global optimum.

For each neuron in the network, the optimization process of its weights can be considered a first-order differential approximation to the optimum of a compound function. Assume there exists a Q layer ANN, for which the mathematical representation of the i th neuron of the k th layer is given as follows:

$$y^k = f^k\left(\sum_{p=1}^{n_{k-1}} y_p^{k-1} W_{ip}^k\right), \tag{2.5}$$

where W_i^k is the weight vector of this neuron, $f^k : \mathbb{R} \to \mathbb{R}$ is the activation function, $y^{k-1} = \left[x_1, x_2, \ldots, x_{n_{k-1}}\right]$ denotes the input vector, and $y^k = [y_1, y_2, \ldots, y_{n_k}]$ refers to the output vector. Given a loss function l, then the loss of the feedforward phase can be represented as follows:

$$l^Q\left(W^Q, y^{Q-1}\right) = l\left[f^Q\left(\sum_{p=1}^{n_{Q-1}} y_p^{Q-1} W_p^Q\right)\right]. \tag{2.6}$$

Similarly, the compound form of the loss regarding the i th neuron of the k th layer is as follows:

$$l^k\left(W_i^k, W^{k+1}, \ldots, W^Q, y^{k-1}\right) = l^Q\left\{W^Q l^{Q-1}\left[W^{Q-1} \ldots f^k\left(\sum_{p=1}^{n_{k-1}} y_p^{k-1} W_{ip}^k\right)\ldots\right]\right\}. \tag{2.7}$$

According to the chain rule, for $1 \leq k \leq Q$, the gradient of this neuron can be calculated according to the following equation set:

$$\begin{cases} \frac{\partial l^k}{\partial W_i^k} = \frac{\partial l^Q}{\partial l^{Q-1}} \cdot W^Q \circ \frac{\partial l^{Q-1}}{\partial l^{Q-2}} \cdot W^{Q-1} \circ \ldots \circ \frac{\partial l^{k+1}}{\partial f^k} \cdot W^{k+1} \circ \frac{df^k}{dx} \cdot y^{k-1} \\ \frac{\partial l^k}{\partial y^{k-1}} = \frac{\partial l^Q}{\partial l^{Q-1}} \cdot W^Q \circ \frac{\partial l^{Q-1}}{\partial l^{Q-2}} \cdot W^{Q-1} \circ \ldots \circ \frac{\partial l^{k+1}}{\partial f^k} \cdot W^{k+1} \circ \sum_{p=1}^{n_k}\left(\frac{df^k}{dx} \cdot W_{ip}^k\right). \end{cases} \tag{2.8}$$

Hence, if the vanilla gradient descent method is applied, then the weight update rule

for this neuron at the t th iteration is obtained below:

$$^{t+1}W_i^k = {}^tW_i^k - \eta \frac{\partial l^k}{\partial {}^tW_i^k}, \tag{2.9}$$

where η is the learning rate or step size. It is worth noting that although there is a backpropagation process, this step only changes the weight without any impact on the input signal, for which, from the perspective of control theory, the feedforward network is still an open-loop system regardless of the use of backpropagation.

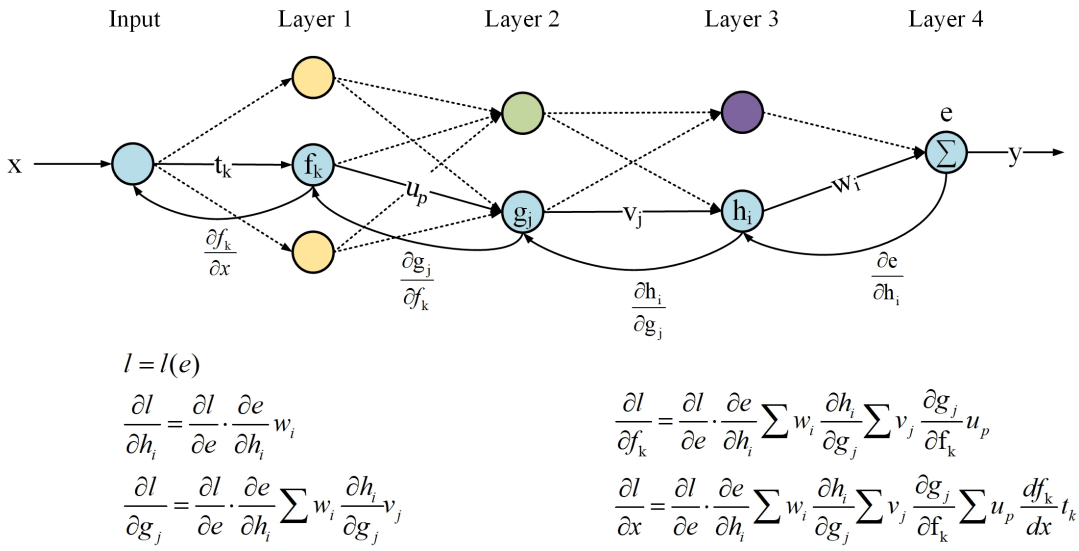

Figure 2.5: Backpropagation in a feedforward neural network.

2.4.2 Radial Basis Function Network and Fuzzy System

Powell [56] summarized the previous studies of function interpolation methods and pioneered the term “radial basis function (RBF)" to describe a category of functions that are applied for multivariate interpolation purposes. Inspired by Powell’s research, Broomhead and Lowe [57] proposed the prototype of the RBF neural network in 1988. Subsequently, in 1989, Moody and Darken [53] developed a network structure powered by locally tuned processing units, which is what is known today as the RBF network. The most notable difference between the RBF network and other feedforward networks is that most feedforward architectures are global approximation systems, while the RBF network is a local approximation design. In a global approximation network such as

the BP network, any individual parameter would affect the global performance, leading to slow convergence as well as obtaining the local minimum in optimization. The local approximation characteristic of the RBF network makes the parameters only affect its corresponding output without influencing the rest of the network, which avoids the problem of adjusting all weights in the model only to adapt a subtle change in the input in the global approximation model, enabling a faster convergence and a better approximation.

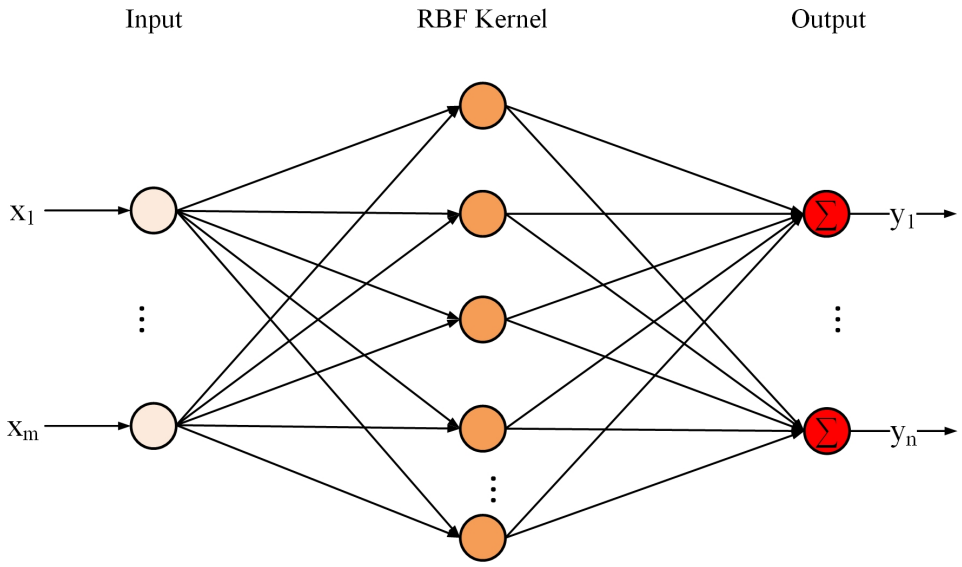

Figure 2.6: The radial basis function network.

The RBF network consists of two layers, among which the node in the hidden layer is composed of a radial basis function, while the nodes of the output layer are usually simple linear functions. The Gaussian function is the most popular basis function for RBF networks. The radial basis function in a hidden layer node plays the role of a kernel function, which produces a local response to the input signal. To be more specific, if the input signal is close to the center of the radial basis function, a relatively large output will be generated. On the contrary, if this signal is far from the kernel center, the response will become very small. Moreover, by setting a threshold, the node can automatically ignore the irrelevant input outside of its Mapping area. The RBF network is based on function interpolation to achieve the approximation of an arbitrary objective function. Interpolation is a category of statistical methods by which associated known samples are utilized to depict an unknown regular pattern [58]. Given an input dataset

with n known sample points $X^1, X^2, \ldots, X^n$, of which each point has m dimensions. To implement RBF interpolation, every input vector is designated a basis function which is defined as $\varphi\left(\|X - X^p\|\right), 1 \leq p \leq n$, where $\|X - X^p\|$ denotes the Euclidean distance between the m-dimensional variable X and the sample $X^p (m < p)$. The role of the radial basis function is to map lower m-dimensional data points to a higher p-dimensional Hilbert space for better linear separability, such that the interpolation function $f(X)$ can be obtained in a linear mapping form as follows:

$$f(X) = \sum_{p=1}^{n} w_p \varphi_p \left(\|X - X^p\|\right) + g(X), \tag{2.10}$$

where $g(X)$ is a low degree polynomial with m variables. Substitute the training labels $b_1, b_2, \ldots, b_p$ of each data point into the interpolation function, and the following equation set is obtained:

$$\begin{cases} w_1\varphi_1\left(\left\|X^1 - X^1\right\|\right) + w_2\varphi_2\left(\left\|X^1 - X^2\right\|\right) + \cdots + w_p\varphi_p\left(\left\|X^1 - X^p\right\|\right) = b_1 \\ w_1\varphi_1\left(\left\|X^2 - X^1\right\|\right) + w_2\varphi_2\left(\left\|X^2 - X^2\right\|\right) + \cdots + w_p\varphi_p\left(\left\|X^2 - X^p\right\|\right) = b_2 \\ \vdots \\ w_1\varphi_1\left(\left\|X^p - X^1\right\|\right) + w_2\varphi_2\left(\left\|X^p - X^2\right\|\right) + \cdots + w_p\varphi_p\left(\|X^p - X^p\|\right) = b_p, \end{cases} \tag{2.11}$$

Rewrite variables in the matrix form:

$$\Phi = \begin{bmatrix} \varphi_1\left(\left\|X^1 - X^1\right\|\right) \varphi_2\left(\left\|X^1 - X^2\right\|\right) & \cdots & \varphi_p\left(\left\|X^1 - X^p\right\|\right) \\ \varphi_1\left(\left\|X^2 - X^1\right\|\right) \varphi_2\left(\left\|X^2 - X^2\right\|\right) & \cdots & \varphi_p\left(\left\|X^2 - X^p\right\|\right) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1\left(\left\|X^p - X^1\right\|\right) \varphi_2\left(\left\|X^p - X^2\right\|\right) & \cdots & \varphi_p\left(\|X^p - X^p\|\right) \end{bmatrix}, W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix}. \tag{2.12}$$

Then the equation set (2-14) can be rewritten as follows:

$$\Phi W = b. \tag{2.13}$$

For any invertible matrix Φ, the weight vector can be obtained through its inverse matrix:

$$W = \Phi^{-1} b. \tag{2.14}$$

Note that for a generalized RBF network, the number of its basis functions is often less than the number of input data points, for which the pseudo-inverse can be used to calculate the weight vector:

$$W = \left(\Phi^T \Phi\right)^{-1} \Phi^T b. \tag{2.15}$$

In practice, the generalized RBF network is more frequently used, because the number of known samples in a dataset is often large, and it is obviously impractical to assign a basis function to every data point in such cases.

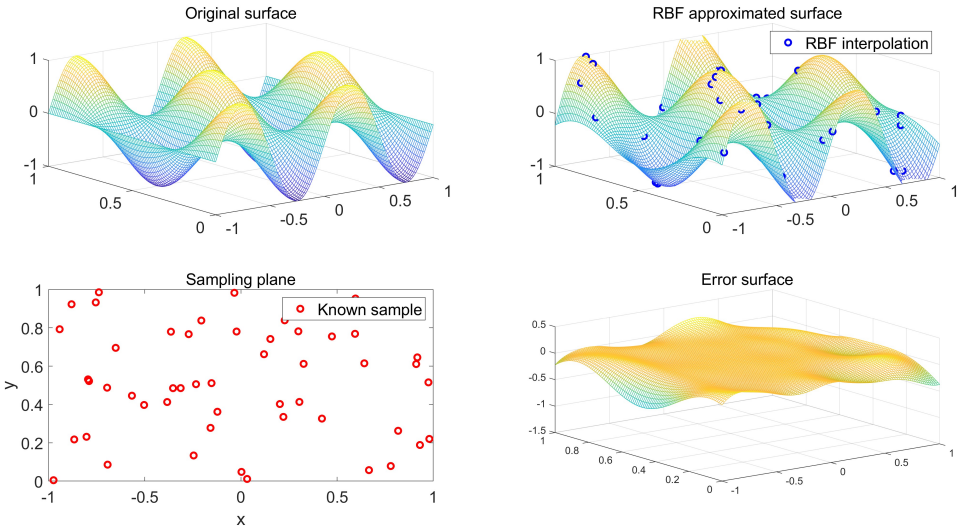

Figure 2.7: RBF interpolation of a non-convex surface.

In the early stages of the study, some researchers noticed the similarity between RBF networks and neuro-fuzzy systems and tried to combine the two to form new machine learning algorithms. Jang et al. [59] proved that the fuzzy system and the RBF network have functional equivalence under certain constraints, revealing the potential of integrating RBF networks into the fuzzy system. Following this train of thought, Cho [60] studied the several applications of RBF networks in fuzzy systems, successfully established a neuro-fuzzy system that utilizes an RBF network as a subsection, and further implemented three different network architectures with extended RBF components. However, these studies did not unveil the intrinsic connection between RBF networks and neuro-fuzzy systems. In fact, subsequent studies have shown that the generalized RBF network is mathematically equivalent to the Sugeno fuzzy inference system. The difference between a Sugeno fuzzy system and an ordinary RBF network is that the

parameters in the fuzzy system are given physical meanings to allow better transparency and interpretability. This conclusion also applies to all neuro-fuzzy systems and neural networks, of which the difference between them is simply the inconsistency in the narrative caused by distinct research angles. The mathematics behind these two fields is the same. Enlightened by this fact, many researchers have taken the RBF network as the prototype to design their neuro-fuzzy systems.

2.4.3 Mamdani Neuro-Fuzzy System

Mamdani fuzzy inference system is the earliest fuzzy inference method that emerged. Naturally, many early neuro-fuzzy systems are designed based on this inference engine, i.e., Mamdani neuro-fuzzy systems [61]. The so-called neuro-fuzzy system is based on the ordinary fuzzy engine with the addition of adaptive optimization policies so that the fuzzy system can also have an end-to-end self-learning ability similar to neural networks. The input and output nodes of the neuro-fuzzy system are equivalent to the counterparts of the neural network, while the hidden layer can play the role of membership functions, fuzzy rules, and defuzzification inference engines. The introduction of neural network structure brings two discernible advantages. Firstly, this weakens the influence of human errors in traditional fuzzy modeling, making the inference product of the algorithm closer to the actual situation and obtaining a more powerful performance. Secondly, the powerful parallel processing capability of the neural network design dramatically enhances the performance of the fuzzy inference system, thus enabling it to resolve more intricate problems. Akin to the RBF network, the Mamdani network is also a local approximation structure. However, each part of the Mamdani structure has interpretable physical meanings, which enables manual intervention under the expert knowledge of the fuzzy system. This property is also the most distinct advantage of neuro-fuzzy systems over pure neural network models because most neural networks are black boxes, for which users are usually unable to explore the connections between internal parameters, causing inconvenience in adjusting the network parameters. In contrast, the parameters of a neuro-fuzzy system can be determined per prior knowledge, giving them performance beyond that of ordinary neural networks in some situ-

ations.

The mathematical expression of the Mamdani neuro-fuzzy system is analogous to the ordinary Mamdani system but with some simplifications to accommodate the framework of neural networks. Given a Mamdani neuro-fuzzy system defined by the following IF-THEN statement:

$$\textit{IF } x_1 \textit{ is } A\left(x_1\right) \textit{ and ... and } x_n \textit{ is } A\left(x_n\right), \textit{ Then } y \textit{ is } B(y),$$

where $x = [x_1, x_2, \ldots, x_n]^T$ is the input vector and for each variable x_i there is a fuzzy antecedent statement $A\left(x_i\right) = \left\{A_1^i, A_2^i, \ldots, A_m^i\right\}, i = 1, 2, \ldots, n$, in which $A_1^i, A_2^i, \ldots, A_m^i$ are the fuzzy sets defined in the universe of discourse U_i. The corresponding fuzzy membership function for each set can be represented as $\mu_{A_j^i}\left(x_i\right), (i = 1, 2, \ldots, n; j = 1, 2, \ldots, p)$. Similarly, if y denotes the output of this system, then the fuzzy statement for the consequent part is $B(y) = \{B_1, B_2, \ldots, B_m\}$, where B_i is the i th set of this statement, of which the membership function $\mu_{B_i}(y)$ is obtained as follows:

$$\mu_{B_i}(y) = \begin{cases} \mu_{A_1^i}\left(x_1\right) \wedge \mu_{A_2^i}\left(x_2\right) \wedge \ldots \wedge \mu_{A_n^i}\left(x_n\right) \wedge \mu_{B_i}(y) \\ \text{or} \qquad\qquad\qquad\qquad\qquad\qquad , i = 1, 2, \ldots, m\,, \\ \mu_{A_1^i}\left(x_1\right) \cdot \mu_{A_2^i}\left(x_2\right) \cdot \ldots \cdot \mu_{A_n^i}\left(x_n\right) \cdot \mu_{B_i}(y) \end{cases} \tag{2.16}$$

where $\wedge$ refers to the fuzzy intersection operator, $\cdot$ denotes the product operator, and $\mu_{A_1^i}\left(x_1\right) \wedge \mu_{A_2^i}\left(x_2\right) \wedge \ldots \wedge \mu_{A_n^i}\left(x_n\right)$ or $\mu_{A_1^i}\left(x_1\right) \cdot \mu_{A_2^i}\left(x_2\right) \cdot \ldots \cdot \mu_{A_n^i}\left(x_n\right)$ is the firing strength of the i th rule. Both methods are popular and should be selected according to the need of the problem. Note that in neuro-fuzzy systems, the product method is often favored. Further, the final fuzzy inference outcome can be calculated using fuzzy union operation, which is given as follows:

$$\mu_B(y) = \mu_{B_1}(y) \vee \mu_{B_2}(y) \vee \ldots \vee \mu_{B_m}(y). \tag{2.17}$$

The end-to-end feature of the neuro-fuzzy system dictates that its output must be a crisp value instead of a fuzzy representation, which indicates that defuzzification is requisite.

In Mamdani models, the weighted average is the most employed defuzzification method and its discrete form is given below:

$$y_c = \frac{\sum_{i=1}^{m} y_i \mu_{B_i}(y_i)}{\sum_{i=1}^{m} \mu_{B_i}(y_i)}, \tag{2.18}$$

where $\mu_{B_i}(y_i) = \max_y \mu_{B_i}(y)$ is the centroid of the consequent membership function, which also equals to the firing strength of the antecedent part. Let α_i represent the firing strength of the i th rule, the final crisp output of the system can also be expressed in the following form:

$$y_c = \sum_{i=1}^{m} y_i \overline{\alpha_i}, \tag{2.19}$$

where $\bar{\alpha}_i = \frac{\alpha_i}{\sum_{i=1}^{m} \alpha_i}$.

The Mamdani neuro-fuzzy system integrates the classic fuzzy reasoning process that applies fuzzy semantics understandable for humans, allowing it to retain the most interpretability. However, the fundamental discrepancy between fuzzy semantics and crisp mathematical logic weakens the precision of the inference model. Additionally, the selection of fuzzy inference operators is excessively dependent on human experience and subjective perception, leading to a further loss in accuracy. Another cost of retaining interpretability is that Mamdani models often contain procedures indispensable for interpretable reasoning but clumsy in terms of getting accurate results, causing decreased efficiency if a large data volume and high data dimensionality are presented. Note that even the interpretability of the Mamdani neuro-fuzzy system is not always guaranteed, and it may fail under some circumstances. For example, when the rule-base is too large to interpret. Such deficiencies determine that the Mamdani model is more suitable for scenarios with low-dimensional, small data volume, and low accuracy requirements. Unfortunately, the application scenarios for today's machine learning algorithms are generally much more complex and more challenging. Therefore, current neuro-fuzzy algorithms rarely adopt the Mamdani method as the inference engine.

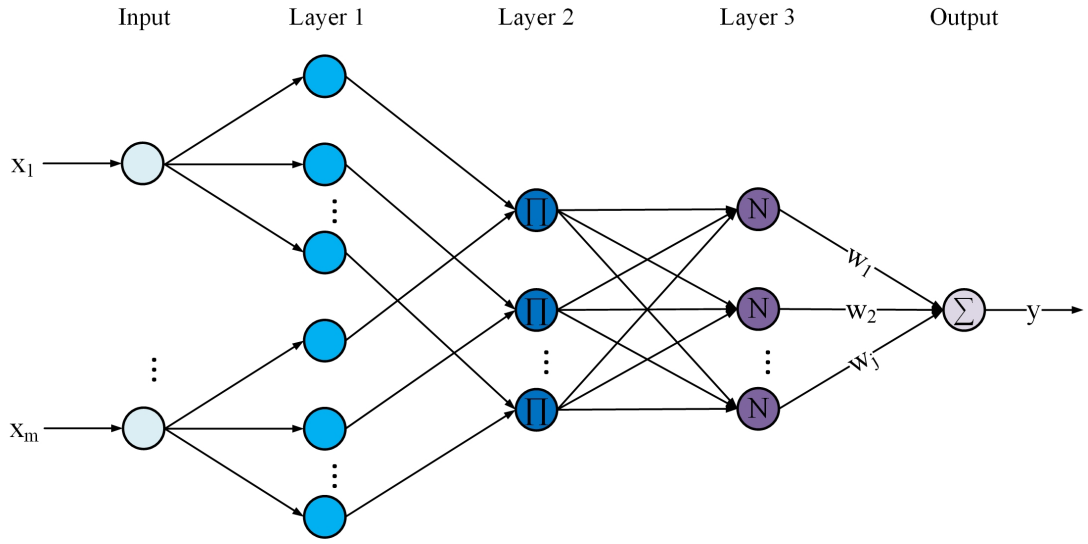

Figure 2.8: A multi-input-single-output Mamdani neuro-fuzzy system.

2.4.4 Sugeno Neuro-Fuzzy System

The Sugeno neuro-fuzzy system is a universal approximator [62] with a local approximation structure, which is de facto equivalent to the RBF network. The property of the Sugeno network is determined by the Sugeno fuzzy inference engine it applied. As opposed to its Mamdani counterpart, the Sugeno fuzzy inference engine does not consider the preservation of semantical interpretability as the top priority. Instead, it reduces the clumsy semantic inference process to the linear combination of first-order or zero-order fuzzy rules, thus significantly improving the accuracy and efficiency of the inference engine. For neuro-fuzzy systems, their application scenarios are usually not fundamentally different from that of ordinary neural networks, i.e., efficiency and accuracy are pursued, while the interpretability requirements are limited to parameter tuning and debugging. The Sugeno engine meets these requirements perfectly, albeit at the expense of certain interpretability. Despite this, it is inexact to say that this design is less interpretable than the Mamdani method. This narrative may be correct if it is purely to understand the semantics, but the semantic interpretation is not identical to the interpretability. The Sugeno model is closer to traditional mathematical models, allowing it to be easily unraveled by mathematical analysis methods. For neuro-fuzzy systems targeting sophisticated data types, the conventional semantic interpretation often fails because the rule-base is too large or intricate, whereas more information is

available if interpreted mathematically, especially from a statistical point of view. It is fair to say that the Sugeno model is an excellent fuzzy inference engine that successfully trades off the performance and the interpretability, leading to more widespread use of Sugeno neuro-fuzzy systems than their Mamdani counterparts. Therefore, most of the emerged neuro-fuzzy systems adopt the Sugeno inference model as the inference engine, including the famous adaptive neuro-fuzzy inference system (ANFIS) [63].

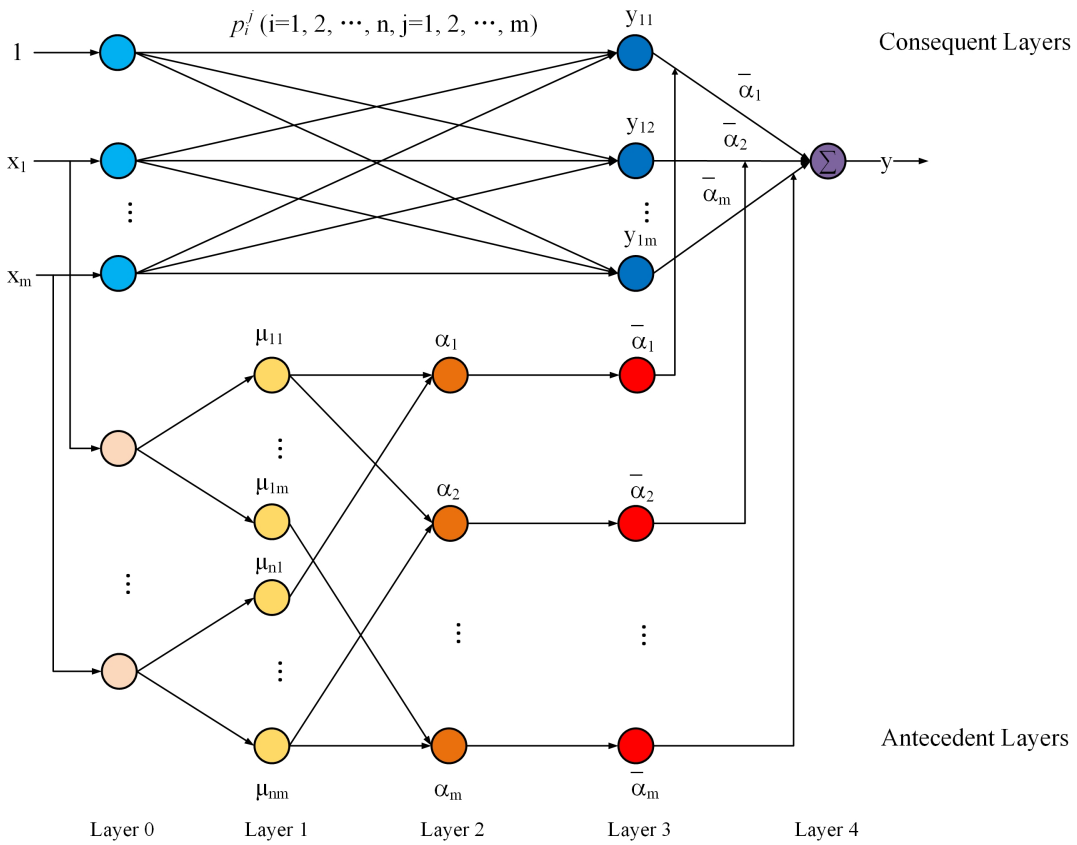

Figure 2.9: The antecedent and consequent part of a single output Sugeno neuro-fuzzy model.

For an n-antecedent and m-rule Sugeno neuro-fuzzy structure, the rule-base is usually organized according to the following IF-THEN form:

Rule 1:IF x_1 *is* A_1^1 *and* x_2 *is* $A_2^1 \ldots$ *and* x_n *is* A_n^1, *then* $f_1 = p_1^1 x_1 + p_2^1 x_2 + \cdots + p_n^1 x_n + q_1$;

Rule 2:IF x_1 *is* A_1^2 *and* x_2 *is* $A_2^2 \ldots$ *and* x_n *is* A_n^2, *then* $f_1 = p_1^2 x_1 + p_2^2 x_2 + \cdots + p_n^2 x_n + q_2$;

$\ldots\ldots$

Rule m:IF x_1 *is* A_1^m *and* x_m *is* $A_2^m \ldots$ *and* x_n *is* A_n^m, *then* $f_1 = p_1^m x_1 + p_2^m x_2 + \cdots + p_n^m x_n + q_m$,

where $x = [x_1, x_2, \ldots, x_n]^T$ denotes the input vector, $A_1^i, A_2^i, \ldots, A_m^i$ are the fuzzy sets defined in the universe of discourse U_i, and $f_1, f_2, \ldots, f_m$ refer to the values of the consequent part of the system. p_j^i and $q_i (i = 1, 2, \ldots, m; j = 1, 2, \ldots, n)$ are the linear coefficients that replace the fuzzy semantic in the Sugeno engine to determine the status of the consequent. Designate $\mu_{A_j^i}(x_j)$ to be the membership function for each input variable, then the firing strength α_i of the i th rule is calculated as follows:

$$\alpha_i = \begin{cases} \mu_{A_1^i}(x_1) \wedge \mu_{A_2^i}(x_2) \wedge \ldots \wedge \mu_{A_n^i}(x_n) \\ \text{or} \qquad\qquad , i = 1, 2, \ldots, m, \\ \mu_{A_1^i}(x_1) \cdot \mu_{A_2^i}(x_2) \cdot \ldots \cdot \mu_{A_n^i}(x_n) \end{cases} \tag{2.20}$$

where $\wedge$ refers to the fuzzy intersection operator, for which the Max-Min approach is often utilized, i.e., $\alpha_i = \min\left\{\mu_{A_1^i}(x_1), \mu_{A_2^i}(x_2), \ldots, \mu_{A_n^i}(x_n)\right\}$, and $\cdot$ denotes the product operator. However, in most Sugeno neuro-fuzzy systems, the product approach is favored because it often shows higher accuracy. Therefore, the output of the consequent part of the system is obtained by calculating the weighted average of its antecedent output:

$$y = \frac{\sum_{i=1}^m \alpha_i f_i}{\sum_{i=1}^m \alpha_i} = \sum_{i=1}^m \bar{\alpha}_i f_i. \tag{2.21}$$

In addition, for a multi-output Sugeno system with r outputs, each individual output can also be obtained in the following form:

$$y_h = \sum_{i=1}^m \bar{\alpha}_i f_{ih}, h = 1, 2, \ldots, r, \tag{2.22}$$

where $\bar{\alpha}_i = \frac{\alpha_i}{\sum_{i=1}^{m} \alpha_i}$.

It is worth noting that although the Sugeno engine is more common in neuro-fuzzy networks, the specific design of a neuro-fuzzy system is not set in stone in terms of which fuzzy inference engine to choose for reasoning. One issue that can never be avoided in fuzzy neural system construction is the trade-off between interpretability and accuracy. In general, it is difficult to reconcile the two in the same model because human logic is vague in its essence, while mathematical language is precise. For a high level of semantic interpretability, the logic engine has to cater to the human way of thinking in system design, which results in a loss of accuracy. Conversely, if the model is designed to achieve a high degree of accuracy, it will need to move away from the human mindset and closer to mathematical logic, which usually leads to a decrease in interpretability. It is accepted that if a high level of interpretability is required, the modeling will focus on the representation of linguistic logic, and the Mamdani inference engine is recommended. If the purpose of modeling is to obtain better accuracy, there is no doubt that the Sugeno inference engine is the most appropriate.

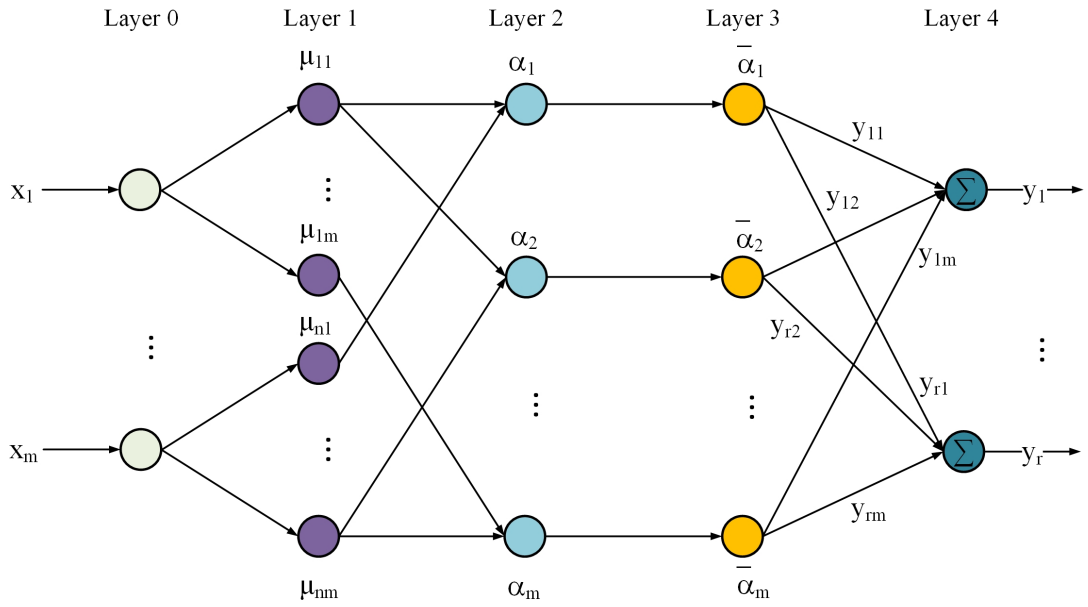

Figure 2.10: The architecture of a multi-input-multi-output Sugeno neuro-fuzzy model.

2.4.5 ANFIS

Traditionally, the establishment of a fuzzy system relies heavily on the experience and knowledge of experts, but for systems with unknown characteristics or intricate mechanisms, human interference seems to lack of edge. In this context, neuro-fuzzy systems with self-learning competence are favored to meet the need for accurate modeling. Of all the designs that have emerged, one deep model has been the most successful, i.e., the adaptive neuro-fuzzy inference system (ANFIS) proposed by Jang [63].

This architecture executes all three basic processes of fuzzy inferencing, i.e., fuzzification, fuzzy reasoning, and defuzzification, in the form of a neural network. It utilizes learning mechanisms in the neural network to adaptively extract rules from input and output data to form an adaptive neuro-fuzzy approximator, i.e., both the antecedent and the subsequent statements are adjusted by adjusting the weights of adaptive nodes during the training process. ANFIS also inherits the interpretability characteristic of the fuzzy inference system, which enables the adjustment of system parameters according to prior knowledge. To achieve better accuracy, ANFIS usually employs the Sugeno model as the fuzzy inference engine. It is worth noting that this structure takes a hybrid of backpropagation and the least square estimator to optimize the network parameters.

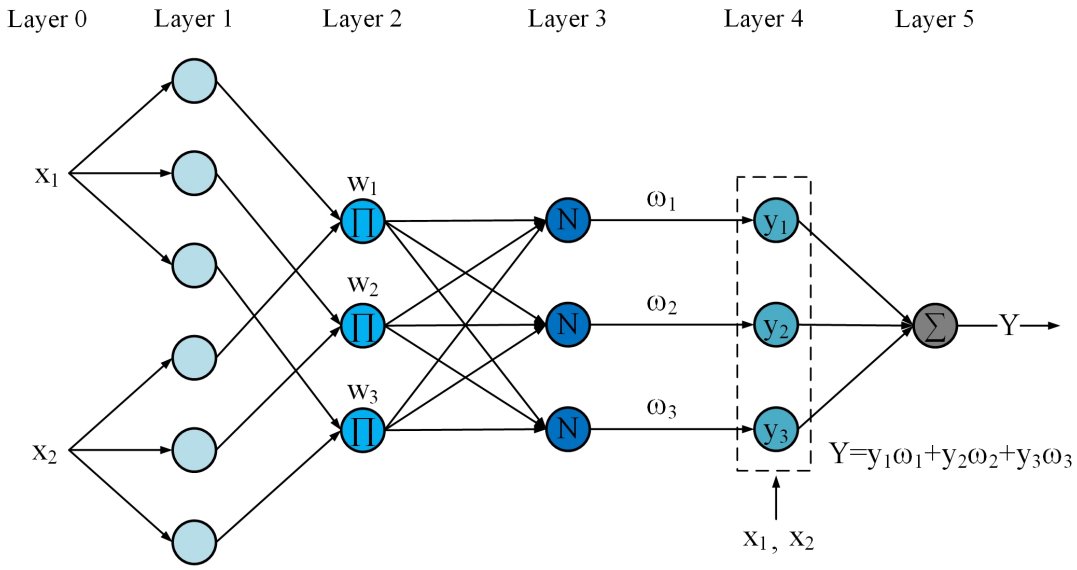

Figure 2.11: ANFIS network with two inputs and single output.

From the perspective of a neural network, the typical ANFIS neuro-fuzzy system is a fivelayer architecture. The first layer is the fuzzification layer, through which the input data is converted into fuzzy membership values by fuzzy membership functions. For an ANFIS model with n inputs and m rules, the membership value of the j th input of the i th rule can be denoted as follows:

$$O_{i,j}^1(x) = \mu_{A_j^i}(x), i = 1,2,\ldots,m; j = 1,2,\ldots,n, \tag{2.23}$$

where $\mu_A(x)$ refers to the membership function. There are many options for the membership functions, such as the Gaussian function, the Polynomial function, the triangular function, and the Cauchy function, among which the Gaussian function is recommended in most cases. The second layer is used to calculate the firing strength of each rule, using the product method to calculate:

$$O_i^2(x) = \prod_{j=1}^{n} O_{i,j}^1(x). \tag{2.24}$$

The third layer is a normalization layer, where the ratio of the firing strength of the ith rule to the sum of the firing strength of all fuzzy rules is calculated:

$$O_i^3(x) = \frac{o_i^2(x)}{\sum_{i=1}^{m} o_i^2(x)}. \tag{2.25}$$

The output of the third layer is also considered the result of the antecedent. By combining the inference result of the consequent, the defuzzified output of each rule is generated in the fourth layer:

$$O_i^4(x) = O_i^3(x) * \left(p_0^i + \sum_{j=1}^{n} p_j^i x_j(x) \right). \tag{2.26}$$

The sum of all inference results is computed at the fifth layer, which is the final output of the network:

$$O_5(x) = \sum_{i=1}^{m} O_i^4(x). \tag{2.27}$$

Deep neuro-fuzzy models represented by ANFIS are equipped with both transparencies from fuzzy systems and strong self-learning competence of the deep neural network, which makes them salient subjects of computational intelligence in recent years. Especially for complex system modeling, deep neuro-fuzzy systems have spectacular advantages over traditional nonlinear modeling methods. Compared with the most fashionable deep learning architectures today, deep neural-fuzzy systems still have the potential to achieve performance that is not inferior to very sophisticated and deep neural networks, despite more compact structures as well as lower computational performance requirements. The shortcoming of ANFIS is mainly due to the harsh selection of optimization methods and the propensity to overfit data. To address these problems, researchers have used interval type- 2 fuzzy logic as inference logic in conjunction with derivativefree optimization methods that significantly improve the algorithm performance.

2.5 Complex Fuzzy Sets and Theories

The evolution of fuzzy theory has never stopped during the past decades. The complex fuzzy set theory is said to be the most significant theoretical breakthrough after the interval type-2 fuzzy theory [16]. Yazdanbakhsh and Dick [16] have conducted a detailed review of different variants of complex fuzzy sets and summarized five categories according to the difference in the selection of common domains. Those common domains include the unit circle, the unit square, the positive quarter of a unit circle, any subset of the complex plane, and a subset of the cross-product within the unit circle. Dick [14] also classified the existing complex fuzzy sets into two factions, i.e., complex fuzzy sets with rotational invariance and complex fuzzy sets without rotational invariance, in which the former only represents the degree of membership by the modulus information, while the latter one is more like the vector logic.

2.5.1 Ramot's Complex Fuzzy Sets

Although similar concepts are mentioned in the work of Kaufman [64], Moses [65], and Nguyen [66] years ago, Ramot [12], [13] first proposed the concept of complex fuzzy sets in the true sense. The core idea is that the range of the fuzzy membership value is extended from the interval $[0,1]$ of the type-1 fuzzy set to the unit circle in the complex plane. Further, the magnitude component of the complex membership function is specified to indicate the degree of membership, and the phase component is defined to be a supplement to membership or contextual information. The definition of the membership degree is as below:

$$\mu_S(x) = r_S(x)e^{j\omega_S(x)}. \tag{2.28}$$

Ramot et al. also defined some basic operators for this new fuzzy set, such as complement, union, intersection, and aggregation, as well as the generation and composition of fuzzy relations. However, the unique nature of this novel fuzzy set caused it to encounter obstacles when defining operators. The first one is how to determine the interactions for phase components. According to the definition, the phase quantity is unambiguous, which means it does not participate in the fuzzy operations. Their solution is to propose two new operators specifically for phase, rotation, and reflection, while suggesting that researchers find operators suitable for specific solutions according to actual problems. Although it is common in the territory of fuzzy systems to select different operators for fuzzy sets accordingly, such an explanation of logic operators for complex fuzzy sets is confusing since there is no theoretical proof to verify the rationality of these operators.

The related fuzzy inference system is also mentioned by Ramot et al. in the paper [13], in which the author uses the Cartesian product to generate fuzzy rules by imitating the classical fuzzy theory, applying the conventional fuzzy composition and implication operators to perform reasoning. It is worth mentioning that when it comes to the aggregation of fuzzy rules, the author emphasizes the use of vector addition so that both the modulus and phase can have a say in the overall result. The introduction

of the vector operation is also considered the most important property of complex fuzzy logic because it is conducive to representing intricate semantics that conventional fuzzy logic cannot tackle. Ramot's definition of complex fuzzy sets has many flaws. It not only lacks rigorous theoretical proof but also does not have a convincing explanation for the phase component in membership, not to mention that the definition of many vital properties relies on intuitive examples rather than mathematical arguments. Such problems have caused disputes for future investigations. Despite this, it is fair to say that as the beginning of complex fuzzy theory, its contribution to later research is immeasurable.

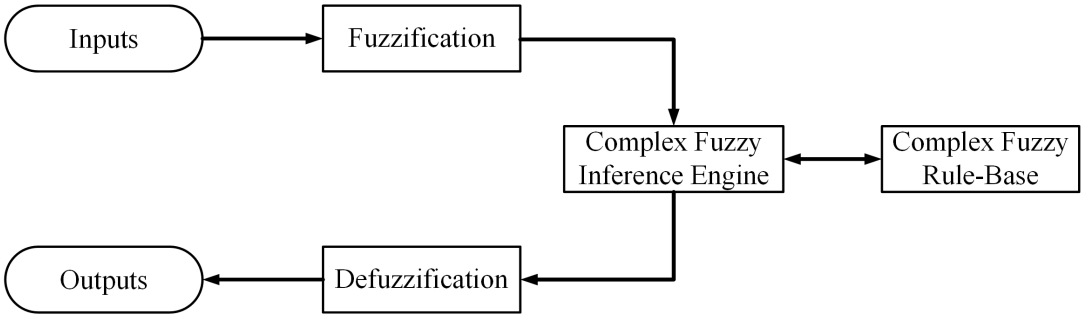

Figure 2.12: The simple illustration of a complex fuzzy logic system. [13]

2.5.2 Pure Complex Fuzzy Sets

Inspired by Ramot's complex fuzzy sets, Tamir [67], [68] also proposed his version of the complex fuzzy theory, called the pure complex fuzzy set. Dissimilar to Ramot's path to define complex fuzzy logic, Tamir's concept is derived from the mathematical axioms of multi-valued logic. This pure complex fuzzy set is defined in the Cartesian form of complex representation:

$$\mu(V,z) = \mu_r(V) + j\mu_i(z), \tag{2.29}$$

where both $\mu_r(V)$ and $\mu_i(z)$ are the real fuzzy membership grade within the interval [0,1], which also indicates that the common domain of this fuzzy set is a unit square located in the first quartile of the complex plane. Specifically, Tamir has also proposed the notion of complex fuzzy classes for this version of complex fuzzy logic. For this reason, $\mu(V,z)$ back in (2.29) represents a fuzzy class which can also be considered as the fuzzy set of a lower order fuzzy set, while $\mu_r(V)$ signifies the membership of V in this

inferior order fuzzy class and $\mu_i(z)$ denotes the membership grade z in V. Compared with Ramot's attempt, this fuzzy theory discusses the definition from the perspective of mathematical axioms, which leads to higher theoretical completeness. However, this idea has not been adopted by any fuzzy inference system at the current stage because the definition of fuzzy classes complicates the semantic interpretation.

2.5.3 Complex Atanassov's Intuitionistic Fuzzy Sets

Alkouri et al. [69] developed the notion of complex Atanassov's intuitionistic fuzzy set (CAIFS) inspired by Atanassov's definition of the intuitionistic fuzzy set (IFS) [70]. The difference between IFS and conventional fuzzy sets is that the concept of non-membership is added. Assume there is a fuzzy set of elements I of the universe of discourse A which is designated by membership function $\mu_I(x)$ valued between 0 and 1. The non-membership function $v_I(x)$ that takes values within the interval $[0,1]$ is defined as well. Then an intuitionistic fuzzy set I is represented in the form below:

$$I = \{[x, \mu_I(x), v_I(x)] : x \in A\}, \tag{2.30}$$

where both the membership and non-membership grade $\mu_I(x), v_I(x) \in [0,1]$ stands for any arbitrary $x \in A$ to the set I. Most importantly, an inequality is specified in the definition of the set, which is $0 \leq \mu_I(x) + v_I(x) \leq 1$. IFS also brings a very special complement operator, which makes the complement operation a direct value swap between the degree of membership and nonmembership, i.e., $\bar{I} = \{[x, v_I(x), \mu_I(x)] : x \in A\}$. And this also leads to a very interesting set of intersection and union operations. To realize a union of IFS, one only needs to find out the maximum membership value as well as the minimum non-membership value among sets, while for the intersection operation the outcome is composed of a minimum membership value and a maximum non-membership value over sets.

The core idea of CAIFS is essentially just the replacement of the real membership and nonmembership functions of IFS with complex values. In this situation, the inequality condition that needs to be satisfied becomes $|\mu_I(x) + v_I(x)| \leq 1$ which is virtually a

modulus inequality. Though the operation principles are basically the same as the original IFS, the use of complex values increases the dimensionality of the membership function, which will undoubtedly help express and process richer information. Notably, this CAIFS has been employed for individualobjective or group-objective decision-making [71], [72] and cellular network [73].

2.5.4 Pythagorean Fuzzy Sets

Yager [74], [75] developed the idea of Pythagoras fuzzy set (PFS) inspired by Atanassov's IFS and interval-valued type-2 fuzzy set, which seems to be a splendid multivariate decision-making algorithm in the case of uncertainty. Assume P is a PFS of the support A, then this PFS can be expressed in the following form:

$$P = \{[x, \mu_Y(x), \mu_N(x)] : x \in A\}, \tag{2.31}$$

where $\mu_Y(x)$ is said to be the membership value of x in P and $\mu_N(x)$ is the non-membership value, which is very much like the case in IFS. PFS is very similar to IFS, and their complement, union, and intersection operators are exactly the same. The main difference between the two definitions is that the common domain of PFS is in the quarter of the unit circle sector located in the first quadrant, i.e., $\mu_Y^2(x) + \mu_N^2(x) \leq 1$ and $\mu_Y(x), \mu_N(x) \in [0,1]$. This divergence in PFS enlarges the area that IFS can cover, leading to better performance in solving practical decision-making problems.

Based on PFS, Dick et al. [17] combined the definition with the existing complex fuzzy theory, then extended the co-domain of PFS from the unit sector in the first quadrant to the entire unit circle. Furthermore, Dick also defined the concepts of anti-membership and anti-non-membership for PFS, proving that negation and complementation are different operations in the case of PFS. This attempt makes PFS another logical operator that can be deployed to perform complex fuzzy logic operations and enables the representation of the semantics of negation and antonyms, which is impossible for conventional type-1 fuzzy logic. Therefore, PFS logic deserves further investigation in the future as the most potential version of complex fuzzy theory.

2.5.5 Other Complex Fuzzy Sets

Many other researchers also made valuable explorations in this direction and demonstrated their understanding of CFS. Kumar et al. [76] put forward a complex intuitionistic fuzzy soft set (CIFSS) as an attempt to extend the original intuitionistic fuzzy soft set (IFSS) mentioned by Maji [77]. Thirunavukaras et al. [78] proposed a parameterized complex fuzzy soft set which is an upgrade of its real number counterpart. Li et al. [79], [80] have been investigating the property of sphere complex fuzzy sets, in which the truth-valuation domain is a high-dimensional hypersphere, enabling the modeling of multiple paralleling variables at a time. Ali et al. [81], [82] further expanded the concept of PFS by developing a 3-element membership function composed of membership, non-membership, and indeterminacy, which leads to a neutron-sophic set for decision-making. Liu et al. [83] developed a complex q-rung orthopair fuzzy set for multivariate and multiple-feature group decision-making. Singh [84] created the bipolar δ-equal complex fuzzy set to describe the inconsistency and the completeness of real-world data. Greenfield [85] defined an interval-valued complex fuzzy set by combining the concept of type-2 fuzzy sets. The above definitions of complex fuzzy sets have not been widely discussed and verified and will not be illustrated in detail.

2.6 Fuzzy Inference Systems for Complex Fuzzy Sets

For any fuzzy set theory, the key to application lies in the successful establishment of corresponding fuzzy logic and reasoning system. At present, inference systems for complex fuzzy theory are all neuro-fuzzy systems. There are two main reasons held responsible for this fact. First, although nearly 20 years have passed since the first complex fuzzy theory was proposed, this field is still in the early exploratory stage, and the research in all aspects is far from mature, especially when the semantic representations of complex fuzzy logic are yet to unravel. Such theoretical incompleteness prevents researchers from acquiring a comprehensive understanding of the properties of complex fuzzy logic systems, which hinders attempts to build such a system. Another significant reason is that the two-dimensional attribute of the complex membership function

of CFS causes intricate semantic interpretations. Even if the semantic representation of CFS is fully deciphered, manually building a fuzzy rule base for CFS logic systems through expert knowledge is still seriously counterintuitive. Therefore, it is not difficult to understand why the neuro-fuzzy system turned out to be such an attractive solution here. In the current stage, several neuro-fuzzy systems based on CFS have emerged. It is worth noting that existing complex neuro-fuzzy systems are all established based on Ramot's definition of complex-fuzzy sets, and there have been no attempts so far to construct neuro-fuzzy systems using other CFSs.

2.6.1 ANCFIS

In 2007, Man et al. [86] replaced type-1 fuzzy logic with complex fuzzy logic in a single input ANFIS architecture and created a six-layer neuro-fuzzy system, known as the earliest complex fuzzy inference architecture, which sparked the discussion over adaptive neuro complex fuzzy inference system (ANCFIS). Later, Chen et al. [87] further improved this concept based on the previous network model in [86], leading to a relatively complete ANCFIS framework. This network is relatively simple compared with various deep network structures commonly applied, in which the whole design is very similar to the traditional ANFIS, as the same TSK method is used for fuzzy inference. Since the architecture is defined to solve the prediction of quasi-periodic problems, a sinusoidal fuzzy membership function is deployed for this purpose, as shown below:

$$r(\theta) = d\sin(a\theta + b) + c, \tag{2.32}$$

where the parameters a, b, c, d are said to be premise parameters.

The first layer of the ANCFIS is to measure the difference between the membership function and the input vector. For this purpose, three measurement operations are recommended, including Euclidean distance calculated through the L-2 norm, convolution, and Elliot function. Among the three alternatives, the Elliot function is considered

the best option, which is given as follows:

$$\frac{z}{1+|z|}, \tag{2.33}$$

where $z = \sum_{k=1}^{n} x_k * [r_i(\theta_k)\cos(\theta_k) + jr_i(\theta_k)\sin(\theta_k)]$, $\theta_k = k\frac{2\pi}{n}$ and n represents the number of elements inside the input vector x. The second layer is designed to determine the firing strength by multiplying the signals from the preceding, then output the resulting product to the next layer. The third layer is a procedure to normalize the outcomes of the previous step to prevent the potential exceeding of tolerance. The purpose of the fourth layer is to reflect the contribution of the phase component in the membership function for the CFS. Consequent parameters are generated in the fifth layer through the least square method. Finally, all the fuzzy inference outputs from different rules are aggregated via the weighted sum in the sixth layer, which is also the overall output of this network. The original network employs the vanilla gradient descent method for backpropagation to train the parameters.

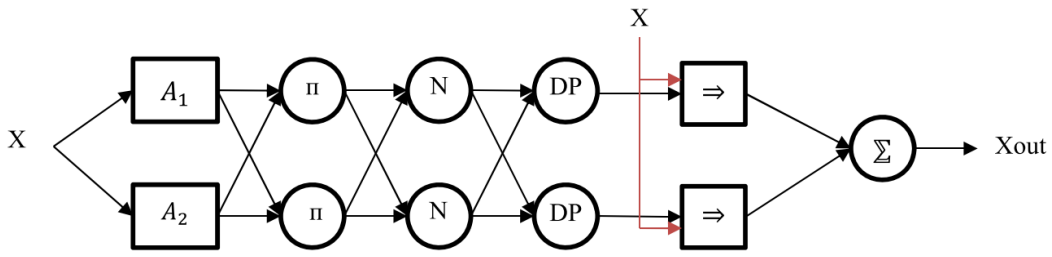

Figure 2.13: The architecture of a single-input two-rule ANCFIS [21].

Yazdanbakhsh et al. further investigated this architecture and introduced it into several application scenarios, including solar power prediction [88] and time-series prediction [89]. They also implemented modifications such as combining ANCFIS with an extreme learning machine [90]; altering ANCFIS for multi-input-multi-output (MIMO) tasks [91] and creating a fast adaptive version of ANCFIS which is named FANCFIS [21]. Yeganejou et al. even developed a classifier based on ANCFIS for condition monitoring [92]. However, because of the limitation of the sinusoidal membership function, such an algorithm is not suited to general function approximation problems. Therefore, there are not many options for application scenarios that ANCFIS can employ, which

hinders its application for diverse intents and purposes. At this stage, the main application for this algorithm is to model and forecast objects with regularity, such as the trend of the stock market or the activity of sunspots, while another scenario is streaming data processing, such as data mining.

2.6.2 CNFS

Analogous to ANCFIS, Chen et al. [93], [94] also developed a neuro-fuzzy system using complex fuzzy logic inspired by ANFIS and CFS, which is named complex neuro-fuzzy self-learning (CNFS) approach. In terms of the reasoning method, such an algorithm still employs the TSK fuzzy inference system due to its relatively better accuracy against its Mamdani counterpart and lower computational complexity. The network is a five-layer structure, but unlike ANCFIS, this architecture utilizes a complex Gaussian membership function instead of a sinusoidal one, which is shown below:

$$\mu\,\text{Gaussian}(x,m,\sigma)=\exp\left[-0.5\left(\frac{x-m}{\sigma}\right)^2\right]-j\left(\frac{x-m}{\sigma^2}\right)\exp\left[-0.5\left(\frac{x-m}{\sigma}\right)^2\right], \quad (2.34)$$

where x is said to be the base variable, m is the mean of a gaussian function while σ is the spread. Thus, the parameter set for this membership function is $\{m,\sigma\}$, which is also the premise parameter set for a node. Assume the objective network has k inputs and one output, then the i th fuzzy rule can be represented as follows:

$$\textit{Rule } i:\ \textit{IF } l_1 \textit{ is } A_1^i(x_1) \textit{ and } l_2 \textit{ is } A_2^i(x_2) \ldots \textit{ and } l_k \textit{ is } A_k^i(x_k).$$

According to the defuzzification strategy of the Sugeno fuzzy system, the output z^i of this rule is:

$$z^i = a_0^i + \sum_{j=1}^{k} a_j^i x_j, \quad (2.35)$$

where $A_j^i(x_j)$ is the j th antecedent of the i th fuzzy rule, a_j^i is the corresponding consequent parameter for $i = 1,2,\ldots,n$. For the objective CNFS, the complex fuzzy inference system is cast into a five-layered network. A sketch demonstration of the structure of the entire CNFS network is shown in the figure below:

Layer 0 is employed as the input layer, of which the function is to receive the input signals and further transmit them to the following layer without any operation. Layer 1 is said to be the fuzzification layer, in which the inputs are fuzzified via the Gaussian membership function mentioned above. Layer 2 is designed to determine the firing strengths for each fuzzy rule, and in this case, complex fuzzy intersection operators, also called t-norms, are applied. Layer 3 is a conventional normalization step where the outcome of the preceding step is mapped into the interval $[0, 1]$. Inside Layer 4, the normalized firing strength is multiplied by the consequent obtained via (2.35) to generate a normalized consequent. Ultimately all the results are aggregated in Layer 5 by operating algebraic sum. Note that the final output at this stage is complex-valued, which is erroneous for real-valued function approximation problems. Therefore, according to the properties of the complex Gaussian membership function in polynomial form, the real part of the output, i.e., the projection of the complex fuzzy set on the real axis, is output as the final defuzzification result.

With the help of the PSO-RLSE hybrid optimization algorithm that applies the PSO algorithm to optimize the antecedent parameters and the RLSE algorithm to optimize subsequent parameters, the network can achieve excellent nonlinear approximation performance. It has been confirmed that the CNFS architecture has greater effectiveness in adaptive capability than its type-1 fuzzy logic counterpart [95]. Owing to the two-dimensional rule-base of complex fuzzy sets that allows a single rule to involve more information, though only very few fuzzy rules are employed in the network, CNFS still shows better regression performance than most similar architectures that only apply type-1 fuzzy logic. At present, this design has been successfully used in several application scenarios, such as image restoration [96], image noise canceling [97], knowledge discovery [98], time series forecasting [95], ARIMA forecasting [99], and multi-class prediction [100]. The main problem with this algorithm is that the derivative-free optimization method employed often increases the computational complexity by at least one order of magnitude compared to the gradient method, making its training process less efficient.

2.6.3 ACNFIS

Shoorangiz et al. [101] also proposed an adaptive complex neuro-fuzzy inference system (ACNFIS) to target function approximation purposes. They introduced a polar form complex Gaussian membership directly inspired by the intersection operator from Ramot's complex fuzzy theory. This function is given as follows:

$$\mu(\theta) = \exp\left[-\left(\frac{\theta - c_A}{a_A}\right)^2\right] \angle \left\{2\pi \exp\left[-\left(\frac{\theta - c_P}{a_P}\right)^2\right]\right\}, \tag{2.36}$$

where $\{c_A, a_A\}$ and $\{c_P, a_P\}$ are parameter sets for amplitude term and phase term respectively. Unfortunately, this study has not been tested in practice, likely due to the lack of simplicity as well as effective defuzzification methods for this polar form function compared to the previous two cases. Therefore, a detailed discussion is not provided here.

2.7 Complex-Valued Structures for Real-Valued Approximation

Apart from the above three algorithms, some researchers have also proposed neuro-fuzzy systems using complex-valued fuzzy membership functions by introducing the notion of CVNN [18] (Complex-valued neural networks) to the fuzzy inference system. CVNN is a kind of deep network architecture in which both the input and output values are complex numbers. Traditionally, such network structures are used to handle problems that involve complex-valued elements in the data to facilitate signal processing of the network. In recent years, however, some researchers have proposed that similar designs are also conducive to tackling pure real-valued problems [18]. One situation is that some connections are presented between the real-valued variables of the data and combining them into the form of complex numbers for input to the network can improve the performance. Alternatively, it can achieve a certain degree of dimensionality reduction by combining real-valued variables into complex numbers when dealing with high-dimensional datasets.

The first of the preceding cases is easier to explain since the independent real-valued

variables in some datasets may actually have an underlying complex-number-defined connection, such as the relationship between resistance and capacitance in a circuit, both of which can affect the voltage and current of the circuit and thus determine the operating power. However, the impact of the two on electrical energy is very different. Resistance converts electrical energy into internal energy, while capacitive reactance only stores electrical energy, which means that their effects on output power are not in the same dimension. In this case, it is more advantageous to combine both into a complex-valued form, i.e., impedance, as the input of the machine learning algorithm than simply using two real-valued variables. There are many similar examples, and as a matter of fact, real number physics describes linear motions while complex number physics describes periodic behaviors, suggesting all physical quantities in the periodicity pattern are subject to a potential complex-number form. Using the CVNN architecture on such problems is more reflective of their nature, resulting in better performance.

In the second case, complex numbers are applied for dimensionality reduction and it may be necessary to combine two unrelated real-valued variables into a complex-valued input, which seems to lack plausibility. Despite this, some scholars investigated this idea and pointed out that for machine learning algorithms, the ramification of combining real-valued variables into complex-valued ones only means a reduction of the degree of freedom to a certain extent [18]. Based on this conclusion, it is feasible to approximate real-valued inputs with fewer complex-valued inputs, except that accuracy may suffer from a trade-off due to the loss of the degrees of freedom [18]. However, for scenarios with high dimensionality, as the curse of dimensionality itself significantly reduces model performance and weakens the nonlinear correlation between variables, the trade-off of taking such a design may be shifted. This design makes sense if the loss caused by the curse of dimensionality is greater than the performance loss caused by using complex values to approximate the real-valued problem. This narrative is especially true for the neuro-fuzzy system, because the scale of its rule-base is often exponential to the input dimension, and the use of CVNN structures can reduce the rule-base to the square root size of its original counterpart, which may lead to a very significant efficiency gain.

Therefore, from the perspective of dimensionality reduction and combating the curse of dimensionality, the introduction of complex numbers in neural network structures can significantly reduce the size of the network, making it a more streamlined structure. On the basis of CVNN, if operation laws of fuzzy logic are added to the network, it becomes a complex-valued neuro-fuzzy system. Currently, two network structures using this concept have emerged, which are CNFIS [19] by Subramanian et al., and CVNF [20] by Ryusuke et al. It is worth mentioning that although such architectures contain terms like "complex" and "fuzzy" in the name, of the essence they cannot be considered veritable neuro-fuzzy inference systems that apply complex fuzzy logic. The trick is that in CVNN-like neuro-fuzzy architectures, the complex-valued structure is just the form to help build a more compact network, whereas the fuzzy logic applied is still subject to type-1 fuzzy theory. The real component and the imaginary component of complex-valued inputs only have interactions at the signal processing level without involving deeper logic, which is very different from the cases involving authentic complex fuzzy logic. Consequently, this category of networks cannot take full advantage of the complex-valued membership over some applications with periodicity, which brings about a relatively limited performance improvement. However, when it comes to applications that are not very demanding in terms of accuracy but have huge dimensionality, two real-valued inputs can be approximated by one complex-valued input to reduce the system dimensionality and dramatically increase the efficiency of the algorithm.

2.8 Chapter Summary

Fuzzy inference logic overturns the traditional way of analyzing problems in crisp mathematical logic, creating a new type of infinite-valued logic in addition to probabilistic methods, which enables the mathematical modeling of imprecise objects. Fuzzy algorithms can be designed along the lines of conventional mathematical methodologies to obtain the best performance, or they can perform in the way humans understand the world to establish a model that human semantics can interpret, which corresponds to two differentiated fuzzy inference engines, i.e., the Sugeno engine and the Mamdani engine. The former handles fuzzy inference in a way similar to the weighted method in crisp mathematics, obtaining models with high accuracy but weak semantic interpretability. The latter retains the complete fuzzy semantic inference process and therefore possesses better interpretability, but its accuracy is relatively inferior with a higher model complexity due to the intrinsic dissimilarity between human semantics and exact mathematical language.

The neuro-fuzzy system is a fuzzy system with self-learning capability constructed by combining the fuzzy inference engine with the architecture of a feed-forward neural network. Functionally, neuro-fuzzy systems are similar to RBF interpolation networks, both of which are local approximation networks. In recent years, neuro-fuzzy systems have become the mainstream method because traditional manual modeling methods are increasingly unable to deal with the growing complexity of applications. Deep neuro-fuzzy systems, represented by ANFIS, are favored today when deep learning methods are prevalent. The advantage of such models is that it is more transparent than traditional deep learning models because fuzzy logic endows physical meaning to each part of the architecture. Increasing model depth improves performance but also weakens interpretability, and application scenarios for deep neuro-fuzzy systems are not that different compared to general deep learning models, which determines that the priority to improve performance potentiality in this area outweighs the interpretability. Even the interpretability is mainly utilized to serve the model performance, such as parameter tuning and model adjustment, rather than extracting expert knowledge from the model.

The practice of deep neuro-fuzzy systems has also increasingly exposed the shortcomings of the traditional type-1 fuzzy logic and stimulated the research of new types of fuzzy logic, for which the two most mentioned fuzzy logic theories are type-2 fuzzy logic and complex fuzzy logic. The type-2 logic system enables both the intra-individual uncertainty and the inter-individual uncertainty in a single rule. Currently, the interval type-2 fuzzy logic is commonly applied to avoid the unnecessary complexity of the generalized type-2 logic. However, the gradient optimization policy is not available for an interval type-2 model due to the lack of a closed-form solution for the first-order derivative of its membership function, which poses a challenge to the optimization of the model. The complex fuzzy set theory extends the co-domain of type-1 logic from the unit interval to the entire unit circle of the complex plane. Complex fuzzy theory is a profound breakthrough from a mathematical point of view. It extends the reasoning logically from one dimension to two dimensions and introduces abstract algebra into the discussion of fuzzy logic. Unfortunately, due to the counterintuitive nature of two-dimensional logic, the semantic interpretation of the complex fuzzy theory is not yet clear, and there is no academic consensus even though many scholars have offered various insights. This issue hinders the use of this logic for manual modeling but does not prevent its design as an adaptive neuro-fuzzy system. Considering that complex fuzzy membership functions generally have closed-form first-order derivatives, the optimization designs for such models are more convenient than their type-2 counterparts, which is also an advantage of applying complex fuzzy theory to neuro-fuzzy systems.

Three deep neuro-fuzzy systems using such logic have emerged, namely ANCFIS, CNFS, and ACNFIS, corresponding to three different complex fuzzy membership functions, i.e., the sinusoidal function, the polynomial complex Gaussian function, and the polar complex Gaussian function. Nevertheless, current approaches suffer from multifaceted imperfections and do not fully exploit the potential of complex fuzzy logic, leaving room for subsequent research. It is worth noting that another class of deep neuro-fuzzy models convert real-valued inputs into fewer complex-valued inputs to approximate real-valued problems, but they are not consistent with complex neuro-fuzzy models. The key is whether complex fuzzy logic plays the role of inference logic in the

model. In a purely complex-valued neuro-fuzzy system, the complex-valued structure is applied only at the signal processing aspect, without any involvement of complex fuzzy logic. Although having no access to the benefits of complex fuzzy theory, such systems can still show advantages for problems where complex-number-defined relations are presented between some real-valued variables. Further, such designs also have the potential for dimensionality reduction to counter the curse of dimensionality in neuro-fuzzy systems, which is extremely meaningful for high-dimension scenarios. All the above arguments will be discussed and analyzed in later chapters. Novel algorithms based on such ideas will also be illustrated and tested to validate the conclusions.

3

Real-World Regression Dataset for Model Validation

> *Be extremely subtle even to the point of formlessness. Be extremely mysterious even to the point of soundlessness. Thereby you can be the director of the opponent's fate.*
>
> **– Sun Tzu –**

Three real-world numerical regression datasets are expounded in this chapter, including a Sunspot time series dataset and two high-dimensional metallurgy datasets for function approximation tests. All these datasets are adopted in subsequent sections of this thesis to validate the proposed model.

3.1 Sunspot Time Series Data

Starspot activities are the most patently visible events of stellar magnetic field concentrations and their interaction with the energetic plasma, usually observed as structures that appear dark on the stellar surface [102]. Although the detailed mechanism of the starspot occurrence is still a matter of underway research, it is widely accepted that they are the reflections of magnetic flux tubes in the convective zone of the star casting through the photosphere within active regions. Such a strong magnetic field inhibits convection in the photosphere leading to the weakening of its characteristics. Consequently, the energy flux from the stellar body interior diminishes, which lowers the surface temperature, causing the exterior area through which the magnetic field covers to look darker against the bright surroundings of photospheric granules. The entire process can take weeks or even months, with the spots shrinking and expanding as they move across the stellar surface. Starspots normally do not appear in an isolated manner, but in groups. Such activities are very common during the lifetime of most stars and vary widely in scale, in some cases up to 30% of their surface area [103]. Starspot occurrence in stars is as frequent as seismicity on terrestrial planets, but unlike quakes, which are highly unpredictable, Starspots are mostly regular with a relatively stable cyclicality. Because of these properties, the starspot phenomenon can be easily observed by humans. Even for star systems far away from the solar system, although the current technology is not advanced enough to monitor the condition of their orbiting planets, there is no barrier to detecting their starspots. An example of the starspot observed on the star HD 12545 is shown in Figure 3.1.

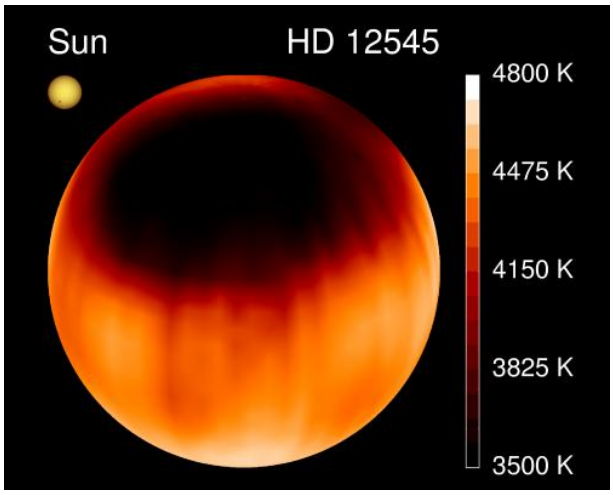

Figure 3.1: Starspot observed on the star HD 12545 (https://apod.nasa.gov/apod/ap000712.html).

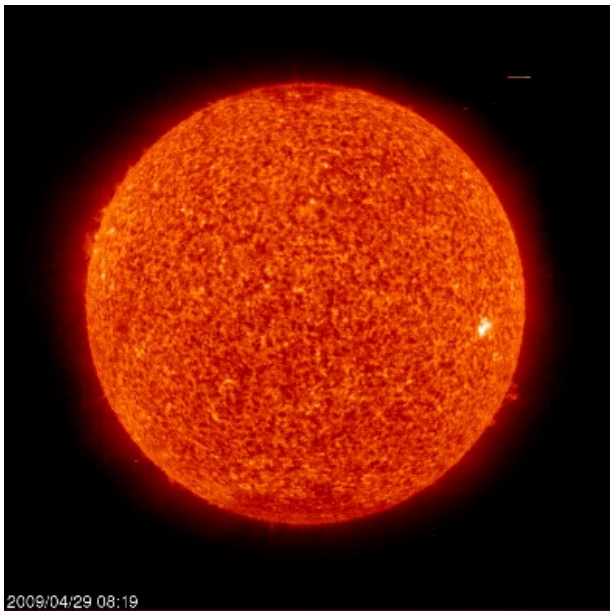

Figure 3.2: Sunspot observed under the ultraviolet spectrum (https://earthobservatory.nasa.gov/images/38380/sunspots-on-april-29-2009).

Similar events on the Sun are called sunspots, for which the earliest record can date back to 800 BC in ancient China [104]. The incipient mention of this phenomenon in Western literature is around 300 BC, by the ancient Greek philosopher Theophrastus, a student of Plato and Aristotle and successor to the latter [105]. It is generally believed that the cycle of sunspot activity is about 11.2 years, and its occurrence is not uniform, with distinct peaks and valleys. Sunspots and the accompanying corona activity also have certain irregularities. For instance, from 1900 to 1960, the number of sunspots observed during the peaks was on the rise, but after the 1960s it became a downturn, of which the pattern is quite similar to the situation 8000 years ago [106]. Some researchers believe that some factors may be hidden behind these irregularities, leading to the difficulty in accurately predicting the intensity of the next peak. However, such a feature also makes the phenomenon a perfect subject for machine learning algorithms, not the least of which the rich historical record also provides ample training and validation data samples. Figure 3.2 displays the sunspot activity under the ultraviolet spectrum.

The Sunspot time-series dataset [107] is introduced to validate the performances of various machine learning models. It includes sunspot activity since 1700 and is available from the website of Sunspot Index and Long-term Solar Observations (SILSO). It is fair to say that the dataset is of very high quality in terms of accuracy of observations and amount of data. The daily, monthly, and monthly smoothed sunspot numbers for the last 13 years are given in Figure 3.3, from which it is evident that despite its overall periodicity, the motion of sunspots has a significant fluctuation within a period measured in days. If observed on a monthly basis, a more cyclical nature is presented, which can be reflected in Figure 3.4, where the historical situation of the monthly mean and 13-month smoothed sunspot activities over the recent several decades are compared. If observed on an annual basis, the fluctuations are almost negligible, and the whole process shows a strong regularity which is displayed in Figure 3.5, in which the yearly mean and 13-month smoothed sunspot records since the year 1700 are provided.

For regression prediction tasks of time-series data, it is necessary to use several previous data points as input quantities to predict current or future states. In this paper, sunspot observations of the three-day basis are used in experiments in the form of dual-

input single-output data pair, i.e., $\{x(\tau-2), x(\tau-1); x(\tau)\}$, where $x(\tau)$ denotes the current observation, and $x(\tau-2), x(\tau-1)$ represent historical observations two intervals and one interval before, respectively. It is worth noting that the sunspot time series is subject to strict causality, which means that the traditional training set and test set validation method should be applied in the test. The cross-validation approach commonly used in some literature is not considered, given that it may destroy the causality of the time series, resulting in a preposterous situation by using future data to predict the past. Additionally, due to the chaotic nature of sunspot activity in a relatively short period of time, the data used for testing has been smoothed to reduce the interference of these unnecessary uncertainties on the model. The smoothed Sunspot time series is a perfect benchmark dataset for validating the performance of machine learning algorithms. First, it is not too difficult for most of these algorithms, which ensures that the algorithm used for testing converges successfully on this dataset. Secondly, algorithms with different performances can easily show differences in this dataset, mainly in prediction errors, which is very beneficial for benchmarking. Finally, the data have a sufficient sample size to meet the requirements of deep models that often require large amounts of data.

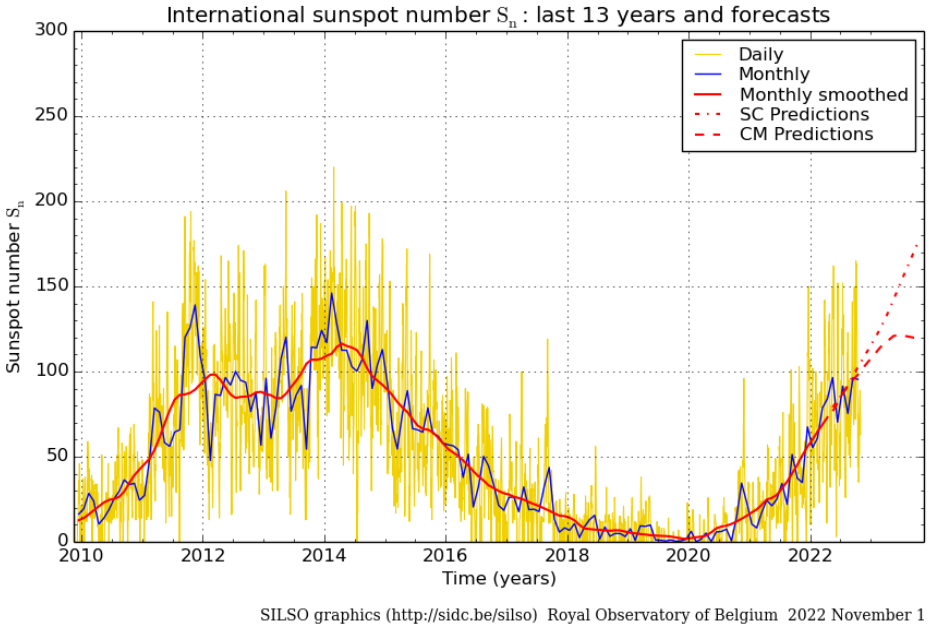

Figure 3.3: Daily and monthly sunspot number (last 13 years) [107]

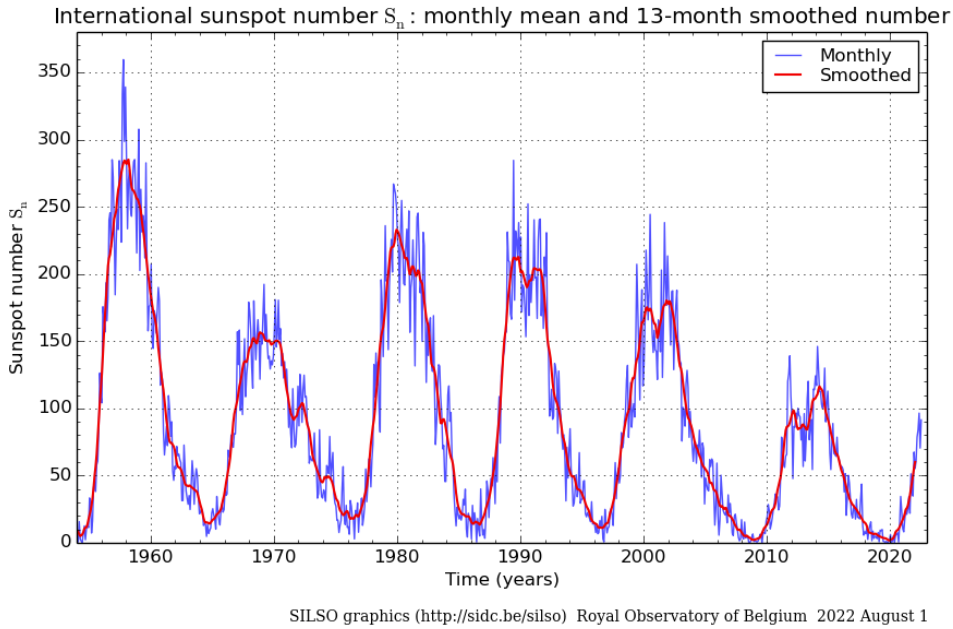

Figure 3.4: Monthly and smoothed sunspot number [107]

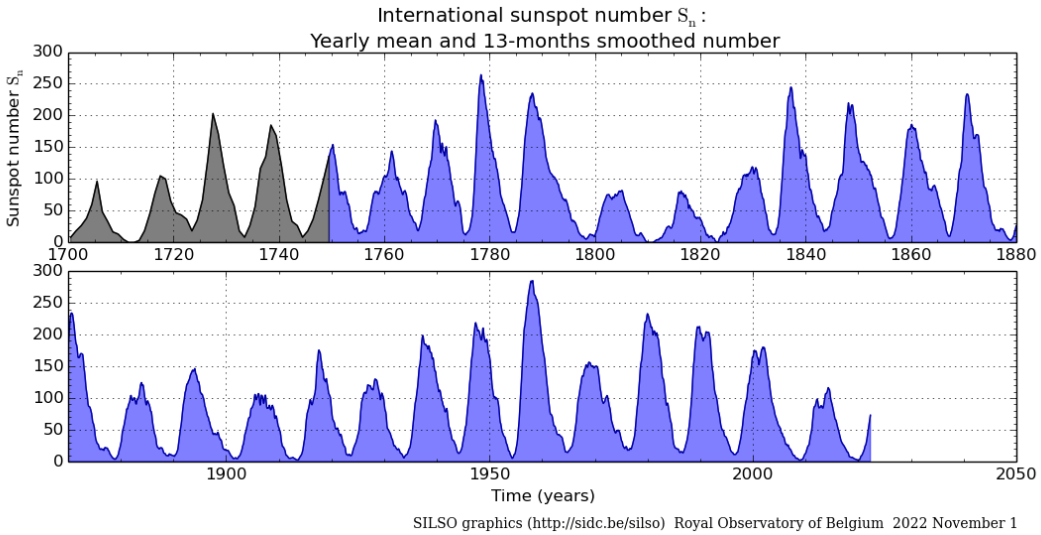

Figure 3.5: Yearly mean and monthly smoothed sunspot number [107].

3.2 Charpy Impact Data for High Strength Steel

Metallurgy is a traditional but vital industrial sector in manufacturing, of which the most crucial product is alloy steel. Metal materials are crystalline substances, where physical and chemical properties depend on the structure and type of atoms that constitute their crystal lattice [108]. Since all this happens on the level of atomic size, even subtle changes in alloy composition may dramatically impact its attributes, indicating that the influence of the formula and processes on characteristics is highly non-linear. The production of alloy steel usually requires the involvement of a variety of chemical elements and a remarkable amount of heat/cold treatment processes. Sometimes it may also require complex procedures such as repeated pressing. These characteristics determine that the material properties of alloy steel are often subject to numerous variables, and the interaction between variables is highly unpredictable. In addition, all these aspects of data rely on massive experiments to obtain, which are costly and time-consuming and often lead to limited and incomplete data in the industry. The above factors make it a tricky assignment to model such processes using traditional mathematical methods.

The Charpy impact test [109], also known as the Charpy V-notch test, is one of the most widely used traditional mechanical property tests to evaluate the impact toughness of metal materials. Metal materials require sufficient toughness in addition to strength and plasticity to serve the purpose. The so-called toughness is the ability of a material to absorb energy during elastic deformation, plastic deformation, and fracture [110]. The Charpy impact test measures the material toughness under impact loading and multiaxial stress. The test places a sample with a specified shape, size, and notch type on the support of an impact tester and then delivers a one-time impact to the material with a pendulum of a specified height, measuring the work absorbed when the material breaks under this impact [109]. The impact on material generally leads to two kinds of fractures, i.e., the ductile fracture and the brittle fracture, where the ductile fracture suggests higher absorption of impact energy. For materials with various levels of strength, inconsistencies in standards when implementing the test are common. For instance, low-strength materials generally have good toughness and ductility,

such materials have high Charpy impact energy, and there is no excessive requirement for the temperature during testing, which can be controlled within the range of 10-35 degrees Celsius. However, some materials require stringent testing conditions, such as materials whose toughness is sensitive to temperature changes or high-strength materials with low Charpy impact energy. In these cases, the temperature should be controlled at around 20 degrees Celsius, i.e., the standard room temperature. For some special-purpose materials, performance at low temperatures is vital, and the Charpy test of such materials is often carried out at a very low temperature, such as below -20 degrees Celsius [111]. It is worth noting that the results of Charpy impact tests are often discrete and random, which often requires at least three repeated experiments on a material to obtain the typical value that can reflect the toughness.

The Charpy impact dataset applied in this thesis records information on the Charpy impact energy of 830 groups of high-strength alloy steels, each of which differs in the formulation and cold/heat treatment conditions. It is a high-dimensional dataset with 16 input variables and a single output, where the input variables include common alloy compositions such as carbon, silicon, and manganese, as well as heat treatment conditions such as tempering temperature. The basics of the data is given in Table 3.1. Among these input factors, 14 are numerical variables, including the content of chemical elements in the material, the heat treatment parameters, and the set conditions for the Charpy impact test, such as test depth and sample size. The remaining two are categorical variables, where the cooling medium variable contains five categories, and the coded site variable includes three types. Note that such variables do not cover all factors that may affect the Charpy impact energy of the high-strength steel, and even the variables included are not necessarily related to the Charpy impact energy, either. For example, Aluminum content is included in the data, but this element is mainly used in steel to enhance corrosion resistance and has little effect on its ductility and toughness. Instead, some relevant indicators do not appear in the data because they are difficult to measure. All of this makes the dataset very different from the carefully selected ones and very close to the situations in real-world scenarios, where relevant factors are missing or overlooked, but irrelevant factors are mistakenly taken.

Table 3.1: The basics of the Charpy impact data

Numerical Variable	**Mean**	**Median**	**Range**
C	0.3942	0.42	0.13-0.52
Si	0.2548	0.25	0.11-0.38
Mn	0.8409	0.82	0.41-1.75
S	0.0167	0.019	0.0008-0.052
Cr	1.0752	1.08	0.11-3.25
Mo	0.2394	0.23	0.02-0.98
Ni	0.3683	0.2	0.03-4.21
Al	0.027	0.026	0.003-0.047
V	0.0077	0.005	0.001-0.26
Hardening Temp	864.02	864.02	810-980
Tempering Temp	647.19	650	190-730
Impact Temp	-5.7869	-10	-53-23
Test Depth	20.8	12.7	5.5-146.05
Sample Size	172.49	155	11-381
Charpy Energy	89.642	89.333	3.46-245.33
Category symbol			**Categories**
Cooling Medium			5
Coded Site			3

Such data poses significant difficulties for modeling. Apart from the reason mentioned above, the difficulty for modeling is even further due to the sparsity caused by the scatter in measurements and inconsistencies related to samples with similar feature variables but different outputs. The inconsistency comes from missed features, such as grain size and other microscopic indicators, for which the measurement demands expensive high-technical equipment and facilities, and thus it is common for these variables to be ignored in industrial datasets. Moreover, the effect of formulation and treatment processes on alloy steels is highly nonlinear, for example, the influence of carbon element on steel, where increasing the carbon content within a certain range helps increase ductility, but if the content is too high, it turns the steel into pig iron that lacks ductility. Similarly, tempering, a heat treatment procedure that heats the material to a set temperature and then rapidly cools it at a controlled temperature, plays an irreplaceable role in the ductility of steel. However, the effect of different tempering temperatures is also highly nonlinear. The nonlinearity and high dimensionality of the data further complicate accurate modeling, not to mention the presence of human error in the measurement, such as recording errors, or variances caused by differed experimental equipment and procedures, suggesting the presence of outliers and weak noise

in the data. Therefore, current attempts to model such a process manually using mathematical methods have not yet resulted in a high degree of accuracy. Even for machine learning algorithms, the sparsity, curse of dimensionality, and noise brought about by the data characteristics also lead to great difficulty for the method to implement. The data density plot between some variables for the Charpy impact data of high-strength steel is given in Figure 3.6. The data density plot between Charpy energy and some input variables are given in Figure 3.7.

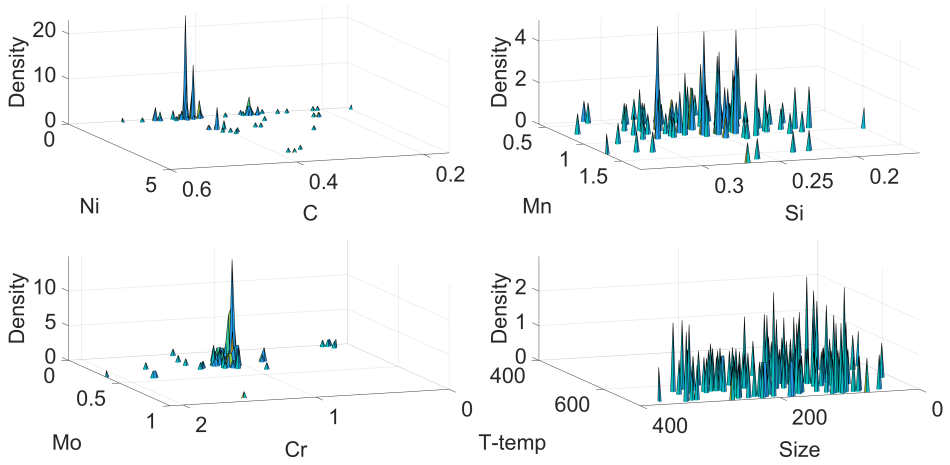

Figure 3.6: The data density plots between "Ni and C", "Mn and Si", "Mo and Cr", and "T-temp and Size" for Charpy impact data.

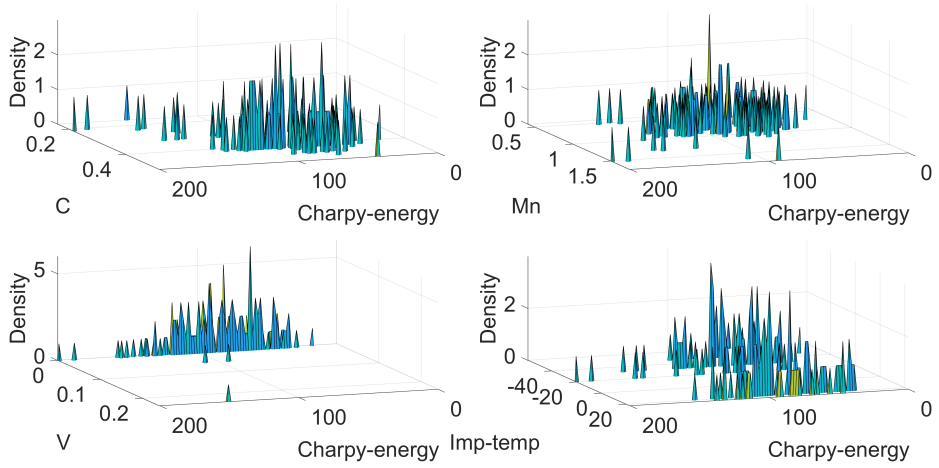

Figure 3.7: The data density plot between Charpy energy and "C", "Mn", "V", and "Imp-temp".

3.3 Ultimate Tensile Strength (UTS) Data for Steel

In materials science, one of the most popular methods used to determine the tensile strength of a material is the tensile test [112], during which a cylindrical specimen is grasped at one end while applying axial tension at the other. The sample is then stretched slowly and continuously at a standard rate until it fails. Three different tensile strength values can be recorded by conducting the above procedure, namely yield strength, ultimate tensile strength, and fracture strength. Yield strength is defined as the maximum stress a material can withstand without permanent deformation [113]. The first phase of material stretching is the elastic deformation phase, which follows Hooke's law, i.e., the stretching length has a linear and positive correlation to the applied force. Plastic deformation occurs when the applied force exceeds a threshold such that the material loses the ability to regain its original shape. The strength measured at this critical point is the yield strength, but it is not the maximum tension that such material can withstand before it fails. By further increasing the stress on the material, the degree of stretching of the test object will show a non-linear relationship with the force applied until it reaches the limit that the material can withstand. The value measured at this point is the ultimate tensile strength (UTS) [114]. For most ductile materials, the ultimate tensile strength is usually 1.5 - 2.0 times higher than the yield strength. Once the threshold of ultimate tensile strength is breached, the material will rapidly lose its integrity, and the stress level also decreases until it finally fractures, at which point the measured value is called the fracture strength [115]. The stresses recorded in the test can be used to plot the stress-strain curve. It is worth noting that the above behavior is only applicable to the ductile material because the brittle material does not have a significant yield point, and its ultimate tensile strength is essentially equivalent to the fracture strength.

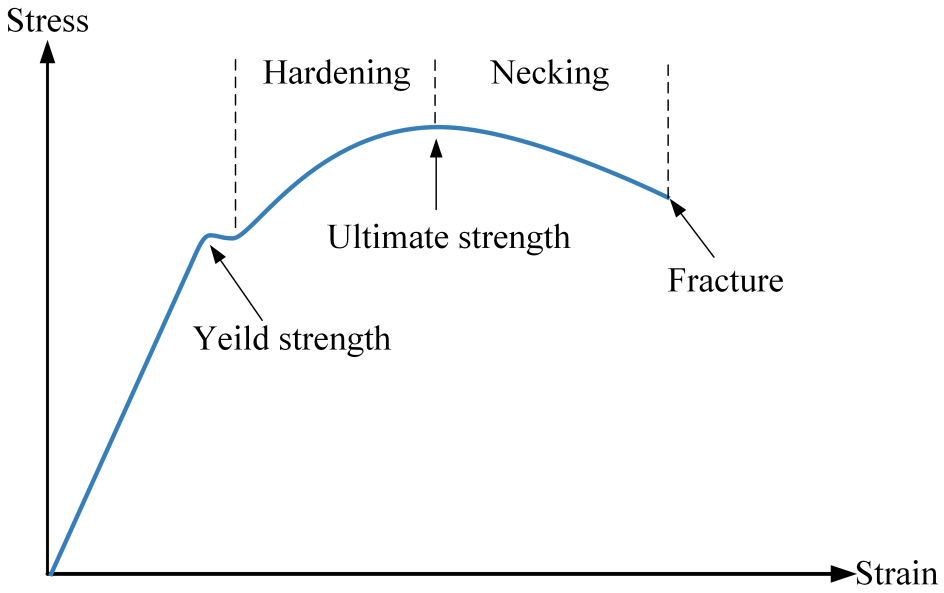

Figure 3.8: The stress-strain curve for low carbon steel. (Figure inspired by [116])

Most metals are ductile materials since the bonds between metal atoms are metallic bonds that can be re-established even after a change of position, which ensures good ductility [116]. Compared with elemental metals, alloy materials have more diverse properties. In general, alloys are less ductile than the pure elementary substance of their constituent metals, and some types of alloys are even brittle materials [117]. However, it is the composition and production process that ultimately determine the properties of the alloy. Steel is an alloy of carbon and iron, which has better ductility in comparison with brittle pig iron. In the presence of an applied external force, the intermolecular forces within the steel first maintain the structural integrity of the material, which corresponds to the elastic deformation phase mentioned above, except that this ability is limited. After the steel yields to a certain point, it is no longer able to maintain the absolute integrity of the material, and the internal grains will regroup themselves to increase the resistance to tensile stress again to stop the structure from fracturing, which leads to a limited but irreversible deformation of the material, i.e., plastic deformation of the steel. The increase in tensile strength caused by internal grain regrouping is also not infinite. After reaching its peak, the weakest point undergoes rapid deformation, and the cross-sectional area shrinks rapidly until fracture. Such process is also known as “necking". The stress-strain curve reflecting the tensile performance of low-carbon

steel is given in Figure 3.8. The maximum tensile strength recorded during this entire process is the ultimate tensile strength of the steel, which is one of the most crucial indicators of tensile strength for steel materials. In most engineering applications, the ultimate tensile strength of steel materials is considered an indispensable factor in the design and analysis of structural reliability. The ultimate tensile strength is an intensive property for which its value does not depend on the size of the test specimen. It can be calculated by the following equation:

$$\sigma_{\max} = \frac{f_{\max}}{s}, \tag{3.1}$$

where $f_{\max}$ denotes the maximum force that the structure bears after entering the plastic deformation stage, S is the original cross-sectional area of the specimen, and $\sigma_{\max}$ refers to the ultimate tensile strength.

Table 3.2: The basics of the ultimate tensile strength data.

Numerical Variable	**Mean**	**Median**	**Range**
C	0.3942	0.41	0.12-0.62
Si	0.2546	0.25	0.11-0.35
Mn	0.7524	0.73	0.35-1.72
S	0.021	0.023	0.0005-0.21
Cr	1.053	1.07	0.05-3.46
Mo	0.2631	0.23	0.01-1
Ni	0.8039	0.25	0.02-4.16
Al	0.036	0.027	0.005-1.08
V	0.0075	0.005	0.001-0.27
Hardening Temp	856.81	850	820-980
Tempering Temp	604.18	610	170-730
Sample Size	156.93	150	8-381
Test Depth	16.08	12.7	4-140
UTS	932.09	912.9	516.2-1842
Category symbol			**Categories**
Site			6
Cooling Medium			3

The UTS dataset utilized in this thesis contains the ultimate tensile strength of 3760 groups of high-strength steels, each of which differs in the formulation and heat treatment settings. This dataset has 15 input variables, where 13 of which are numerical and 2 are categorical. The basics of this dataset are provided in Table 3.2. Akin to the Charpy impact dataset, the relationship between inputs and outputs in the UTS dataset is highly nonlinear, and its modeling process also suffers from missing key variables and interference from uncorrelated variables. Owing to the relatively high certainty in the UTS test, this dataset is less chaotic than the Charpy impact data. However, the sparsity of the data increases and the missing variables seem to play a more significant role in test results, which makes modeling even more difficult. For machine learning algorithms, high dimensionality and sparsity are immense challenges. Not only too many input variables itself can bring about the problem of the curse of dimensionality, but the interference of irrelevant variables can also affect the modeling outcome. Sparsity often leads to the incomplete training of machine learning models and reduces the accuracy of the trained model. Besides, the absence of variables leads to further difficulty in modeling, which can only be compensated by the generalization ability of the algorithm to a limited extent. As per experience, many machine learning algorithms perform poorly on this dataset, requiring pre-processing of the data such as filtering variables or dimensionality reduction to reduce the data difficulty to obtain results of practical significance. The above characteristics make the data very challenging but also make it an ideal candidate for testing the extreme performance of the proposed algorithm. The data density plot between several input variables for the UTS data of high-strength steel is given in Figure 3.9. The data density plot between the UTS value and some input variables is also shown in Figure 3.10.

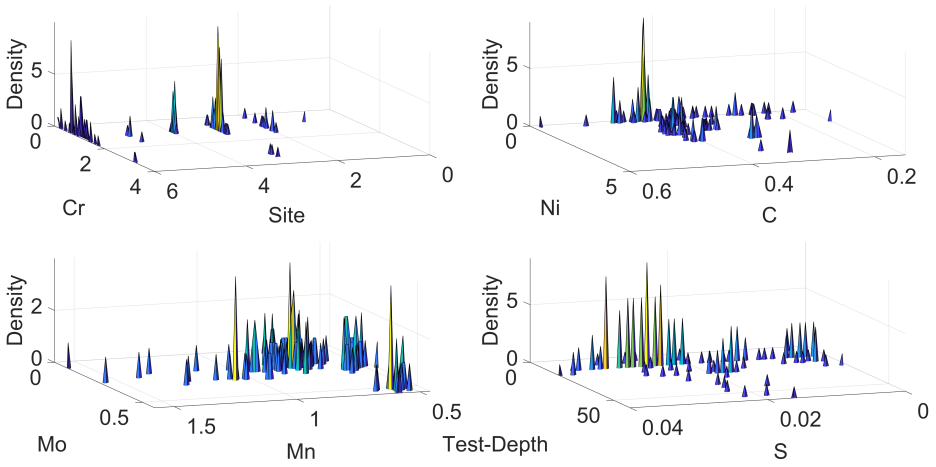

Figure 3.9: The data density plot between “Cr and Site”, “Ni and C”, “Mo and Mn”, and “Test-Depth and S” for the UTS data.

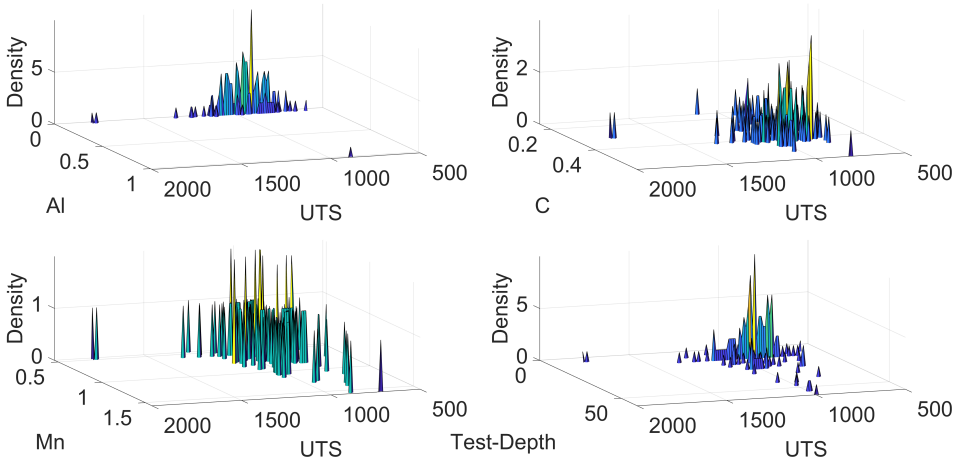

Figure 3.10: The data density plot between UTS and “Al”, “C”, “Mn”, and “Test-Depth”.

3.4 Performance Indices for Benchmark Tests

Establishing the benchmark for regression models requires, in addition to an appropriate validation dataset, reliable performance metrics that objectively and comprehensively reflect how the model performs. For regression applications, most of the commonly used performance indicators compare the statistical difference between the model outputs and their corresponding labels as the approach to evaluate the approximation efficacy. The most widely used performance index is the mean square error (MSE), which describes the performance of an algorithm based on the quadratic of the residuals between model predictions and label values. Considering that the value of MSE is often very large in practice, the square root form of MSE is introduced, i.e., root-mean-square error (RMSE) [118], which can be calculated as follows:

$$RMSE = \left(\frac{1}{N} \sum_{i=1}^{N} (y_i - w_i)^2 \right)^{1/2}, \tag{3.2}$$

where N is the number of samples, w_i represents the actual value or the label of each sample, and y_i denotes the observed value. RMSE is a measure of overall residuals to compare forecasting deviations of different algorithms for a specific dataset, for which the lower the RMSE, the better the model accuracy. It is important to note that this metric cannot score the performance of a particular algorithm across different datasets, as it is scale-dependent and differentiated datasets are unlikely to share an identical scale. The other drawback of RMSE is that its method of calculating the residual quadratic is too sensitive to errors or outliers in the data, resulting in decoupling from the actual performance of the algorithm in such cases.

The mean absolute error (MAE) [119] is another residual-based performance indicator, which uses the average sum of the absolute values of the residuals instead of the square to describe the difference between two sets of values, of which the mathematical expression of is given as follows:

$$MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - w_i| . \tag{3.3}$$

Akin to MSE and RMSE, the MAE is a scale-dependent accuracy measure subject to the scale as presented by the data and cannot be used to provide well-founded comparisons between objects with diverse scales. But using the absolute residual approach to describe the error brings added benefits. Firstly, the metric is less sensitive to outliers in the data, making it a perfect substitute for RMSE in the presence of high noise levels. Secondly, the relationship between MAE and residuals is linear, and its statistical implications are easier to understand compared to RMSE, thus facilitating some analytical tasks. However, the MAE calculated by the absolute value method has difficulty reflecting subtle differences when the residuals between values are relatively small, which causes it less popular than the RMSE in most cases.

Considering that the above residual-based performance metrics depend on the data scale and cannot evaluate the performance of a single algorithm on different datasets, the symmetric mean absolute percentage error (SMAPE) [120] is introduced into the benchmark test. SMAPE is an improved version of the mean absolute percentage error (MAPE) [121]. The proposal of MAPE does address the limitations of previous performance metrics governed by data scales, but it also has many shortcomings. Firstly, MAPE uses the residuals divided by the absolute value of the label to eliminate the effect of scale, leading to its inability to handle the case when the label value is zero. Secondly, despite MAPE being measured by the percentage, its value can be greater than 100% without an upper bound, which may easily lead to misunderstanding of the information it conveys. Thirdly, MAPE is asymmetric due to a more pronounced penalty for situations where the observed value is larger than the label, causing it to favor underfitting models. SMAPE overcomes the above problems and thus becomes a popular model performance indicator, which is obtained according to the following formula:

$$SMAPE = \frac{100\%}{N} \sum_{i=1}^{N} \frac{2 * |y_i - w_i|}{|y_i| + |w_i|}. \tag{3.4}$$

SMAPE can tackle the case where the actual value is zero by replacing the absolute actual value on the denominator with the sum of the absolute value of the label and the observation. This modification also limits the SMAPE between 0% and 200%, sig-

nificantly reducing the potential for confusion and misunderstanding. The addition of the upper bound also alleviates the problem of an unbalanced penalty against negative errors in the original MAPE, allowing SMAPE to be a more reliable option in evaluating the model. However, this indicator is also deficient because its solution to mitigate the MAPE asymmetry introduces another asymmetry, i.e., the denominator is influenced by the scale of the observed value and the actual value, which partially offsets the previous effort to eliminate the impact of data scale on metrics. Additionally, given that SMAPE values defined between 0% and 200% may still cause inconvenience, researchers prefer to remove the "2" item from the equation when using it as a benchmark test metric so that the interval lies perfectly between 0% and 100%.

In addition to residual-based statistics, a statistic based on the degree of discreteness of a numerical sequence itself, i.e., the standard deviation (SD/STD) [122], is also introduced in this thesis to assess the statistical differences that exist between the model outputs and the label values. In mathematical statistics, the standard deviation, defined as the square root of the variance, quantifies the variation or dispersion of a series of values. A low standard deviation suggests that the values are distributed closely around the mean, while a high standard deviation indicates that the values scatter over a relatively broad range. Ideally, the standard deviation can be acquired if every member of the entire population is known. However, it is rarely the case in the real world because it may only be possible to access a subset of the statistical population, which makes it unfeasible to obtain the actual sample mean to calculate the exact variance when only a portion of the population is available. In such cases, only estimated standard deviations can be obtained based on the estimated mean and variance of the population, and the estimates of such statistics can be determined according to a finite random sample, for which the formula is given as follows:

$$STD = \left(\frac{1}{N-1} \sum_{i=1}^{N} (y_i - u)^2 \right)^{1/2}, \tag{3.5}$$

where N indicates the size of a random sample, u represents the mean of this sample, and y_i denotes a single value in the sample. For the assessment of model regression

effectiveness, the significance of introducing standard deviation is that residual-based metrics focus on the difference between the predicted values and the actual ones, but ignore the distribution of both themselves, resulting in an incomplete reflection of the model's prediction confidence. There exists a situation where the prediction model can obtain good residual performance, but the statistical distribution of the prediction results differs significantly from the labels. Such problems are more common in machine learning-based algorithms since the optimizers of these algorithms are designed based on residuals, and their goal is to obtain minimized residual statistics rather than maximum authenticity. Even for simple classification problems, it occurs that models with high accuracy in the sense of performance indices output completely wrong classification results. Thus, the introduction of standard deviations in benchmark tests can help detect the presence of such problems to ensure that flawed models do not "get away with it". Moreover, the standard deviation can also help to identify the better one if two models share a similar residual performance.

According to the above perspectives, it is believed that jointly utilizing performance metrics RMSE, MAE, SMAPE, and STD in the benchmark test can largely compensate for the shortcomings caused by merely applying a single index, thus providing a more reliable assessment of the actual performance of the model. However, given that the Sunspot time series dataset is relatively simple and the RMSE index alone is enough to evaluate the predictive performance of the model on it, this four-index-strategy will only be applied in the evaluation of the two complex metallurgical datasets.

3.5 Chapter Summary

This chapter provides a brief overview of the real-world datasets applied in the thesis. Three independent subchapters are created to present the Sunspot time series dataset, the Charpy impact dataset for high-strength steels, and the ultimate tension dataset for steel materials, respectively. Related background knowledge is demonstrated at the beginning of each subsection; information such as data dimensionality, variable details, and data density distribution are also included; at the end of each subsection, descriptions of the characteristics and difficulties of each dataset are pointed out and discussed. For sunspot time series data, the overall periodic regularity is present along with limited chaotic properties, making it an ideal dataset for machine learning benchmark tests. Given that the time series prediction cannot violate causality, the cross-validation method is not available for this dataset. The Charpy impact dataset is high-dimensional, highly nonlinear, sparse, and mildly noisy. As a rough dataset obtained directly from the industrial sector, some relevant quantities are missing because they are difficult to measure, and irrelevant variables are also falsely included. These features cover common problems that machine learning algorithms are likely to encounter in real-world scenarios, which provides a better simulation of real-world situations than carefully refined ideal datasets. The ultimate tensile strength dataset is also an industrial dataset, of which the characteristics are close to the Charpy impact dataset but with a significantly higher sparsity. Sparsity immensely increases the difficulty of modeling, causing many machine learning models to perform poorly on this data. Therefore, the ultimate tensile strength dataset will serve as the ultimate challenge for the proposed algorithms in future chapters. For the convenience of readers unfamiliar with the field, statistical performance indicators involved in evaluating model performance regarding regression tasks are also clarified at the end of this section.

4

DCVSF: A Deep Complex-Valued Single Iteration Fuzzy System for Predictive Modelling

> *I cannot help fearing that men may reach a point where they look on every new theory as a danger, every innovation as a toilsome trouble, every social advance as a first step toward revolution, and that they may absolutely refuse to move at all.*
>
> **– Alexis de Tocqueville –**

4.1 Introduction

Numerical prediction is a traditional application of regression analysis and a class of the functional regression problem, i.e., using a model to fit the abstractive mapping relationships between input and output data sets to predict numerical outputs with given input vectors [123]. Classic predictive models are usually built manually and often with poor performance when significant nonlinearity is presented in the data. Not to mention that applying such methods requires considerable expert knowledge of users, which further narrows its already limited applicability. The emergence of machine learning brings a solution with self-learning and adaptive modeling capabilities, but the shortcomings of the current machine learning methods are also evident. Firstly, the human brain only requires inputting a small number of samples once or a few times to complete the learning of a concept, but it usually takes a machine learning method hundreds of iterations to train on a much larger sample set to achieve similar outcomes or worse [124]. Secondly, many learning algorithms are troubled by the so-called curse of dimensionality [22], which usually leads to the rocketing computational resources requirement as well as a steady drop in the model accuracy as the data dimension expands [125]. Thirdly, interpretability is also one of the most challenging problems of machine learning algorithms, given that many models with excellent performance are "hard to explain" black boxes [126]. The most significant impact of the black-box property on model training is that it is difficult to determine the initial parameters, and it is not convenient to quickly identify the problem when the exception occurs.

The deep neural network is currently the most popular machine learning technology. On the one hand, to increase the training speed and reduce the number of iterations, concepts such as transfer learning [127]–[129], meta-learning [130]–[132], and reinforcement learning [133] have been successively introduced to improve the training efficiency and enhance the performance of the network model. These attempts lead to more flexible models but fail to tackle the problems associated with the high computational demand of deep network algorithms. On the other hand, some numerical prediction algorithms based on deep neural networks can indeed achieve relatively good accuracy when it comes to data with higher dimensionality but also at the cost of sig-

nificantly increased computational complexity [134]. In fact, the fundamental logic behind deep learning algorithms is to solve the problem by massively piling up computing resources, which means advances in hardware performance play a more vital role than algorithm improvement regarding performance enhancement. Therefore, approaches from the deep neural network perspective are mostly “the expensive solutions,” which are suitable for cases where accuracy is paramount and sufficient hardware performance is guaranteed.

Fuzzy systems possess very different characteristics, of which the architectures are often much more compact and require lower computational performance than deep networks. Fuzzy models with self-learning capabilities merely demand a small amount of data points to train and can converge quickly. Interpretability is also an advantage of fuzzy systems compared to deep neural architectures [135]. Such distinguishing features make adaptive fuzzy systems potentially a class of "affordable solutions" for fast machine learning scenarios where the demand for accuracy is not extreme, but the computational resource is rather limited. However, fuzzy systems also suffer from the curse of dimensionality, similar to other machine learning technologies. The increase in dimensionality causes the fuzzy system with complete rule partitions to expand its rule-base exponentially, which may significantly increase its memory usage and hinder its potential, especially for incremental learning in memory-constrained systems. Therefore, mitigating this problem should become a research vector.

In this chapter, a complex-valued Wang-Mendel (CVWM) method is proposed by combining the core ideas of complex-valued neural networks (CVNNs) [18] and Wang-Mendel (WM) algorithm [136], [137]. On the one hand, this new model inherits the most significant advantage of the classic WM method, i.e., enabling the algorithm to complete training in only one iteration, which leads to a faster training process over traditional adaptive fuzzy systems based on neural networks or evolutionary algorithms. On the other hand, the introduction of the new idea that applying complex-valued structures to deal with the purely real-valued function mapping problem dramatically alleviates the explosive growth of the WM method’s rule-base as the input dimension increases, reducing its rule base to the square root size of the original method. For higher

dimensional scenarios, the concept of hierarchical deep fuzzy systems [138] is also introduced to form a deep complex-valued single-iteration fuzzy system (DCVSF), which can mitigate the impact of higher dimensionality even further. This hierarchical system utilizes CVWM as the basic unit. By piling up multiple CVWM units in a way similar to a convolutional neural network, such architecture reduces the dimensionality of the input layer by layer and finally obtains a one-dimensional output. In addition, to overcome the problems caused by the sparsity of the data, the t-distributed stochastic neighbor embedding (t-SNE) [139] is included to mitigate the sparsity by dimensionality reduction. DCVSF also requires only a single iteration to obtain the results, and its rule-base grows with the input dimension at a rate of only the square root level compared to the conventional fuzzy algorithms. The above characteristics make the algorithm an "affordable solution" particularly suitable for some higher-dimensional scenarios where achieving real-time performance with limited memory and computational performance is required, such as streaming data processing and incremental learning. For prediction accuracy, despite the slight performance loss caused by the complex-valued structure, CVWM and DCVSF still exhibit competitive nonlinear modeling capabilities in experiments. Finally, as a fuzzy system, DCVSF has better interpretability than most conventional deep learning models. Although the complex-valued structure destroys the natural connections between variables and makes knowledge extraction infeasible, its interpretability is still meaningful for debugging and parameter tuning.

4.2 Methodological Premises

4.2.1 CVNNs and CVNF

Complex-valued neural networks (CVNNs) are a category of artificial neural networks in which an operation object is a complex number. According to Hirose *et al* [140]–[142], the similarity shared by the multiplication operator of complex numbers and vector operations can bring statistical correlation between the real and imaginary parts, which limits the degree of freedom of numerical changes in the operation process. Due to the loss of degrees of freedom, a complex-valued input is not equivalent to two real-

valued inputs. But if the two real-valued variables are related in phase or frequency domain or share some correlations, operating the two as united as a complex number may lead to better performance [143]. Such a performance improvement is often reflected in the aspect of generalization capability. It is well-known that the more parameters the network model has, the easier it is for the network to fall into overfitting. For traditional real-valued models, operations between real numbers are utilized to approximate the complex connections between variables, which may lead to the introduction of redundant variables. Not to mention that each variable also has redundant degrees of freedom, consequently overfitting the data and reducing generalization performance even further. Therefore, CVNNs are theoretically practical and meaningful.

For fuzzy systems, complex arithmetic logic even brings a more valuable feature, i.e., alleviating the effects of the curse of dimensionality. Ryusuke *et al* [20] combined the concept of CVNN with the basic structure of the Mamdani fuzzy neural network, introducing the idea of CVNN into the construction of a fuzzy system for the first time, and named the product as the complex-valued neural fuzzy (CVNF) system. In the CVNF model, two real-valued inputs go through the network in the form of a single complex number, which significantly narrows the size of the fuzzy rule-base when computing [20]. Although a complex-valued input item is not equivalent to two real-valued inputs, thanks to the unique fault tolerance of the fuzzy intersection operation, the fuzzy rule-base generated by complex inputs can still largely reflect the whole picture even if there is no connection between the two variables [20]. In fact, the network achieves rather good nonlinear mapping capabilities when tested on cases of two inputs and four inputs. Unfortunately, such design requires backpropagation for optimization, and backpropagation of complex-valued networks is exceptionally intricate, which basically neutralizes its practical value. It should be clarified that the complex-valued architecture mentioned here is not the same as the concept called complex fuzzy theory (CFT) [16]. The CFT utilizes a complex-valued structure only for its complex-valued fuzzy rule-base to obtain the information implied by real-valued variables, for which the complex numbers only involve rule-base-related operations. The notion of CVNF is to combine two real-valued variables into a complex number for dimensionality reduction, and

it still belongs to the realm of type-1 fuzzy logic.

4.2.2 The Wang-Mendel (WM) Method

The seminal WM method was proposed by Wang *et al* [136] in 1992, and its final form was subsequently determined in 2003 by Wang [137]. This algorithm first measures the scales of input and output values as the reference to evenly generate fuzzy divisions for all variables. Once all divisions are determined, a membership function will be assigned to match every division in a way that the endpoint of the division coincides with the vertex of a membership function. The above step is said to be the initialization of the rule space. The next step is the training process, during which each membership function is assigned a firing strength according to each input-output pair. After finishing the entry of all training data points, one IF-THEN statement, i.e., one fuzzy rule, can be obtained by applying an intersection operation for all firing strengths generated using this membership function. Similarly, the algorithm can achieve all the other rules by performing the above procedure on all membership functions. The collection of such rules forms the trained rule base. For a single-output dataset with d input variables and n sample points, the set of the kth input-output data pair is given as follows:

$$\left(\alpha_1^k,\ \alpha_2^k, \ldots, \alpha_d^k; \gamma^k\right), k = 1, 2, \ldots, n, \tag{4.1}$$

where $\alpha_1^k,\ \alpha_2^k, \ldots, \alpha_d^k \in R,\ \gamma^k \in R$. Assume that the space of each input variable is divided into m fuzzy sets and the rule set for the ith variable is $\{\tilde{A}_i^1, \tilde{A}_i^2, \ldots, \tilde{A}_d^{m^d}\}, i = 1, 2, \ldots, d$. Then, the fuzzy IF-THEN rule-base can be generated as follows:

Rule 1: IF α_1 *is* $\tilde{A}_1^1$ *and* α_2 *is* $\tilde{A}_2^1 \ldots$ *and* α_d *is* $\tilde{A}_d^1$*, then* γ *is* $\mu(\gamma; \gamma_a^1, \sigma^1)$;

Rule 2: IF α_1 *is* $\tilde{A}_1^2$ *and* α_2 *is* $\tilde{A}_2^2 \ldots$ *and* α_d *is* $\tilde{A}_d^2$*, then* γ *is* $\mu(\gamma; \gamma_a^2, \sigma^2)$;

.....

Rule m: IF α_1 *is* $\tilde{A}_1^m$ *and* α_2 *is* $\tilde{A}_2^m \ldots$ *and* α_d *is* $\tilde{A}_d^m$*, then* γ *is* $\mu(\gamma; \gamma_a^m, \sigma^m)$;

.....

Rule m^d*: IF* α_1 *is* $\tilde{A}_d^{m^d}$ *and* α_2 *is* $\tilde{A}_{d-1}^{m^d} \ldots$ *and* α_d *is* $\tilde{A}_1^{m^d}$*, then* γ *is* $\mu(\gamma; \gamma_a^{m^d}, \sigma^{m^d})$.

where $\gamma_a^1, \gamma_a^2, \ldots, \gamma_a^m, \ldots, \gamma_a^{m^d}$ and $\sigma^1, \sigma^2, \ldots, \sigma^m, \ldots, \sigma^{m^d}$ are constant values to be determined by training. $\mu(\gamma; \gamma_a, \sigma) = \frac{\sigma - |\gamma - \gamma_a|}{\sigma}$, $\gamma \in [\gamma_a - \sigma, \gamma_a + \sigma]$ or $\mu(\gamma; \gamma_a, \sigma) = 0$, if $\gamma \notin [\gamma_a - \sigma, \gamma_a + \sigma]$ denotes a triangular membership function.

The WM algorithm only requires one full iteration of training set to complete the training, which comes with the cost. It possess three inherent flaws that cannot be ignored:

1. As a result of the necessity to initialize a considerable number of fuzzy sets for each variable, the WM algorithm can be easily plagued by the curse of dimensionality.

2. The completeness of the rule-base of this algorithm depends heavily on the distribution of the data set, which significantly limits its performance against sparsity.

3. The excessive rules of the WM method often give the algorithm a propensity to overfit the data.

4.2.3 Hierarchical Deep Fuzzy Systems

In simple terms, the core idea of a hierarchical network is to divide a multi-dimensional input into several smaller input vectors and assign them to different units for processing so that the information from each vector is aggregated to obtain a new output vector with a lower dimension than the original input [138]. Then the output vector of the previous level is used as the input for the next level to generate an even lower dimensional vector. The above steps should be repeated until the final output vector only has one dimension. Sometimes, the input variables from previous levels can be added to the input vector for the later levels to improve the accuracy. This process is similar to the realization process of the convolutional neural network, i.e., the information from the raw data is abstracted level-by-level through a multi-layer network to obtain the final result. The difference is that convolutional neural networks process image information,

while hierarchical fuzzy systems process high-dimensional vectors. The mathematics of each level is equivalent to using m separate subfunctions with q input variables to mimic an n input m output multivariable function $G(x_1, x_2, \ldots, x_n)$, which is expressed as follows:

$$\{g_1, g_2, \ldots, g_m\} \rightarrow G(x_1, x_2, \ldots, x_n), \tag{4.2}$$

where $m, n, q \in \mathbb{N}^+$ and $g_1, g_2, \ldots, g_m$ are the functions of each unit in this level.

Raju *et al* [138] first introduced the concept of the hierarchical deep fuzzy system in 1991 as an attempt to mitigate the curse of dimensionality for fuzzy control models. Hereafter, Wang proved that the hierarchical structure is strictly a universal nonlinear approximator through the mathematical derivation in [144] and demonstrated a general approach to designing hierarchical systems in [145]. Subsequently, other new hierarchical fuzzy systems were proposed despite a large proportion of these architectures being based on ANFIS [63]. For obvious reasons, researchers tend to use a level-by-level training policy for hierarchical algorithms rather than the traditional end-to-end training applied by most deep networks. This strategy leads to a new problem, given that the models used to construct the basic operation unit for the deep architecture, such as ANFIS, usually require repeated iterations to complete the training. If each layer demands dozens or hundreds of iterations to converge, the efficiency of the objective hierarchical architecture will be significantly degraded. Hence, Wang integrated the idea of hierarchy with his earlier proposed WM algorithm to create a DCFS (deep convolutional fuzzy system) [135]. Given that the basic unit of this model is the WM model, each level can finish training in only one iteration and avoids the trouble of repeated iterations, thus dramatically increasing the algorithm efficiency. However, despite the adoption of the hierarchical architecture, the training speed and memory usage of this DCFS model is still unsatisfactory in many scenarios due to the initialized rule-base of the original WM algorithm being huge, which leaves room for further improvements.

4.3 Methodology

4.3.1 Complex-Valued Wang-Mendel (CVWM)

The process of the CVWM algorithm generating a rule-base can be divided into five steps. To increase the robustness as well as the performance of the nonlinear mapping of the algorithm, CVWM uses the Gaussian membership function to replace the triangular membership function used by the original WM method. Given the kth input vector $a(k) = [a_1(k), a_2(k), \ldots, a_{2n}(k)]^T$ and output $y(k)$, where $k = 1, 2, \ldots, L$. Note that all the data points must be normalized between 0 and 1 first before further steps.

Step 1: Measure the maximum and minimum values of a given input variable and equally split the value range of each input into m fuzzy sets.

$$\begin{aligned} \min a_1 &= \min\{a_1(k)|k = 1, 2, \ldots, L\} \\ \max a_1 &= \max\{a_1(k)|k = 1, 2, \ldots, L\} \\ &\ldots\ldots \\ \min a_{2n} &= \min\{a_{2n}(k)|k = 1, 2, \ldots, L\} \\ \max a_{2n} &= \max\{a_{2n}(k)|k = 1, 2, \ldots, L\} \end{aligned} \tag{4.3}$$

Hence, the membership function of the initialized rule-base can be presented as follows:

$$O_j^p(k) = \exp\left(-\frac{(x(k) - \mu_j^p)^2}{2\sigma_j^2}\right), \tag{4.4}$$

where $\mu_j^p = j\frac{\max a_p - \min a_p}{m-1}$, $\sigma_j = \sqrt{\frac{\max a_p - \min a_p}{m-1}\frac{1}{8log2}}$, $p = 1, 2, \ldots, 2n$ and $j = 1, 2, \ldots, m$.

Step 2: Encapsulate membership functions into complex numbers, we have:

$$Ocp_j^q(k) = \exp\left(-\frac{(x(k)-\mu_j^{2q-1})^2}{2\sigma_j^2}\right) + i\exp\left(-\frac{(x(k)-\mu_j^{2q})^2}{2\sigma_j^2}\right), \tag{4.5}$$

where $q = 1, 2, \ldots, n$, $Ocp_j^q(k)$ is the operation object to be used to generate the rule-base in Step 3.

Step 3: First, calculate the weight using complex multiplication:

$$\omega(k) = \prod_{q=1}^{n} Ocp_j^q(k), \tag{4.6}$$

Then, substitute $y(k)$ to calculate the weighted average:

$$av = \begin{cases} \frac{\sum_{k=1}^{L} y(k) * \omega(k)}{\sum_{k=1}^{L} \omega(k)}, & \sum_{k=1}^{L} \omega(k) \neq 0 \\ 0, & \sum_{k=1}^{L} \omega(k) = 0, \end{cases} \tag{4.7}$$

where av_j denotes the weighted average of the firing strength of the jth rule and $av_j = 0$ means the corresponding rule is not triggered. Given that av_j is a complex value not usable for a real-valued operation, a real-valued term that represents the information from its corresponding rule should be generated to proceed with the following step, which is defined as the support and can be obtained as follows:

$$\text{sup} = \parallel av \parallel_2. \tag{4.8}$$

Step 4: The support values of some rules can be 0 because those rules are not triggered, which leads to the incomplete generation of the fuzzy rule base. Such an issue is common when dealing with sparse and non-uniformly distributed real-world datasets. Therefore, a strategy is necessary to estimate the support value for an untriggered rule according to its adjacent rules. A rule-base can be considered a high-dimensional tensor, where each rule is one of the elements. If the dimension of the input vector is $2n$, and each variable is assigned m fuzzy divisions, then the rule-base of CVWM can be represented as an n-dimensional tensor with m elements for each dimension, i.e., m^n elements in total. A method is employed here, i.e., the zero items are replaced with the mean of all non-zero support directly adjacent to this element to generate an estimated but complete rule-base.

$$\text{sup}_s = \begin{cases} \text{sup}_s, & \text{sup}_s \neq 0 \\ \frac{\text{sum}(\text{sup}_{s\pm1}, \text{sup}_{s\pm m}, \ldots, \text{sup}_{s\pm m^n-1})}{\parallel \text{sup}_{s\pm1}, \text{sup}_{s\pm m}, \ldots, \text{sup}_{s\pm m^n-1} \parallel_0}, & \text{sup}_s = 0 \end{cases} \tag{4.9}$$

where $s = 1, 2, \ldots, m, \ldots, m^n$; $\sup_{s\pm1}, \sup_{s\pm m}, \ldots, \sup_{s\pm m^{n-1}}$ are the support values of adjacent elements of sth grid from different dimensions. Note that if any expected adjacent element of the target element in equation (4.9) does not exist or exceeds the scope of the rule-base, then the support value of such adjacent rule is set to be zero. In the above steps, since the original input vector with $2n$ real-valued variables is substituted by a vector with n complex variables, the size of the expected rule-base is reduced from m^{2n} to m^n.

Step 5: This step generates the output of the trained model, which is shown as follows:

$$y_c = \| \frac{\langle \vec{\omega}_c, s\vec{u}p \rangle}{\sum \vec{\omega}_c} \|_2, \tag{4.10}$$

where $\vec{\omega}_c$ is the vector of the firing strength generated by input x_c according to equation (4.6), $s\vec{u}p$ is the vector of support values, $\langle \vec{\omega}_c, s\vec{u}p \rangle$ means the inner product of two vectors, and y_c denotes the model output.

4.3.2 Deep Complex-Valued Single-Iteration Fuzzy (DCVSF) System

For deep convolutional networks, a coverage area needs to be specified for each convolution kernel, i.e., the receptive field [146]. A similar concept also applies to each unit of the hierarchical fuzzy system, named the receptive window in this work. The receptive window is not for pixels but to cover a part of the variables from the input vector. Considering that the application of complex multiplication in CFWM limits the degree of freedom of the rule-base, which leads to the loss of precision, the receptive window should be as small as possible to reduce further performance loss. But if the window is too small, it may demand too many units for the final hierarchical system to have full coverage of data, which decreases the efficiency. Also, for obvious reasons, the window size should be even valued. Hence, it is recommended that the size of the receptive window for a single unit should not exceed six, and four is the ideal size in most cases. In addition, to improve the accuracy, overlapping is necessary between different receptive windows, and the number of variables to overlap for each window can be freely selected according to actual needs.

If given the input vector $x = [x_1, x_2, \ldots, x_n]^T$ and output y. To establish a DCVFS architecture to solve the nonlinear mapping problem for this dataset, the receptive window, and the overlap are assigned 4 and 2, respectively. Then, the mapping relationship of the first level of the system can be expressed as follows:

$$\begin{aligned} x_1^1 &= f_1^1\left(x_1^0, x_2^0, \ldots, x_4^0\right) \\ x_2^1 &= f_2^1\left(x_3^0, x_4^0, \ldots, x_6^0\right) \\ &\cdots\cdots \\ x_{\frac{n-2}{2}}^1 &= f_{\frac{n-2}{2}}^1\left(x_{n-3}^0, x_{n-2}^0, \ldots, x_n^0\right), \end{aligned} \tag{4.11}$$

where $\{x_1^1, x_2^1, \ldots, x_{\frac{n-2}{2}}^1\}$ is the output vector of the first level, which is used to play the input vector for the second level. $f_1^1, f_2^1, \ldots, f_{\frac{n-2}{2}}^1$ represent predictive models obtained by approximating training data using CVWM algorithm. According to the level-by-level training strategy, the dependent variable, i.e., y, is used as the label for the consequent part of each unit. The visualization of the first level of the architecture is given in Figure 4.1.

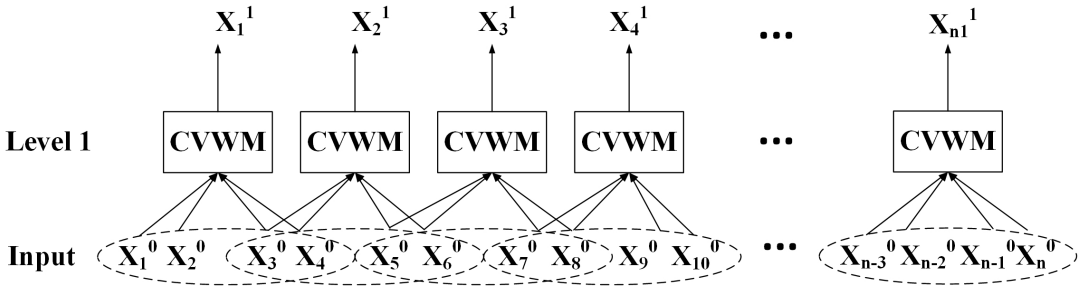

Figure 4.1: The first level of the DCVSF architecture.

The construction and training of every subsequent level are the same as those of the first level until the output becomes one-dimensional. If this deep architecture ends with L levels in total, then the entire mapping process can be expressed as follows:

$$f_1^L \circ \{f_1^{L-1} \ldots \circ [f_1^2 \circ (f_1^1, f_2^1, \ldots), \ldots], \ldots\} \rightarrow \quad f(x_1, x_2, \ldots, x_n), \tag{4.12}$$

where $\circ$ denotes the operator for function composition, $\{f_1^1, \ldots, f_1^2, \ldots, f_1^{L-1}, \ldots, f_1^L\}$

is the set of all predictive models generated in this DCVSF structure, $f(x_1, x_2, \ldots, x_n)$ denotes the actual functional relation between input and output of the data set. A demonstration for output level is shown in Figure 4.2.

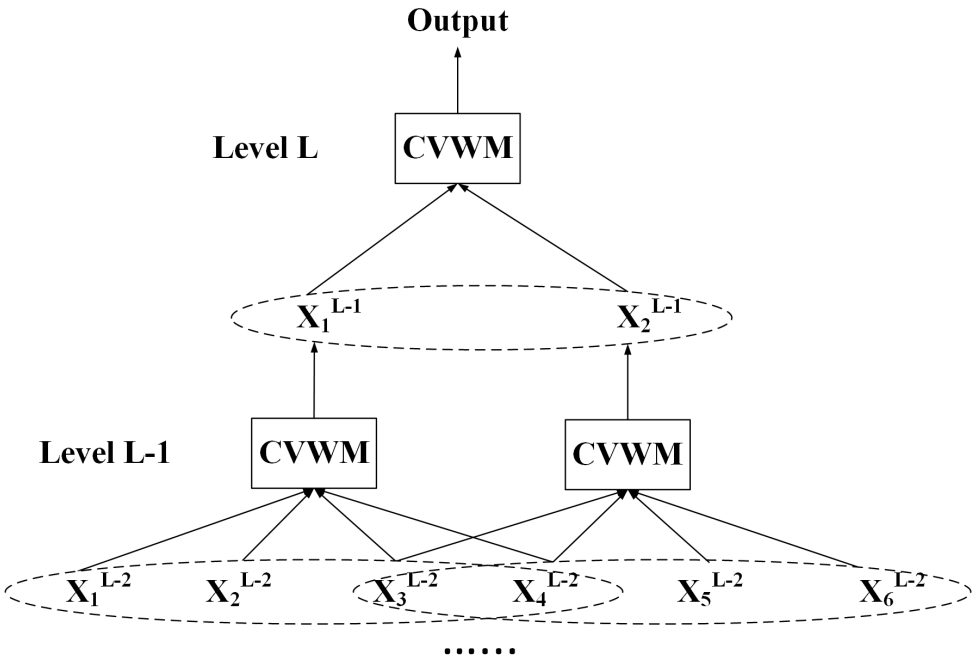

Figure 4.2: The final level of the DCVSF architecture.

It is worth mentioning that both CVWM and DCVSF are transparent. Although complex arithmetic makes the rule-base infeasible to interpret, such a property is still valuable for debugging and problem identification. The user can quickly locate the problematic units or variables based on this feature and correct them.

4.3.3 t-Distributed Stochastic Neighbor Embedding

The most applied dimensionality reduction (DR) algorithm is principal component analysis (PCA) [147], but as a linear method, it does not work well when encountering highly nonlinear datasets. The t-distributed stochastic neighbor embedding (t-SNE) is an effective DR algorithms for nonlinearity. Compared with vanilla SNE [148], t-SNE introduces t-distribution to reduce the crowding problem [149], which dramatically enhances the performance. The principle of the algorithm is as follows:

Assume there exists an m dimensional dataset $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}^m$. The goal is to reduce the dimensionality and convert it into a low-dimensional dataset $\mu_1, \mu_2, \ldots, \mu_n \in \mathbb{R}^d$, where $d < m$. Randomly select a point λ_i, use the Gaussian distribution to estimate its conditional probability $P_{j|i}$ with respect to its neighbor λ_j:

$$P_{j|i} = \frac{\exp\left(-\frac{\|\lambda_i - \lambda_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(-\frac{\|\lambda_i - \lambda_k\|^2}{2\sigma_i^2}\right)}. \tag{4.13}$$

The probability density function for Student's t-distribution is usually given as:

$$f_t(\tau) = \frac{\Gamma\left(\frac{v+1}{2}\right)}{\sqrt{v\pi}\Gamma\left(\frac{v}{2}\right)}\left(1 + \frac{\tau^2}{v}\right)^{-\frac{v+1}{2}}, \tag{4.14}$$

where v denotes the degree of freedom and $\Gamma(\tau) = \int_0^\infty x^{\tau-1}e^{-x}dx$ is the gamma function. Assign the value of v to 1, the t-distribution reduces to Cauchy distribution:

$$f_c(\tau) = \frac{1}{\pi\left(1 + \tau^2\right)}. \tag{4.15}$$

Hence, the conditional probability of a low-dimensional data point μ_i against its neighbor μ_j is obtained as:

$$Q_{j|i} = \frac{\left(1 + \| \mu_i - \mu_j \|^2\right)^{-1}}{\sum_{k \neq i}\left(1 + \| \mu_k - \mu_j \|^2\right)^{-1}}. \tag{4.16}$$

The difference between the two distributions is measured with Kullback-Leibler divergence:

$$KL = \sum_i \sum_j P_{j|i} \log \frac{P_{j|i}}{Q_{j|i}}. \tag{4.17}$$

The expected low-dimensional vector can be obtained by minimizing the difference in distribution between high-dimensional and low-dimensional data.

4.4 Experimental Results and Analyses

4.4.1 Input-to-Output Numerical Example for CVWM

The numerical demonstration of the CVWM model employs a sample selected from the non-linear function data generated according to the following equation,

$$f(x_1, x_2) = \sin(10x_1) + \cos(4x_2) - \cos(3x_1x_2), 0 \leq x_1 \leq 0.5, 0.5 \leq x_2 \leq 1, \quad (4.18)$$

for which $x_1 = 0.0275, x_2 = 0.775$ and its label $y = 1.5801$. A CVWM model with two inputs, one output, and nine divisions is trained on a training set with 125 data samples to implement this test. The parameter information of the objective model is provided as follows:

$$\begin{aligned}
\vec{sup} =& [1.70944818652361, 1.95047589647353, 2.14906806456774, \ldots \\
& 2.07884862080738, 1.75120532034107, 1.30816708116708, \ldots \\
& 0.968113546202332, 0.939498876617390, 1.149415117164901], \\
\vec{\mu^1} =& [0, 0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5], b^1 = 0.0265, \\
\vec{\mu^2} =& [0.5, 0.5625, 0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375, 1], b^2 = 0.0265.
\end{aligned}$$

First, the value of all complex-valued membership functions can be calculated according to equation (4.5).

$$\vec{Ocp} = \begin{bmatrix}
(4.87794765690766 + 4.87794765690766j) \times 10^{-24} \\
(1.20329746147621 + 1.20329746147621j) \times 10^{-14} \\
(1.15949505259070 + 1.15949505259070j) \times 10^{-7} \\
0.00436440288309460 + 0.00436440288309460j \\
0.641712948781452 + 0.641712948781452j \\
0.368567304322776 + 0.368567304322776j \\
(8.26899719104033 + 8.26899719104033j) \times 10^{-4} \\
(7.24684407869198 + 7.24684407869198j) \times 10^{-9} \\
(2.48087581161067 + 2.48087581161067j) \times 10^{-16}
\end{bmatrix},$$

The numerator for Mamdani defuzzification is then available by computing the inner product of the vector $\vec{Ocp}$ (values of the membership functions) and the vector $\vec{sup}$ (supports):

$$numerator = \vec{Ocp} \cdot \vec{sup} = 1.61579246647277 + 1.61579246647277j.$$

The denominator of the defuzzification formula can also be obtained by summing over all elements of the vector $\vec{Ocp}$:

$$denominator = \sum_{i=1}^{9} \vec{Ocp} = 1.01547167890279 + 1.01547167890279j.$$

Hence, one can acquire the final prediction result by calculate the L2 norm of the following expression:

$$y = \| \frac{numerator}{denominator} \|_2 = 1.59117432818867 \approx 1.5912.$$

The difference between the predicted result $y = 1.5912$ and the ground truth $y = 1.5801$ is not significant, which is acceptable considering that the function is non-convex and the number of rules assigned for this test is low according to the standard of WM or CVWM method. It is worth noting that if the traditional WM approach is used to solve this problem, the number of rules generated for the same setting should be $9^2 = 81$ due to the presence of two variables, which implies a significant increase in memory usage compared to CVWM.

4.4.2 CVWM on Synthetic Function Modelling

The 2-input synthetic dataset is generated by the "Sawtooth" function as follows:

$$f(x_1, x_2) = x_1^2 + 4\sin(2\pi x_1) - x_2^4 - 6\cos(2\pi x_2), -2 \leq x_1, x_2 \leq 2, \tag{4.19}$$

in which 1000 training data points are uniformly sampled between [-2,2] and 800 test data points are also taken from the same range but sparser. More details about this test

are provided in Table 4.1 , and the visualization of the forecasting outcome is given in Figure 4.3. As it can be seen from Figure 4.3, despite the target function being highly nonlinear and non-convex, both algorithms achieve high prediction accuracy. The approximation results of the two algorithms are so close that only a minor difference to discern from the given plot. The RMSE of the two in the table also shows that the errors of the two algorithms are in the same order of magnitude, while the WM method has only a negligible lead. Given that WM initializes up to 2500 rules to achieve this performance and CVWM uses only 50, it is sufficient to demonstrate that the proposed CVWM algorithm can obtain similar accuracy as the original WM algorithm in a data-dense function approximation task with an immensely reduced rule-base.

Table 4.1: The settings and testing RMSE of WM and CVWM over the "Sawtooth" function test.

	Testing RMSE	**Divisions**	**Rules**	**Iterations**
WM	1.965×10^{-1}	50	2500	1
CVWM	2.640×10^{-1}	50	50	1

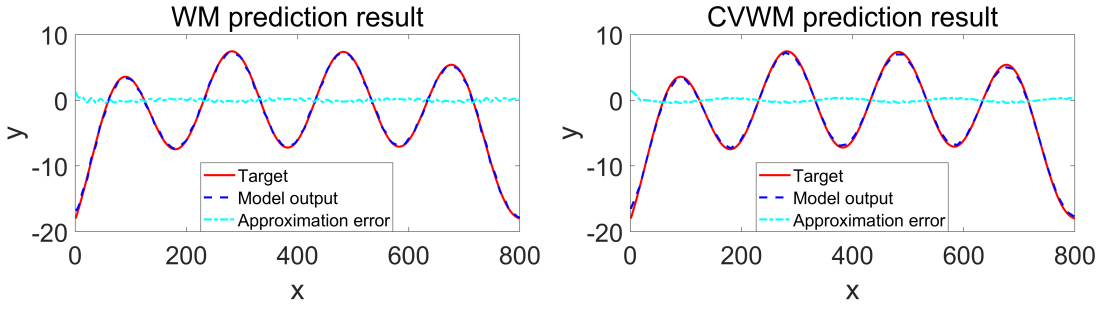

Figure 4.3: The comparison between WM and CVWM outputs on the "Sawtooth" function test.

4.4.3 CVWM on Sunspot Prediction

The Sunspot time series data used in this chapter contains sunspot activity sampled from 1984 to 2000, which is free to access from the World Data Center for the Sunspot Index (SIDC) [107]. The dataset consists of 2000 data samples in total. For the experiment, each input vector is constructed in the form of $\{y(\tau-2), y(\tau-1); y(\tau)\}, \tau = 3, 4, \ldots, 2000$, in which the current sunspot value is forecast according to the earlier two moments. Note that time series forecasting cannot violate its intrinsic causality, i.e., one cannot use future values to predict the past, so the first half of the 2000 data points are

employed to construct the training set, while the second half is utilized as the test set, to ensure that all data points in the training set precede the data points in the test set. The detailed information and the comparison of the result between benchmark models are shown in Table 4.2, and Figure 4.4 shows the visualized prediction outcome of CVWM on the test set.

Table 4.2: A comparison between benchmark algorithms over the Sunspot test.

	MSE (Normalized)	**Divisions**	**Rules**	**Iterations**
SARIMA [150]	6.5733×10^{-3}	–	–	–
BP [52]	1.4264×10^{-4}	–	–	200
LSTM [151]	3.4530×10^{-4}	–	–	200
Elman-NARX [152]	1.4078×10^{-4}	–	–	200
NFS [95]	8.5112×10^{-5}	5	25	100
WM [137]	2.5775×10^{-5}	20	400	1
CVWM	6.6587×10^{-5}	20	20	1

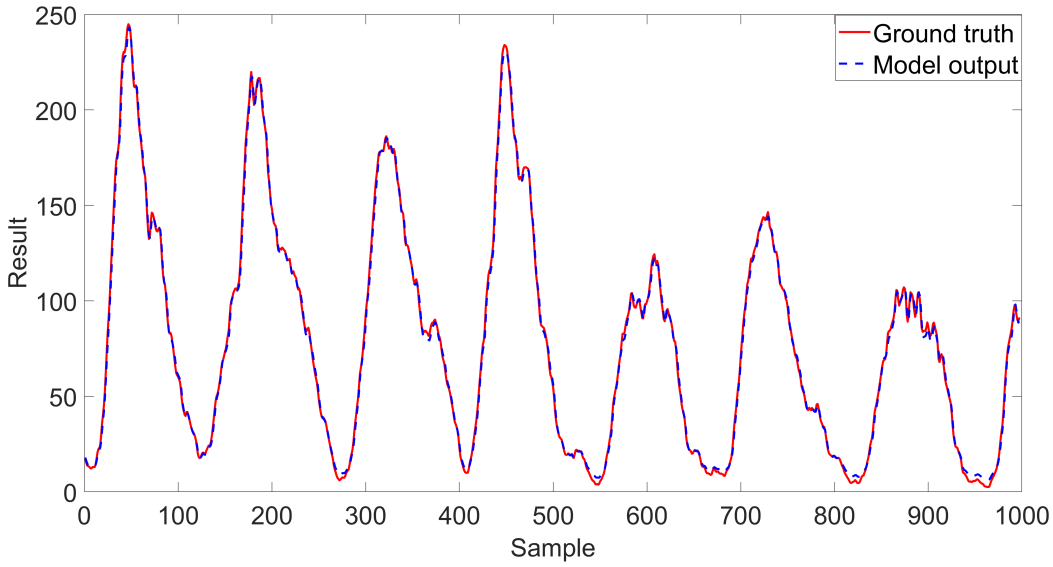

Figure 4.4: The CVWM prediction result on the Sunspot dataset.

Table 4.2 shows that the prediction accuracy of CVWM outperforms that of the recurrent Elman network [152] and NFS (ANFIS with PSO-RLSE optimization) [95] and is only inferior to the original WM algorithm. Considering that the complex arithmetic involved in CVWM reduces precision, this is in line with reality. However, when comparing the size of the generated rule-base, it is not surprising that the WM algorithm performs the best as it initialized 400 rules to achieve such a performance. In contrast, CVWM achieves a prediction accuracy close to that of WM using only 20 rules, even

fewer than the 25 rules used by the NFS. Such a result suggests that CVWM has the potential in time series forecasting, not to mention that it only needs to be trained once, which makes it more advantageous than algorithms that require repeated iterations.

4.4.4 DCVSF on Charpy Impact Data Modelling

The Charpy impact dataset is a 16-dimensional metallurgical dataset with high non-linearity, sparsity, and stochasticity, which is challenging even for many powerful deep neural network models. Given that directly substituting such data into the model may prove unwise, the t-SNE algorithm is employed to reduce it to 10 dimensions to alleviate the sparsity (only for DCFS and DCVSF). This time the DCVSF model designed for high-dimensional scenarios is applied to generate the predictive model for this tricky dataset. Table 4.3 gives the values of performance indicators obtained during the experimental process for all benchmark models. To fully access the performance of involved algorithms on this dataset, standard deviation (STD), mean absolute error (MAE), symmetric mean absolute percentage error (SMAPE), and root mean square error (RMSE) are employed as performance indicators. Additionally, Figure 4.5 displays the regression plot between model prediction and the actual Charpy impact energy.

According to Table 4.3, DCVFS still achieves a competitive outcome on this tricky dataset. It is worth noting that this algorithm was initially designed as an "affordable solution" to provide near-mainstream model performance with limited computational resources. Although the regression accuracy is inferior to other deep network models, this result is acceptable considering DCVSF's single-iteration training and lower computational demand. Concomitantly, the introduction of t-SNE does improve the ability of DCVSF when dealing with sparsity. Even when compared with DCFS, i.e., the counterpart using WM as the basic unit, DCVSF constructed by CVWM does not show a significant loss of accuracy on the Charpy impact test. All these results suggest the feasibility of using a complex-valued input to approximate two real-valued inputs, even if it may lead to a loss of accuracy to some degree compared with pure real-valued methods. In addition, the Charpy impact dataset used in the experiments is designed for testing

high-performance deep neural network models, which have high requirements on the nonlinear mapping ability, generalization ability, and noise immunity of the models. Therefore, it is reasonable to believe that the DCVSF algorithm may perform better on less challenging cases, which this algorithm more often encounters.

Table 4.3: Performance regarding several benchmark models over the Charpy impact data test.

	STD	MAE	SMAPE	RMSE	Divisions	Iterations
BP [52]	25.2233	16.9010	21.2874	20.8389	–	200
RBF [53]	30.9238	14.7211	19.5694	20.1443	–	100
GRNN [153]	22.7647	16.0598	19.0444	22.0339	–	100
LSTM [151]	24.9117	17.0228	21.5417	21.2242	–	200
DBN [154]	27.2085	16.2536	20.2341	20.1211	–	50
DCFS [135]	29.4341	17.4645	21.8562	23.7818	30	1
DCVSF	25.3238	17.3158	21.6963	23.1534	30	1

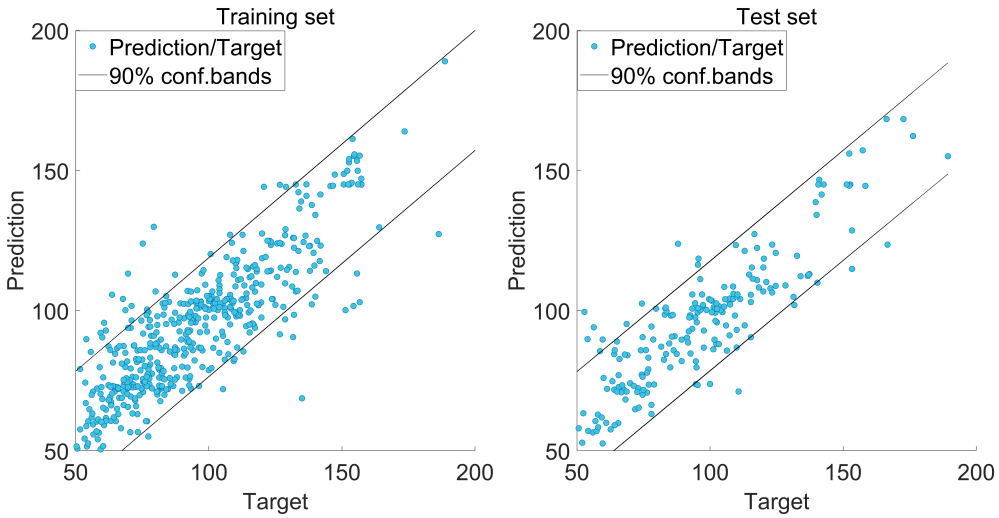

Figure 4.5: The result regression plot for Charpy impact energy prediction by DCVSF.

4.4.5 DCVSF on Ultimate Tensile Strength Data Regression

The ultimate tensile strength (UTS) dataset is another metallurgical dataset with 15 input variables and a single output. As mentioned in Chapter 3, the biggest obstacle to modeling this dataset is its high-level sparsity, which poses more challenges to the algorithm than the Charpy impact dataset. Given that the size of the dataset is slightly larger than the algorithm requires, 1500 data points, i.e., 40% of the entire dataset, are randomly selected for this experiment, of which 1000 are to form the training set, while

the remaining 500 are as samples for the test set. Similar to the case in the Charpy impact data test, the t-SNE algorithm is applied to reduce the dimensionality of the input vectors from 15 dimensions to 10 dimensions to ease its sparsity. Although it is theoretically possible to shrink the data dimensionality to an under 10 level to obtain a denser data matrix, the final dimensionality of the t-SNE output is set to be consistent with the case in the Charpy impact data test to investigate the effect of sparsity on the DCVSF algorithm. Table 4.4 shows the performance metrics obtained during the test, and Figure 4.5 gives the regression plot.

Table 4.4: Performance regarding several benchmark models over the UTS data test.

	STD	**MAE**	**SMAPE**	**RMSE**	**Divisions**	**Iterations**
BP [52]	150.9674	32.1335	3.4460	44.4965	–	200
RBF [53]	149.2079	41.2173	4.4341	54.1319	–	100
GRNN [153]	153.8146	39.2092	4.1841	56.5168	–	100
LSTM [151]	137.5868	42.4865	4.5160	56.5765	–	200
DBN [154]	142.1892	36.1535	3.9260	47.7999	–	50
DCFS [135]	150.8623	35.8515	3.8851	47.3721	30	1
DCVSF	135.9964	53.7214	5.7822	78.7232	60	1

A significant weakness of the original WM approach and its hierarchical structure version DCFS is that they are highly susceptible to sparsity, which is inevitable considering the way the WM method generates its rule-base. Most machine learning algorithms approximate the data by relying on optimization algorithms to tune their parameters. The WM approach, however, achieves self-learning by initializing a large rule-base that covers all possibilities regardless of relevance and then using the data to trigger the rules within it and assign them firing strength. During training, the rules with high firing strength acquire higher weights, while rules with low relevance are assigned low weights. Such an approach enables the model to complete training in a single iteration, but at the cost that the pre-initialized rule-base may not match the data well. Metaphorically, the rule-base is like a net, and the training data is the prey. The more the number of fuzzy divisions, the smaller the holes in the net. The higher the sparsity of the data, the smaller the size of the prey. Sparse data can easily pass through the "net" of weaved by the WM algorithm and make it unable to capture the "prey." Thus, in order to obtain as much information as possible from the sparse data, it is necessary to increase the fuzzy intervals assigned to each variable, but considering that the size of

the rule-base of such a model grows exponentially and the fuzzy intervals are the base of this exponential, doing so would quickly lead to an immensely large rule-base and seriously affect efficiency. A huge rule-base can also cause serious overfitting problems. Even if the algorithm succeeds in matching sparse data by increasing the number of fuzzy partitions to achieve good results on the training set, it may still perform poorly on the test set. Theoretically, once the sparsity of the data reaches a critical point, all WM-like algorithms will fail. For CVWM and DCVSF, although a complex-valued structure reduces the size of its rule-base, the consequent decrease in the degrees of freedom leads to a further reduction in its ability to handle sparsity compared to DCFS. Such an improvement to counter the curse of dimensionality even reduces the threshold at which the algorithm fails on sparsity, which is clearly exposed in the ultimate tensile strength data test.

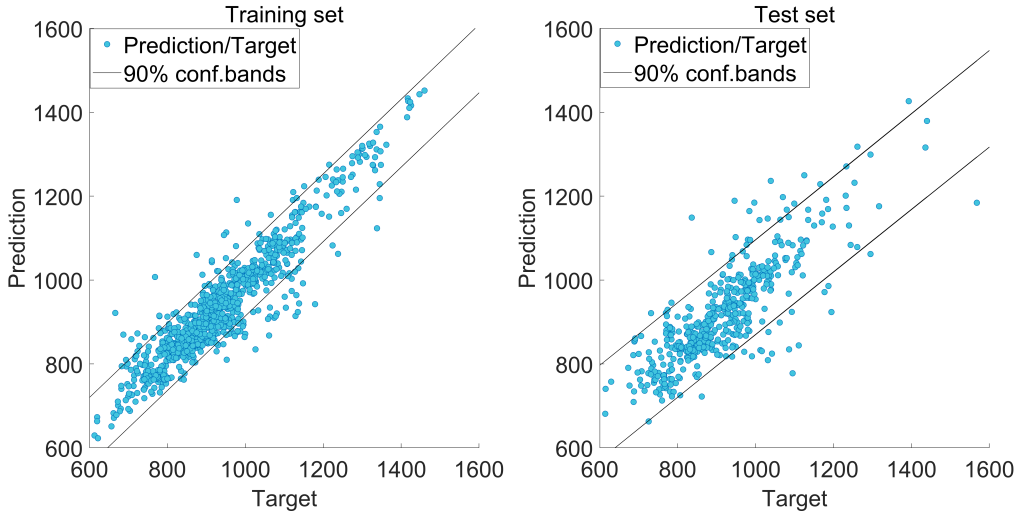

Figure 4.6: The result regression plot for the UTS prediction by DCVSF.

According to Table 4.4, despite the dimensionality reduction algorithm, DCVSF still performs poorly on the UTS dataset. Not only far worse than other neural network models but also inferior to DCFS. The difference between the performance of the test and training sets can also be seen in Figure 4.6 Even if the fuzzy division increased to 60 in the experiment, which is a very high number for any fuzzy model, its accuracy on the test set is still not ideal, despite a relatively good result on the training set. The experimental results are highly consistent with the conclusions of the theoretical analysis, i.e., the DCVSF algorithm is not a feasible option for data with high sparsity.

4.5 Chapter Summary

In this chapter, CVWM and DCVSF are proposed for the purpose of numerical prediction. The CVWM algorithm aims to alleviate the problem that the rule base of traditional WM methods grows too fast with increasing input dimensions by introducing the idea of converting two real-valued signals into one complex-valued signal for processing, thus slowing down the increase of the rule base as the input dimensions grow. For general low-dimensional dense datasets, CVWM reduces the generated rule base to the square root scale of the original WM method with only a minor loss of prediction accuracy, which enormously improves the efficiency of the algorithm. On this basis, by introducing the concept of a hierarchical fuzzy system, CVWM is further developed as the new algorithm DCVSF to process higher-dimensional data. Both CVWM and DCVSF inherit the feature of completing training in only a single iteration from the traditional WM algorithm. Both approaches are valuable for memory-constrained systems owing to the significant reduction of the size of the rule-base, thus reducing the memory demand of the system. In particular, its ability to complete training in a single iteration makes it well-suited for some online machine learning scenarios, such as streaming data processing, time series forecasting, and even incremental learning.

However, such algorithms are relatively weak in dealing with sparse data. Attempts to mitigate the sparsity of the data using the t-SNE dimensionality reduction algorithm in this chapter have yielded positive results on high-dimensional data with relatively low sparsity. Unfortunately, on datasets with extremely high sparsity, even dimensionality reduction cannot compensate for the inherent deficiencies of CVWM and DCVSF algorithms in dealing with sparsity. Considering that all algorithms have their strengths and weaknesses and there is no one-size-fits-all solution for all problems, the attempt to use the complex-valued structure to alleviate the curse of dimensionality of fuzzy algorithms, especially for the WM method, can be considered a success. In addition, the inefficiency of tackling the data type with significant sparsity is an inherent drawback of WM-like algorithms and cannot be solved by simply improving the WM algorithm itself. Thus, a new algorithm with a very different way of training will be proposed in the next chapter to cover the field of sparsity.

5

RACFIS: A New Rapid Adaptive Complex Fuzzy Inference System

> *Reality is created by the mind, we can change our reality by changing our mind.*
>
> **– Plato –**

5.1 Introduction

The modeling of real-world data has always been a challenging problem. Unlike those popular datasets that are purposefully selected as examples to test the performance of specific algorithms, datasets that users are more likely to encounter in scientific research or industry are often not so ideally presented. Real-world datasets may lead to processing difficulties, including redundancy, imbalance, incompleteness of the data, unrelated variables, missing dimensions, human entry errors or measurement errors, and so on, which hinder the efficient implementation of machine learning algorithms. In such scenarios, it is often impossible for users to know where the missing parts or the problems are for manual correction. Consequently, the generalization capability of an algorithm is the crucial element to be relied upon to overcome such obstacles. Moreover, regression tasks, such as data analysis and prediction in various application areas, lead to some real-time requirements. A heavy and sophisticated algorithm that usually indicates a dramatic increase in hardware costs is less likely to be the ideal solution that can be widely accepted. Occasionally, researchers may also hope their models have a certain degree of interpretability to adjust the errors more easily or even extract expert knowledge from the model. Fuzzy systems have clear advantages in this respect.

Traditionally, fuzzy systems rely on manually constructed rule-bases using expert knowledge or hypothetical mathematical models to deal with regression problems, but this train of thought has long been unable to meet the demand nowadays. Naturally, self-learning fuzzy models represented by neuro-fuzzy inference systems (NFIS) have become mainstream for current research, especially NFISs with deep network architectures. Apart from the depth, the most relevant factors determining the NFIS performance include its fuzzy logic and optimization algorithm. On the one hand, as the logic engine driving NFIS, the inference logic used in the model contributes fundamentally to the model's performance. Type-1 NFIS models that are first to emerge have obvious drawbacks because of the limited expressive ability of type-1 rules, i.e., a relatively large rule-base is often required to describe a complex problem, making it extremely inefficient and vulnerable to the curse of dimensionality when dealing with real-world situations with many variables. Furthermore, since type-1 logic can only represent re-

latively superficial information, this often leads to overfitting and poor generalization capability of the model. Subsequently, type-2 fuzzy logic [11], [39], especially interval type-2 logic [155], has been introduced to build NFIS, which significantly improves the accuracy, rule-base size, and generalization capability of NFIS models. Nevertheless, in recent years, the related architectures of type-2 fuzzy logic are also close to optimum. It is necessary to introduce a more powerful fuzzy logic if further improvements in the performance are required. Type-N fuzzy logic [3] and complex fuzzy logic [12], [13] are two promising options that follow this train of thought. However, neuro-fuzzy models based on type-N logic rarely mentioned in the literature. In contrast, more progress has been made for complex neuro-fuzzy systems [16].

On the other hand, appropriate optimization policies are also crucial for NFISs. There are two main categories [156] of solutions in general: backpropagation and hybrid optimization. Currently, the strategy of merely applying backpropagation is considered suboptimal [63], while most newly proposed NFIS models adopt the hybrid optimization tactic. The "rationale" behind the hybrid method is to optimize the antecedent and consequent parameters separately. Given that the antecedent part of the network includes nonlinear fuzzy membership functions, whereas the consequent part only exists linear mappings, the use of different methods can help facilitate faster training and obtaining the global optimum. The consequent part is relatively simple to train, for which the solutions include RLS [157], extended Kalman filter [158], ELM [159] or SVM (for classification purposes) [160]. For the training of the antecedent part, the gradient methods were popular in the early years, but recently researchers prefer meta-heuristic solutions, such as particle swarm [161], evolutionary algorithms [162], or artificial bee colonies [163]. Blindly opting for a derivative-free method may not be wise because the gradient is actually the prior knowledge of the optimization surface. The implementation of population methods ignores this vital information, which often leads to increased training complexity and decreased efficiency. Such a train of thought is unfavorable to real-world application scenarios that pursue concomitantly accuracy and efficiency.

In recent years, breakthroughs such as multi-parameter gradient-momentum methods have been made in the practices of gradient optimization [164]. In the meantime,

the theory of complex fuzzy sets [16] has also received increasing attention, which brings new possibilities to the NFIS. Complex fuzzy theory is considered a breakthrough after type-2 fuzzy theory. It extends the common domain of fuzzy membership to the entire unit circle in the complex plane, allowing a single rule to accommodate more information over previous fuzzy logic. Such a property is conducive to enhancing the generalization performance of the system and managing the size of the rule-base. Also, complex fuzzy logic can describe semantics that the previous fuzzy logic cannot represent, which helps to expand the application scenarios of fuzzy systems. The frequency-domain nature of complex numbers also makes CFT more effective in dealing with periodicity.

This chapter proposes a new rapid adaptive complex neural-fuzzy system (RACFIS) by redesigning the optimization framework of the CNFS [94], leading to a faster convergence and more accurate model that far exceeds its original counterpart. Note that the word "adaptive" in this definition means self-learning. Such an algorithm utilizes the two-dimensional character of the complex fuzzy rule-base, which allows a single rule to contain more information, enabling better generalization capability in scenarios of periodicity and stochasticity. Although the semantics of the complex fuzzy set is yet to reveal, this network model is still as transparent as other neural-fuzzy architectures, which may be meaningful in some specific situations. In terms of the optimization strategy, bisecting K-Means clustering (unsupervised learning) [165], Quasi-hyperbolic momentum [166], and RLS estimation together as a joint optimization strategy, assuring that the network can quickly converge to a better global minimum. Given that the so-called "curse of dimensionality" often leads to a sharp increase in computational complexity, the sparseness of information, and the weakening of non-linear connections between data points, a mechanism capable of accurately excluding insignificant variables is necessary for higher dimensional tasks. Thus, the mean impact value (MIV) [167] algorithm based on the RBF network is applied. Finally, the algorithm is tested on a synthetic dataset, a Sunspot time series dataset, and two metallurgical datasets. Experimental results show that RACFIS shows superior performance over other well-known benchmark algorithms.

5.2 Methodological Premises

5.2.1 Complex Fuzzy Set

Ramot *et al.* [12] created the original complex fuzzy theory by extrapolating the concept of fuzzy membership into the complex plane, for which the membership grade is defined as follows:

$$\phi_s(x) = \varphi_s(x) \cdot e^{j\omega_s(x)}, j = \sqrt{-1}, \tag{5.1}$$

where $\varphi_s(x)$ and $\omega_s(x)$ are respectively the modulus and the phase of the membership function. According to the definition, the degree of membership is restricted in the unit circle of the complex plane, which delimits $\varphi_s(x) \in [0,1]$, whereas $\omega_s(x)$ can be an arbitrary value. This theory also indicates that the ordinary type-1 fuzzy set is a case where the phase component of the complex fuzzy set is zero. As for the fuzzy inference logic of the above complex fuzzy sets, Ramot *et al.* explained it in the subsequent article [13] and clarified some basic operators for complex fuzzy reasoning.

Assume that there are two complex fuzzy sets A and B within the universe of discourse U, where set A is the aggregation of a series of sets $A_1, A_2, A_3, \cdots, A_n$. The operators for complement, union, intersection, and aggregation operations can be given respectively as equations (5.2) (5.3) (5.4) (5.5) as follows:

$$\phi_{\bar{A}}(x) = C[\varphi_A(x)] \cdot e^{(j\omega_{\bar{A}}(x))}, \tag{5.2}$$

$$\phi_{A\cup B}(x) = [\varphi_A(x) \bigoplus \varphi_B(x)] \cdot e^{(j\omega_{A\cup B}(x))}, \tag{5.3}$$

$$\phi_{A\cap B}(x) = [\varphi_A(x) \star \varphi_B(x)] \cdot e^{(j\omega_{A\cap B}(x))}, \tag{5.4}$$

$$\phi_A(x) = aggregate[\phi_{A_1}(x), \phi_{A_2}(x), \ldots, \phi_{A_n}(x)] = \sum_{i=1}^{n} \omega_i \varphi_{A_i}(x), \tag{5.5}$$

where $\bigoplus$ is said to be t-conorm, $\star$ is the representation of t-norm. In abstract mathematics, a t-norm denotes intersection in a lattice and conjunction in logic, whereas a t-conorm behaves as a disjunction logic or a union operator. $\omega_{A\cup B}(x)$ and $\omega_{A\cap B}(x)$ can be calculated in numerous ways, such as the sum of $\omega_A(x)$ and $\omega_B(x)$, the maximum or the minimum between these two. The aggregation operator is identical to

the inner product of a vector, which suggests that Ramot's complex fuzzy membership degree shares a vector property.

Dick [14] introduced the notion of lattice [24] into the definition of the theory of complex fuzzy sets. According to Dick's interpretation, complex fuzzy sets can be divided into two categories, i.e., the ones with rotational invariance and without rotational invariance. A complex fuzzy set is considered as rotationally invariant if and only if function $L : \Gamma \times \Gamma \to \Gamma, L(pe^{j\alpha} \cdot e^{jk}, qe^{j\beta} \cdot e^{jk}) = e^{jk} \cdot L(pe^{j\alpha}, qe^{j\beta})$ stands for all elements $pe^{j\alpha}, qe^{j\beta} \in \Gamma$, in which Γ is the lattice where this CFS belongs to. Dick also noticed that possible candidates for implication operators, including conjunction, disjunction, and negation, are conspicuously restricted if a CFS is rotationally invariant. For instance, the algebraic product should not be used as the conjunction operator for a CFS with rotational invariance since the algebraic product does not satisfy the definition of rotational invariance. Instead, when it comes to a CFS without the rotational invariance, the algebraic product may well apply to conjunction operations.

5.2.2 Bisecting K-Means

The primeval k-means algorithm was first proposed by Lloyd [168] and Kanungo *et al*, respectively. [169]. To divide a dataset into k clusters, the basic k-means first initializes k centroids $\xi_1, \xi_2, \ldots, \xi_k$, then computes the distance between each point x_i and the centroid ξ_j. Every point is assigned to the nearest cluster according to the point-to-center distance. Therefore, the updated centroids are as follows:

$$\xi_j = \frac{1}{|c_j|} \sum_{x \in c_j} x \tag{5.6}$$

where c_j denotes the jth cluster and $j = 1, 2, \ldots, k$. The algorithm can realize the optimized clusters by repeating the above process when the stopping condition is satisfied. The traditional k-means algorithm has a manifest flaw in that manual determination of the cluster centers is required, but manually selected centroid is not often precise enough, which offsets the accuracy. Therefore, the k-means++ algorithm [170] is developed with the capability of autonomously positioning the centers. This method

randomly selects cluster centers, then refines the centroids through multiple iterations to obtain the optimal clustering scheme. However, this solution causes randomness as well as local optima.

Steinbach *et al*. [165] introduced the idea of hierarchical clustering into the modification of basic k-means, leading to a new top-down clustering algorithm named bisecting k-means. Akin to k-means++, this algorithm also stochastically chooses clustering centers, but it effectively avoids local optima with the help of a binary decision tree. This algorithm first regards all data as a big cluster and dichotomizes it into two smaller ones using ordinary k-means. Then the cluster with the larger SSE (sum of the squared errors) is split again according to the maximum SSE principle. Finally, the above procedure is repeated until the kth independent cluster emerges. Although the result still has a certain degree of randomness, it can ensure that the outcome is closer to the true global optimum. Bisecting k-means is a fast and efficient method. The computational speed is faster than traditional k-means or k-means++, especially when k is large. A complete demonstration of the bisecting k-means is as follows:

Algorithm 5.1 Bisecting K-Means.

Step 1. Manually set up the number of clusters k. (In this chapter, k must be identical to the number of fuzzy rules.)
Step 2. Initialize the objective data set, and generalize a cluster that contains all data points.
Step 3. Bisect the cluster using basic k-means $(k = 2)$ by randomly opting for two centroids.
Step 4. Estimate the SSE of two clusters, and pick up the one with the larger SSE as the next target to split.
Step 5. Go back to Step 3 and carry out such steps repeatedly until the overall cluster number equals k.

5.2.3 Quasi-Hyperbolic Momentum (QHM) Optimization

Currently, there are two types of popular gradient optimization methods. One is the stochastic gradient methods such as SGD [171] and Adam [172]. The other is the gradient-momentum method, such as heavy-ball [173] and Nesterov [174]. The above approaches usually have two hyperparameters to determine, one is the learning

rate, and the other is related to the momentum. Hu *et al.* [175] compared the first-order optimization method with the classic control theory, confirming that the gradient optimization process is essentially a PID controller. A PID controller has three linear independent parameters, namely proportional (P), integral (I), and derivative (D) terms. The essence of the momentum is the influence of historical gradients on the current process, which is analogous to the function of a lead-lag compensator in a PID control problem. This finding reveals that first-order gradient methods can possess additional degrees of freedom for parameters, leading to a new train of thought regarding the design of such algorithms. Subsequently, the multi-parameter gradient-momentum optimization methods are receiving attention since they may be better adapted to the optimization surface, especially when it is non-convex, thereby improving the algorithm's performance.

The quasi-hyperbolic momentum [166] algorithm is one of them, where the learning rate α, the immediate discount factor v, and the momentum discount factor β constitute the hyperparameter space of the algorithm. Note that one can rewrite the QHM as a PID controller by considering β as a free variable. The idea of quasi-hyperbolic momentum originates from an informal and conjectural variance reduction analysis, for which a plain explication is an average of the momentum and regular gradient drop weighted by v. The ordinary definition of the heavy-ball momentum firmly relates the exponential discount factor β to the contribution of the immediate gradient. But the role played by the QHM directly comes from decoupling the momentum from the impact of the current gradient value (with deterministic setting), as well as decoupling the gradient square mean from the current gradient square (with stochastic settings) when updating the weights. This method is believed to converge faster than the traditional heavy ball gradient-momentum method with a smoother process, while the sensitivity to individual parameters diminished, reducing the difficulty of parameter tuning. The update rule for the QHM optimization using the deterministic setting is as follows:

$$g(k+1) = \beta g(k) + (1-\beta)\nabla f(x(k)) \tag{5.7}$$

$$x(k+1) = x(k) - \alpha[(1-v)\nabla f(x(k)) + v g(k+1)], \tag{5.8}$$

where $\alpha, \beta, v \in R, \nabla f(x(k))$ represent the first-order derivative of the optimization object,$(1-v)\nabla f(x(k)) + vg(k+1)$ denotes the modulo initialization bias which is a gradient estimator in short, and $g(k)$ can be viewed as the momentum buffer. Under normal circumstances, the value of v is 0.7 and the value of β is 0.999, which means that in most cases such algorithm can be applied like a single parameter method by only adjusting the learning rate.

5.2.4 Recursive Least Square (RLS) Estimation

Traditionally, the consequent part of a Sugeno fuzzy system is updated utilizing the offline least square (LS) method, which is less effective than the online method. Although the Kalman filter can also be a solution to update the parameters, in the absence of any additional knowledge of the target, this only increases the computational complexity without leading to any significant benefits. Therefore, the RLS that satisfies the requirement for online learning, as well as computational efficiency, is employed here as the updating method for the consequent part of the RACFIS network. If given the following model equation:

$$Y_L = \phi_L \hat{\theta} + \omega_L, \tag{5.9}$$

where ϕ_L is the system matrix, $\hat{\theta}$ is the input vector, Y_L is the output vector, and ω_L is the vector of system error or noise. To estimate $\hat{\theta}$ at the kth recursion, according to the principle of the least square estimation, the following equation set is obtained:

$$\begin{cases} \hat{\theta}(k) = (\phi_k^T \phi_k)^{-1} \phi_k^T Y_k \\ \phi_k = [\phi_{k-1}^T, \varphi(k)]^T \\ Y_k = [Y_{k-1}^T, y(k)]^T, \end{cases} \tag{5.10}$$

where $\varphi(k)$ and $y(k)$ are the system vector and output vector for the kth input-output pair, respectively. Here, it is specified that the equation starts from $\varphi(0)$, i.e., $k = 0, 1, \ldots, n-1$. Further, the following equation can be easily obtained by defining $P(k) = (\phi_k^T \phi_k)^{-1}$:

$$P(k) = [P^{-1}(k-1) + \varphi(k)\varphi^T(k)]^{-1}. \tag{5.11}$$

Substitute equations (5.10) and (5.11) into the matrix inverse formula $(A+BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1}+DA^{-1}B)^{-1}DA^{-1}$, where $A = P^{-1}(k-1), C = I, B = \varphi(k)$, and $D = \varphi^T(k)$. Then, the following equation can be achieved:

$$P(k) = \left[I - \frac{P(k-1)\varphi(k)\varphi^T(k)}{1+\varphi^T(k)P(k-1)\varphi(k)}\right] P(k-1), \tag{5.12}$$

Given that $\hat{\theta}(k) = (\phi_k^T\phi_k)^{-1}\phi_k^T Y_k$ and $P^{-1}(k)\hat{\theta}(k) = \phi_k^T Y_k$, therefore the ultimate form of $\hat{\theta}$ at the kth recursion is:

$$\hat{\theta}(k) = \hat{\theta}(k-1) + P(k)\varphi(k)[y(k) - \varphi^T(k)\hat{\theta}(k-1)], \tag{5.13}$$

For the implementation of RLS estimation, equations (5.12) and (5.13) constitute the update rule. Regarding the initialization of the algorithm, $\varphi(0)$ is set to be zero vector while the start of P is given as $P(0) = \alpha I$, where α usually takes a large positive number, I is the identity matrix.

5.3 Methodology

5.3.1 RACFIS Overview

The RACFIS network uses complex fuzzy logic for fuzzy inference and uses the Sugeno method as the inference engine for defuzzification. It is worth noting that although the algorithmic process of RACFIS involves complex signals, it is limited to the antecedent part of the fuzzy inference system. Once the defuzzification is completed, the network signals become real-valued before the consequent operation is performed to determine the final output of the architecture. The input of the RACFIS network is completely real-valued and the same as the network output. The complex-valued terms are fuzzy membership values that contain the information of the data but do not directly react with the network input or output. Therefore, from the function mapping point of view, RACFIS still belongs to the category of real-valued function mapping, which is fundamentally different from the models designed for complex-valued inputs or out-

puts. Assume the objective network has k inputs and one output, then the ith fuzzy rule can be represented as follows:

$$Rule\ \ i:\ IF\ \ l_1\ is\ \ A_1^i(x_1)\ and\ \ l_2\ is\ \ A_2^i(x_2),\ldots,\ and\ \ l_k\ is\ \ A_k^i(x_k).$$

According to the defuzzification strategy of the Sugeno method, the output z^i of this rule is as follows:

$$z^i = a_0^i + \sum_{j=1}^{k} a_j^i x_j, i = 1,2,\ldots,n, \tag{5.14}$$

where$A_j^i(x_j)$ is the jth antecedent of the ith fuzzy rule, a_j^i is the corresponding consequent parameter. The three-parameter gradient momentum method, i.e., Quasi-hyperbolic momentum, is used to optimize the antecedent parameters, while the recursive least square is the optimization policy for the consequent part. Moreover, Bisecting K-Means as an unsupervised learning method is applied to pre-train the antecedent parameters. The RBFN-based MIV algorithm is also utilized to select the variables to mitigate the curse of dimensionality. A general schematic diagram of the RACFIS is given in Figure 5.1.

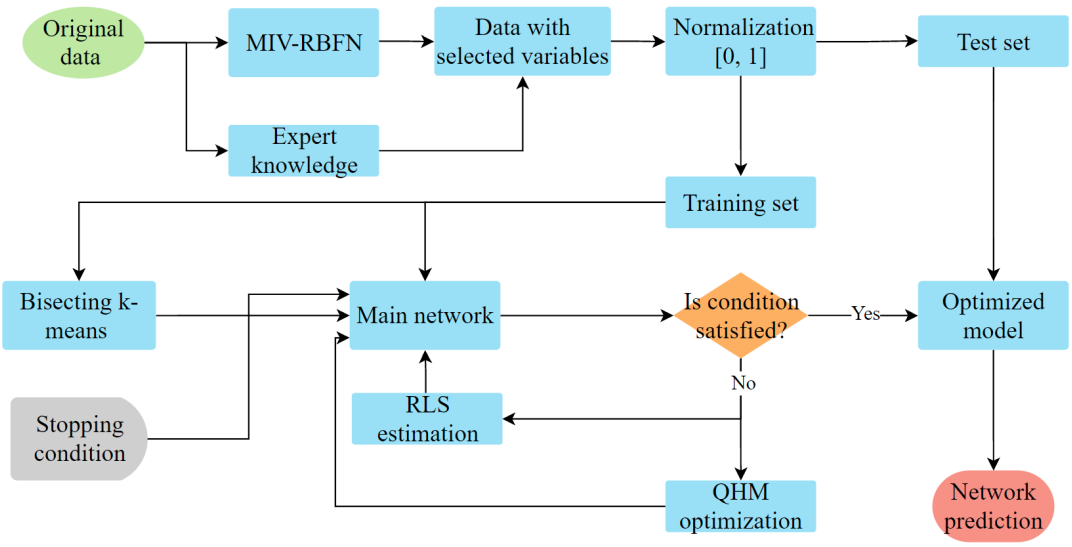

Figure 5.1: RACFIS algorithm flow.

5.3.2 Network Structure of RACFIS

The RACFIS network is a five-layer structure as shown in Figure 5.2. The operation of each layer of the network with the input vector $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^T$ and output $y(t)$ at time t is presented as follows:

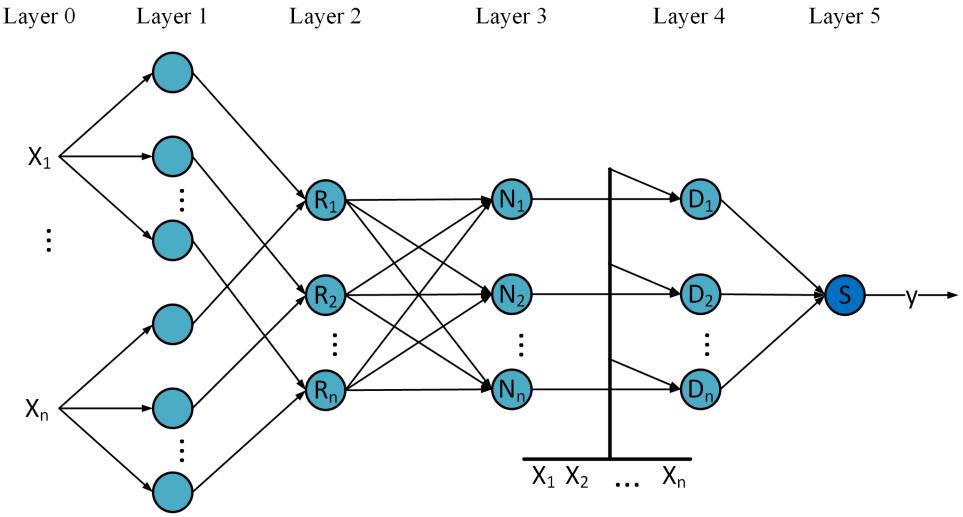

Figure 5.2: The illustration of the main structure of RACFIS. X_i in the figure represents input variable, R_i denotes a complex fuzzy rule, N_i is the normalization of the product of each rule, D_i is the outcome of Sugeno defuzzification, and S denotes the sum of all results which is identical to the output of the network, i.e., y.

Layer 1: It is the fuzzification layer in which the real-valued inputs are transferred into fuzzy membership grades with the simplified complex Gaussian membership function as follows:

$$O^i_{1,j}(t) = \exp\left(-\frac{(x(t)-\mu^i_j)^2}{2b^i_j}\right) - j\exp\left(-\frac{(x(t)-\mu^i_j)^2}{2b^i_j}\right)\frac{x(t)-\mu^i_j}{b^i_j}\delta^i_j, \tag{5.15}$$

where $O^i_{1,j}(t)$ denotes the membership of the ith rule of the jth input, and $\{\mu^i_j, b^i_j, \delta^i_j\}$ is the antecedent parameter set for each rule.

Layer 2: This layer is for calculating the firing strength of the inference system, in which a complex multiplication is applied as follows:

$$O^i_2(t) = \prod_{j=1}^{n} O^i_{1,j}(t) =: \alpha^i_j(t) + j\beta^i_j(t), \tag{5.16}$$

$$\alpha_j^i(t) = \exp\left(-\sum_{j=1}^{n}\frac{(x_j(t)-\mu_j^i)^2}{2b_j^i}\right)\left[1-\prod_{j=1}^{n}\frac{(x_j(t)-\mu_j^i)}{b_j^i}\delta_j^i\right], \tag{5.17}$$

$$\beta_j^i(t) = -\exp\left(-\sum_{j=1}^{n}\frac{(x_j(t)-\mu_j^i)^2}{2b_j^i}\right)\sum_{j=1}^{n}\frac{x_j(t)-\mu_j^i}{b_j^i}\delta_j^i, \tag{5.18}$$

where $O_2^i(t)$ is the strength of the ith firing rule. $\alpha_j^i(t)$ and $\beta_j^i(t)$ represent the value of real part and imaginary part of the fuzzy membership grade, respectively.

Layer 3: As the normalization layer for relevant firing strengths, the complex division performs the normalization operation so that the two dimensions of information can fully interact. For simplicity, equations (5.17) and (5.18) are used here to simplify the expression:

$$O_3^i(t) = \frac{O_2^i(t)}{\sum_{r=1}^{K} O_2^r(t)} = \frac{\alpha^i(t)+j\beta^i(t)}{\sum_{r=1}^{K}\alpha^r(t)+j\sum_{r=1}^{K}\beta^r(t)}, \tag{5.19}$$

where $O_3^i(t)$ is the normalized value of the ith node in this layer.

Layer 4: Each node in layer 4 is an adaptive node, which utilizes the Sugeno method to calculate the output of each fuzzy reasoning with the consequent parameters:

$$O_4^i(t) = O_3^i(t)\cdot(p_0^i + \sum_{j=1}^{n} p_j^i x_j(t)), \tag{5.20}$$

where $O_4^i(t)$ is the output of each node and $\{p_0^i, p_1^i, p_2^i, \ldots, p_n^i\}$ is the set of consequent parameters.

Layer 5: Only one single node is in this layer, of which the function is to obtain the real-valued overall output of the network. Here, only the real part of the complex output is utilized.

$$O_5(t) = \mathrm{Re}\sum_{i=1}^{K} O_4^i(t), \tag{5.21}$$

where "Re" means the real part of the complex number, $O_5(t)$ is the overall output of the neuro-fuzzy system and K is the number of fuzzy rules assigned for each input. To perform a real value regression task, it would be incorrect if the output result is a complex number. Thus, an operation to convert the complex-valued inference result defined in the complex plane into the real-valued output is necessary, and it is also essentially a

step of CFS defuzzification. Some methods have emerged to achieve this goal, including the dot product (ANCFIS), the projection on the real axis (CNFS), or the modulo operation. For a complex fuzzy membership function defined in a polynomial form, its real axis projection equals the value of the real component, making it the best candidate to be the final defuzzified output. Additionally, employing the real component as a network result does not change the natural mapping relation between the antecedent and the outcome, which enables the possibility of using the unsupervised convex clustering method to pre-train the antecedent parameters.

Similarly to many other neural networks based on Gaussian functions, the mean square error (MSE) function is used as the cost function for network optimization. There are three reasons for this choice. First, the use of the MSE function is convenient since we can easily find its derivative. Second, assuming that the error between the predicted output and the real output complies with a Gaussian distribution, the minimum MSE error is essentially consistent with their maximum likelihood estimation [176], which makes the MSE the ideal candidate for measuring the loss of a regression process. Finally, the MSE cost function generally leads to faster convergence. Assume that there are N input-output pairs $\{x_1(t), x_2(t), \ldots, x_n(t), y(t)\}_{t=1}^{N}$, the corresponding MSE function is as follows:

$$\text{MSE} = \frac{1}{2N}\sum_{t=1}^{N}(O_5(t) - y(t))^2, \tag{5.22}$$

It is worth mentioning that from the control theory point of view, RACFIS is an open-loop system. Although a back-propagation algorithm is applied to optimize the parameters, it does not cause any impact on the input signal and therefore does not constitute closed-loop feedback.

5.3.3 Network Propagation and Gradient of RACFIS

The training process of RACFIS has two phases akin to all backpropagation networks, i.e., forward propagation and backpropagation. For the forward propagation stage, assume the network applies the complex Gaussian membership function, and the input-

output set $\{x_1(t), x_2(t), \ldots, x_n(t), y(t)\}_{t=1}^{N}$, the following equation can be obtained:

$$g_{1,j}^{i}(t) = \exp\left(-\frac{(x(t)-\mu_j^i)^2}{2b_j^i}\right) - j\exp\left(-\frac{(x(t)-\mu_j^i)^2}{2b_j^i}\right)\frac{x(t)-\mu_j^i}{b_j^i}\delta_j^i, \tag{5.23}$$

where $g_j^i(t)$ denotes the membership of the ith rule of the jth input, and $\left\{\mu_j^i, b_j^i, \delta_j^i\right\}$ are the antecedent parameters for each rule. One can calculate the firing strength of the inference system using complex multiplication, which is obtained in the form of the following equation set:

$$\begin{cases} \beta^i(t) = \prod_{j=1}^{n} g_j^i(t) =: u_j^i(t) + jv_j^i(t) \\ u_j^i(t) = w^i(t)\left[1 - \prod_{j=1}^{n}\frac{\left(x_j(t)-\mu_j^i\right)}{b_j^i}\delta_j^i\right] \\ v_j^i(t) = -w^i(t)\sum_{j=1}^{n}\frac{x_j(t)-\mu_j^i}{b_j^i}\delta_j^i \\ w^i(t) = \exp\left(-\sum_{j=1}^{n}\frac{\left(x_j(t)-\mu_j^i\right)^2}{2b_j^i}\right), \end{cases} \tag{5.24}$$

where $\beta^i(t)$ is the strength of the ith firing. $u_j^i(t)$ and $v_j^i(t)$ represent the value of real part and imaginary part of the fuzzy membership grade, respectively. Thus, the normalized firing strength $\gamma^i(t)$ can be further derived as follows:

$$\begin{gathered} \gamma^i(t) = \frac{\beta^i(t)}{\sum_{r=1}^{K}\beta^r(t)} = \frac{u^i(t)+jv^i(t)}{\sum_{r=1}^{K}\mu^r(t) + j\sum_{r=1}^{K}v^r(t)} = \\ \frac{u^i(t)\sum_{r=1}^{K}\mu^r(t) + v^i(t)\sum_{r=1}^{K}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} + j\frac{v^i(t)\sum_{r=1}^{K}\mu^r(t) - u^i(t)\sum_{r=1}^{K}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} \end{gathered} \tag{5.25}$$

Then, the complex-valued inference outcome is obtained as follows:

$$\begin{gathered} \xi(t) = \sum_{i=1}^{K}\xi_i(t) = \sum_{i=1}^{K}\gamma^i(t)\left(a_0^i + \sum_{j=1}^{n}a_j^i x_j(t)\right) \\ = \sum_{i=1}^{K}\gamma_R^i(t)\left(a_0^i + \sum_{j=1}^{n}a_j^i x_j(t)\right) + j\sum_{i=1}^{K}\gamma_I^i(t)\left(a_0^i + \sum_{j=1}^{n}a_j^i x_j(t)\right) \end{gathered} \tag{5.26}$$

Finally, the cost function of the network is obtained at the end of the feedforward phase:

$$Loss = \frac{1}{2N}\sum_{t=1}^{N}(\mathrm{Re}\,\xi(t) - y(t))^2 =: \frac{1}{2N}\sum_{t=1}^{N}e(t)^2 \tag{5.27}$$

Regarding the backpropagation phase, the partial derivatives of $u_i(t)$ and $v_i(t)$ are firstly derived. For $i = 1, \ldots, K, j = 1, 2, \ldots, n$, one can acquire the following first-order partial derivatives:

$$\begin{aligned}
\frac{\partial u^i(t)}{\partial \mu_j^i} &= u^i(t)\frac{x_j(t)-\mu_j^i}{b_j^i}\delta_j^i + \frac{w^i(t)}{\left(x_j(t)-\mu_j^i\right)}\prod_{j=1}^{n}\frac{\left(x_j(t)-\mu_j^i\right)}{b_j^i}\delta_j^i,\\
\frac{\partial v^i(t)}{\partial \mu_j^i} &= v^i(t)\frac{x_j(t)-\mu_j^i}{b_j^i}\delta_j^i + \frac{w^i(t)}{b_j^i}\delta_j^i,\\
\frac{\partial u^i(t)}{\partial b_j^i} &= u^i(t)\frac{\left(x_j(t)-\mu_j^i\right)^2}{2\left(b_j^i\right)^2}\delta_j^i + \frac{w^i(t)}{b_j^i}\prod_{j=1}^{n}\frac{\left(x_j(t)-\mu_j^i\right)}{b_j^i}\delta_j^i,\\
\frac{\partial v^i(t)}{\partial b^i} &= v^i(t)\frac{\left(x_j(t)-\mu_j^i\right)^2}{2\left(b_j^i\right)^2}\delta_j^i + w^i(t)\frac{x_j(t)-\mu_j^i}{\left(b_j^i\right)^2}\delta_j^i,\\
\frac{\partial u^i(t)}{\partial \delta_j^i} &= -\frac{w^i(t)}{\delta_j^i}\prod_{j=1}^{n}\frac{\left(x_j(t)-\mu_j^i\right)}{b_j^i}\delta_j^i,\\
\frac{\partial v^i(t)}{\partial \delta_j^i} &= -w^i(t)\frac{x_j(t)-\mu_j^i}{b_j^i},
\end{aligned} \tag{5.28}$$

Then, the partial derivative $\frac{\partial e(k)}{\partial \mu_j^i}$ can be written in the following form:

$$\begin{aligned}
\frac{\partial e(t)}{\partial \mu_j^i} = &\left\{\frac{\frac{\partial u^i(t)}{\partial \mu_j^i}\sum_{r=1}^{K}\mu^r(t) + \frac{\partial v^i(t)}{\partial \mu_j^i}\sum_{r=1}^{K}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} + \sum_{r=1}^{K}\frac{\frac{\partial u^i(t)}{\partial \mu_j^i}\mu^r(t) + \frac{\partial v^i(t)}{\partial \mu_j^i}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2}\right.\\
&\left. - 2\frac{\frac{\partial u^i(t)}{\partial \mu_j^i}\mu^r(t) + \frac{\partial v^i(t)}{\partial \mu_j^i}v^r(t)}{\left[\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2\right]^2}\sum_{r=1}^{K}\left[\mu^r(t)\sum_{r=1}^{K}\mu^r(t) + v^r(t)\sum_{r=1}^{K}v^r(t)\right]\right\}\\
&\left(a_0^i + \sum_{j=1}^{n}a_j^i x_j(t)\right),
\end{aligned} \tag{5.29}$$

$\frac{\partial e(t)}{\partial b_j^i}$ can be obtained by replacing $\frac{\partial \mu^i(t)}{\partial \mu_j^i}$ and $\frac{\partial v^i(t)}{\partial \mu_j^i}$ in $\frac{\partial e(t)}{\partial \mu_j^i}$ with $\frac{\partial u^i(t)}{\partial b_j^i}$ and $\frac{\partial v^i(t)}{\partial b^i}$:

$$\frac{\partial e(t)}{\partial b_j^i} = \left\{ \frac{\frac{\partial u^i(t)}{\partial b_j^i}\sum_{r=1}^{K}\mu^r(t) + \frac{\partial v^i(t)}{\partial b_j^i}\sum_{r=1}^{K}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} + \sum_{r=1}^{K}\frac{\frac{\partial u^i(t)}{\partial b_j^i}\mu^r(t) + \frac{\partial v^i(t)}{\partial b_j^i}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} \right.$$
$$\left. -2\frac{\frac{\partial u^i(t)}{\partial b_j^i}\mu^r(t) + \frac{\partial v^i(t)}{\partial b_j^i}v^r(t)}{\left[\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2\right]^2}\sum_{r=1}^{K}\left[\mu^r(t)\sum_{r=1}^{K}\mu^r(t) + v^r(t)\sum_{r=1}^{K}v^r(t)\right] \right\}$$
$$\left(a_0^i + \sum_{j=1}^{n}a_j^i x_j(t)\right), \tag{5.30}$$

Similarly, one can also obtain partial derivative $\frac{\partial e(k)}{\partial \delta_j^i}$:

$$\frac{\partial e(t)}{\partial \delta_j^i} = \left\{ \frac{\frac{\partial u^i(t)}{\partial \delta_j^i}\sum_{r=1}^{K}\mu^r(t) + \frac{\partial v^i(t)}{\partial \delta_j^i}\sum_{r=1}^{K}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} + \sum_{r=1}^{K}\frac{\frac{\partial u^i(t)}{\partial \delta_j^i}\mu^r(t) + \frac{\partial v^i(t)}{\partial \delta_j^i}v^r(t)}{\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2} \right.$$
$$\left. -2\frac{\frac{\partial u^i(t)}{\partial \delta_j^i}\mu^r(t) + \frac{\partial v^i(t)}{\partial \delta_j^i}v^r(t)}{\left[\left(\sum_{r=1}^{K}\mu^r(t)\right)^2 + \left(\sum_{r=1}^{K}v^r(t)\right)^2\right]^2}\sum_{r=1}^{K}\left[\mu^r(t)\sum_{r=1}^{K}\mu^r(t) + v^r(t)\sum_{r=1}^{K}v^r(t)\right] \right\}$$
$$\left(a_0^i + \sum_{j=1}^{n}a_j^i x_j(t)\right). \tag{5.31}$$

The above equations (5.29), (5.30), and (5.31) are partial derivatives of the network output error regarding the antecedent parameters, which are also the directional derivatives of the optimization surface for the RACFIS network, i.e., the gradient:

$$\nabla f = \begin{bmatrix} \frac{\partial e(t)}{\partial \mu} \\ \frac{\partial e(t)}{\partial b} \\ \frac{\partial e(t)}{\partial \delta} \end{bmatrix}. \tag{5.32}$$

5.3.4 Joint Optimization Strategy for RACFIS

The optimization policy for the RACFIS model is a joint method utilizing bisecting k-means, QHM, and RLS. A key premise for this method to work is that all data should be normalized within the interval $[0,1]$ before feeding into the model because different scales between data variables can cause problems in the deep architecture backpropagation, such as gradient explosion or gradient vanishing. It is worth noting that one should determine the training and test sets before the normalization and normalize the two individually. If the training and test sets are determined after the normalization, the training set may carry some data distribution information from the test set, resulting in a less rigorous evaluation of the model. After the normalization process, bisecting k-means clustering is used to pre-train the antecedent part of the network to have a faster convergence and a better result closer to the true global optimum. $\{\mu^i, b^i, \delta^i\}, i = 1,2,\ldots,N$ is the set of the antecedent parameters, where μ^i is the center and b^i is the width of each complex Gaussian membership function. One can obtain a set of cluster centroids $C = \{c_1, c_2, \ldots, c_i, \ldots, c_N\}$ by applying bisecting k-means to the entire training set. Such centroids can be further utilized to calculate the sum of the distances from all points in each cluster to the centroid denoted as $D = \{d_1, d_2, \ldots, d_i, \ldots, d_N\}$, where $d_i = \sum_{k=1}^{n}(x_i^k - c_i)$. Thus, the pre-train process of antecedent parameters $\{\mu^i, b^i, \delta^i\}$ is over, and the values can be determined as follows:

$$\mu^i = c_i, b^i = \rho \sum_{k=1}^{n}(x_i^k - c_i), \delta^i = 1, \tag{5.33}$$

where x_i^k is the coordinate of each individual data point in the ith cluster. ρ denotes the expansion coefficient, which is default to be 0.95. n is the number of points in this cluster.

As a result of clustering, the network is initialized at a position closer to the global minimum than using a random initialization, but a gap between the current status and the optimal still exists. Hence, an error backpropagation solution is required to refine the parameters, and here the QHM gradient optimization is employed. To implement QHM, combine the equations (5.7) and (5.8), and the following expression of the up-

date rule is achieved as follows:

$$x(k+1) = x(k) - \alpha[(1-v\beta)\nabla f(x(k)) + v\beta g(k)], \tag{5.34}$$

where α is the initial learning rate, β is the exponential discount factor, v is the immediate discount factor, $g(k) = \beta g(k-1) + (1-\beta)\nabla f(x(k-1))$ is the momentum buffer of each parameter, and $e(k) = \text{Output}(k) - y(k)$ is the error of the kth iteration. As mentioned above, each rule has three different antecedent parameters$\{\mu_j^i, b_j^i, \delta_j^i\}$. For each step of the iteration, the updated parameters can be calculated according to the equation (5.34). The expression for the parameter set at iteration $k+1$ is as follows:

$$\begin{cases} \mu^i(k+1) = \mu^i(k) - \alpha[(1-v\beta)e(k)\left.\frac{\partial e(k)}{\partial \mu^i}\right|_{\mu^i=\mu^i(k)} + v\beta g_\mu^i(k)] \\ b^i(k+1) = b^i(k) - \alpha[(1-v\beta)e(k)\left.\frac{\partial e(k)}{\partial b^i}\right|_{b^i=b^i(k)} + v\beta g_b^i(k)] \\ \delta^i(k+1) = \delta^i(k) - \alpha[(1-v\beta)e(k)\left.\frac{\partial e(k)}{\partial \delta^i}\right|_{\delta^i=\delta^i(k)} + v\beta g_\delta^i(k)]. \end{cases} \tag{5.35}$$

After updating the non-linear antecedent parameters by QHM, the linear consequent part also requires a refresh using RLS. For $i = 1, 2, \ldots, q$, to update the consequent part at kth iteration, the following two variables are derived:

$$\begin{cases} \hat{\theta} = [(\theta^1)^T, (\theta^2)^T, \ldots, (\theta^q)^T] \\ \varphi(k) = [(\varphi^1(k))^T, (\varphi^2(k))^T, \ldots, (\varphi^q(k))^T], \end{cases} \tag{5.36}$$

where $\theta^i = [a_0^i, a_1^i, \ldots, a_q^i]^T$ and

$$\varphi^i(k) = [\lambda_R^i, \chi_1(k)\lambda_R^i, \ldots, \chi_{q-1}(k)\lambda_R^i]^T. \tag{5.37}$$

λ_R^i is the real part output of the corresponding normalization node. By substituting $\hat{\theta}$ and $\varphi(k)$ into equations (5.12) and (5.13), the updated parameters can be determined. Such parameters constitute the new weights of the consequent part of the network for the next iteration. To obtain the minimum error measured by the MSE objective function (5.22), the clustering process only needs to be implemented once, whereas the QHM-RLS refining process is supposed to be repeated iterations until the model satisfies

the error requirement or reaches the maximum epoch setting. The entire parameter adapting operation is shown as Algorithm 5.2.

Algorithm 5.2 Bisecting K-Means, QHM and RLS joint optimization.

Step 1. Split the data set into a training set and a testing set.
Step 2. Normalize the data set to the interval [0, 1]. Note that one should normalize the training set and the testing set separately.
Step. 3 Initialize the antecedent part of the parameter set $\{\mu^i, b^i, \delta^i\}$ using the information obtained by clustering.
Step. 4 Calculate the output of the architecture with the original settings, and derive the error by $e(k) = \text{Output}(k) - y(k)$.
Step. 5 Refine the antecedent part (non-linear) for the network by the QHM optimization method.
Step. 6 Estimate the consequent part (linear) with RLS estimation.
Step. 7 Calculate the output of the architecture, derive the error by$e(k) = \text{Output}(k) - y(k)$.
Step. 8 Repeat Step. 5 Step. 7, until the stopping criterion is satisfied.

5.3.5 MIV-RBFN Algorithm for Variable Analysis

Traditionally, the principal component analysis (PCA) [177]–[179] method is applied to select variables. However, as a linear method, it does not perform well on highly non-linear data sets. Kernel PCA [180] is an extension of the conventional PCA, which can linearize non-linear features by mapping them into the Hilbert space via the kernel functions. Although this method is more effective with non-linear data, difficulties still exist when determining the suitable parameter configurations, in the absence of prior knowledge of the data set, due to the need to manually adjust the settings.

For RACFIS, a MIV [167] algorithm integrated with the RBF network is employed to evaluate the variables for RACFIS when it comes to real-world data. The RBF network has an intuitive structure, easy training, and fast convergence. It can approximate arbitrary non-linear functions and shares inherent similarities with neuro-fuzzy systems, making it a suitable candidate as a referee network for the variable selection algorithm of the RACFIS network. The MIV algorithm here is not a dimensionality reduction algorithm, as it does not change the manifold of the data itself. Instead, it outputs a score for each variable to measure the relevance of this dimension in a specific case,

which generally relies on a pre-trained neural network as a supervisor. The algorithm judges the importance of variables by comparing the difference between the two network output vectors. One of the two vectors is achieved by an input vector that is a certain percentage larger in value than the original input, and the other is obtained by an input vector that is the same proportion smaller. The formula for calculating MIV is as follows:

$$MIV = \frac{abs\,|ref_{\text{increase}} - ref_{\text{decrease}}\,|}{N} \tag{5.38}$$

where ref_{increase} and ref_{decrease} denote referee network output under the two input vectors mentioned above, and N is the number of input vectors. When implementing the algorithm, one only needs to train the referee network without the involvement of prior knowledge of the data. The mean impact value is considered one of the most effective indicators for evaluating variable importance. The complete implementation process is given by Algorithm 5.3.

Algorithm 5.3 MIV-RBFN Algorithm.

Step 1. Initialize an RBF network, train the network with original input set $\chi^T = \{X_1, X_2, \ldots, X_q\}, q \in N^+$, where $X_k^T = \{x_1^k, x_2^k, \ldots, x_i^k, \ldots, x_n^k\}, k = 1, 2, \ldots, q, i = 1, 2, \ldots, n; n$ is the dimension of an input vector.

Step 2. Gradually increase the number of RBF kernels until the network accuracy reaches the setting value, denote the optimized network structure as a function $G_{op}(X, a)$, where x is the input set, and a is the parameter vector.

Step 3. Generate two new input sets, where $\chi_{increase}^T = \chi^T \cdot (1 + \eta)$ and $\chi_{decrease}^T = \chi^T \cdot (1 - \eta)$ and η is the adjustment rate.

Step 4. Run the network $G_{op}(X, a)$ with inputs $\chi_{increase}^T$ and $\chi_{decrease}^T$ respectively, obtain the output vectors $G_{op}(\chi_{increase}^T, a)$ and $G_{op}(\chi_{decrease}^T, a)$.

Step 5. Calculate the MIV of each variable: $MIV(x_i) = abs[G_{op}(\chi_{increase}^T, a) - G_{op}(\chi_{decrease}^T, a))]/q$.

5.4 Experimental Results and Analyses

5.4.1 Input-to-Output Numerical Example for RACFIS

The numerical demonstration of the RACFIS network uses a sample selected from the normalized Sunspot time series data, i.e., $x_1 = 0.0497, x_2 = 0.0476$ along with its label $y = 0.0461$. A RACFIS model with two inputs, one output, and three divisions is trained on the normalized Sunspot training set to solve this problem, for which the detailed parameter information is available as follows:

$$
\begin{aligned}
&\mu_1^1 = 0.424551458642240, \quad \mu_2^1 = 0.150213429327331, \quad \mu_3^1 = 0.720083762391771,\\
&\mu_1^2 = 0.423668503213857, \quad \mu_2^2 = 0.149835747936988, \quad \mu_3^2 = 0.720079920080232,\\
&b_1 = 1.46094548410898, \quad b_2 = 11.8157142115210, \quad b_3 = 1.46653839221557,\\
&\delta_1^1 = 0.999999999999999, \quad \delta_2^1 = 0.9999999999999991, \quad \delta_3^1 = 0.999999999999977,\\
&\delta_1^2 = 0.999999999999999, \quad \delta_2^2 = 0.999999999999995, \quad \delta_3^2 = 1.0000000000000,\\
&p_1^0 = 14.2524658415543, \quad p_2^0 = -1.51117410860539, \quad p_3^0 = -15.4775533604290,\\
&p_1^1 = 8.25080133537064, \quad p_2^1 = -16.5236682853100, \quad p_3^1 = 8.18123993194625,\\
&p_1^2 = -1.51918200681092, \quad p_2^2 = 13.8658045141122, \quad p_3^2 = -4.67938399579610.
\end{aligned}
$$

According to equation (5.15), one can calculate the output of the neurons in the first layer of the network, i.e., $O_{1,j}^i$. For instance,

$$
\begin{aligned}
O_{1,1}^1 &= \exp\left(-\frac{(x_1-\mu_1^1)^2}{2b_1}\right) - j\exp\left(-\frac{(x_1-\mu_1^1)^2}{2b_1}\right)\frac{x_1-\mu_1^1}{b_1}\delta_1^1\\
&= 0.953035906292485 + 0.244563689791652j,
\end{aligned}
$$

Similarly, the outputs of the other neurons in this layer can be obtained as follows:

$$
\begin{aligned}
O_{1,2}^1 &= 0.999572147343944 + 0.00850731929503057j,\\
O_{1,3}^1 &= 0.857919035126373 + 0.392200838129195j,\\
O_{1,1}^2 &= 0.952738339457055 + 0.245279642677795j,\\
O_{1,2}^2 &= 0.999557387818326 + 0.00865271677357599j,\\
O_{1,3}^2 &= 0.857096854711326 + 0.393048817862650j.
\end{aligned}
$$

Substituting the above results into equation (5.16), the output of the second layer neurons can be obtained as follows:

$$O_2^1 = 0.848007352359992 + 0.466765510258118j,$$
$$O_2^2 = 0.857919035126373 + 0.392200838129195j,$$
$$O_2^3 = 0.952738339457055 + 0.245279642677795j.$$

Therefore, the neuron outputs of the third layer can be calculated by further substituting the above values into equation (5.19):

$$O_3^1 = 0.359242938460111 + 0.0210120993358160j,$$
$$O_3^2 = 0.338022071771127 - 0.154035031690365j,$$
$$O_3^3 = 0.302734989768762 + 0.133022932354549j.$$

Combining the output of the third layer, the input data x_1 and x_2, and the linear parameter p, one can further find the output values of neurons of the fourth layer of the network according to equation (5.20), i.e., the consequent output:

$$O_4^1 = 5.24131143125436 + 0.306564011850975j,$$
$$O_4^2 = -0.565249660210355 + 0.257581550421428j,$$
$$O_4^3 = -4.62998900255801 - 2.03443518160890j.$$

Consequently, The final output of the network is available by applying the sum of the real component of the previous layer:

$$O_5 = \text{Re} \sum_{i=1}^{3} O_4^i(t) = 0.0460727684860016 \approx 0.0461$$

Hence, the predicted Sunspot time series result at this moment is 0.0461. Note that the ground truth is also 0.0461, indicating the high accuracy of the RACFIS network on this data point prediction.

5.4.2 Synthetic Data Test

The synthetic data is produced by the "tooth" function,

$$f(x) = 0.08 \times \{1.2 \times [(x-1) \times (\cos(3x))]$$

$$+[(x-(x-1) \times (\cos(3x))) \times \sin(x))]\}, 3 \leq x \leq 7, \tag{5.39}$$

in which 200 training data points were sampled from [3,7] with an interval of 0.02, and 50 test data points were sampled with an interval of 0.08 within the same scale. In this test, the CNFS [94] model is used as a control because it applies the same Gaussian complex affiliation function as RACFIS but uses a very different PSO derivative-free optimization strategy based on random initialization to optimize the network. Although the PSO method is easy to implement, the computational complexity is much higher than that of the gradient method, and the random initialization often causes the model to fall into a local optimum and makes the final result quite uncertain. It is also harder to achieve the global optimum for PSO due to the lack of information the gradient provides as a priori knowledge for the optimization surface. By comparison, the advantages of the RACFIS model over the previous CNFS can be intuitively demonstrated.

More details of the test are available in TABLE 5.1, and the visualized outcome is shown in Figure 5.3. Both models have achieved good results. However, RACFIS is superior to CNFS in nearly all aspects under the premise of the same number of rules. RACFIS performs better in areas with steeper changes, and the testing error and the number of epochs required are tenfold smaller than its CNFS counterpart. In addition, given that the optimization methods of RACFIS and CNFS both have a certain degree of randomness, 20 random tests are conducted to record the MSE values. The Kruskal-Wallis [181] test was employed to compare the obtained MSEs of those two models from the statistical aspect. In this test, we assume that two objects have no statistical difference, and a p-value less than 0.05 suggests a statistically significant difference between the two models. Finally, it achieves a p-value of 0.0138, which is enough to reject the null hypothesis, i.e., it is conclusive that the performance of RACFIS is better than that of CNFS, statistically.

Table 5.1: Comparison between CNFS and RACFIS over the synthetic data test.

	Testing MSE	Rules/Clusters	Learning settings	Max epoch
CNFS [94]	1.8691×10^{-4}	9	650 (population size)	200
RACFIS	1.3445×10^{-5}	9	10^{-6} (step size)	10

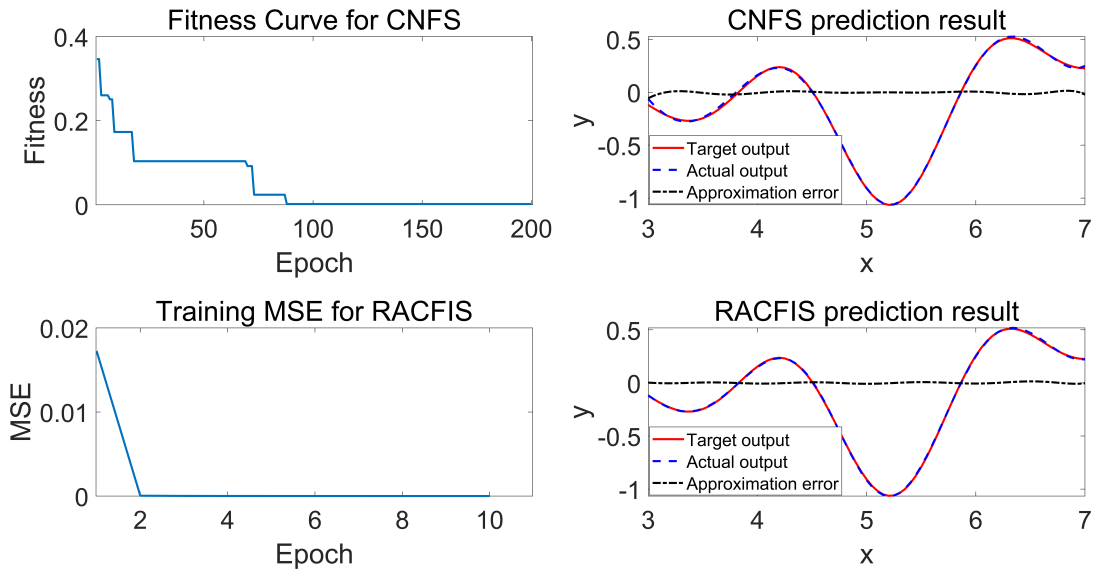

Figure 5.3: Comparison between CNFS and RACFIS on the "Tooth" function test for $x \in [3,7]$.

5.4.3 Sunspot Time Series Test

Sunspot data [107] adopted in this test include sunspot activity recorded between 1976 to 1992. This dataset contains a total of 2000 single-value samples, and each input-output data pair is in the form of $\{y(\tau-1), y(\tau); y(\tau+1)\}, \tau = 2,3,\ldots,2000\}$, in which the next sunspot state is predicted by the value of the current moment as well as the previous one. As always required, to avoid the failure of the gradient back-propagation of the network, one should normalize these data points between 0 and 1, then equally divide them to get the training set and test set. The experimental results of RACFIS and other benchmark models are available in Table 5.2. The visualized prediction outcome is given in Figure 5.4.

According to Table 5.2, the accuracy of RACFIS is far beyond SARIMA [150] (an ARIMA variant for periodic time series forecast). Elman-NARX [152] (a branch of recurrent neural network) and NFS [95] (ANFIS with PSO-RLSE optimization) are also significantly inferior to RACFIS regarding sunspot forecasting, which is in line with the

excellence that complex fuzzy models show when dealing with periodicity. Compared with CNFS, which is also a complex fuzzy model, the performance between the two is very close. However, RACFIS only applies 3 rules and 30 iterations of training to achieve the time series prediction accuracy that takes the CNFS model 300 iterations and 5 rules to realize. Given that the difference between CNFS and RACFIS is only the optimization strategy, it is fair to say that the proposed hybrid optimization algorithm is paramount for improving the model performance in time series forecasting tasks.

Table 5.2: Comparisons between the benchmark algorithms over the Sunspot test.

	Testing MSE	**Rules/Clusters**	**Iterations**
SARIMA [150]	6.5733×10^{-3}	–	–
BP [52]	1.4264×10^{-4}	–	200
LSTM [151]	3.4530×10^{-4}	–	200
Elman-NARX [152]	1.4078×10^{-4}	–	200
NFS [95]	8.5112×10^{-5}	5	300
CNFS	4.1490×10^{-5}	5	300
RACFIS	4.1045×10^{-5}	3	30

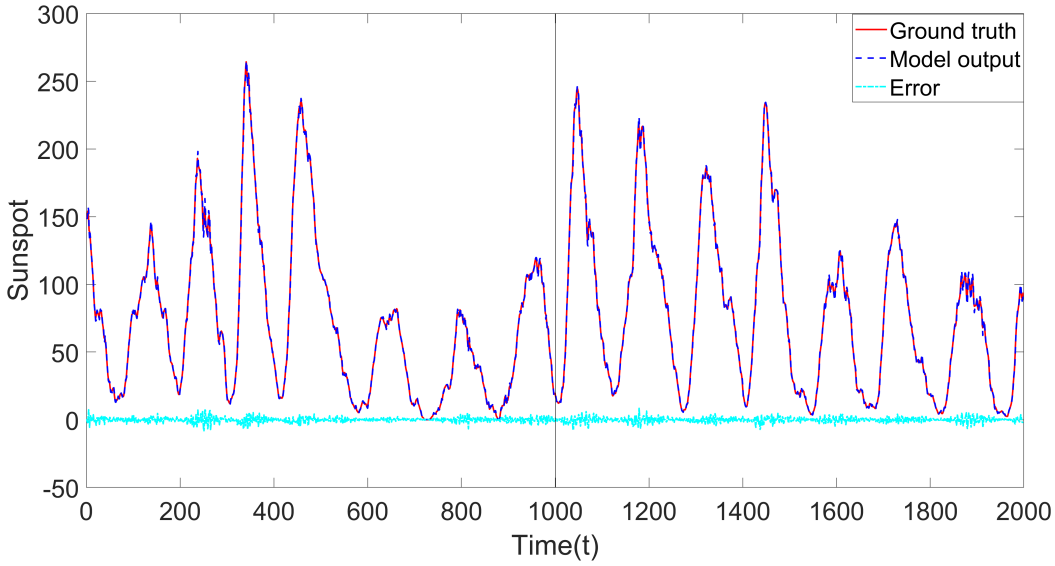

Figure 5.4: RACFIS performance on Sunspot test. The left side of the black dividing line in the figure is the prediction effect on the training set, and the right side is the performance on the test set.

5.4.4 Charpy Impact Data Test

Metals are crystalline substances whose physical and chemical properties depend on the structure and type of atoms that constitute their crystal lattice. Since all phenomena happen on the level of atomic size, even subtle changes in alloy composition may significantly affect its attributes, indicating that the influence of the components on the metal characteristic is highly non-linear. The Charpy impact energy measures the material capacity to assimilate the fracture energy. A Charpy impact test dataset with 16 input dimensions, single output, and 830 data points is applied for this data regression experiment. Such a data set is tricky for traditional modeling methods due to its discontinuity, scatter in measurements, and nonlinearity. In addition, the MIV algorithm is utilized to rank the contribution of each variable so that the less related ones can be opted out. The MIV of each variable under the adjustment rate of 10%, 20%, and 30% are given in Table 5.3.

Table 5.3: MIV of each variable for the Charpy Impact Test.

Input Variable	**Adjustment Rate**		
	10%	**20%**	**30%**
C	0.9866	1.9678	2.9380
Si	0.2548	-2.5154	-3.7635
Mn	1.6863	3.3569	4.9896
S	0.0565	0.1129	0.1694
Cr	3.5733	6.9862	10.0912
Mo	1.4240	2.8452	4.2610
Ni	0.6321	1.2629	1.8956
Al	-0.0041	-0.0082	-0.0123
V	-0.2822	-0.5644	-0.8465
Hardening Temp	0	0	0
Tempering Temp	148.9641	-45.7167	1.5452×10^{-6}
Impact Temp	12.6622	29.7544	-46.9750
Sample Size	-2.0193	-1.8219	0.0686
Test Depth	0.6174	0.0207	0.0207
Category symbol			**Categories**
Cooling Medium	-0.7666	-1.2536	-1.3499
Coded Site	-0.2611	-0.4680	-0.5806

Table 5.4: The parameter setting of RACFIS for the Charpy impact data test.

Bisecting k-means clustering for antecedence initialization			
Number of clusters for each variable k	6	Expansion coefficient ρ	0.95
QHM		RLS estimator	
Learning rate α	10^{-5}	Initial consequent parameter a_i, for $i = 1, 2, \ldots, q$	0.3
Immediate discount factor v	0.7	P_0	$\alpha * I$
Momentum discount factor β	0.999	α	10^6
Max epoch	10	I	60*60 Identity matrix
Initial momentum buffer $g(0)$	0	θ_0	60-dimension zero vector

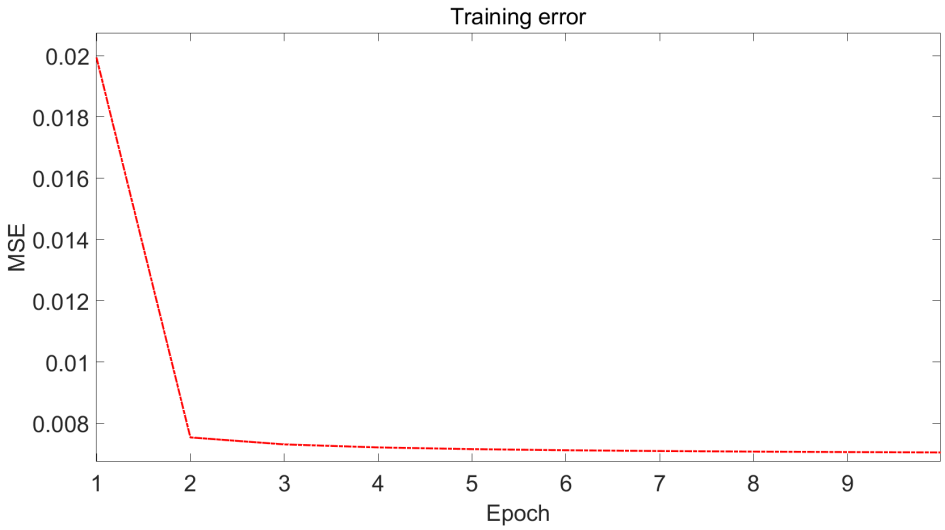

Figure 5.5: The curve of the MSE cost function over the learning process for Charpy data test.

According to the absolute value of MIV, the following 9 variables are more significantly influencing factors, including C, Si, Mn, Cr, Mo, Ni, Sample Size, Tempering Temperature, and Impact Temperature. The remaining variables are excluded due to their introduction into the model only bringing insignificant positive effects in prediction accuracy but significantly increasing the computational burden. After completing the variable analysis for the dataset, the first 600 sample points of the dataset are used as the training set, while the rest 230 are as the test set. In addition, the network setting for this experiment is given in Table 5.4.

As it can be seen from Figure 5.5, after pre-training of the antecedent parameters by the bisecting k-means clustering, plus the proposed QHM-RLS hybrid algorithm for refining, the network converged to a small MSE value after 10 iterations. Note that the QHM-RLS algorithm is an online optimization method, which is different from the traditional batch optimization method. The former updates the network parameters based on each input data point, while the latter renews the network only after a batch of data inputs or even after searching the entire dataset. There are three important reasons why RACFIS was able to converge so quickly. Firstly, the convex clustering method determines the network parameters to a position close to the global optimum, avoiding the possibility of large deviations due to random initialization. Secondly, the QHM-RLS online optimization strategy avoids the lag of the batch renewal approach, which significantly improves the training efficiency. Thirdly, the QHM gradient optimization solution can better adapt to the optimization surface, reducing the situation of getting trapped in the local optimum in repeated iterations.

The comparison between the network prediction and the target output value is available in Figure 5.6, and the result regression plot under 90% confidence is given in Figure 5.7. Table 5.5 reveals the overall performance of the proposed RACFIS regarding performance indices in the experiment compared with the other benchmark algorithms. RACFIS outperforms all benchmark algorithms listed in the table from the perspective of RMSE, MAE, and SMAPE using only 10 iterations. STD indicator reflects the degree of dispersion of the output data itself. The outputs from The prediction results of RACFIS have a lower error with a broader distribution scale, which indicates a remarkable generalization capability. In terms of the comparisons with other fuzzy systems, the performance of RACFIS also surpasses GrC-NF [182] (granular modeling); Q-ANFIS [183] (fuzzy C-Means clustering and quantum membership functions); interval type-2 model [184] (Sugeno defuzzification); and CNFS which has the similar complex fuzzy logic. Except for the same number of rules as Q-ANFIS, the number of fuzzy rules required by RACFIS is less than that of all non-complex fuzzy inference systems.

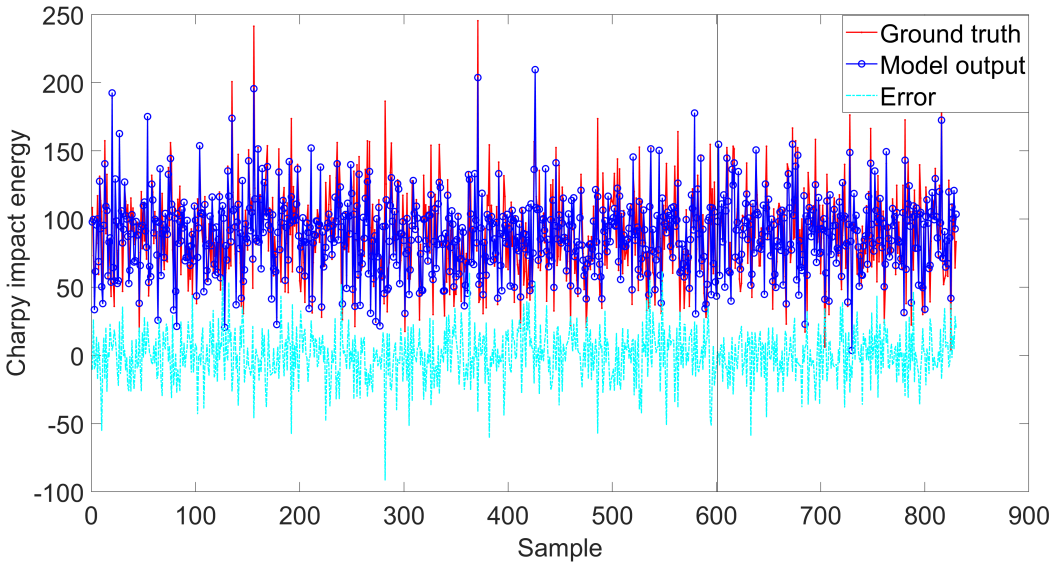

Figure 5.6: Performance of RACFIS in Charpy energy prediction. The part on the left of the black line is the comparison between the predicted results of the training set and the actual value. The right side is the same but for the performance on the test set. Approximation error is also given in the figure.

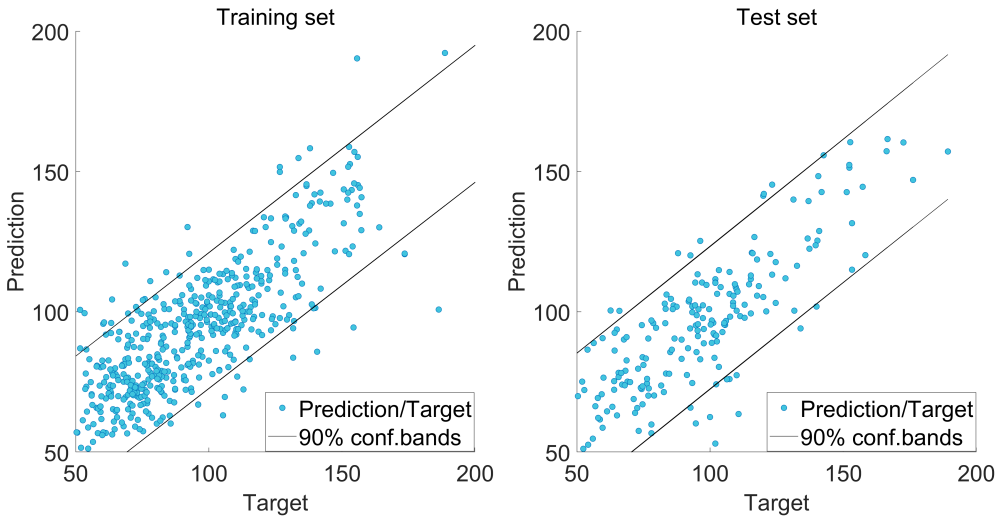

Figure 5.7: Result regression plot for Charpy energy prediction.

Table 5.5: Comparison of the performance (Charpy data test).

	STD	MAE	SMAPE	RMSE	Rules	Iterations
BP [52]	25.2233	16.9010	21.2874	20.8389	–	200
RBF [53]	30.9238	14.7211	19.5694	20.1443	–	100
GRNN [153]	22.7647	16.0598	19.0444	22.0339	–	100
LSTM [151]	24.9117	17.0228	21.5417	21.2242	–	200
DBN [154]	27.2085	16.2536	20.2341	20.1211	–	50
GrC-NF [182]	–	–	–	20.4200	9	200
Q-ANFIS [183]	–	–	–	18.1700	6	100
IT2Sugeno [184]	–	–	–	19.6500	8	100
CNFS	25.7738	16.3243	22.4266	20.5506	6	100
RACFIS	27.2434	13.5313	17.8632	17.1872	6	10

The excellent performance of RACFIS on prediction accuracy can also be attributed to three aspects. First, the QHM-RLS online optimization strategy improves the network performance on the test set, as batch optimization tends to overfit the training set. The non-fuzzy deep network models in the benchmark test are almost all batch-based, which is a crucial reason for their inferior performance on this data compared to RACFIS. Secondly, the introduction of complex fuzzy sets also contributes to its performance. On the one hand, complex fuzzy sets can accommodate more information, and it not only improves the generalization capability but also reduces the number of fuzzy rules required, which is quite evident compared with other fuzzy models. On the other hand, another advantage of the complex fuzzy model over most type-2 models is that its membership function has closed-form solutions for gradient optimization. The first-order derivatives of the type-2 membership function do not have algebraically closed first-order derivatives. Consequently, such models often use non-gradient optimization methods with higher computational complexity, which significantly affects the efficiency. Online learning strategy of RACFIS is also conducive to convergence speed. Finally, the use of convex clustering methods and QHM pushes the training result closer to the global optimum, which explains why RACFIS uses a similar complex fuzzy membership function as the CNFS model, but the performance is much better. Moreover, the above results also indicate that RACFIS can deal with the challenges posed by the Charpy impact dataset. Given that many real-world datasets are below the difficulty of this dataset, it is reasonable to conclude that RACFIS is suitable for the most practical application scenarios.

5.4.5 Ultimate Tensile Strength (UTS) Data Test

The ultimate tensile strength is defined as the peak of engineering stress in a stress-strain curve. Within the data set, 15 input variables correspond to an output dimension, with 3760 data points. This high dimensional data is tricky for its non-linearity, intricate interaction between input variables, measurement error of industrial process, and data sparsity. Additionally, extra 12 data points with similar input values but different outputs are employed as the validation set to measure the generalization capacity of the model. Regarding these 12 *abnormal* points, there are three reasons for them to exist, including human errors in data entry, missing dimensions, and measurement errors, which are also inevitable in other real-world datasets. This validation process can further test the performance from a higher level of universal applicability. The result of the MIV analysis for the UTS dataset is available in Table 5.6.

Table 5.6: MIV of each variable for the Ultimate Tensile Strength Test.

Input Variable	**Adjustment Rate**		
	10%	**20%**	**30%**
C	8.1864	16.3605	24.5097
Si	0.0794	0.1594	0.2404
Mn	-10.3150	-20.4747	-30.3276
S	-0.0723	-0.1445	-0.2168
Cr	-15.0656	-29.7948	-43.8615
Mo	-3.0059	-6.0067	-8.9937
Ni	-37.5670	-74.1685	-108.8752
Al	-0.3721	-0.7443	-1.1164
V	-0.1084	-0.2169	-0.3253
Hardening Temp	-0.0017	0	0
Tempering Temp	224.3830	274.6474	-482.3313
Sample Size	-830.1928	-624.7182	1321.7
Test Depth	-25.7766	-35.7784	-64.4520
Category symbol			**Categories**
Site	-242.2889	-454.9862	-609.9973
Cooling Medium	-179.3221	-344.4547	-482.3313

According to the MIV value, one can separate 10 variables with the highest relevance to the UTS as the model input variables, i.e., C, Mn, Cr, Mo, Ni, Site, Tempering Temperature, Cooling Medium, Sample Size, and Test Depth, respectively. Given the excessive number of data points in this dataset, which far exceeds the number needed to train the model, 1000 random data points selected from the original dataset are used as the training set, while another 500 as the test set. The network parameter settings of the experiment are shown in Table 5.7.

Table 5.7: The parameter setting of RACFIS for the UTS test.

Bisecting k-means clustering for antecedence initialization			
Number of clusters for each variable k	5	Expansion coefficient ρ	0.95
QHM		RLS estimator	
Learning rate α	10^{-6}	Initial consequent parameter a_i, for $i = 1, 2, \ldots, q$	0.3
Immediate discount factor v	0.7	P_0	$\alpha * I$
Momentum discount factor β	0.995	α	10^6
Max epoch	10	I	55*55 Identity matrix
Initial momentum buffer $g(0)$	0	θ_0	55-dimension zero vector

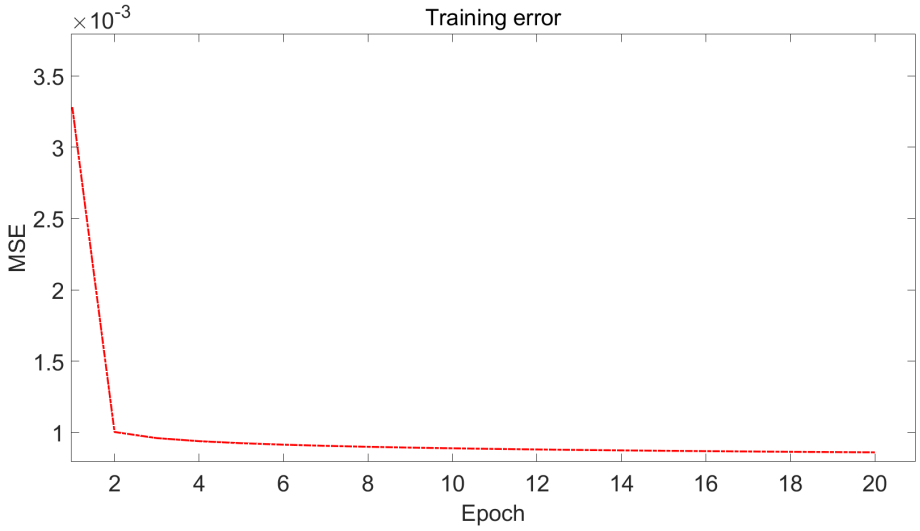

Figure 5.8: The curve of the MSE cost function over the learning process for UTS data test.

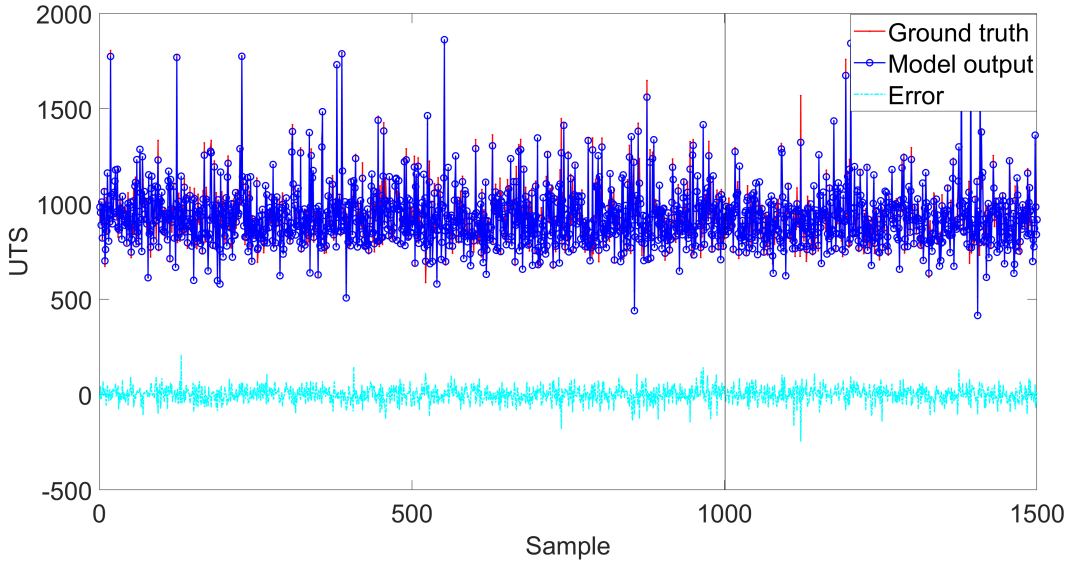

Figure 5.9: RACFIS network output for the UTS data test. The left side of the black line is the comparison between the network outputs of the 1000 training samples and the target values. The right side is with the same narrative for 500 testing data points. Approximation error is shown in the figure as well.

According to Figure 5.8, the RACFIS model only requires 20 iterations to converge in this dataset, which is only one-tenth of the level of most other network models. Table 5.8 compares the performance of RACFIS with several benchmark algorithms, from which RACFIS reveals a competitive performance, surpassing all the non-fuzzy algorithms mentioned in the list. Even in comparison with fuzzy algorithms, RACFIS only uses 5 rules to obtain a comparable prediction accuracy that is achieved by IMOFM [185] (Mamdani type-1), MOIT2FM [186] (Mamdani type-2) and IT2-Sugeno with 6 rules. Although CNFS adopts a similar complex fuzzy logic as RACFIS, it is not as comparable to RACFIS, owing to the limitations of its optimization policy. RACFIS has proven to achieve excellent performance with the smallest number of iterations compared with other algorithms listed in the case of the UTS test. The prediction result of the network is available in Figure 5.9, and the result regression plot with 90% confidence bands is given in Figure 5.10.

Table 5.8: Comparison of the performance (UTS data test).

	STD	MAE	SMAPE	RMSE	Rules	Iterations
BP	150.9674	32.1335	3.4460	44.4965	–	200
RBF	149.2079	41.2173	4.4341	54.1319	–	100
GRNN	153.8146	39.2092	4.1841	56.5168	–	100
LSTM	137.5868	42.4865	4.5160	56.5765	–	200
DBN	142.1892	36.1535	3.9260	47.7999	–	50
IMOFM [185]	–	–	–	45.5200	6	200
MOIT2FM [186]	–	–	–	40.5200	6	100
IT2Sugeno	–	–	–	38.7600	6	100
CNFS	142.8011	37.6474	4.0346	51.3500	5	100
RACFIS	151.4953	30.4602	3.3001	39.4170	5	20

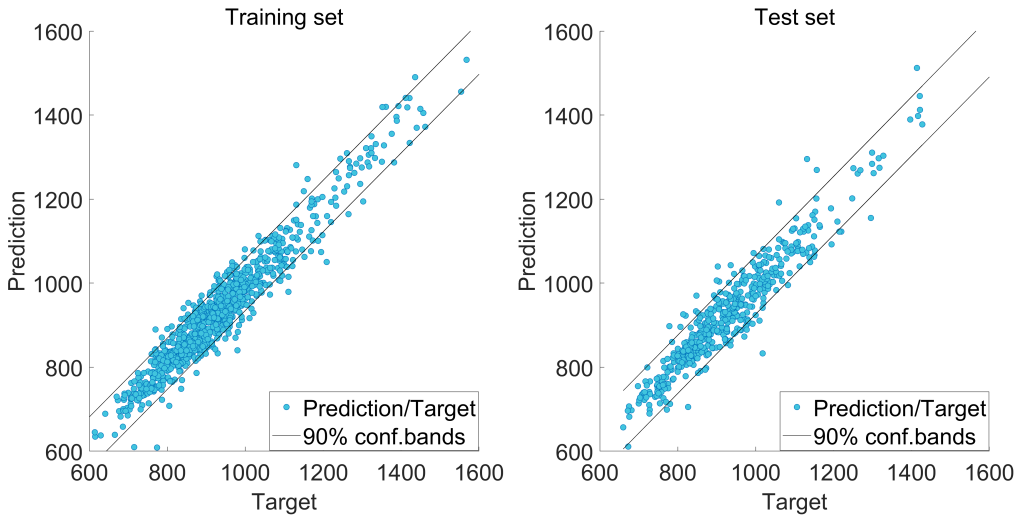

Figure 5.10: Result regression plot for UTS prediction.

The UTS data test is the most challenging dataset in this thesis, mainly because of its sparsity, which significantly increases the modeling difficulty. In the previous chapter, the proposed DCVSF performed disappointedly on sparse data due to its design, but the sparsity does not seem to be a trouble for RACFIS. The principle of RACFIS differs from DCVSF because it only initializes a relatively small rule base for each variable and adjusts the parameters and weights of each rule by an optimization algorithm to complete the training. This design can be better adapted to different data distributions, not to mention that there is a clustering algorithm to pre-train the antecedents, which further reduces the effect of sparsity. It is fair to conclude that RACFIS has advantages in terms of prediction accuracy, convergence speed, and adaptability to data, making it a machine learning modeling approach suitable for most real-world scenarios.

As already stated, the UTS data set also has a validation set consisting of 12 *abnormal* data points to test the generalization capacity of the network. Figure 5.11 displays the linear regression between the target output and the prediction with the 90% confidence band. The experimental results of all benchmark algorithms and RACFIS are available in Table 5.9, in which STD is an important indicator. According to the table, deep learning models GRNN, LSTM, and DBN have almost no processing capabilities for this validation set since the STDs of the model output of those models are close to 0. The STD indicator measures the degree of dispersion of the model output sequence against its mean. An STD close to 0 indicates that the algorithm can hardly respond to the subtle changes of variables in the validation set as it fails to distinguish the difference between each input data point. However, the other four algorithms can process these data more effectively, where RACFIS has the lowest prediction error, which is surprisingly good for this tricky test.

The advantage of this generalization ability may come from three aspects. Firstly, the use of complex fuzzy logic enables each rule of the network to express richer information and thus reduces the tendency of overfitting. Secondly, the introduction of online learning also significantly improves the algorithm performance on the validation set, especially in this extreme scenario. Online learning adjusts the network parameters after each data point is fed in, allowing it to learn more subtle information about the data, which also explains why deep learning algorithms such as GRNN, LSTM, and DBN have almost no processing power for this validation set, as all three algorithms rely on batch optimization. Thirdly, the clustering method pre-trains the antecedent parameters, effectively mitigating the local optimum caused by random initialization and ensuring that the algorithm is closer to the theoretical global optimum, which is evident in the comparison of RACFIS and CNFS. Although RACFIS and CNFS use similar complex Gaussian membership functions, the generalization performance of RACFIS still has a significant ascendancy, which is entirely owing to the difference in optimization strategies.

Table 5.9: Comparison of the performance (12 abnormal data points test).

	STD	MAE	SMAPE	RMSE
BP	227.1069	40.5753	4.4768	53.0111
RBF	165.6330	44.7274	4.4971	55.8941
GRNN	5.9167	250.5156	28.0084	316.9053
LSTM	9.1936	276.9733	26.4932	400.6833
DBN	0.8827	244.8736	27.2302	312.2535
CNFS	184.4885	46.9601	4.8166	59.3112
RACFIS	202.1760	33.0322	3.6206	39.0780

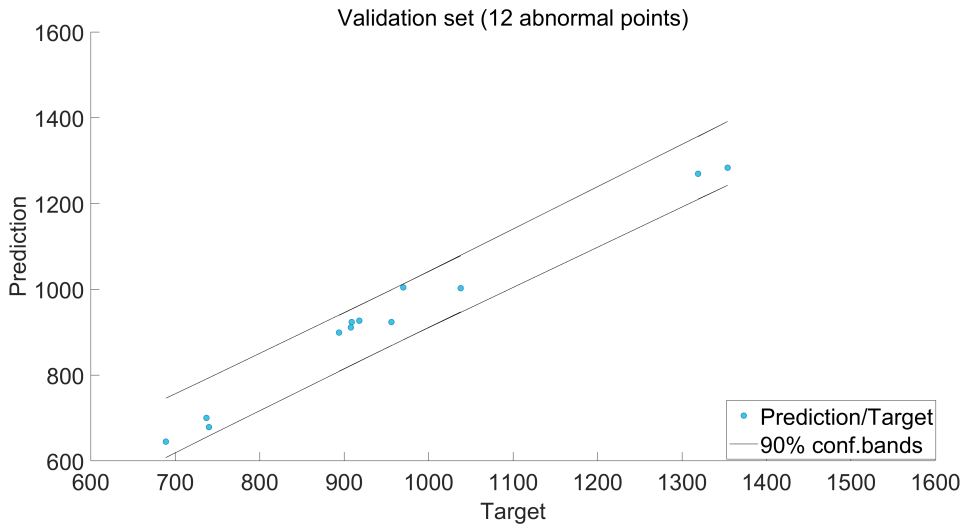

Figure 5.11: Results regression plot for UTS prediction (12 abnormal data points).

Notably, the UTS data set does have missing variables because some physical quantities are difficult to measure in industrial processes and are therefore not included in the data set used for testing, which may prohibit accurate prediction for many machine learning models. RACFIS exhibits excellent generalization capabilities, which not only learn information about the data but also make predictions as close as possible to the ground truth with key variables missing. Such an algorithm has the potential to meet the most demanding situations in everyday applications, enabling it to be a competitive solution for real-world data modeling tasks.

5.5 Chapter Summary

The RACFIS network is proposed for real-world regression modeling scenarios in this chapter. On the one hand, it adopts a newly designed joint optimization method, leading to surprising algorithm efficiency. The optimization method includes the use of convex clustering to pre-train the antecedent parameters of the network so that it locates at a position close to the global optimum before the start of the iteration, thus speeding up the training and effectively reducing the possibility that the model falls into local optima. In the iterative training step, a newly designed QHM-RLS online optimization strategy is applied to refine the network parameters, where the QHM method is a novel three-parameter gradient optimization method characterized by faster convergence and smoother training, and RLS is a traditional linear parameter estimator for optimizing the linear part of the network. This new joint optimization approach allows the RACFIS model to converge with only one-tenth of the number of iterations compared to other benchmark algorithms and also leads to a result closer to the global optimum.

On the other hand, the combination of complex fuzzy logic and gradient-based online learning methods also contributes to the good performance of RACFIS. The rule-base of complex fuzzy sets is defined in the unit circle of the complex plane, which has a better information capacity than the traditional type-1 fuzzy sets, making it possible to achieve better generalization capability with fewer rules. The rule-base of RACFIS can be even smaller than most type-2 fuzzy models when solving the same problem, owing to the application of complex fuzzy sets. Also, one can optimize a complex fuzzy model using gradient methods because the closed-form solutions exist for the first-order derivatives of membership functions, which is an advantage over type-2 fuzzy models since it is well-known that there are no such solutions for most interval type-2 membership functions. The essence of the gradient is the prior knowledge of the optimization surface, which is conducive to speeding up the convergence and reducing computational consumption. The gradient-based online learning strategy is more flexible than traditional batch optimization, alleviating the model's tendency to overfit the training set and improving its generalization capability on the testing set. Combining these advantages, the RACFIS model outperformed all benchmark models in almost all tests regarding

prediction accuracy, convergence speed, and generalization performance.

Moreover, given that real-world datasets often contain variables with little relevance, the RACFIS algorithm also includes an RBF network-based MIV algorithm for variable analysis. This method differs from the dimensionality reduction method in that it does not change the manifold of the data since the result is simply a "score" for the variable, and the user can determine whether the variable has value based on this. Irrelevant variables can be identified and excluded by ranking the scores that relate their contribution to the model output. MIV is considered the best of such methods, and its accuracy is 100% in the experiments of this chapter. With this algorithm, one can reduce the data dimensionality to mitigate the problems caused by the curse of dimensionality, and it also helps to eliminate interference from irrelevant factors to improve prediction accuracy.

The RACFIS algorithm is highly adaptable to tricky data types, including datasets with high sparsity, a significant advantage over the DCVSF algorithm presented in the previous chapter. RACFIS only needs to initialize a relatively mini rule-base to achieve high-accuracy nonlinear approximation with a small number of iterations, which makes it a promising solution for real-world application scenarios that require better prediction accuracy, faster convergence, and a smaller computational resource footprint at the same time. However, the RACFIS architecture and its online optimization approach do not fit the purpose for data types with significant outliers and noises, for which it may fail to learn data information or even be unable to converge. Considering that noise is inevitable in many application scenarios, it is necessary to design an algorithm that is more robust against such abnormality in the data. Therefore, a robust machine learning architecture will be presented to neutralize this issue in the next chapter.

6

ENCFIS: An Exclusionary Neural Complex Fuzzy Inference System for Robust learning

> *My goal is simple. It is a complete understanding of the universe, why it is as it is and why it exists at all.*
>
> **– Stephen Hawking –**

6.1 Introduction

For a long time, numerical modeling overridingly relied on primitive statistical models such as ordinary least squares (OLS) [187] or hypothetical models based on tenable assumptions, expert knowledge, and deductive reasoning. However, the reliability of these methods is often inadequate, and, in many cases, they can only be employed to describe certain phenomena qualitatively, which hinders any future attempts to perform regression tasks based on the information obtained. With the rise of machine learning, inductive statistical models represented by artificial neural networks (ANN) are rapidly replacing traditional methods as the dominant approach in the field. As technology continues to permeate this quickly digitizing world where the internet of things (IoT), deep learning, and big data technologies are massively and increasingly applied, machine learning-based numerical modeling is now indispensable among the incalculable amount of research and industrial sectors.

Nevertheless, conventional machine learning methods, including fuzzy logic-based ones, rely highly on the tidiness of the dataset. As such algorithms are all designed under the hypothetical premise that the data distribution is clean, even the most preeminent algorithms can be confused if there are statistically existential noises or outliers in the data. Unfortunately, real-world data is rarely ideal. Indeed, in many cases, one can remove the noise and outliers with pre-processing methods, such as filtering, smoothing, or anomaly detection [188]. However, these techniques are hardly 100% effective, and anomalous data points will inevitably be fed into the model, leading to a reduction in performance. Some designs with good generalization capability may be resistant to such perturbations to a certain degree, but as long as such disturbances play a role in the final result, the regression accuracy of the model will more or less be affected. Even worse, if undetectable but ruinous adversarial perturbation [189] is planted in the data, it will lead to a calamitous impact on model performance. In fact, practical attack methods [190] have already been developed by creating malicious data samples with specially designed adversarial perturbations for the purpose of sabotaging machine learning models. Conventional machine learning models are utterly defenseless against this type of attack. Naturally, the performance and adaptability in real-world scenarios

are likely to be enhanced if a machine learning regression model can initiatively counter outliers. This category of approaches is also referred to as “robust machine learning" [191].

Support vector regressions (SVRs) [192] can be considered a paradigm following this line of thought. From the perspective of statistics, the ultimate objective of all machine learning algorithms is to achieve the minimized expected risk. However, considering that the expected risk is difficult to measure, most algorithms seek to minimize the empirical risk, i.e., minimizing the mean loss over all samples in the training set. It also poses a problem that pure empirical risk minimization is likely to contribute to the overfitting of the model and causes it to be sensitive to noise. Hence, structural risk minimization is introduced in SVR as a compromise between empirical risk and expected risk, which can be captured by adding an L2 regularization in its loss function. Subsequently, by introducing a slack variable to this regularization term, a “hard/soft margin" is created to pardon the errors arising from incorrect data points and thus significantly avoid the effects of noises [193]. Despite the fact that SVR has been extensively popular in dealing with noise, such a method is not perfect. Firstly, although SVR can avoid the noise in many cases by setting a “margin", it does not recognize the differences between the noise and the data from the statistical aspect, which may cause the false adoption of noisy data points while some genuine data points are mistakenly excluded. Secondly, even a sparse representation is available for the SVR model, the complexity of training such a model can be excessive especially for nonlinear problems with a significant number of training samples, as it requires initializing a remarkable number of support vectors to obtain linearly separable results on the Hilbert space and calculate kernel functions for each support vector, which results in its exceptional inefficiency under such circumstances. In the worst case, if sparse representations are difficult to obtain, the number of support vectors for the SVR model may even approach the number of training samples, which is unacceptable.

Other attempts to improve the noise immunity of machine learning algorithms are also notable. The simplest way is to replace the objective function with a robust loss function, such as Huber loss [194], log-cosh loss, or quantile loss [195]. Barron [196]

proposed a generalized adaptive robust loss function that is considered superior to traditional robust loss functions. This approach has the advantage of simplicity but with negligible improvements in algorithm robustness. Further, some researchers enhanced robustness by introducing a noise adaptation mechanism into the model, i.e., label-noise representation learning (LNRL) [197]. Goldberger *et al*. [198] increased the algorithmic robustness against outliers by adding a noise adaptation layer to the deep network. Patrini *et al*. [199] introduced loss correction methods in deep learning models to form a more robust model. Wang *et al*. [200] and Ren *et al*. [201] applied data reweighting techniques to facilitate the noise tolerance of network structures. Jiang *et al*. [202] employed a small-loss technique to suppress noise but resulted in cumulative errors. Han *et al*. [203] ameliorated the small-loss trick by training two neural networks synchronously to create a difference channel structure, which successfully cancels out cumulative noise. However, although LNRL methods can enhance the robustness of models, such a train of thought also leads to a dramatic increase in training difficulty. Subsequently, Wang *et al*. [204] compensated for the shortcomings of previous LNRL methods by employing a meta-learning approach to estimate the noise transition matrix, which significantly ameliorates the efficiency of robust learning systems. This attempt also created a brand-new class of meta-learning-based robust learning methods. Carmon *et al*. [205] proposed a semi-supervised learning strategy to allow networks to increase robustness by simply adding more unlabeled data samples. Nevertheless, semi-supervised learning and meta-learning require pure validation sets to pre-train the model, whereas clean data may not be available for some scenarios.

In addition, some non-architectural level techniques can also improve the robustness of the training process, including regularization, loss designing, or even manually excluding some corrupted parts of data. Such methods are not strictly enhancements to the model robustness but merely improve the robustness of the model for specific problems. Although many exploratory attempts have been made, conspicuous shortcomings exist in the present robust machine learning methods: a) The robustness comes at the expense of significantly offsetting the algorithm's accuracy. b) The training process is often sophisticated and unfriendly to applications requiring real-time adjustments. c)

Mostly tailor-made to classification and pattern recognition tasks with uncertain effectiveness over numerical regression application scenarios. Notably, robust learning frameworks for fuzzy models have not yet emerged to our best knowledge.

Driven by the above motivations, an exclusionary neural complex fuzzy inference system (ENCFIS) that learns from both clean and contaminated datasets with a robust, accurate, and expeditious regression performance has been proposed in this chapter. Note that the word “exclusionary" in this context characterizes the model’s capability of ruling out the obfuscation caused by outliers during training. Complex fuzzy sets and logic (CFS&T) [16] is also introduced in this work to replace the traditional type-1 or type-2 fuzzy logic for constructing the fuzzy inference system. The membership degree of a complex fuzzy set is defined in a unit circle on the complex plane, with two-dimensional properties that enable less sensitivity to the disturbance than the one-dimensional type-1 logic, contributing to improved robustness. Additionally, complex numbers are algebraically closed structures, and the first-order derivatives of the membership functions of complex fuzzy sets can guarantee a closed-form solution, which makes gradient optimization possible. It is an overt advantage in contrast to the interval type-2 fuzzy logic because an algebraic solution does not exist for the derivatives of most interval type-2 membership functions. Considering that derivatives are actually the prior knowledge of the optimization surface, by applying gradient optimization policy, the complex fuzzy model can be potentially more efficient and precise than a typical interval type-2 model.

Regarding robust learning, the partition-based fuzzy c-means clustering method [206] is employed to pre-train the antecedent parameters of membership functions, which not only ensures that the model is placed close to its optimum before the iterative optimization process but also helps to avoid the interference from label noises as clustering is performed only on input variables. For robustly optimizing the consequent part, a Welsch function-based M-estimator [207] is applied to exclude the impact of outliers on linear parameters. Given that the residuals are often decoupled from the actual target under strong noise conditions, compromising the reliability of ordinary loss functions. Therefore, the ENCFIS network adopts a Huber loss function that can generate pseudo-

residuals as the loss function. Finally, such a robust design is validated on a highly contaminated synthetic dataset, a severely corrupted Sunspot time series dataset, and a metallurgical dataset that simulates the real-world scenario. Experimental results indicate that the ENCFIS model is robust to extremely noisy datasets and adaptable and competitive to real-world datasets.

6.2 Methodological Premises

6.2.1 Fuzzy C-Means Clustering

The k-means clustering is the best-known convex clustering method, simple but effective, and is massively applied in many fields. However, it must assign data samples to crisp clusters, which often leads to unreasonable partitioning at the boundary of each one. Principally, fuzzy c-means clustering [206] as a convex clustering method can play a similar role, but with the help of the fuzzy theory, it can provide a more flexible solution, i.e., soft-clusters, compared to hard-clusters generated by the traditional k-means algorithm. The soft clustering solution does not force the data into a particular cluster but provides the membership vector of that sample against all partitions. Subsequently, the fuzzy partition matrix, as the clustering outcome, is formed by integrating all membership vectors. Suppose that a dataset with n samples needs to be divided into c clusters, and $\xi_1, \xi_2, \ldots, \xi_c$ are the pre-determined centroids. The normalized membership value u_{ij} that reflects the degree of association between the ith data point and the jth cluster can be calculated as follows:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\|x_i - \xi_j\|}{\|x_i - \xi_k\|} \right)^{\frac{2}{c-1}}}, \tag{6.1}$$

where u_{ij} is an element of the fuzzy partition matrix U, and notably, $\sum_{j=1}^{c} u_{ij} = 1$. Then, the updated centroid ξ_j can be obtained under the new membership values:

$$\xi_j = \frac{\sum_{i=1}^{n} u_{ij} \cdot x_i}{\sum_{i=1}^{n} u_{ij}}, \tag{6.2}$$

The algorithm repeats the above steps several times until the objective function reaches the global minimum or satisfies the stopping condition. The sum of the squared error (SSE) is often the option as the objective function, which can be computed according to the following equation:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij} \left\| x_i - \xi_j \right\|^2, \tag{6.3}$$

In practice, however, it is customary to use the following stop condition:

$$\|U(k+1) - U(k)\| < \epsilon \tag{6.4}$$

where k denotes the iteration count, and ϵ represents the threshold to stop the iteration, which is usually a small number. Although the centroids are pre-determined as the information to initialize the clustering process in this section, one can also use the pre-determined fuzzy partition matrix to start the algorithm. A complete demonstration of the fuzzy c-means is as follows:

Algorithm 6.1 Fuzzy C-Means.

Step 1. Manually determine the number of clusters c.
Step 2. Initialize the fuzzy partition matrix $U(0)$ or the centroid vector $\xi(0)$.
Step 3. Calculate the renewed fuzzy partition matrix $U(k)$.
Step 4. Compute the updated centroid vector $\xi(k)$.
Step 5. Go back to Step 3 and repeat the above processes several times until the stopping condition $\|U(k+1) - U(k)\| < \epsilon$ is satisfied.

6.2.2 DEMON Momentum Decaying Optimization

The gradient-momentum method [208] is widely adopted amongst the brigade of gradient descent schemes. The idea of momentum is introduced to facilitate optimization in directions of low curvature while avoiding fluctuations in directions of high curvature. Compared with the vanilla gradient solution, the involvement of momentum can smooth the optimization process and help to avert some possible local optima. Assume $x(k) \in \mathbb{R}^d$ is the parameter vector of the kth iteration for a network, then the iterative expression for the classical gradient-momentum optimization is as follows:

$$\begin{cases} x(k+1) = x(k) + \alpha v(k) \\ v(k) = \eta v(k-1) - \nabla f(x(k)), \end{cases} \tag{6.5}$$

where $\alpha \in \mathbb{R}$ represents the learning rate, $\nabla f(x(k))$ refers to the vector of the gradient, η denotes the momentum coefficient and the item $v(k) \in \mathbb{R}^d$ accumulates momentum over the iterations. However, such a technique is unduly sensitive to the initial para-

meter setting, and even subtle changes can make an appreciable difference in the final optimization outcomes. In addition, unchanged momentum also causes it easy for the model to miss the optimum over the last few iterations. Consequently, methods of gradually reducing the momentum item during training have emerged, in which DEMON [209] is currently considered state-of-the-art.

It is well-known that the momentum is essentially the impact of historical gradients on the present learning process. For a constant momentum training process, the common iterative expression is as follows:

$$\begin{aligned} x(k+1) &= x(k) - \alpha\nabla f(x(k)) - \alpha\eta\nabla f(x(k-1)) - \cdots - \alpha\eta^{k-1}\nabla f(x(1)) \\ &= x(k) - \alpha\nabla f(x(k)) - \alpha\sum_{i=1}^{k}\eta(i)\nabla f(x(k-i)) \end{aligned} \tag{6.6}$$

where $\eta(i)$ signifies the momentum of the i th iteration and in this case $\eta(i) = \eta^i$. It should be noted that if $k \to \infty$, then $\eta(k) \to 0$. According to the sum of geometric series, we have:

$$\lim_{k\to\infty}\sum_{i=1}^{k}\eta(i) = \eta\lim_{k\to\infty}\sum_{i=0}^{k}\eta(i) = \frac{\eta}{1-\eta}. \tag{6.7}$$

The DEMON momentum decaying is enlightened by the above expression such that the momentum item can reduce to zero within finite T steps. Following this idea, the decayed momentum at the kth step is given as:

$$\eta(k) = \eta_{init}\frac{1-\frac{k}{T}}{(1-\eta_{\text{init}}) + \eta_{init}\left(1-\frac{k}{T}\right)}, \tag{6.8}$$

where η_{init} is the initial momentum. Thus, the renewed gradient-momentum update equation applying DEMON momentum decay is as follows:

$$\begin{cases} x(k+1) = x(k) - \alpha\nabla f(x(k)) + \eta(k)v(k) \\ v(k) = \eta(k-1)v(k-1) - \alpha\nabla f(x(k-1)) \end{cases}, \tag{6.9}$$

As the momentum decays to zero after finite iterations, the network sensitivity to the initial value of the momentum parameter sharply decreases, which is an evident advant-

age for models seeking more robustness during training, especially when it is hyperparameter sensitive. This approach also enables a better optimization process around the optimal point.

6.2.3 Robust Estimators

In the field of parameter estimation, many simple yet excellent methods are available if the actual problem is consistent with the assumed parametric model and no statistically significant outliers exist in the observed values, such as the least squares estimation, method of moments, and maximum likelihood estimation [187]. However, in practice, idealized parametric models are rarely identical to real problems, and observations are often inaccurate, noisy, or distorted. In this context, the concept of “robust statistics" [210] came into being, characterized by its tolerance to imprecision in hypothetical models and the interference of abnormal observations. Robust estimators, designed based on robust statistics, strive to approach traditional statistical methods but are more resistant to misleading outliers or deviations from the reference distribution.

A variety of robust estimators have emerged during the past few decades, which belong to three main categories, i.e., L-estimators, R-estimators, and M-estimators. Specifically, L-estimation is a linear combination of the observation statistics, for example, the sample median and α-trimmed mean, which is extraordinarily straightforward but statistically robust. The most famous practice of L-estimation is the Theil-Sen estimator [211], in which the mean of a least square estimator is replaced by the multivariate median, significantly increasing the robustness. Such a method has no additional parameters, and its performance is almost comparable to OLS, with a high breakdown point of 29.3%. However, due to the sharp escalation in the difficulty of determining the median value as the dimension increases, the efficiency and accuracy of the L-estimator represented by the Theil-Sen estimator plummet under high-dimensional conditions. Therefore, L-estimators are rarely taken into practice since high-dimensional scenarios are often involved in linear estimations. L-estimators based on α-trimmed mean are also not preferred because of their dependence on prior knowledge. Other than L-

estimation, R-estimation is realized by compartmentalizing the residuals into several classes, and each class is assigned a unique weight, therefore achieving a robust statistic. However, the R-estimator is even less popular than the L-estimator because its construction relies more on prior knowledge of the problem than the L-estimator, which vastly increases the arduousness of its applications.

Currently, the most mainstream robust estimation method is the M-estimation. The difference between this method and the previous two is that it simulates the possible distribution of the data by formulating an influence function, thus avoiding the requirement for prior knowledge. Such a solution, as a generalization of the maximum likelihood estimator, is susceptible to the data distributed near the mean while insensitive to the disturbance of anomalous points (usually situated far from the mean). Thereupon, one can accomplish a reliable and robust estimation even if the priority of the data is unavailable. Two types of M-estimators have emerged so far, i.e., ρ-estimator and ψ-estimator. For ρ-type, assume an n-dimensional measure space $\Lambda \in \mathbb{R}^n$, and $\lambda \in \Lambda$ is the parameter vector of the model. Thus, the representation of this Mestimator $\Xi(G)$ in accordance with the mapping $\rho : \chi \times \Lambda \to \mathbb{R}^n$ is given as follows:

$$\Xi(G) := \text{argmin}_{\lambda \in \Lambda} \int_{\chi} \rho(x, \lambda) dG(x), \tag{6.10}$$

where $\rho(x, \lambda)$ is the influence function, G denotes the distribution of observed values and χ refers to the distribution of estimation. Note that if $\rho(x, \lambda) = -\ln\left(\frac{\partial G(x,\lambda)}{\partial x}\right)$, then the M-estimator will reduce to the ordinary maximum likelihood estimator. For a differentiable and continuous influence function ρ, the M-estimator can be simplified to a more computationally convenient form, i.e., the ψ-type. In this case, the estimator $\xi(G)$ is defined by equation:

$$\int_{\chi} \psi(x, \xi(G)) dG(x) = 0, \tag{6.11}$$

and the estimation $\tilde{\lambda}$ can be obtained by solving the following equation:

$$\int_{\chi} \psi(x, \lambda) dG(x) = 0 \tag{6.12}$$

where $\psi(x,\lambda) = \frac{\partial\rho(x,\lambda)}{\partial\lambda}$. In addition, M-estimators with bounded ψ are typically robust, which suggests the importance of selecting appropriate influence functions. Dozens of influence functions have hitherto been proposed, but only five of them are most applied, i.e., the Huber function, the Hampel function, the Cauchy function, the Bisquare function, and the Welsch function [207].

6.2.4 M-Estimator with Welsch Influence

One can implement most linear estimation tasks utilizing the ordinary LS. However, the OLS is susceptible to abnormality among the training samples, which is inappropriate for a robust learning scenario. An M-estimator with the Welsch influence can be an effective solution under such circumstances. The Welsch influence function is defined as follows:

$$f_W(\varepsilon) = \frac{\gamma^2}{2}\left[1 - \exp\left(-\frac{\varepsilon^2}{\gamma^2}\right)\right], |\varepsilon| \leq \infty \tag{6.13}$$

Given the following system model:

$$y_L = S_L\theta + \omega_L, \tag{6.14}$$

where S_L is the sample matrix, θ denotes the parameter vector, y_L is the vector for output labels, and ω_L is the vector of the system noise independent of the input. The discrete M-estimator can be obtained by minimizing the following objective function:

$$\sum_{i=1}^{n} f_W\left(\frac{y_i - s_i\hat{\theta}}{\hat{\sigma}}\right) = \sum_{i=1}^{n} f_W\left(\varepsilon_i\right), \tag{6.15}$$

where $s_i \in S_L$ denotes the ith data sample, $\hat{\sigma}$ represents the scale parameter, ε_i is ith the residual, f_W refers to Welsch influence. For continuously differentiable function f_W, designate $\psi = \frac{\partial f_W}{\partial\theta}$, then one can further simplify the minimization process by solving the following equation:

$$\sum_{i=1}^{n} \psi\left(\frac{y_i - s_i\hat{\theta}}{\hat{\sigma}}\right) s_i = \sum_{i=1}^{n} \psi\left(\varepsilon_i\right) s_i = 0, \tag{6.16}$$

Define weight function $w(\varepsilon)$ under the Welsch influence:

$$w(\varepsilon) = \frac{\psi(\varepsilon)}{\varepsilon} = \exp\left(-\frac{(\varepsilon/\gamma)^2}{2}\right), \tag{6.17}$$

and substitute it into the equation (6.16) to obtain the following representation:

$$\sum_{i=1}^{n} w\left(\varepsilon_i\right) \varepsilon_i s_i = 0 \tag{6.18}$$

To estimate $\hat{\theta}$, the equation below is derived by substituting $\varepsilon_L = S_L\hat{\theta} - y_L$ into equation (6.18):

$$S_L^T W S_L \hat{\theta} - S_L^T W y_L = 0, \tag{6.19}$$

where W is the weight matrix determined by the weight function $w(\varepsilon)$. Thus, the final estimation of the parameter vector θ is as follows:

$$\hat{\theta} = \left(S_L^T W S_L\right)^{-1} S_L^T W y_L \tag{6.20}$$

Regarding the tuning constant γ in the weight function, this is related to the estimation efficiency. Usually, the efficiency of ordinary LS is considered the reference. The higher the constant tunes, the closer its performance is to LS, but the robustness against outliers decreases. The most encountered relative efficiencies of the Welsch M-estimator against ordinary LS under different γ settings are available in Table 6.1. In most cases, the M-estimator is recommended to run at the efficiency level of 95%.

Table 6.1: The efficiency level under different γ for the Welsch M-Estimator.

γ	2.3831	2.9850	3.9104	4.7407
Efficiency level	90%	95%	98%	99%

6.3 Methodology

6.3.1 ENCFIS Overview

ENCFIS is a robust learning architecture for the nonlinear regression of highly noisy data. Unlike most previous attempts that use filters to eliminate noise before sending the data into the model, this algorithm uses the difference in statistical distribution between the noise samples and the authentic samples to offset the effect of noise. It first maps the nonlinear raw data to a high-dimensional Hilbert space using a kernel function to make the object linearly separable. Then it applies a linear M-estimator in the Hilbert space to obtain robust regression results according to the hypothetical distribution of the data, which is given in the form of an influence function. Specifically, in ENCFIS, the above process is driven by a uniquely designed Sugeno fuzzy inference system based on complex fuzzy logic that perfectly integrates robust learning.

Assume the objective system has m inputs and one output, then the ith fuzzy rule can be represented as follows:

$$Rule\ \ i:\ IF\ \ l_1\ is\ \ A_1^i(x_1)\ and\ \ l_2\ is\ \ A_2^i(x_2)\ ,\ldots,\ and\ \ l_m\ is\ \ A_m^i(x_m).$$

By applying the Sugeno defuzzification, the output ζ^i of this rule is obtained:

$$\zeta^i = a_0^i + \sum_{j=1}^{m} a_j^i x_j, i = 1,2,\ldots,n, \tag{6.21}$$

where $A_j^i\left(x_j\right)$ denotes the jth antecedent of the ith complex fuzzy rule, a_j^i signifies its corresponding consequent parameter. In terms of the optimization of the antecedent parameters, the fuzzy c-means algorithm is utilized first to pre-train the Gaussian kernel parameters to approximate the actual global optimum. Subsequently, the DEMON momentum decaying optimization is taken to fine-tune the parameters. Regarding the consequent part, a ψ-type M-estimator based on the Welsch influence function is applied to conduct a robust linear estimation to determine the weights. Moreover, the Huber loss is selected as the cost function to counteract the disturbance caused by outliers. An overall schematic diagram of the ENCFIS is available in Figure 6.1.

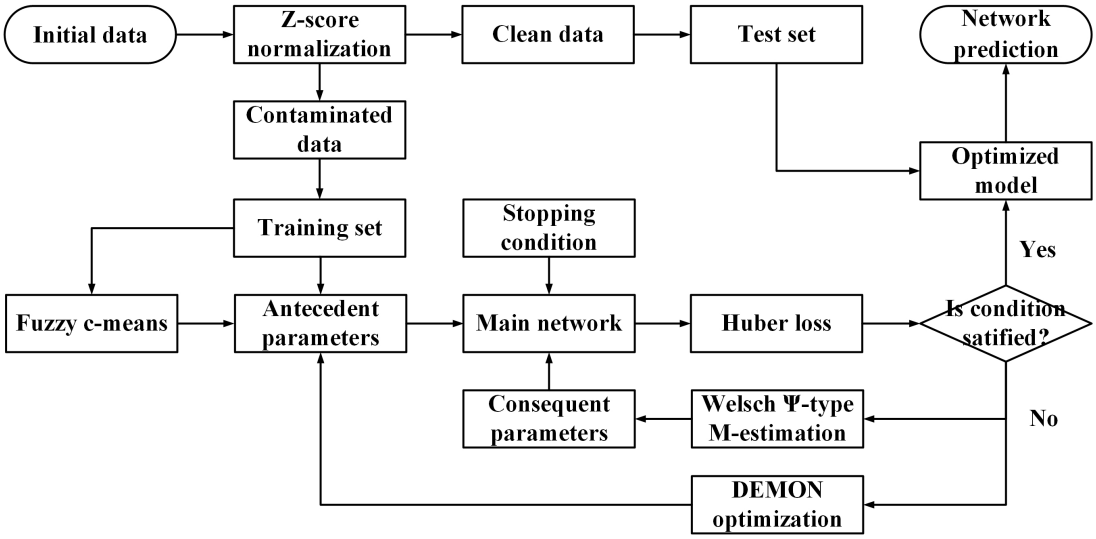

Figure 6.1: ENCFIS algorithm flow.

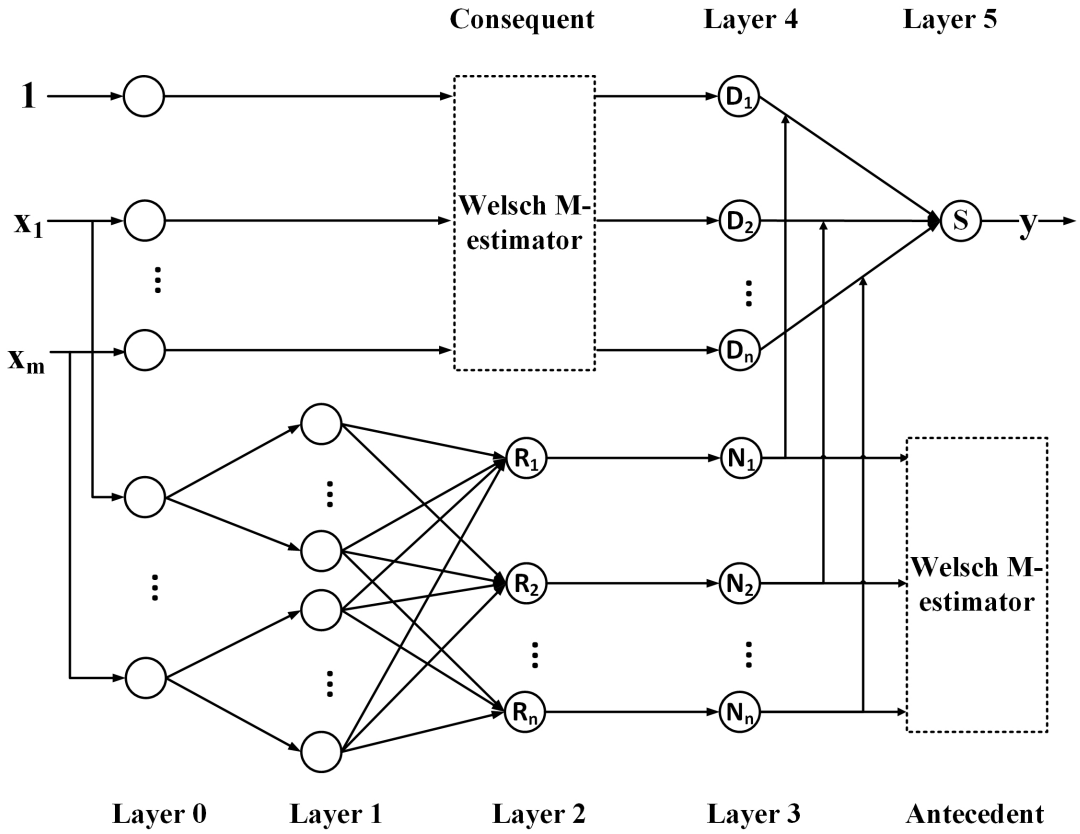

Figure 6.2: The main network structure of ENCFIS. (X_i denotes the input variable, R_i represents the firing strength of each rule, N_i is the normalization layer, D_i is the outcome of Sugeno defuzzification, and S signifies the sum of the previous layer, which is identical to the output of the network, i.e., y.)

6.3.2 Network Structure of ENCFIS

ENCFIS has a five-layer network architecture displayed in Figure 6.2. The operation of each layer under the n-dimensional input vector $x(\tau) = [x_1(\tau), x_2(\tau), \ldots, x_n(\tau)]^T$ at time τ is presented as follows:

Layer 1: The fuzzification is carried out in this layer, where the real-valued inputs are mapped into the complex common domain via the following complex Gaussian membership function:

$$O_{1,j}^i(\tau) = \exp\left(-\frac{\left(x(\tau)-\mu_j^i\right)^2}{2b_j^i}\right) - \text{jexp}\left(-\frac{\left(x(\tau)-\mu_j^i\right)^2}{2b_j^i}\right)\frac{x(\tau)-\mu_j^i}{b_j^i}\lambda_j^i, \tag{6.22}$$

where $O_{1,j}^i(\tau)$ refers to the complex membership of the ith rule by the jth input variable, and $\left\{\mu_j^i, b_j^i, \lambda_j^i\right\}$ denotes the antecedent parameters for each node.

Layer 2: This layer computes the firing strength of each rule, in which a complex multiplication is applied:

$$O_2^i(\tau) = \prod_{j=1}^{n} O_{1,j}^i(\tau) =: \alpha_j^i(\tau) + j\beta_j^i(\tau), \tag{6.23}$$

$$\alpha_j^i(\tau) = \left[1 - \prod_{j=1}^{n}\frac{\left(x_j(\tau)-\mu_j^i\right)}{b_j^i}\lambda_j^i\right]\exp\left(-\sum_{j=1}^{n}\frac{\left(x_j(\tau)-\mu_j^i\right)^2}{2b_j^i}\right), \tag{6.24}$$

$$\beta_j^i(\tau) = -\sum_{j=1}^{n}\lambda_j^i\frac{x_j(\tau)-\mu_j^i}{b_j^i}\exp\left(-\sum_{j=1}^{n}\frac{\left(x_j(\tau)-\mu_j^i\right)^2}{2b_j^i}\right), \tag{6.25}$$

where $O_2^i(\tau)$ is the strength of the ith firing and $\alpha_j^i(\tau), \beta_j^i(\tau)$ represent the values of real component and imaginary component, respectively.

Layer 3: All firing strengths are normalized in this layer via complex division so that the information of two dimensions can thoroughly interact. For brevity, formulas (6.24) and (6.25) are used here to simplify the expression:

$$O_3^i(\tau) = \frac{O_2^i(\tau)}{\sum_{r=1}^{m} o_2^r(\tau)} = \frac{\alpha^i(\tau) + j\beta^i(\tau)}{\sum_{r=1}^{m} \alpha^r(\tau) + j\sum_{r=1}^{m} \beta^r(\tau)}, \tag{6.26}$$

where $O_3^i(\tau)$ denotes the normalized output of ith node in this layer.

Layer 4: This layer consists of adaptive nodes, in which the Sugeno defuzzification is implemented to derive the output of fuzzy reasoning in accordance with corresponding consequent parameters:

$$O_4^i(\tau) = O_3^i(\tau) * \left(p_0^i + \sum_{j=1}^{n} p_j^i x_j(\tau) \right), \tag{6.27}$$

where $O_4^i(\tau)$ is the result of each reasoning and $\{p_0^i, p_1^i, p_2^i, \ldots, p_n^i\}$ are consequent parameters.

Layer 5: The overall output of the network is determined in this layer by a simple sum of node outputs from layer 4. Note that only the real component of the complex number is utilized.

$$O_5(\tau) = \mathrm{Re} \sum_{i=1}^{m} O_4^i(\tau), \tag{6.28}$$

where "Re" denotes the real component of the complex number, $O_5(\tau)$ is the network output, and m equals the number of fuzzy rules for this system. This step maps the complex reasoning result into real numbers as an indispensable defuzzification step for the complex fuzzy sets. For a complex membership function defined in the polynomial form, the complex plane projection on the real axis equals its real component, making it the best candidate to be the defuzzified output. This strategy maintains the nature mapping relation between the antecedent and the network output, enabling the antecedent parameters to pre-train by a partition-based convex clustering method. Given that the feed-forward structure of ENCFIS is consistent with the RACFIS model of the previous section, for the input-to-output numerical demonstration, please see section 5.4.1.

Regarding the selection of objective function, the mean square error (MSE) loss is often preferred owing to its good performance in regression tasks. But the MSE loss is sensitive to outliers, causing a vulnerability against noisy data. Although the mean absolute error (MAE) is considered more robust against disturbances, it is less popular in neural network optimizations due to the poor performance of the constant gradient near the minimum. Hence, Huber loss [194], combining the merits of the two, is taken as the objective function for ENCFIS, which is shown as follows:

$$\widehat{H}(\tau)=\begin{cases} \frac{1}{2}\left(y(\tau)-O_5(\tau)\right)^2, & |y(\tau)-O_5(\tau)| \leq \delta \\ \delta\,|y(\tau)-O_5(\tau)|-\frac{1}{2}\delta^2, & otherwise \end{cases} \tag{6.29}$$

where $\widehat{H}(\tau)$ represents the Huber loss, $|y(\tau)-O_5(\tau)|$ is the residual at the τth input, δ denotes the tuning coefficient. By setting a δ, the Huber function diverts a part of abnormal data points to the MAE loss, which enhances the robustness of the traditional MSE loss. The user should choose the parameter δ cautiously because a too-large δ may lead to reduced robustness against the noise, whereas a too-small one will slow down the algorithm convergence. After deploying the Huber loss, the optimization process no longer relies on the residual, but the pseudo-residual defined as follows:

$$\hat{h}(\tau)=\begin{cases} y(\tau)-O_5(\tau), & |y(\tau)-O_5(\tau)| \leq \delta \\ \delta sign\left(y(\tau)-O_5(\tau)\right), & otherwise \end{cases} \tag{6.30}$$

where $\hat{h}(\tau)$ represents the pseudo-residual generated by the τth data point, $sign()$ is the signum operator that extracts the sign from a real number. Since the noise tends to keep the actual residuals away from the optimization target, the use of pseudo residuals certainly smooths the training process. The merit of the Huber function is its simplicity and the fact that it has only one adjustable parameter. One can tune it easily according to the noise level of the data, or it can also be optimized in an iterative training process with means like the evolutionary algorithm. Note that Huber loss does not have second-order derivatives and therefore is not accessible for algorithms that apply a second-order optimization policy. Fortunately, it is not an issue for ENCFIS as it adopts first-order gradient optimization.

6.3.3 Robust Learning Process for ENCFIS

The robust learning strategy for ENCFIS includes several steps. First, a fuzzy c-means clustering procedure is applied to pre-train antecedent parameters with only input variables to minimize the influence of potential label noise; then, the algorithm undergoes forward propagation to determine the current Huber loss; the DEMON momentum decaying and the Welsch M-estimator are utilized to iteratively optimize the nonlinear antecedent parameters and the consequent parameters, respectively, until reaching the minimum Huber loss. Note that if the variables are too different in scale, it may negatively affect the learning algorithms. Thus, it is crucial to normalize the data according to the Z-score [212] method so that this strategy gets the best performance. The formula for the Z-score normalization is as follows:

$$x_z = \frac{x - \mu_z}{\sigma_z} \tag{6.31}$$

where μ_z is the mean of the samples, σ_z is the standard deviation, and x_z is the normalized value. After the standardization, clustering is employed to pre-train the antecedent part of the network using only input variables. By operating the fuzzy c-means on all training samples, the vector of r cluster centroids $C = \{c_1, c_2, \ldots, c_i, \ldots, c_r\}$ is obtained. Here, r must be identical to the number of fuzzy rules assigned to each input variable. Subsequently, the sum of the distances between samples and their corresponding centroid in every cluster is calculated, i.e., $D = \{d_1, d_2, \ldots, d_i, \ldots, d_r\}$, where $d_i = \sum_{k=1}^{n} \left(x_i^k - c_i\right), n$ denotes the number of samples in this cluster. Given that the fuzzy c-means clustering results in the fuzzy partition matrix rather than explicit clusters, the centers used here come from the clusters with the highest membership values in the matrix. Then, the antecedent parameter set $\{\mu^i, b^i, \lambda^i\}$ is determined as follows:

$$\mu^i = c_i, b^i = \rho \sum_{k=1}^{n} \left(x_i^k - c_i\right), \lambda^i = 1 \tag{6.32}$$

where μ^i denotes the center of each kernel, b^i refers to the width and λ^i represents the scale factor for the complex component. ρ is said to be the expansion factor, which is defaulted to be 1.

For the type-1 Gaussian fuzzy kernels, after the previous step, the antecedent parameters should be very close to the global optimum. But for the complex fuzzy kernels, there is still a non-ignorable difference. At this time, the DEMON gradient optimization strategy can further refine the parameters. Combine the equations (6.8) and (6.9), the expression of the update rule for parameter μ^i at the recursion $k+1$ is given as follows:

$$\begin{cases} \eta(k) = \eta_{init} \frac{1-\frac{k}{T}}{(1-\eta_{\text{init}})+\eta_{\text{init}}\left(1-\frac{k}{T}\right)} \\ \mu^i(k+1) = \mu^i(k) - \alpha h(k) \frac{\partial f(k)}{\partial \mu^i}\Big|_{\mu^i=\mu^i(k)} + \eta(k) v(k) \\ v(k+1) = \eta(k) v(k) - \alpha h(k) \frac{\partial f(k)}{\partial \mu^i}\Big|_{\mu^i=\mu^i(k)} \end{cases} \tag{6.33}$$

where $h(k)$ is the pseudo-residual under the Huber loss defined by equation (6.30). $\frac{\partial f(k)}{\partial \mu^i}$ is the partial derivative with respect to μ^i. Further, one can obtain the update rules for parameters b^i and δ^i by replacing $\frac{\partial f(k)}{\partial \mu^i}\Big|_{\mu^i=\mu^i(k)}$ with $\frac{\partial f(k)}{\partial b^i}\Big|_{b^i=b^i(k)}$ and $\frac{\partial f(k)}{\partial \lambda^i}\Big|_{\lambda^i=\lambda^i(k)}$, respectively.

The linear consequent parameters are estimated using the Welsch M-estimator, which is a vital step to robust learning. Assume there is a set of N data samples involved, then define the equation coefficient matrix S^i, weight matrix W^i, data label vector y and the parameter vector θ^i as follows:

$$S^i = \begin{bmatrix} f_0^i(x_1) & f_1^i(x_1) & \cdots & f_q^i(x_1) \\ f_0^i(x_2) & f_1^i(x_2) & \cdots & f_q^i(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_0^i(x_N) & f_1^i(x_N) & \cdots & f_q^i(x_N) \end{bmatrix},$$

$$W^i = \begin{bmatrix} w_1^i & \cdots & 0 & 0 \\ 0 & w_2^i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_N^i \end{bmatrix},$$

$$y = [y_1, y_2, \ldots, y_N]^T,$$

$$\theta^i = \left[a_0^i, a_1^i, \ldots, a_a^i\right]^T,$$

where q equals to the number of fuzzy rules, $w_0^i, w_1^i, \ldots, w_q^i$ are determined in accordance with formula (6.17). Hence, the estimated consequent parameter at the kth recursion can be calculated using the following equation:

$$\hat{\theta}^k = \left[\left(S^k\right)^T W^k S^k\right]^{-1} \left(S^k\right)^T W^k y. \tag{6.34}$$

The clustering step only needs to be used once, while the DEMON optimizer and M-estimator joint learning approach is supposed to repeat several recursions until the error requirement is satisfied or it reaches the maximum epoch setting. Such a learning process is extremely robust against label noise for the following reasons. Firstly, the clustering process does not involve labels, thereby more accurately estimating the antecedent parameters. Secondly, the genuine residuals are replaced with pseudo residuals to conduct gradient optimization, which offsets the influences of noise in training objectives. Thirdly, the noise-insensitive M estimator is adopted to obtain the consequent parameters, which avoids the noise sensitivity problems caused by ordinary LS. The simplified algorithm procedures are as follows:

Algorithm 6.2 Robust Learning for ENCFIS.

Step 1. Normalize the dataset according to the Z-score standardization formula.
Step. 2 Pre-determine the antecedent parameter set $\{\mu^i, b^i, \lambda^i\}$ using information obtained from fuzzy c-means.
Step. 3 Compute outputs with the initial setting, then generate the pseudo-residual $\hat{h}(\tau)$ under the Huber loss.
Step. 4 Refine the antecedent part (non-linear) of the network by the DEMON momentum decaying method.
Step. 5 Estimate the consequent part (linear) with Welsch M-estimator.
Step. 6 Calculate the output of the network, and obtain the renewed pseudo-residual $\hat{h}(\tau)$.
Step. 7 Repeat Step.4-Step.6 until the stopping condition is satisfied.

6.4 Experimental Results and Analyses

6.4.1 Corrupted Synthetic Data Test

The single-input-single-output synthetic dataset is produced using the following sinusoidal function:

$$f(x) = \sin(x-3), 0 \leq x \leq 10, \tag{6.35}$$

where 201 original data points are generated by isometric sampling from the range [0,10] with an interval of 0.05. Corruption is then added by replacing part of the initial data with random values ranging from [-3, 2]. A total of 45% of the data points were "corrupted" during this process, which poses an enormous challenge to any machine learning model. It is worth noting that there is no testing or validation set for this experiment. The algorithm performance is evaluated based on its ability to "restore" the original data. However, given that the RMSE computed using corrupted data makes no sense, the RMSE value is evaluated according to the original data. The hyperparameter setting for this test is available in Table 6.2, and the comparison between the proposed ENCFIS model and other models is in Table 6.3. The actual MSE and the Pseudo-MSE during the training process are side by side displayed in Figure 6.3. The visualization of the data restored by different models is shown in Figure 6.4.

Table 6.2: ENCFIS hyperparameter setting for the corrupted synthetic data test.

Bisecting k-means clustering for antecedence pre-training			
Cluster number k	9	Expansion factor ρ	1
DEMON		M-estimator and Huber Loss	
Learning rate α	10^{-6}	Tuning constant γ	2.9850
Initial momentum factor β	0.1	α-cut factor δ	0.15

Table 6.3: RMSE of the models over the corrupted synthetic data test.

	BP	RBF	GRNN	EA-SVR	Type-1	Type-2	ENCFIS
RMSE	0.639	0.419	0.380	0.126	0.0271	0.0219	0.0082
Iterations	100	100	100	30	100	–	60

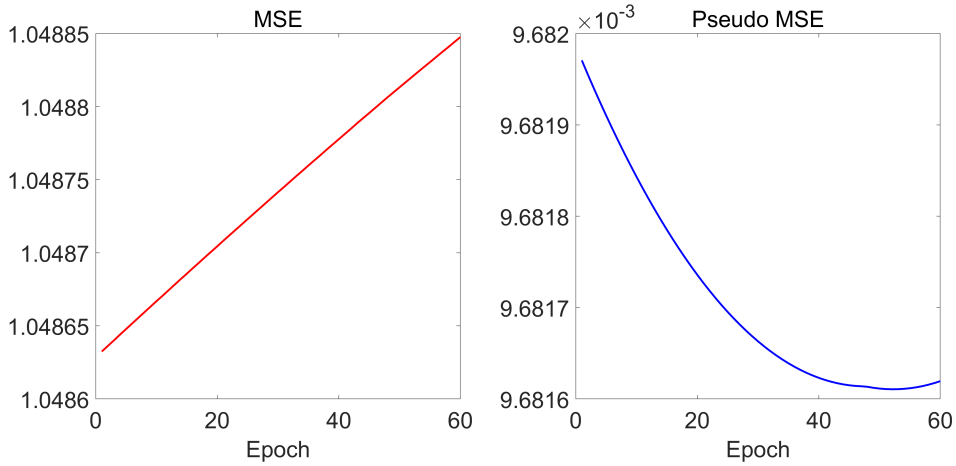

Figure 6.3: MSE and Pseudo-MSE in training.

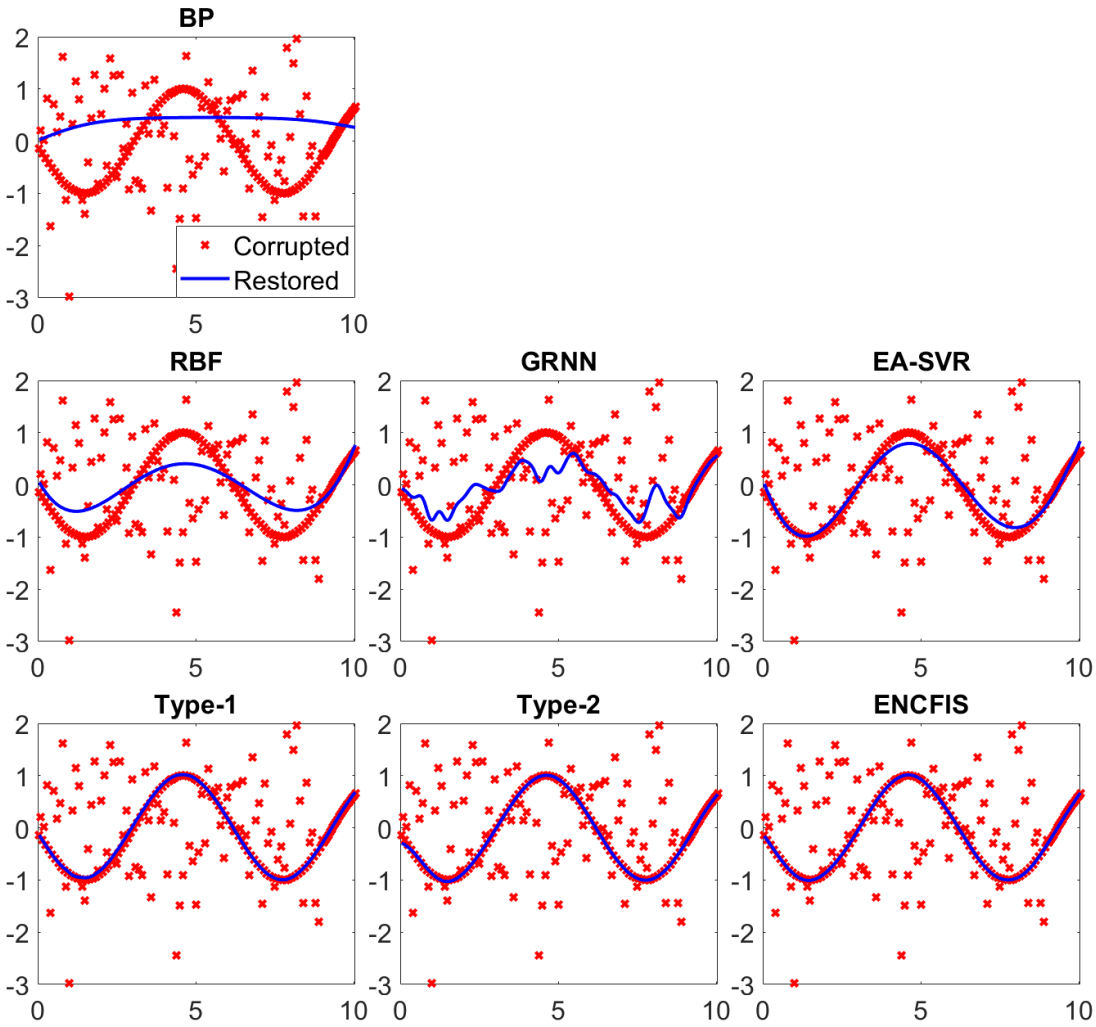

Figure 6.4: Restored sinusoidal curves.

As it can be easily seen from Figure 6.3 that the variation trends of MSE and pseudo-MSE values during training, computed in accordance with residuals and pseudo residuals, respectively, show a remarkable difference. The genuine residuals from the training process lost their authenticity as a measure of the empirical error due to the extensive noisy data points in the dataset. Instead, pseudo-residuals generated by the Huber loss function play a crucial role in measuring the training progress. Regarding the "restoration" performance, seven models, including BP [52], RBF [53], GRNN [54], EA-SVR [193], Type-1, Type-2, and ENCFIS are tested, where Type-1 is a counterpart of ENCFIS that uses exactly the same robust learning strategy but takes the traditional type-1 fuzzy logic instead of the complex fuzzy logic. The Type-2 model is obtained by expanding the membership function on top of the trained type-1 counterpart, where the Lower membership function delay factor is set to 0.1. Both Type-1 and Type-2 models are specially introduced as a control group to validate if the complex fuzzy logic is conducive to the performance and robustness of the model. Furthermore, EA-SVR is an SVR model that applies an evolutionary algorithm [213] to tune the hyperparameters for obtaining the best performance from the SVR model.

As shown in Figure 6.4, the most used but non-robust regression models, such as BP, RBF, and GRNN, are inefficacious under the presence of such a high level of noise. Even the noise-resistant SVR model is influenced, despite the adoption of EA, which also confirms that the SVR strategy of offsetting noises via setting a "margin" has limitations. However, the proposed ENCFIS, Type-1 and Type-2 models obtained the best outcome in this data recovery challenge. Further investigation of the RMSEs in Table 6.3 reveals that ENCFIS not only far outperforms all models, including SVR, but also significantly outperforms Type-1 and Type-2, which are the type-1 and type-2 fuzzy logic counterparts of ENCFIS as mentioned above. Note that the RMSE values were calculated by comparing the restored data with the uncorrupted original data. This test shows a positive result for the effectiveness of the proposed robust learning method, and the introduction of complex fuzzy logic is also immensely conducive to the accuracy and robustness. Thus, it is sufficient to conclude that the ENCFIS model is robust when only high-level label noise is presented.

6.4.2 Corrupted Sunspot Time Series Test

In this experiment, 602 Sunspot observations from 1976 to 1980 are used to create the contaminated training set, while another 602 records from 1984 to 1989 are taken to form the clean validation set. The form of each data point used for the test is as $\{\chi(\tau-2),\chi(\tau-1);\chi(\tau)\}$, in which two previous observations are utilized to create the input vector while the current observation $\chi(\tau)$ is considered the output label. For the training set, around 150 out of the 602 observed values are replaced with random values, which means 25% of the time series is compromised. It also suggests that if the training data is constructed using the above form, 50% of the data points will contain one incorrect input variable, 25% of the data points will have a misleading training label, and only 25% of the data points are intact. This experiment is far more challenging than the previous one because only the label noise exists in that case while the inputs are unaffected. But time series forecasting requires applying the earlier observations to predict the future ones, which gives rise to the inescapable perturbation inside input vectors, bringing daunting challenges to the prediction algorithm. In other words, all benchmark models suffer from together intense label noises and faulty inputs in this test. The hyperparameter setting of ENCFIS is available in Table 6.4, the RMSEs of all benchmark models are given in Table 6.5, and the visualization of model outputs on both the contaminated training set and pure validation set are displayed in Figure 6.5 and Figure 6.6.

Table 6.4: ENCFIS hyperparameter setting for the corrupted Sunspot data test.

Bisecting k-means clustering for antecedence pre-training			
Cluster number k	6	Expansion factor ρ	1
DEMON		M-estimator and Huber Loss	
Learning rate α	10^{-5}	Tuning constant γ	2.9850
Initial momentum factor β	0.3	α-cut factor δ	0.1

Table 6.5: RMSE of the models over the corrupted Sunspot data test.

	BP	RBF	GRNN	EA-SVR	Type-1	Type-2	ENCFIS
RMSE	23.9236	37.5726	75.1519	17.2526	14.1203	17.0519	6.4720
Iterations	100	50	100	30	50	–	30

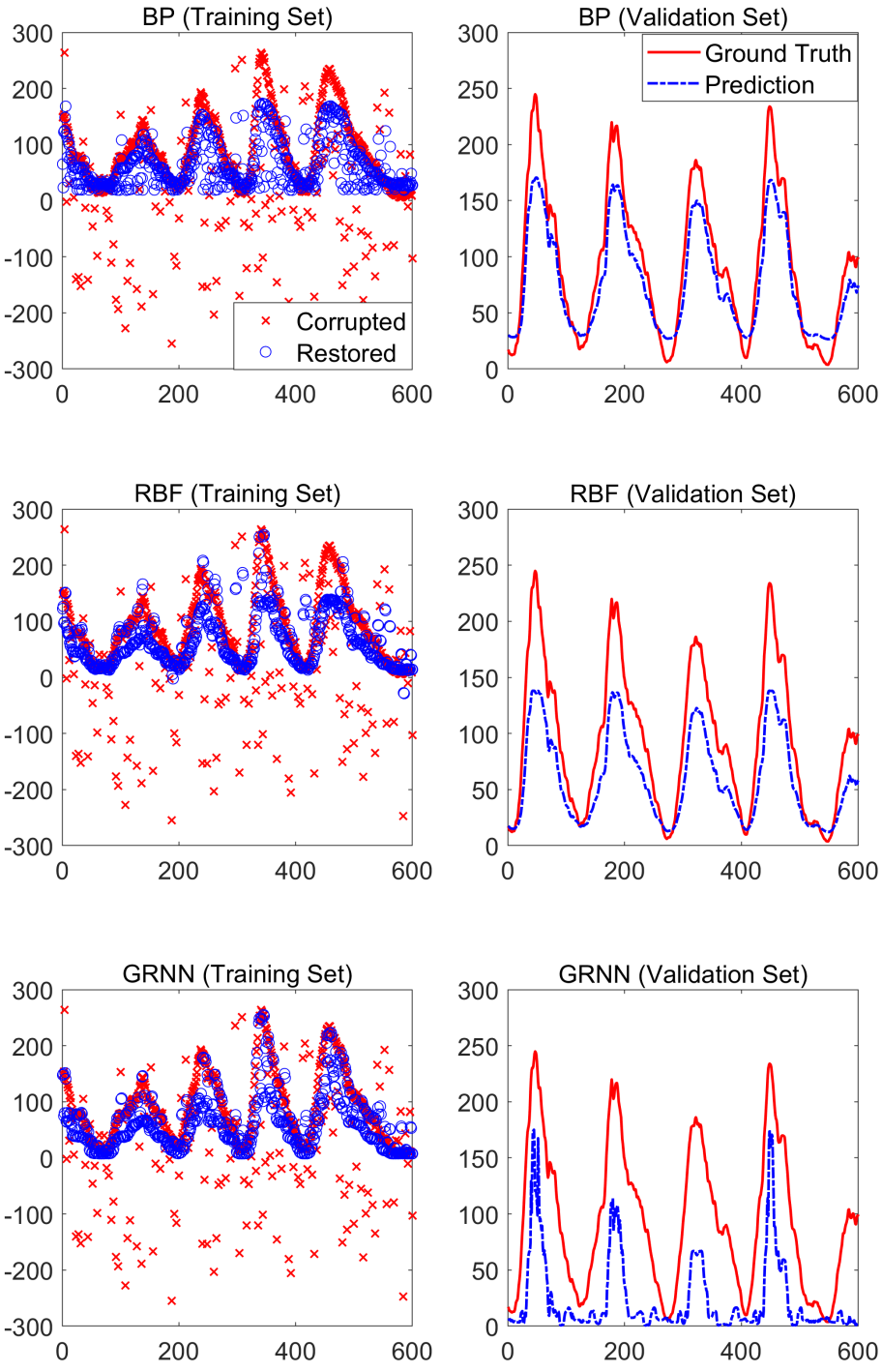

Figure 6.5: Corrupted Sunspot time series prediction on both the training set and validation set. (Conventional models)

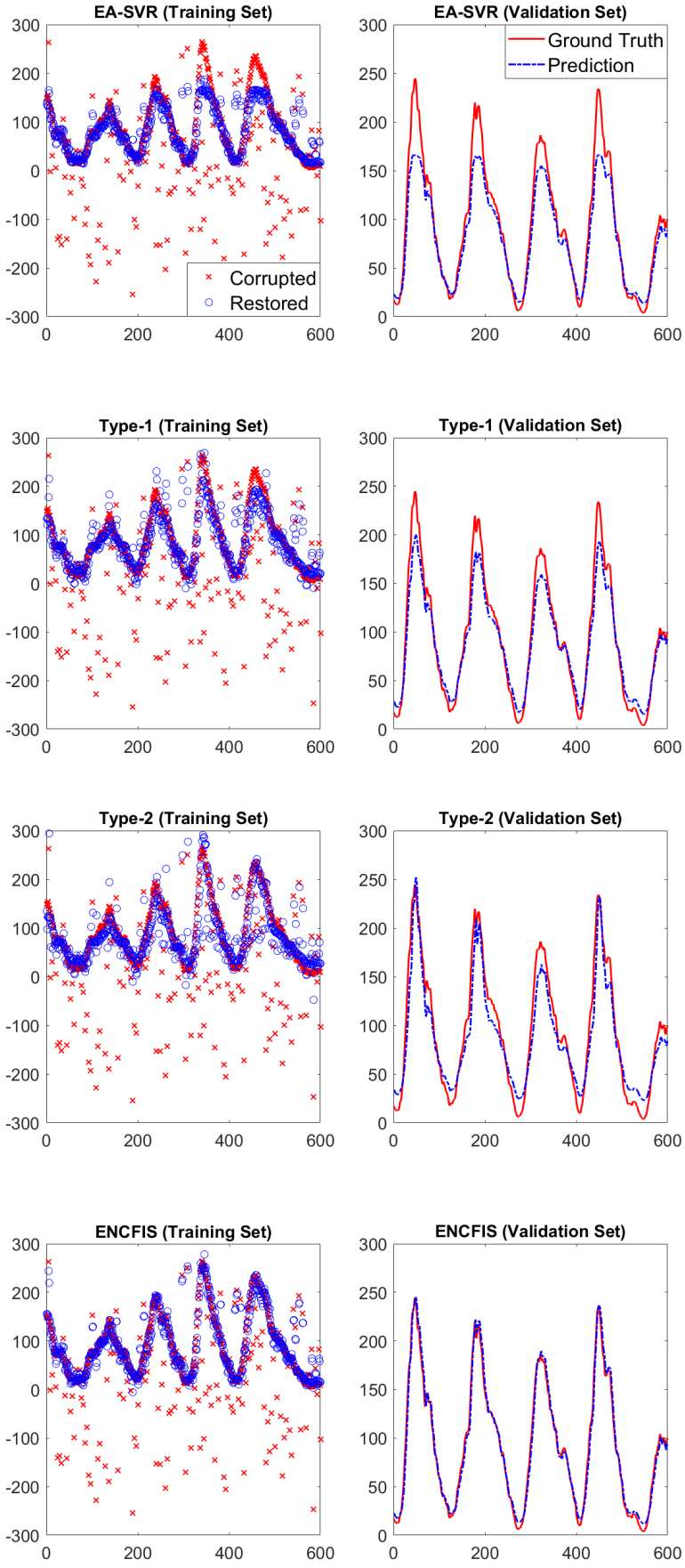

Figure 6.6: Corrupted Sunspot time series prediction on both the training set and validation set. (Noise robust models)

As expected, three non-robust algorithms, such as BP, RBF, and GRNN, perform poorly in this test. Traditional machine learning models have difficulty handling such data types with heavy noise in both the input and the label. These models cannot identify noisy data points and remove their interference but give noises the same credit as the genuine data samples, making the training process deviate from the correct direction. EA-SVR outperforms non-robust models but with evident deviations in the prediction of peak points due to the false exclusion of some valid data points near the peak. Given that the solution of tackling noisy data by the SVR algorithm does not include a mechanism that distinguishes noise and data according to their statistical distributions, the occurrence of such problems is difficult to avoid. Additionally, even the “Type-1” and "Type-2" models that uses the same robust learning strategy as the ENCFIS model failed to accurately predict the Sunspot peaks, making ENCFIS the only winner in this challenge.

In fact, the Sunspot time series is not difficult to forecast, and many models can achieve good performance on this dataset. The only reason for the poor performance is the heavy noise and massive outliers. Thus, it is fair to say that model robustness against disturbances is the key to excelling in this case. From this perspective, the ENCFIS model exhibits spectacular robustness, as it barely gets affected by the outliers, which is not only reflected in Figure 6.5 but also verified from the RMSE values on Table 6.5. The ENCFIS model applies the Welsch influence as a reasonable hypothesis on the probability distribution of valid data points, which designates low weights to noises or outliers that do not conform to this distribution during the training process while assigning high weights to data points that comply to this hypothetic distribution, thus achieving good noise robustness. It also confirms that using the difference in probability distribution to identify noises is more reliable than the SVR solution of simply setting "soft boundaries" for outliers. Moreover, given that ENCFIS’ type-1 and type-2 fuzzy logic counterparts, i.e., the “Type-1” and "Type-2" models, also interfered by the noise, it leads to the conclusion that the two-dimensional properties of complex fuzzy logic and the disturbance insensitivity it brings are vital to the success of ENCFIS as well.

6.4.3 ENCFIS on Ultimate Tensile Strength (UTS) Modeling

The ultimate tensile strength refers to the peak of engineering stress in a stress-strain curve, which is highly nonlinear and sparse. This experiment tests 1,500 data samples from actual industrial processes, of which 1,000 consists of the training set and the other 500 forms the test set. This dataset has 15 input variables, of which 13 are numerical, whereas the rest 2 are categorical. Given the high dimensionality of the data and the presence of data that are not strongly correlated with predicted values, only the 10 most relevant influencing variables are selected to construct input vectors, i.e., C, Mn, Cr, Mo, Ni, Site, Tempering Temperature, Cooling Medium, Sample Size, and Test Depth. In addition, since the data comes directly from industrial processes, it contains an unknown number of outliers caused by measurement errors or incorrect human entry, which, albeit in limited numbers, may still offset model performance. It is believed that such a test is representative of most real-world scenarios to validate the model's potential in applications. The hyperparameter setting of the network is available in Table 6.6, and the performance of all benchmark models is given in Table 6.7.

Table 6.6: ENCFIS hyperparameter setting for the UTS data test.

Bisecting k-means clustering for antecedence pre-training			
Cluster number k	6	Expansion factor ρ	1
DEMON		M-estimator and Huber Loss	
Learning rate α	10^{-6}	Tuning constant γ	3.9104
Initial momentum factor β	0.3	α-cut factor δ	0.2

From the perspective of RMSE values, the performance of different models on this dataset varies widely. As a result of the high dimensionality, data sparsity, and perturbation caused by outliers, traditional non-robust neural network models such as BP, RBF, GRNN, LSTM citeref43, and DBN [66] have underwhelming performance on this dataset. The classical ANFIS neuro-fuzzy model does not achieve the best prediction performance as well due to the limited expressive ability of the type-1 fuzzy logic. Only three models achieved MSE values that are less than 40, namely EA-SVR, IT2-Sugeno [12], and ENCFIS and those are close in performance. The first-rate performance of the SVR model is attributed to its unique vector representation of the input data, which

makes it immune to high dimensionality and sparsity. SVR is also robust to noise owing to its unique design, which further contributes to its advantages. IT2-Sugeno is a type-2 fuzzy model that benefits from the richer semantic representation of the interval type-2 fuzzy logic. The enhanced generalization capability of the interval type-2 rule-base also increases the robustness, as its parameters are less prone to fluctuations due to occasional outliers. However, both methods have conspicuous disadvantages. SVR is inefficient when dealing with large samples because it needs to generate a support vector for almost each data point, whereas the traditional type-2 fuzzy model is not robust enough for heavy noises. Furthermore, both methods require the evolutionary algorithm as the optimization policy, which brings the computational complexity a magnitude higher than gradient optimization.

Regarding ENCFIS, it obtained the best performance thanks to the proposed robust learning strategy and the improved generalization capability brought by complex fuzzy logic. The algorithm efficiency is also much higher than other benchmark algorithms owing to the proposed learning strategy. It is reflected by the training epochs, as ENCFIS achieves the lowest RMSE value with only 20 iterations. It is fair to say that the proposed ENCFIS is an extraordinary model that integrates performance, efficiency, and robustness, making it highly adaptable to intricate numerical regression tasks and real-world application scenarios. It is commendable that, unlike most robust models, ENCFIS does not sacrifice its accuracy for the exchange of extra robustness, which proves that excellent robustness and good model performance can co-exist in the same architecture.

Table 6.7: Comparison of the performance (UTS data test).

	STD	MAE	SMAPE	RMSE	**Rules**	**Epoch**
BP	150.9674	32.1335	3.4460	44.4965	-	200
RBF	149.2079	41.2173	4.4341	54.1319	-	100
GRNN	153.8146	39.2092	4.1841	56.5168	-	100
LSTM	137.5868	42.4865	4.5160	56.5765	-	200
DBN	142.1892	36.1535	3.9260	47.7999	-	50
EA-SVR	153.1510	29.5739	3.1759	39.5723	-	30
IT2-Sugeno	-	-	-	38.7600	6	100
ANFIS	148.6884	36.4036	3.9672	45.4163	6	50
ENCFIS	153.6219	28.8886	3.1456	38.0137	6	20

6.4.4 ENCFIS on Charpy Impact Energy Modeling

The Charpy impact energy dataset is also an industrial dataset with 16 input variables and one dependent variable, including 14 numerical and 2 categorical inputs. Although this dataset is less sparse than the UTS dataset, it has slight stochasticity. The Charpy test suffers from a certain degree of uncertainty, and inconsistent results can occur even with the same initial conditions. Inevitably, the measured impact energy is only a typical value obtained after performing at least three trials instead of the exact value. Rough modeling is relatively easy in such cases, whereas accurate modeling is significantly more challenging. Besides, the data is even trickier in the presence of a small number of outliers in this data due to human error. As a result, many models can exhibit acceptable performance on this dataset but struggle to improve the accuracy further. In this experiment, only the 9 most correlated variables are considered, i.e., C, Si, Mn, Cr, Mo, Ni, Sample Size, Tempering Temperature, and Impact Temperature, to alleviate the problems caused by the curse of dimensionality. The initial hyperparameter settings of the ENCFIS network are shown in Table 6.8, and the experimental result and its comparison with other benchmark models are available in Table 6.9.

Table 6.8: ENCFIS hyperparameter setting for the Charpy impact data test.

Bisecting k-means clustering for antecedence pre-training			
Cluster number k	6	Expansion factor ρ	1
DEMON		M-estimator and Huber Loss	
Learning rate α	1.5×10^{-6}	Tuning constant γ	4.7407
Initial momentum factor β	0.3	α-cut factor δ	0.15

Table 6.9: Comparison of the performance (Charpy impact data test).

	STD	MAE	SMAPE	RMSE	**Rules**	**Epoch**
BP	25.2233	16.9010	21.2874	20.8389	-	200
RBF	30.9238	14.7211	19.5694	20.1443	-	100
GRNN	22.7647	16.0598	19.0444	22.0339	-	100
LSTM	24.9117	17.0228	21.5417	21.2242	-	200
DBN	27.2085	16.2536	20.2341	20.1211	-	50
EA-SVR	25.7636	14.1185	18.0370	19.4008	-	30
IT2-Sugeno	-	-	-	19.6500	8	100
ANFIS	25.2724	16.1874	21.5623	20.1235	6	50
ENCFIS	26.6696	13.0726	17.1354	16.8880	6	30

According to Table 6.9, the RMSE performance of most benchmark models is very close, with only three models having RMSEs less than 20, i.e., EA-SVR, IT2-Sugeno, and ENCFIS, of which ENCFIS performs much better than the previous two and becomes the best in the test. It is worth noting that the RACFIS model presented in the chapter before also achieved good prediction accuracy on this dataset, but the mechanism is not the same as ENCFIS. The excellent performance of RACFIS is attributed to three factors: complex fuzzy logic, online learning, and a hybrid gradient-based optimization algorithm. ENCFIS does not apply an online learning strategy due to the fact that the robust estimator requires a sufficient number of samples to differentiate the difference in distribution between data and noise, for which only batch learning is possible. However, the robust learning strategy of ENCFIS gives the model an additional advantage, as it not only improves the noise tolerance of the model but also enhances its capability to handle "coarse" datasets.

The correlation between the input variables and the label of the Charpy impact dataset is not very strong compared with many other application scenarios due to the influence of the stochasticity of the Charpy impact test. Such a situation is actually the purpose that the robust estimator was initially designed for, i.e., handling data with relatively poor quality. ENCFIS is an advanced extension of the robust estimator in the field of nonlinear regression, which inherits all the properties of the robust estimator. The model applies a weight to each data point to reflect its correlation based on the influence function, which not only can eliminate irrelevant noises but also makes full use of the less correlated data, maximizing the information learned from the data and thus improving the model performance. This mechanism explains why ENCFIS can achieve the lowest regression error compared to all the benchmark algorithms as well as the other models presented in previous chapters on the Charpy impact data test. The above features are proof of the incredible adaptability of ENCFIS to real-world datasets, making it the most successful and promising algorithm presented in this thesis.

6.5 Chapter Summary

In this chapter, the first robust machine learning architecture based on complex fuzzy theory for numerical regression purposes is presented. This algorithm has a specially designed learning strategy to neutralize the effect of noises and outliers. First, it utilizes clustering of input variables to pre-train kernel parameters to avoid the influence of label noises. Then, the momentum decay gradient method, Huber loss, and Welsch M-estimator are applied to form an iterative hybrid robust learning method, in which the M-estimator assigns weights to data samples based on the hypothetical premises provided by the Welsch influence function to reflect their relevance. Not only does this effectively neutralize the effects of massive noise, but it also maximizes the use of less correlated data samples, improving the algorithm's performance on rough datasets. Huber loss combines the advantages of both the MSE and MAE loss functions to generate "pseudo-residuals" during training to replace the genuine residuals that are invalidated by a large amount of noise, thus improving the robustness of the model.

In addition, the existence of closed-form solutions for the first derivatives of complex fuzzy membership functions enables gradient optimization, thereby increasing algorithm efficiency, which is a significant advantage over the type-2 architecture for which gradient solutions are unavailable. The gradient-based momentum decay optimization also effectively avoids the hyperparameter sensitivity of the constant momentum method, reducing the difficulty of parameter tuning and enhancing the convenience of practical scenarios. The two-dimensional attribute and richer information capacity of the complex fuzzy rule-base also increase the generalization capability and anti-disturbance performance. Experimental results also confirm that the proposed ENCFIS algorithm exhibits incredible adaptability for a problem with only label noise and when both input variables and labels are noisy. It also has astonishing performance for some coarse real-world datasets, and its superb noise robustness does not come at the expense of accuracy, which is rare for models with similar purposes. Therefore, it is reasonable to believe that ENCFIS is very successful as a robust learning architecture with the potential to enable machine learning under harsher data conditions.

7

Conclusions and future work

> *There isn't always an explanation for everything.*
>
> **– Ernest Hemingway, A Farewell to Arms –**

7.1 Conclusions

Data modeling algorithms can be divided into three categories, i.e., process-driven, algorithm-driven, and data-driven. Process-driven approaches are rule-based and suitable for static and repeatable scenarios but are less flexible and require customization for each specific application. Algorithm-driven designs rely on abstract mathematical models, which can offer better flexibility than process-driven tools on static data conditions, but are often less accurate, especially when data features are intricate. Both approaches have the advantage of being well-interpretable and the disadvantage of being dependent on expert knowledge. Data-driven algorithms are virtually end-to-end inductive statistical models that only require raw data samples and labels to generate the mapping relations and are considered the most convenient, flexible, and accurate, which can also be applied even when concrete mathematical expression is unavailable but at the expense of reduced interpretability. In recent years, rising end-to-end adaptive learning algorithms are replacing traditional process-driven or algorithm-driven modeling methods in many areas.

Fuzzy algorithms are also moving in a data-driven direction, the importance of interpretability decreases, and model design is biased towards better learning from the data rather than explaining the physical meaning of the rules. Following this trend, self-learning deep neuro-fuzzy models are favored, for which the research points of interest are similar to those for ordinary deep network architectures, i.e., efficiency, accuracy, and data applicability. Such models are expected to have faster convergence, use smaller training sets and achieve better prediction accuracy. However, accomplishing these goals is not easy, and the main challenge comes from the properties of the raw data itself, such as high dimensionality, sparsity, non-linearity, and noises, where high dimensionality triggers the curse of dimensionality, leading to a drop in accuracy of the fuzzy algorithm and an increase in computational complexity. Sparsity and non-linearity can seriously affect the model regression performance, which may even cause convergence failure in extreme cases. Noisy nature of the real world also interferes with the model's ability to acquire information from the data and invalidates the learning model. This thesis explored the outlook of using complex-valued structures and

complex fuzzy theory to enhance the performance of deep neuro-fuzzy data modeling algorithms regarding the above perspectives.

In Chapter 4, a complex-valued Wang-Mendel (CVWM) method and a deep complex-valued single-iteration fuzzy system (DCVSF) were proposed to mitigate the curse of dimensionality of the traditional Wang-Mendel (WM) method. The WM method is known for its single-iteration training feature, which differs from many neuro-fuzzy systems that rely on optimization algorithms to adjust the network parameters to realize a non-linear mapping iteratively. However, such a training process involves assigning different firing strengths to the rules according to their relevance to the problem, which requires the initialization of a remarkable number of rules for each variable to cover all possibilities. Unfortunately, the rule-base of the WM model suffers an exponential dilemma, for which the size is determined by the number of rules for a single variable as the base and the number of input variables as the power, meaning it is highly susceptible to the curse of dimensionality. The CVWM design is innovative in that it introduces a complex-valued structure to the original WM method, using complex-valued arithmetic logic to deal with real-valued problems and successfully bringing its rule base down to the square root size of the original version. The DCVSF architecture handles very high dimensional data types by incorporating CVWM units into a hierarchical fuzzy system while only generating a relatively controllable size of rule-base. Also, both algorithms retain the WM method's property to finish training in a single iteration and are even more efficient due to the reduced rule base. The smaller rule-base coupled with the one-pass training property makes the above models ideal for memory-constrained scenarios.

Experimental results in Chapter 4 indicated that the newly proposed algorithms are successful, as they significantly reduce the size of the rule-base while the performance loss is almost negligible for dense data problems. Notably, both algorithms also inherit the shortcoming of the original WM algorithm, i.e., the inability to tackle the sparsity, which is even exacerbated by the introduction of the complex-valued structure. Attempts to alleviate this problem using dimensionality reduction methods have had limited success and remain ineffective for highly sparse datasets. It is an intrinsic

flaw of the WM-like algorithm that is impossible to resolve by improvement because the sparsity often leads to many untriggered rules in the rule-base, preventing it from reflecting the whole picture of the data. Therefore, it is better to seek other solutions for sparse data types.

In Chapter 5, a rapid adaptive complex neuro-fuzzy system (RACFIS) was developed to target real-world application situations, which uses a new complex fuzzy theory as the inference logic, leading to significant advantages over the traditional type-1 model and a competitive performance compared with the type-2 model as well. The membership for complex fuzzy sets is a two-dimensional complex-valued notion defined in the unit circle in the complex plane, allowing them to accommodate more information and enabling better generalization capabilities of the algorithm on regression modeling. Such a property can also help reduce the number of rules required for the model, improving efficiency and mitigating over-fitting. At the same time, the periodic nature of complex numbers increases the algorithm's performance against quasi-periodic problems, which is conducive to tasks such as time series forecasting and stream data processing. Furthermore, another advantage of complex fuzzy membership functions over their interval type-2 counterparts is that they have algebraically closed-form first-order derivatives, making first-order gradient optimization possible. The gradient is a priori knowledge of the optimization surface and has positive implications for the efficiency of the optimization algorithm. Interval type-2 networks are forced to apply derivative-free optimization schemes because their membership functions do not exist in representable first-order derivative form, which increases the computational complexity and weakens the efficiency of the optimizer.

The proposed RACFIS utilizes a hybrid gradient-based optimization policy and incorporates unsupervised learning as a pre-train method. It first uses convex clustering methods to pre-train the antecedent parameters of the network so that they are closer to the global optimum than the randomly initialized ones. The model then applies quasi-hyperbolic momentum (QHM) and regression least squares methods (RLS) to iteratively optimize the network's antecedent and consequent parameters to obtain the final global optimum. QHM is a new multi-parameter gradient optimization solution, faster and

smoother than traditional gradient-momentum solutions, and its combination with RLS results in a very efficient hybrid online optimization strategy. The meaning of introducing online learning into the RACFIS model is that it provides better test set prediction accuracy than traditional batch learning and further accelerates network convergence. Experimental results in Chapter 5 showed that the RACFIS model is highly efficient and can converge using only a fraction of the number of iterations required by other benchmark algorithms. The size of the rule-base is also competitive due to the use of complex fuzzy sets. RACFIS exhibits impressive accuracy and generalization capability on tricky real-world datasets, putting it well ahead of all benchmark models employed for comparison. RACFIS can accommodate most data types in real-world application scenarios, including sparse data. However, there are also cases where massive noises and outliers exist in the data for real-world situations, demanding a more robust and specialized architecture.

In Chapter 6, an exclusionary neural complex fuzzy inference system (ENCFIS) was presented as a robust machine learning solution. Robust learning is a promising area of advanced machine learning, primarily for scenarios in which massive noises and outliers are existential. Traditionally, one can remove noise from data through means such as filtering, but in many cases, it is impossible to completely prevent noise-containing data from feeding into the model. The learning process of conventional machine learning algorithms relies on the hypothetical premise that the target data is clean, whereas the presence of statistically significant noises or outliers can cause performance degradation or even model failure. Robust learning models have unusual mechanisms that lead to less susceptibility to outliers, thus maximizing the valid information obtained from the data.

The proposed ENCFIS is the first robust neuro-fuzzy system, and the attempt is unprecedented. Firstly, it utilizes clustering methods to pre-train the antecedent parameters of the network, which increases robustness in the presence of label noise, as clustering is an unsupervised learning method and does not require the involvement of data labels. Secondly, the linear parameters are optimized using an M-estimator based on the Welsch influence, which assigns weights to each data sample according to the difference

between it and the assumed statistical distribution, thus marginalizing outliers. Thirdly, ENCFIS employs the Huber robust loss function as the objective function instead of the traditional MSE loss function. The pseudo-residuals generated by the Huber function replace the genuine residuals, which lose credibility to reflect the training error due to high noise conditions. This design significantly neutralizes the interference of the label noise on the optimization target and ensures that the optimization proceeds in the right direction. In addition, complex fuzzy logic also has a positive effect on the robustness of the model, as the complex operations lead to an increased generalization capability, reducing its sensitivity to outliers to a certain extent. Also, given that robust models are often sensitive to hyperparameters, a gradient-based momentum decay algorithm is employed to refine the antecedent parameters. The decaying momentum algorithm is considerably less susceptible to the initial parameter settings than its constant gradient counterpart, thus enhancing the convenience of the ENCFIS architecture under real-world conditions.

Experimental results in Chapter 6 confirmed that ENCFIS is surprisingly resilient to high-noise environments. For data with only label noise, it can complete modeling almost unaffected on a synthetic dataset with 45% data corruption. It also exhibits excellent performance for tricky datasets where the label and input noises are present at the same time. The performance of ENCFIS on the tests involved even exceeds that of support vector machines that are considered highly robust to noise, demonstrating the superiority of the algorithm. It is also remarkable that the performance of ENCFIS for regular clean data does not diminish despite its high robustness, which is rare even for robust models. It is reasonable to believe that ENCFIS is a successful attempt at robust machine learning with promising research and application value.

Nevertheless, not much work on the interpretability of the proposed models in this thesis, and several reasons hinder the discussion in this direction. For CVWM and DCVSF models, the complex-valued structure destroys the physics of the fuzzy membership functions, making the rule-base uninterpretable. The RACFIS and ENCFIS models are also not feasible for interpretation at the current stage because the research of complex fuzzy logic is still in its infancy, with no mature and widely accepted theory to

decipher its semantic representations. In addition, the increased network depth and dimensionality may also undermine the model's interpretability. Despite this, the models presented in this thesis are still transparent, with apparent advantages over black-box models in debugging and parameter tuning, which is good enough for most data-driven scenarios.

7.2 Future work

The application of complex-valued structures and complex fuzzy theory is successful in this thesis. For future research, there is little room for complex-valued architectures solely for mitigating the curse of dimensionality due to obvious reasons, whereas the complex fuzzy theory holds vast promise. Currently, two key issues need to be resolved urgently to make complex fuzzy theory better serve the neuro-fuzzy systems. The first is that it is necessary to develop a complex membership function superior to existing solutions, as the three emerged versions are rather primitive and cannot maximize the benefits of complex fuzzy logic. Also, the first-order derivatives of these functions are too sophisticated, which causes inconvenience in network design and development. Second, although interpretability can be weak for many deep neuro-fuzzy systems, it is still necessary to unravel the semantic interpretation of complex fuzzy logic, which could extend the theory for applications such as data mining and knowledge extraction. More knowledge of the mechanism by which its rules work may also help design new neuro-fuzzy systems based on such theory.

For network architecture, several promising research directions are also available for exploration. Firstly, many studies show that increasing the network depth can enhance the information capacity of the learning system, thus boosting its non-linear mapping performance, which means it is possible to further improve the accuracy following this route. Secondly, popular concepts in machine learning, such as meta-learning, reinforcement learning, and semi-supervised learning, may lead to unexpected effects and discoveries in the model. The successful attempt at robust machine learning in Chapter 6 can be considered an example of this case. Thirdly, some fundamental elements of

networks, such as objective functions and optimization algorithms, are always worth investigating. The gradient optimization policy is recommended in this thesis, but it does not mean this is the only solution, and better performance may be possible by trying other optimization algorithms or objective functions. Finally, it is necessary to note that although the algorithms proposed in this thesis are general-purpose designs targeting most situations, they can also be customized according to the characteristics of application scenarios, thereby improving the performance in specific fields.

References

[1] L. Zadeh, 'Fuzzy sets', *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965 (cit. on pp. 2, 20).

[2] L. Zadeh, 'Fuzzy algorithms', *Information and Control*, vol. 12, no. 2, pp. 94–102, 1968 (cit. on pp. 2, 20).

[3] L. Zadeh, 'The concept of a linguistic variable and its application to approximate reasoning—i', *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975 (cit. on pp. 2, 20, 22, 108).

[4] A. Pollack, *Fuzzy Computer Theory: How to Mimic the Mind?* 1989 (cit. on p. 2).

[5] K. Åström, 'Directions in intelligent control', *IFAC Proceedings Volumes*, vol. 24, no. 1, pp. 1–9, 1991, IFAC Symposium on Intelligent Tuning and Adaptive Control, Singapore, 15-17 January 1991 (cit. on p. 2).

[6] D. Silver, A. Huang, C. J. Maddison *et al.*, 'Mastering the game of go with deep neural networks and tree search', English, *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016 (cit. on p. 2).

[7] D. Duvenaud, O. Rippel, R. P. Adams and Z. Ghahramani, 'Avoiding pathologies in very deep networks', eng, *Journal of machine learning research*, vol. 33, pp. 202–210, 2014 (cit. on p. 2).

[8] H. Takagi, 'Fusion technology of neural networks and fuzzy systems: A chronicled progression from the laboratory to our daily lives', *International Journal of Applied Mathematics and Computer Science*, vol. 10, pp. 647–673, 2000 (cit. on p. 2).

[9] N. Talpur, S. J. Abdulkadir, H. Alhussian *et al.*, 'A comprehensive review of deep neuro-fuzzy system architectures and their optimization methods', *Neural Computing and Applications*, vol. 34, no. 3, pp. 1837–1875, Feb. 2022 (cit. on pp. 2, 4).

[10] D. Wu, 'On the fundamental differences between interval type-2 and type-1 fuzzy logic controllers', *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 5, pp. 832–848, 2012 (cit. on pp. 4, 23).

[11] N. Karnik, J. Mendel and Q. Liang, 'Type-2 fuzzy logic systems', *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 643–658, 1999 (cit. on pp. 4, 22, 23, 108).

[12] D. Ramot, R. Milo, M. Friedman and A. Kandel, 'Complex fuzzy sets', *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 2, pp. 171–186, 2002 (cit. on pp. 4, 46, 108, 110, 174).

[13] D. Ramot, M. Friedman, G. Langholz and A. Kandel, 'Complex fuzzy logic', *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 4, pp. 450–461, 2003 (cit. on pp. 4, 46, 47, 108, 110).

[14] S. Dick, 'Toward complex fuzzy logic', *IEEE Transactions on Fuzzy Systems*, vol. 13, no. 3, pp. 405–414, 2005 (cit. on pp. 5, 45, 111).

[15] W. Szmielew, 'Elementary properties of abelian groups', eng, *Fundamenta mathematicae*, vol. 41, no. 2, pp. 203–271, 1955 (cit. on p. 5).

[16] O. Yazdanbakhsh and S. Dick, 'A systematic review of complex fuzzy sets and logic', *Fuzzy Sets and Systems*, vol. 338, pp. 1–22, 2018, Theme: Fuzzy Systems (cit. on pp. 5, 6, 45, 86, 108, 109, 150).

[17] S. Dick, R. R. Yager and O. Yazdanbakhsh, 'On pythagorean and complex fuzzy set operations', *IEEE Transactions on Fuzzy Systems*, vol. 24, no. 5, pp. 1009–1021, 2016 (cit. on pp. 6, 49).

[18] J. Bassey, L. Qian and X. Li, 'A survey of complex-valued neural networks', 2021 (cit. on pp. 6, 55, 56, 84).

[19] K. Subramanian, R. Savitha and S. Suresh, 'A complex-valued neuro-fuzzy inference system and its learning mechanism', *Neurocomputing*, vol. 123, pp. 110–120, 2014, Contains Special issue articles: Advances in Pattern Recognition Applications and Methods (cit. on pp. 6, 57).

[20] H. Ryusuke and M. Kazuyuki, 'Generation of fuzzy rules by a complex-valued neuro-fuzzy learning algorithm', jpn ; eng, *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, vol. 27, no. 1, pp. 533–548, 2015 (cit. on pp. 6, 57, 86).

[21] O. Yazdanbakhsh and S. Dick, 'Fancfis: Fast adaptive neuro-complex fuzzy inference system', *International Journal of Approximate Reasoning*, vol. 105, pp. 417–430, 2019 (cit. on pp. 7, 52).

[22] R. E. Bellman, *A Guided Tour*. Princeton: Princeton University Press, 1961 (cit. on pp. 7, 83).

[23] K. Meinke and J. V. Tucker, 'Universal algebra', in *Handbook of Logic in Computer Science (Vol. 1): Background: Mathematical Structures*. USA: Oxford University Press, Inc., 1993, pp. 189–368 (cit. on p. 15).

[24] G. Grätzer, 'General lattice theory: 1979 problem update', *algebra universalis*, vol. 11, no. 1, pp. 396–402, Dec. 1980 (cit. on pp. 16, 111).

[25] G. Schmidt, 'Orders and lattices', in *Relational Mathematics* (Encyclopedia of Mathematics and its Applications), Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2010, pp. 183–199 (cit. on p. 16).

[26] J. Pykacz, 'Łukasiewicz operations in fuzzy set and many-valued representations of quantum logics', *Foundations of Physics*, vol. 30, no. 9, pp. 1503–1524, Sep. 2000 (cit. on p. 17).

[27] L. Gamut, *Logic, Language, and Meaning: Introduction to logic* (Logic, Language, and Meaning). University of Chicago Press, 1991 (cit. on p. 18).

[28] H. B. Enderton, 'Chapter one - sentential logic', in *A Mathematical Introduction to Logic (Second Edition)*, H. B. Enderton, Ed., Second Edition, Boston: Academic Press, 2001, pp. 11–66 (cit. on p. 18).

[29] L. M. Augusto, *Many-Valued Logics. A Mathematical and Computational Introduction.Second Edition*. London: College Publications, 2017, pp. 1–9 (cit. on p. 19).

[30] G. Priest, 'Many-valued logics', in *An Introduction to Non-Classical Logic: From If to Is* (Cambridge Introductions to Philosophy), 2nd ed., Cambridge Introductions to Philosophy. Cambridge University Press, 2008, pp. 120–141 (cit. on p. 19).

[31] A. Rose, 'Systems of logic whose truth-values form lattices', *Mathematische Annalen*, vol. 123, no. 1, pp. 152–165, Dec. 1951 (cit. on p. 19).

[32] T. J. Ross, '5 logic and fuzzy systems - fuzzy logic', in *Fuzzy Logic with Engineering Applications.* Fourth Edition, Newark, UNITED STATES: John Wiley & Sons, Incorporated, 2017, pp. 122–132 (cit. on p. 19).

[33] L. A. Zadeh and R. R. Yager, 'Fuzzy sets and applications : Selected papers', 1987 (cit. on p. 20).

[34] A. Kandel, *Fuzzy Mathematical Techniques with Applications*. USA: Addison-Wesley Longman Publishing Co., Inc., 1986 (cit. on p. 20).

[35] B. Kosko and S. Isaka, 'Fuzzy logic', *Scientific American*, vol. 269, no. 1, pp. 76–81, Jul. 1993 (cit. on p. 20).

[36] T. J. Ross, '11 fuzzy control systems - examples of fuzzy control system design', in *Fuzzy Logic with Engineering Applications.* Fourth Edition, Newark, UNITED STATES: John Wiley & Sons, Incorporated, 2017, pp. 393–404 (cit. on p. 20).

[37] T. J. Ross, '8 fuzzy systems simulation - nonlinear simulation using fuzzy systems', in *Fuzzy Logic with Engineering Applications.* Fourth Edition, Newark, UNITED STATES: John Wiley & Sons, Incorporated, 2017, pp. 243–246 (cit. on p. 21).

[38] J. Carvajal, G. Chen and H. Ogmen, 'Fuzzy pid controller: Design, performance evaluation, and stability analysis', *Information Sciences*, vol. 123, no. 3, pp. 249–270, 2000 (cit. on p. 21).

[39] J. M. Mendel, 'Type-2 fuzzy sets', in *Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions, 2nd Edition*. Cham: Springer International Publishing, 2017, pp. 259–306 (cit. on pp. 22, 108).

[40] L. A. Zadeh, 'A note on z-numbers', *Information Sciences*, vol. 181, no. 14, pp. 2923–2932, 2011 (cit. on p. 24).

[41] L. A. Zadeh, 'Outline of a new approach to the analysis of complex systems and decision processes', *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, no. 1, pp. 28–44, 1973 (cit. on p. 24).

[42] Mamdani, 'Application of fuzzy logic to approximate reasoning using linguistic synthesis', *IEEE Transactions on Computers*, vol. C-26, no. 12, pp. 1182–1191, 1977 (cit. on p. 24).

[43] E. MAMDANI and S. ASSILIAN, 'An experiment in linguistic synthesis with a fuzzy logic controller', *International Journal of Human-Computer Studies*, vol. 51, no. 2, pp. 135–147, 1999 (cit. on p. 25).

[44] M. H. Eghbal Ahmadi, S. J. Royaee, S. Tayyebi and R. Bozorgmehry Boozarjomehry, 'A new insight into implementing mamdani fuzzy inference system for dynamic process modeling: Application on flash separator fuzzy dynamic modeling', *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103 485, 2020 (cit. on p. 25).

[45] M. Sugeno, 'An introductory survey of fuzzy control', *Information Sciences*, vol. 36, no. 1, pp. 59–83, 1985 (cit. on p. 26).

[46] M. Sugeno and G. Kang, 'Structure identification of fuzzy model', *Fuzzy Sets and Systems*, vol. 28, no. 1, pp. 15–33, 1988 (cit. on p. 26).

[47] Y. Tsukamoto, 'An approach to fuzzy reasoning method', *Readings in Fuzzy Sets for Intelligent Systems*, pp. 523–529, 1993 (cit. on p. 27).

[48] F. Rosenblatt, 'The perceptron: A probabilistic model for information storage and organization in the brain.', *Psychological review*, vol. 65, no. 6, pp. 386–408, Nov. 1958 (cit. on p. 29).

[49] L. Arnold, S. Rebecchi, S. Chevallier and H. Paugam-Moisy, 'An introduction to deep learning', vol. 1, Jan. 2011, pp. 477–488 (cit. on p. 29).

[50] F. Rosenblatt, 'The perceptron: A probabilistic model for information storage and organization in the brain', eng, *Psychological review*, vol. 65, no. 6, pp. 386–408, 1958 (cit. on p. 30).

[51] M. Minsky, S. Papert and L. Bottou, 'The Diameter-Limited Perceptron', in *Perceptrons: An Introduction to Computational Geometry, [2017 edition]*, The MIT Press, Sep. 2017 (cit. on p. 30).

[52] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning Representations by Back-propagating Errors', *Nature*, vol. 323, no. 6088, pp. 533–536, 1986 (cit. on pp. 30, 99, 101, 102, 131, 136, 169).

[53] J. Moody and C. J. Darken, 'Fast learning in networks of locally-tuned processing units', *Neural Computation*, vol. 1, no. 2, pp. 281–294, Jun. 1989 (cit. on pp. 30, 32, 101, 102, 136, 169).

[54] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks', *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017 (cit. on pp. 30, 169).

[55] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986 (cit. on p. 30).

[56] M. J. D. Powell, 'Radial basis functions for multivariable interpolation: A review', in *Algorithms for Approximation*. USA: Clarendon Press, 1987, pp. 143–167 (cit. on p. 32).

[57] D. Broomhead and D. Lowe, 'Multivariable functional interpolation and adaptive networks', *Complex Systems*, vol. 2, pp. 321–355, 1988 (cit. on p. 32).

[58] R. Kress, 'Interpolation', in *Numerical Analysis*. New York, NY: Springer New York, 1998, pp. 151–188 (cit. on p. 33).

[59] J.-S. Jang and C.-T. Sun, 'Functional equivalence between radial basis function networks and fuzzy inference systems', *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 156–159, 1993 (cit. on p. 35).

[60] K. B. Cho and B. H. Wang, 'Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction', *Fuzzy Sets and Systems*, vol. 83, no. 3, pp. 325–339, 1996 (cit. on p. 35).

[61] D. Rutkowska, 'Neuro-fuzzy architectures based on the mamdani approach', in *Neuro-Fuzzy Architectures and Hybrid Learning*. Heidelberg: Physica-Verlag HD, 2002, pp. 105–126 (cit. on p. 36).

[62] J. Buckley, 'Sugeno type controllers are universal controllers', *Fuzzy Sets and Systems*, vol. 53, no. 3, pp. 299–303, 1993 (cit. on p. 39).

[63] 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 (cit. on pp. 40, 43, 89, 108).

[64] A. (Kaufmann, *Introduction to fuzzy arithmetic : theory and applications / Arnold Kaufmann, Madan M. Gupta.* (Van Nostrand Reinhold electrical/computer science and engineering series), eng. New York, N.Y: Van Nostrand Reinhold Co., 1985 (cit. on p. 46).

[65] D. Moses, O. Degani, H.-N. Teodorescu, M. Friedman and A. Kandel, 'Linguistic coordinate transformations for complex fuzzy sets', in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No.99CH36315)*, vol. 3, 1999, 1340–1345 vol.3 (cit. on p. 46).

[66] H. Nguyen, A. Kandel and V. Kreinovich, 'Complex fuzzy sets: Towards new foundations', in *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No.00CH37063)*, vol. 2, 2000, 1045–1048 vol.2 (cit. on pp. 46, 174).

[67] D. E. Tamir, L. Jin and A. Kandel, 'A new interpretation of complex membership grade', *International Journal of Intelligent Systems*, vol. 26, no. 4, pp. 285–312, 2011 (cit. on p. 47).

[68] D. E. Tamir and A. Kandel, 'Axiomatic theory of complex fuzzy logic and complex fuzzy classes', English, *International Journal of Computers, Communications and Control*, vol. 6, no. 3, pp. 562–576, Sep. 2011 (cit. on p. 47).

[69] A. S. Alkouri and A. R. Salleh, 'Complex intuitionistic fuzzy sets', eng, vol. 1482, pp. 464–470, 2012 (cit. on p. 48).

[70] K. T. Atanassov, 'Intuitionistic fuzzy sets', *Fuzzy Sets and Systems*, vol. 20, no. 1, pp. 87–96, 1986 (cit. on p. 48).

[71] D. Rani and H. Garg, 'Complex intuitionistic fuzzy preference relations and their applications in individual and group decision-making problems', *International Journal of Intelligent Systems*, vol. 36, no. 4, pp. 1800–1830, 2021 (cit. on p. 49).

[72] R. T. Ngan, L. H. Son, M. Ali *et al.*, 'Representing complex intuitionistic fuzzy set by quaternion numbers and applications to decision making', *Applied Soft Computing*, vol. 87, p. 105 961, 2020 (cit. on p. 49).

[73] N. Yaqoob, M. Gulistan, S. Kadry and H. A. Wahab, 'Complex intuitionistic fuzzy graphs with application in cellular network provider companies', *Mathematics*, vol. 7, no. 1, 2019 (cit. on p. 49).

[74] R. R. Yager and A. M. Abbasov, 'Pythagorean membership grades, complex numbers, and decision making', *International Journal of Intelligent Systems*, vol. 28, no. 5, pp. 436–452, 2013 (cit. on p. 49).

[75] R. R. Yager, 'Pythagorean membership grades in multicriteria decision making', *IEEE Transactions on Fuzzy Systems*, vol. 22, no. 4, pp. 958–965, 2014 (cit. on p. 49).

[76] T. Kumar and R. K. Bajaj, 'On complex intuitionistic fuzzy soft sets with distance measures and entropies', English, *Journal of Mathematics*, vol. 2014, 2014 (cit. on p. 50).

[77] P. K. Maji, 'More on intuitionistic fuzzy soft sets', eng, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, ser. Lecture Notes in Computer Science, vol. 5908, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 231–240 (cit. on p. 50).

[78] P. Thirunavukaras and S. Rengarajulu, 'Parameterized soft complex fuzzy sets', *Journal of Progressive Research in Mathematics*, vol. 4, no. 2, pp. 303–308, Jun. 2015 (cit. on p. 50).

[79] C. Li and C.-H. Tu, 'Complex neural fuzzy system and its application on multiclass prediction — a novel approach using complex fuzzy sets, iim and multiswarm learning', *Applied Soft Computing*, vol. 84, p. 105 735, 2019 (cit. on p. 50).

[80] C.-H. Tu and C. Li, 'Multitarget prediction—a new approach using sphere complex fuzzy sets', *Engineering Applications of Artificial Intelligence*, vol. 79, pp. 45–57, 2019 (cit. on p. 50).

[81] A. Mumtaz and F. Smarandache, 'Complex neutrosophic set', English, *Neural Computing & Applications*, vol. 28, no. 7, pp. 1817–1834, 2017 (cit. on p. 50).

[82] Z. Ali and T. Mahmood, 'Complex neutrosophic generalised dice similarity measures and their application to decision making', eng, *CAAI Transactions on Intelligence Technology*, vol. 5, no. 2, pp. 78–87, 2020 (cit. on p. 50).

[83] P. Liu, T. Mahmood and Z. Ali, 'Complex q-rung orthopair fuzzy aggregation operators and their applications in multi-attribute group decision making', English, *Information*, vol. 11, no. 1, p. 5, 2020 (cit. on p. 50).

[84] P. K. Singh, 'Bipolar δ-equal complex fuzzy concept lattice with its application', English, *Neural Computing & Applications*, vol. 32, no. 7, pp. 2405–2422, Apr. 2020 (cit. on p. 50).

[85] S. Greenfield, F. Chiclana and S. Dick, 'Interval-valued complex fuzzy logic', in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 2014–2019 (cit. on p. 50).

[86] J. Y. Man, Z. Chen and S. Dick, 'Towards inductive learning of complex fuzzy inference systems', in *NAFIPS 2007 - 2007 Annual Meeting of the North American Fuzzy Information Processing Society*, 2007, pp. 415–420 (cit. on p. 51).

[87] Z. Chen, S. Aghakhani, J. Man and S. Dick, 'Ancfis: A neurofuzzy architecture employing complex fuzzy sets', *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 2, pp. 305–322, 2011 (cit. on p. 51).

[88] O. Yazdanbaksh, A. Krahn and S. Dick, 'Predicting solar power output using complex fuzzy logic', in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, 2013, pp. 1243–1248 (cit. on p. 52).

[89] O. Yazdanbakhsh and S. Dick, 'Time-series forecasting via complex fuzzy logic', in *Frontiers of Higher Order Fuzzy Sets*, A. Sadeghian and H. Tahayori, Eds. New York, NY: Springer New York, 2015, pp. 147–165 (cit. on p. 52).

[90] O. Yazdanbakhsh and S. Dick, 'Ancfis-elm: A machine learning algorithm based on complex fuzzy sets', in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 2007–2013 (cit. on p. 52).

[91] O. Yazdanbakhsh and S. Dick, 'Forecasting of multivariate time series via complex fuzzy logic', *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 8, pp. 2160–2171, 2017 (cit. on p. 52).

[92] M. Yeganejou and S. Dick, 'Inductive learning of classifiers via complex fuzzy sets and logic', in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2017, pp. 1–6 (cit. on p. 52).

[93] C. Li and T.-W. Chiang, 'Complex neuro-fuzzy self-learning approach to function approximation', N. T. Nguyen, M. T. Le and J. Świątek, Eds., pp. 289–299, 2010 (cit. on p. 53).

[94] C. Li and T.-W. Chiang, 'Function approximation with complex neuro-fuzzy system using complex fuzzy sets – a new approach', *New Generation Computing*, vol. 29, no. 3, p. 261, Aug. 2011 (cit. on pp. 53, 109, 129, 130).

[95] C. Li, T.-W. Chiang and L.-C. Yeh, 'A novel self-organizing complex neuro-fuzzy approach to the problem of time series forecasting', *Neurocomputing*, vol. 99, pp. 467–476, 2013 (cit. on pp. 54, 99, 130, 131).

[96] C. Li and F. Chan, 'Complex-fuzzy adaptive image restoration – an artificial-bee-colony-based learning approach', N. T. Nguyen, C.-G. Kim and A. Janiak, Eds., pp. 90–99, 2011 (cit. on p. 54).

[97] C. Li, T. Wu and F.-T. Chan, 'Self-learning complex neuro-fuzzy system with complex fuzzy sets and its application to adaptive image noise canceling', *Neurocomputing*, vol. 94, pp. 121–139, 2012 (cit. on p. 54).

[98] C. Li and F.-T. Chan, 'Knowledge discovery by an intelligent approach using complex fuzzy sets', in *Intelligent Information and Database Systems*, J.-S. Pan, S.-M. Chen and N. T. Nguyen, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 320–329 (cit. on p. 54).

[99] C. Li and T.-W. Chiang, 'Complex neurofuzzy arima forecasting—a new approach using complex fuzzy sets', *IEEE Transactions on Fuzzy Systems*, vol. 21, no. 3, pp. 567–584, 2013 (cit. on p. 54).

[100] C. Li and C.-H. Tu, 'Complex neural fuzzy system and its application on multi-class prediction — a novel approach using complex fuzzy sets, iim and multi-swarm learning', *Applied Soft Computing*, vol. 84, p. 105 735, 2019 (cit. on p. 54).

[101] R. Shoorangiz and M. H. Marhaban, 'Complex neuro-fuzzy system for function approximation', English, *International Journal of Applied Electronics in Physics and Robotics*, vol. 1, no. 2, pp. 5–9, 2013 (cit. on p. 55).

[102] S. K. Solanki, 'Sunspots: An overview', eng, *The Astronomy and astrophysics review*, vol. 11, no. 2-3, pp. 153–286, 2003 (cit. on p. 62).

[103] S. V. Berdyugina, 'Starspots: A key to the stellar dynamo', eng, *Living reviews in solar physics*, vol. 2, no. 1, pp. 1–62, 2005 (cit. on p. 62).

[104] Xu Zhen-tao, 'The hexagram "feng" in "the book of changes" as the earliest written record of sunspot', *Chinese Astronomy*, vol. 4, no. 4, p. 406, 1980 (cit. on p. 64).

[105] G. Galilei, E. Scheiner Christoph andReeves and A. Van Helden, '8 Sunspots before the telescope', in *On Sunspots*, University of Chicago Press, Oct. 2010 (cit. on p. 64).

[106] S. K. Solanki, I. G. Usoskin, B. Kromer, M. Schussler and J. Beer, 'Unusual activity of the sun during recent decades compared to the previous 11,000 years', English, *Nature*, vol. 431, no. 7012, pp. 1084–7, Oct. 2004 (cit. on p. 64).

[107] *Sunspot Index and Long-term Solar Observations (SILSO)*. World Data Center (cit. on pp. 64–66, 98, 130).

[108] D. Brandt and J. Warner, *Metallurgy Fundamentals: Ferrous and Nonferrous*. Goodheart-Willcox Company, Incorporated, 2019 (cit. on p. 67).

[109] N. Saba, M. Jawaid and M. Sultan, '1 - an overview of mechanical and physical testing of composite materials', Woodhead Publishing Series in Composites Science and Engineering, M. Jawaid, M. Thariq and N. Saba, Eds., pp. 1–12, 2019 (cit. on p. 67).

[110] D. R. Askeland, 'Mechanical testing and properties', pp. 140–187, 1996 (cit. on p. 67).

[111] M. Stewart, '5 - mechanical design of pressure vessels', in *Surface Production Operations*, M. Stewart, Ed., Boston: Gulf Professional Publishing, 2021, pp. 117–196 (cit. on p. 68).

[112] S. Wiederhorn, R. Fields, S. Low *et al.*, 'Mechanical properties', in *Springer Handbook of Materials Measurement Methods*, H. Czichos, T. Saito and L. Smith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 303–304 (cit. on p. 71).

[113] H. A. Barnes, 'The yield stress—a review or "$\pi\alpha\nu\tau\alpha\ \rho\varepsilon$" — everything flows?', *Journal of Non-Newtonian Fluid Mechanics*, vol. 81, no. 1, pp. 133–178, 1999 (cit. on p. 71).

[114] W. F. Hosford, 'Mechanical testing', in *Mechanical Behavior of Materials*, 2nd ed. Cambridge University Press, 2009, pp. 36–64 (cit. on p. 71).

[115] J. T. Black and R. A. Kohser, '2.6 fracture toughness and the fracture mechanics approach', eng, in *DeGarmo's materials and processes in manufacturing*, Thirteenth edition. Hoboken: Wiley, 2019, pp. 164–169 (cit. on p. 71).

[116] W. D. Callister and D. G. Rethwisch, 'Atomic bonding in solids, atomic structure and interatomic bonding', eng, in *Materials science and engineering an introduction*, 10th edition. Hoboken, NJ: John Wiley & Sons, Inc., 2018, pp. 30–47 (cit. on p. 72).

[117] W. D. Callister and D. G. Rethwisch, 'Applications and processing of metal alloys', eng, in *Materials science and engineering an introduction*, 10th edition. Hoboken, NJ: John Wiley & Sons, Inc., 2018, pp. 349–372 (cit. on p. 72).

[118] R. J. Hyndman and A. B. Koehler, 'Another look at measures of forecast accuracy', *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006 (cit. on p. 76).

[119] C. J. Willmott and K. Matsuura, 'Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance', *Climate Research*, vol. 30, no. 1, pp. 79–82, 2005 (cit. on p. 76).

[120] S. Makridakis, 'Accuracy measures: Theoretical and practical concerns', *International Journal of Forecasting*, vol. 9, no. 4, pp. 527–529, 1993 (cit. on p. 77).

[121] B. E. Flores, 'A pragmatic view of accuracy measurement in forecasting', *Omega*, vol. 14, no. 2, pp. 93–98, 1986 (cit. on p. 77).

[122] J. M. Bland and D. G. Altman, 'Measurement error', *BMJ: British Medical Journal*, vol. 313, no. 7059, pp. 744–744, 1996 (cit. on p. 78).

[123] D. A. Freedman, 'Observational studies and experiments', in *Statistical Models: Theory and Practice*, 2nd ed. Cambridge University Press, 2009, pp. 1–17 (cit. on p. 83).

[124] S. Sardi, R. Vardi, Y. Meir *et al.*, 'Brain experiments imply adaptation mechanisms which outperform common ai learning algorithms', *Scientific Reports*, vol. 10, no. 1, p. 6923, Apr. 2020 (cit. on p. 83).

[125] A. Zimek, E. Schubert and H.-P. Kriegel, 'A survey on unsupervised outlier detection in high-dimensional numerical data', *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012 (cit. on p. 83).

[126] F. Doshi-Velez and B. Kim, 'Towards a rigorous science of interpretable machine learning', eng, 2017 (cit. on p. 83).

[127] L. Y. Pratt, 'Discriminability-based transfer between neural networks', in *Proceedings of the 5th International Conference on Neural Information Processing Systems*, ser. NIPS'92, Denver, Colorado: Morgan Kaufmann Publishers Inc., 1992, pp. 204–211 (cit. on p. 83).

[128] L. Pratt and B. Jennings, 'A survey of transfer between connectionist networks', *Connection Science*, vol. 8, no. 2, pp. 163–184, 1996 (cit. on p. 83).

[129] S. J. Pan and Q. Yang, 'A survey on transfer learning', *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010 (cit. on p. 83).

[130] J. Schmidhuber, 'Evolutionary principles in self-referential learning. on learning now to learn: The meta-meta-meta...-hook', Diploma Thesis, Technische Universitat Munchen, Germany, May 1987 (cit. on p. 83).

[131] S. Bengio, Y. Bengio and J. Cloutier, 'On the search for new learning rules for anns', *Neural Processing Letters*, vol. 2, no. 4, pp. 26–30, Jul. 1995 (cit. on p. 83).

[132] T. Hospedales, A. Antoniou, P. Micaelli and A. Storkey, 'Meta-learning in neural networks: A survey', eng, *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 9, pp. 5149–5169, 2022 (cit. on p. 83).

[133] V. Mnih, K. Kavukcuoglu, D. Silver *et al.*, 'Playing atari with deep reinforcement learning', eng, 2013 (cit. on p. 83).

[134] A. E. Teschendorff, 'Avoiding common pitfalls in machine learning omic data science', *Nature Materials*, vol. 18, no. 5, pp. 422–427, May 2019 (cit. on p. 84).

[135] L.-X. Wang, 'Fast training algorithms for deep convolutional fuzzy systems with application to stock index prediction', *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 7, pp. 1301–1314, 2020 (cit. on pp. 84, 89, 101, 102).

[136] L.-X. Wang and J. Mendel, 'Generating fuzzy rules by learning from examples', *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992 (cit. on pp. 84, 87).

[137] L.-X. Wang, 'The wm method completed: A flexible fuzzy system approach to data mining', *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 768–782, 2003 (cit. on pp. 84, 87, 99).

[138] G. V. S. Raju, J. Zhou and R. A. Kisner, 'Hierarchical fuzzy control', *International Journal of Control*, vol. 54, no. 5, pp. 1201–1216, 1991 (cit. on pp. 85, 88, 89).

[139] L. Van Der Maaten and G. Hinton, 'Visualizing data using t-sne', eng, *Journal of machine learning research*, vol. 9, pp. 2579–2625, 2008 (cit. on p. 85).

[140] A. Hirose, 'Applications of complex-valued neural networks to coherent optical computing using phase-sensitive detection scheme', *Information Sciences - Applications*, vol. 2, no. 2, pp. 103–117, 1994 (cit. on p. 85).

[141] A. Hirose and S. Yoshida, 'Comparison of complex- and real-valued feedforward neural networks in their generalization ability', in *Neural Information Processing*, B.-L. Lu, L. Zhang and J. Kwok, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 526–531 (cit. on p. 85).

[142] A. Hirose, 'Continuous complex-valued back-propagation learning', English, *Electronics Letters*, vol. 28, 1854–1855(1), 20 Sep. 1992 (cit. on p. 85).

[143] A. Hirose and S. Yoshida, 'Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence', *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 4, pp. 541–551, 2012 (cit. on p. 86).

[144] Li-Xin Wang, 'Universal approximation by hierarchical fuzzy systems', *Fuzzy Sets and Systems*, vol. 93, no. 2, pp. 223–230, 1998 (cit. on p. 89).

[145] L.-X. Wang, 'Analysis and design of hierarchical fuzzy systems', *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp. 617–624, 1999 (cit. on p. 89).

[146] J.-M. Alonso and Y. Chen, 'Receptive field', eng, *Scholarpedia journal*, vol. 4, no. 1, p. 5393, 2009 (cit. on p. 92).

[147] H. Abdi and L. J. Williams, 'Principal component analysis', *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010 (cit. on p. 94).

[148] G. E. Hinton and S. Roweis, 'Stochastic neighbor embedding', in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun and K. Obermayer, Eds., vol. 15, MIT Press, 2002 (cit. on p. 94).

[149] L. Van Der Maaten and G. Hinton, 'Visualizing data using t-sne', eng, *Journal of machine learning research*, vol. 9, pp. 2579–2625, 2008 (cit. on p. 94).

[150] R. Hyndman and G. Athanasopoulos, '8.9 seasonal arima models', English, in *Forecasting: Principles and Practice*, 2nd. Australia: OTexts, 2018, pp. 252–261 (cit. on pp. 99, 130, 131).

[151] S. Hochreiter and J. Schmidhuber, 'Long short-term memory', *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997 (cit. on pp. 99, 101, 102, 131, 136).

[152] M. Ardalani-Farsa and S. Zolfaghari, 'Chaotic time series prediction with residual analysis method using hybrid elman–narx neural networks', *Neurocomputing*, vol. 73, no. 13, pp. 2540–2553, 2010, Pattern Recognition in Bioinformatics Advances in Neural Control (cit. on pp. 99, 130, 131).

[153] D. Specht, 'A general regression neural network', *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991 (cit. on pp. 101, 102, 136).

[154] G. E. Hinton, S. Osindero and Y.-W. Teh, 'A fast learning algorithm for deep belief nets', *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006 (cit. on pp. 101, 102, 136).

[155] J. Mendel, H. Hagras, W.-W. Tan, W. W. Melek and H. Ying, 'Introduction to type-2 fuzzy sets', in *Introduction To Type-2 Fuzzy Logic Control: Theory and Applications*. 2014, pp. 32–79 (cit. on p. 108).

[156] K. Shihabudheen and G. Pillai, 'Recent advances in neuro-fuzzy system: A survey', *Knowledge-Based Systems*, vol. 152, pp. 136–162, 2018 (cit. on p. 108).

[157] L. Ljung, '11 recursive estimation methods', in *System identification : theory for the user*, 2nd ed. Upper Saddle River, N.J. : London: Prentice Hall PTR ; Prentice-Hall International, 1999, pp. 361–398 (cit. on p. 108).

[158] D. Simon, 'Training fuzzy systems with the extended kalman filter', *Fuzzy Sets and Systems*, vol. 132, no. 2, pp. 189–199, 2002 (cit. on p. 108).

[159] Z.-L. Sun, K.-F. Au and T.-M. Choi, 'A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 5, pp. 1321–1331, 2007 (cit. on p. 108).

[160] Y. Chen and J. Wang, 'Support vector learning for fuzzy rule-based classification systems', *IEEE Transactions on Fuzzy Systems*, vol. 11, no. 6, pp. 716–728, 2003 (cit. on p. 108).

[161] M. Shoorehdeli, M. Teshnehlab and A. Sedigh, 'A novel training algorithm in anfis structure', in *2006 American Control Conference*, 2006, p. 6 (cit. on p. 108).

[162] E. G. Carrano, R. H. C. Takahashi, W. M. Caminhas and O. M. Neto, 'A genetic algorithm for multiobjective training of anfis fuzzy networks', in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 3259–3265 (cit. on p. 108).

[163] D. Karaboga and E. Kaya, 'Training anfis using artificial bee colony algorithm', in *2013 IEEE INISTA*, 2013, pp. 1–5 (cit. on p. 108).

[164] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016 (cit. on p. 108).

[165] M. Steinbach, G. Karypis and V. Kumar, 'A comparison of document clustering techniques', *Proceedings of the International KDD Workshop on Text Mining*, Jun. 2000 (cit. on pp. 109, 112).

[166] J. Ma and D. Yarats, 'Quasi-hyperbolic momentum and adam for deep learning', 2018 (cit. on pp. 109, 113).

[167] G. W. Dombi, P. Nandi, J. M. Saxe, A. M. Ledgerwood and C. E. Lucas, 'Prediction of rib fracture injury outcome by an artificial neural network', *Journal of Trauma and Acute Care Surgery*, vol. 39, no. 5, 1995 (cit. on pp. 109, 125).

[168] S. Lloyd, 'Least squares quantization in pcm', *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982 (cit. on p. 111).

[169] T. Kanungo, D. Mount, N. Netanyahu *et al.*, 'An efficient k-means clustering algorithm: Analysis and implementation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, 2002 (cit. on p. 111).

[170] D. Arthur and S. Vassilvitskii, 'K-means++: The advantages of careful seeding', in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07, New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035 (cit. on p. 111).

[171] A. Krizhevsky, I. Sutskever and G. Hinton, 'Imagenet classification with deep convolutional neural networks', eng, *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017 (cit. on p. 112).

[172] D. P. Kingma and J. Ba, 'Adam: A method for stochastic optimization', 2014 (cit. on p. 112).

[173] B. Polyak, 'Some methods of speeding up the convergence of iteration methods', *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964 (cit. on p. 112).

[174] Y. Nesterov, 'A method for solving the convex programming problem with convergence rate $O(1/k^2)$', *Proceedings of the USSR Academy of Sciences*, vol. 269, pp. 543–547, 1983 (cit. on p. 112).

[175] B. Hu and L. Lessard, 'Control interpretations for first-order optimization methods', in *2017 American Control Conference (ACC)*, 2017, pp. 3114–3119 (cit. on p. 113).

[176] I. Goodfellow, Y. Bengio and A. Courville, '5 machine learning basics', in *Deep Learning*. Cambridge, Massachusetts: MIT Press, 2016, pp. 129–133 (cit. on p. 119).

[177] K. P. F.R.S., 'Liii. on lines and planes of closest fit to systems of points in space', *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901 (cit. on p. 125).

[178] 'Principal component analysis and factor analysis', in *Principal Component Analysis*. New York, NY: Springer New York, 2002, pp. 150–166 (cit. on p. 125).

[179] I. T. Jolliffe and J. Cadima, 'Principal component analysis: A review and recent developments', eng, *Philosophical transactions of the Royal Society of London. Series A: Mathematical, physical, and engineering sciences*, vol. 374, no. 2065, pp. 20 150 202–20 150 202, 2016 (cit. on p. 125).

[180] B. Schölkopf, A. Smola and K.-R. Müller, 'Nonlinear component analysis as a kernel eigenvalue problem', *Neural Computation*, vol. 10, no. 5, pp. 1299–1319, Jul. 1998 (cit. on p. 125).

[181] W. H. Kruskal and W. A. Wallis, 'Use of ranks in one-criterion variance analysis', *Journal of the American Statistical Association*, vol. 47, no. 260, pp. 583–621, 1952 (cit. on p. 129).

[182] G. Panoutsos and M. Mahfouf, 'A neural-fuzzy modelling framework based on granular computing: Concepts and applications', *Fuzzy Sets and Systems*, vol. 161, no. 21, pp. 2808–2830, 2010, Theme: Fuzzy Control and Fuzzy Systems (cit. on pp. 134, 136).

[183] R. Muscat and M. Mahfouf, 'Predicting charpy impact energy for heat-treated steel using a quantum-membership-function-based fuzzy model', *IFAC-PapersOnLine*, vol. 49, no. 20, pp. 138–142, 2016, 17th IFAC Symposium on Control, Optimization and Automation in Mining, Mineral and Metal Processing MMM 2016 (cit. on pp. 134, 136).

[184] Q. Liang and J. Mendel, 'An introduction to type-2 tsk fuzzy logic systems', in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No.99CH36315)*, vol. 3, 1999, 1534–1539 vol.3 (cit. on pp. 134, 136).

[185] C. Jun, 'Biologically inspired optimisation algorithms for transparent knowledge extraction allied to engineering materials processing', 2010 (cit. on pp. 139, 140).

[186] S. Wang and M. Mahfouf, 'Multi-objective optimisation for fuzzy modelling using interval type-2 fuzzy sets', in *2012 IEEE International Conference on Fuzzy Systems*, 2012, pp. 1–8 (cit. on pp. 139, 140).

[187] D. A. Freedman, 'Multiple regression: Special topics', in *Statistical Models: Theory and Practice*, 2nd ed. Cambridge University Press, 2009, pp. 61–80 (cit. on pp. 147, 155).

[188] V. Chandola, A. Banerjee and V. Kumar, 'Anomaly detection: A survey', eng, *ACM computing surveys*, vol. 41, no. 3, pp. 1–58, 2009 (cit. on p. 147).

[189] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi and P. Frossard, 'Universal adversarial perturbations', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 86–94 (cit. on p. 147).

[190] N. Papernot, P. McDaniel, I. Goodfellow *et al.*, 'Practical black-box attacks against machine learning', in *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, Apr. 2017, pp. 506–519 (cit. on p. 147).

[191] M. Shafique, M. Naseer, T. Theocharides *et al.*, 'Robust machine learning systems: Challenges,current trends, perspectives, and the road ahead', *IEEE Design & Test*, vol. 37, no. 2, pp. 30–57, 2020 (cit. on p. 148).

[192] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola and V. Vapnik, 'Support vector regression machines', in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan and T. Petsche, Eds., vol. 9, MIT Press, 1996 (cit. on p. 148).

[193] A. J. Smola and B. Schölkopf, 'A tutorial on support vector regression', *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, Aug. 2004 (cit. on pp. 148, 169).

[194] P. J. Huber, 'Robust estimation of a location parameter', *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964 (cit. on pp. 148, 163).

[195] Q. Wang, Y. Ma, K. Zhao and Y. Tian, 'A comprehensive survey of loss functions in machine learning', *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, Apr. 2022 (cit. on p. 148).

[196] J. T. Barron, 'A general and adaptive robust loss function', in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4326–4334 (cit. on p. 148).

[197] B. Han, Q. Yao, T. Liu *et al.*, 'A survey of label-noise representation learning: Past, present and future', 2020 (cit. on p. 149).

[198] J. Goldberger and E. Ben-Reuven, 'Training deep neural-networks using a noise adaptation layer', in *International Conference on Learning Representations*, 2017 (cit. on p. 149).

[199] G. Patrini, A. Rozza, A. K. Menon, R. Nock and L. Qu, 'Making deep neural networks robust to label noise: A loss correction approach', in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2233–2241 (cit. on p. 149).

[200] Y. Wang, A. Kucukelbir and D. M. Blei, 'Robust probabilistic modeling with bayesian data reweighting', in *Proceedings of the 34th International Conference on Machine Learning*, ser. ICML'17, vol. 70, Sydney, NSW, Australia: JMLR.org, 2017, pp. 3646–3655 (cit. on p. 149).

[201] M. Ren, W. Zeng, B. Yang and R. Urtasun, 'Learning to reweight examples for robust deep learning', in *International Conference on Machine Learning (ICML)*, 2018 (cit. on p. 149).

[202] L. Jiang, Z. Zhou, T. Leung, L.-J. Li and L. Fei-Fei, 'Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels', in *International Conference on Machine Learning (ICML)*, 2018 (cit. on p. 149).

[203] B. Han, Q. Yao, X. Yu *et al.*, 'Co-teaching: Robust training of deep neural networks with extremely noisy labels', in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, ser. NIPS'18, Montréal, Canada: Curran Associates Inc., 2018, pp. 8536–8546 (cit. on p. 149).

[204] Z. Wang, G. Hu and Q. Hu, 'Training noise-robust deep neural networks via meta-learning', in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4523–4532 (cit. on p. 149).

[205] Y. Carmon, A. Raghunathan, L. Schmidt, P. Liang and J. C. Duchi, 'Unlabeled data improves adversarial robustness', in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019, vol. 32 (cit. on p. 149).

[206] J. C. Bezdek, R. Ehrlich and W. Full, 'FCM: The fuzzy c-means clustering algorithm', *Computers & Geosciences*, vol. 10, no. 2, pp. 191–203, 1984 (cit. on pp. 150, 152).

[207] D. de Menezes, D. Prata, A. Secchi and J. Pinto, 'A review on robust m-estimators for regression analysis', *Computers & Chemical Engineering*, vol. 147, p. 107 254, 2021 (cit. on pp. 150, 157).

[208] S. Ruder, 'An overview of gradient descent optimization algorithms', 2016 (cit. on p. 153).

[209] J. Chen, C. Wolfe, Z. Li and A. Kyrillidis, 'Demon: Improved neural network training with momentum decay', in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3958–3962 (cit. on p. 154).

[210] P. J. Huber, 'Robust statistics', in *International Encyclopedia of Statistical Science*, M. Lovric, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1248–1251 (cit. on p. 155).

[211] X. Dang, H. Peng, X. Wang and H. Zhang, 'Theil-sen estimators in a multiple linear regression model', (cit. on p. 155).

[212] E. Kreyszig, H. Kreyszig and E. J. Norminton, '24.1 Data representation. average. spread', eng, in *Advanced engineering mathematics, 10th edition*. Hoboken, N.J.: Wiley, 2011, pp. 1011–1015 (cit. on p. 164).

[213] M. Jaderberg, V. Dalibard, S. Osindero *et al.*, 'Population based training of neural networks', 2017 (cit. on p. 169).

List of Abbreviations

ACNFIS	adaptive complex neuro-fuzzy inference system
AI	artificial intelligence
ANCFIS	adaptive neuro complex fuzzy inference system
ANFIS	adaptive neuro-fuzzy inference system
ANN	artificial neural network
ARIMA	autoregressive integrated moving average
BP	backpropagation
CAIFS	complex Atanassov's intuitionistic fuzzy set
CFS&T	Complex fuzzy sets and logic
CFS	complex fuzzy set
CFT	complex fuzzy theory
CIFSS	complex intuitionistic fuzzy soft set
CNFIS	complex-valued neuro-fuzzy inference system
CNFS	complex neuro-fuzzy self-learning system
CRI	compositional rule of inference
CVNF	complex-valued neuro-fuzzy system
CVNN	complex-valued neural network

CVWM	complex-valued Wang-Mendel
DBN	deep belief network
DCFS	deep convolutional fuzzy system
DCVSF	deep complex-valued single-iteration fuzzy system
DEMON	decaying momentum hyperparameter rule
DR	dimensionality reduction
EA-SVR	support vector regression using evolutionary algorithms
Elman-NARX	Elman-Narx neural network
ELM	extreme learning machine
ENCFIS	exclusionary neural complex fuzzy inference system
FLS	fuzzy logic system
FMP	fuzzy modus ponens
FOU	footprint of uncertaint
GrC-NF	granular computing and neural-fuzzy modelling
GRNN	generalized regression neural network
IFSS	intuitionistic fuzzy soft set
IFS	intuitionistic fuzzy set
IMOFM	immune inspired multi-objective fuzzy modeling
IOT	internet of thing
IT2Sugeno	interval type-2 sugeno fuzzy inference system
LNRL	label-noise representation learning
LS/OLS	ordinary least square
LSTM	long short-term memory

MAE	mean absolute error
MAPE	mean absolute percentage error
MIV	mean impact value
MOIT2FM	multi-objective interval type-2 fuzzy modelling
MP	modus ponens
MSE	mean square error
MT	modus tollens
NFIS	neuro-fuzzy inference system
NFS	neuro-fuzzy system
PCA	principal component analysis
PFS	Pythagorean fuzzy set
PID	proportional-integral-derivative
Pseudo-MSE	pseudo mean square error
PSO	particle swarm optimization
Q-ANFIS	quantum adaptive neuro complex fuzzy inference system
QHM	quasi-hyperbolic momentum
RACFIS	rapid adaptive complex neuro-fuzzy inference system
RBF	radial basis function
RLSE	recursive least square estimation
RLS	recursive least square
RMSE	root-mean-square error
SARIMA	seasonal autoregressive integrated moving average
SD/STD	standard deviation

SILSO	sunspot index and long-term solar observation
SMAPE	symmetric mean absolute percentage error
SVM	support vector machine
SVR	support vector regression
t-conorm	triangular conorm
t-norm	triangular norm
t-SNE	t-distributed stochastic neighbor embedding
TSK	Takagi-Sugeno-Kang
UTS	ultimate tensile strength
WM	Wang-Mendel method