# C-loss based Higher-order Fuzzy Inference Systems for Identifying DNA N4-methylcytosine Sites

Yijie Ding, Prayag Tiwari, Quan Zou, Fei Guo, and Hari Mohan Pandey

***Abstract*—DNA methylation is an epigenetic marker, that plays an important role in the biological processes of regulating gene expression, maintaining chromatin structure, imprinting genes, inactivating X chromosomes, and developing embryos. The traditional detection method is time-consuming. Currently, researchers have used effective computational methods to improve the efficiency of methylation detection. This study proposes a fuzzy model with correntropy induced loss (C-loss) function to identify DNA N4-methylcytosine (4mC) sites. To improve the robustness and performance of the model, we use kernel method and the C-loss function to build a higher-order fuzzy inference systems (HFIS). To test performance, our model is implemented on six 4mC and eight UCI data sets. The experimental results show that our model achieves better prediction performance.**

***Index Terms*—DNA N4-methylcytosine, 4mC, Fuzzy model, Kernel method, Sequence classification.**

## I. Introduction

DNA N4-methylcytosine (4mC) is a form of DNA chemical modification that can change genetic performance without changing the DNA sequence. A large number of studies have shown that DNA methylation can control gene expression by changing the chromatin structure, DNA conformation, DNA stability, and DNA interactions with proteins. In the development of malignant tumors, the state of methylation is not static. The degree of hypomethylation of the whole genome in tumor cells is closely related to disease progression, tumor size, and malignancy. DNA methylation detection is effective for assessing tumor malignancy. The degree of judgment is of great significance. However, the traditional detection method is a wet experiment, which is time-consuming and labor-intensive. Therefore, it is necessary to propose an effective computational method for 4mC site identification.

This work is supported in part by the National Natural Science Foundation of China (NSFC 62172076, 62172296 and 61902271), the Natural Science Research of Jiangsu Higher Education Institutions of China (19KJB520014) and the Special Science Foundation of Quzhou (2021D004). (*Corresponding Author: Quan Zou, Fei Guo, Hari Mohan Pandey, and Prayag Tiwari*)

Y. Ding is with Yangtze Delta Region Institute (Quzhou), University of Electronic Science and Technology of China, Quzhou, 324000, P.R.China; E-mail: wuxi_dyj@163.com

P. Tiwari is with the Department of Computer Science, Aalto University, Espoo, Finland; Email: prayag.tiwari@aalto.fi

Q. Zou is with the Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu, 610054, P.R.China; E-mail: zouquan@nclab.net

F. Guo is with the School of Computer Science and Engineering, Central South University, Changsha, 410083, P.R.China; E-mail: guofeieileen@163.com

H.M. Pandey is with Data Science and Artificial Intelligence, Department of Information and Computing, Bournemouth University, UK; Email: profharimohanpandey@gmail.com

Manuscript received Nov 28, 2021; revised , .

In recent years, computational methods based on machine learning (ML) have been proposed to solve 4mC recognition [1], [2], [3]. Conventional statistical learning and deep learning are the two main methods used to solve the 4mC identification problem. For statistical learning methods, manual feature extraction was used to represent DNA sequences. The support vector machine (SVM) [4], random forest (RF), nave Bayes, extremely randomized tree, AdaBoost and logistic regression were utilized to build a predictive model. The iDNA4mC model was first proposed to identify 4mC sites by Chen et al. [5]. The iDNA4mC used nucleotide chemical properties and frequency to represent features of DNA and fed them into SVM for prediction. He et al. [6] developed 4mCPred via SVM and position-specific trinucleotide sequence propensity (PSTNP), which can extract key information of DNA sequences. Wei et al. [7] proposed a two-step feature optimization strategy to construct a predictive model. This method was called 4mcPred-SVM. Based on multiple features of DNA sequences, Hasan et al. developed two types of predictors, i4mC-ROSE[8] and i4mC-Mouse [9], to identify 4mC sites in Rosaceae and mouse genomes. To further improve the predictive performance of the model, iDNA-MS[10], Meta-4mCpred [11] and DNA4mC-LIP [12] integrated existing predictors to identify 4mC sites.

Deep learning can represent the features of DNA sequences through multilayer networks. The 4mCCNN model, which was based on one-dimensional convolutional neural network (CNN), was proposed by Khanal et al. [13]. The DNC4mC-Deep model [14] employed nucleotide frequency (NCPNF), nucleotide chemical property, binary encoding (BE), nucleotide chemical property (NCP) and Kmer as input features for CNN. The long short-term memory (LSTM) was also used to develop an effective deep model, called DeepTorrent [15].

Fuzzy inference system (FIS) is an effective calculation model to solve uncertain and vague problems. Zero-order and first-order FIS (1-FIS), which are two of the more popular FIS models, had been employed in data mining, pattern recognition and automatic control. Classical fuzzy inference systems have three types of models: Mamdani-Larsen [16], Takagi-Sugeno-Kang (TSK) [17], [18] and generalized fuzzy systems [19]. Among them, the TSK model is a popular fuzzy system. Chen [20] and Chiang [21] proposed the zero-order TS fuzzy systems based on SVM. Support vectors were used to construct the antecedent and subsequent parts of fuzzy rules; the kernel function of SVM was composed of fuzzy basis functions, and the number of fuzzy rules was determined by the number of support vectors (consistent). This method can improve generalization ability, but with an increase in the support vector, the fuzzy rules will also increase. Therefore, this strategy increased the complexity of the systems. Xu et al. [22] also developed a zero-order TS fuzzy systems, called enhanced soft subspace clustering and sparse learning-based concise TSK fuzzy systems (ESSC-SL-CTSK-FS), could generate the sparse subspace and antecedent in each fuzzy rule. Kerk et al. [23] developed a monotone zero-order TSK-based FIS via monotone fuzzy rule interpolation. The first-order TSK fuzzy systems used fuzzy clustering algorithm to construct the rule antecedent, and the latter was a linear function. When the data is insufficient, or a fuzzy system trained with an incomplete data set, the generalization ability of the model will be affected. To overcome this problem, Deng et al. [24] studied a fuzzy system based on knowledge-levers (KL-FS) from the perspective of transfer learning. Gu et al. [25] proposed Bayesian TSK fuzzy classifier (B-TSK-FC), which estimated the parameters by Markov-Chain Monte-Carlo technique. Rezaee [26] developed a data-driven TSK systems that automatically obtained the fuzzy rules and optimized parameters. To address the problem of regression, Zuo et al. [27] proposed a TS fuzzy regression transfer learning model. For the constrained monotonic scenarios, a monotonic relation-constrained TSK systems were proposed by Deng [28]. For the data sequence, it is more difficult to determine the fuzzy set and adapt changes in the data distribution. Yu et al. [29] designed a topology learning-based fuzzy random neural network (TLFRNN) to solve this problem. To improve the performance of predictive model, a patch learning (PL) algorithm was proposed to build fuzzy systems by Wu et al. [30]. For TSK systems, Wu et al. [31] also proposed an efficient and effective training algorithm with minibatch gradient descent (MBGD), AdaBound and regularization. For identification of epileptic EEG signals, Jiang et al. designed a multiview TSK fuzzy systems (MV-TSK-FS) [32], which weighted outputs of different views. Jiang also employed TSK fuzzy systems, semisupervised learning and transductive transfer learning models to detect epileptic seizures via EEG signals [33]. Wiktorowicz [34] constructed a high-order TSK fuzzy systems by the particle swarm optimization (PSO) and batch least squares (BLS).

Although, FIS (based on TSK fuzzy systems) has been significantly developed, the above works lack consideration for the processing of high-dimensional feature spaces and noise samples. As the dimension of the feature space increases, the complexity of FIS will also increase. Noise samples will also affect the decision hyperplane of the model. Inspired by Chen [20], Chiang [21] and Wiktorowicz's [34] works, we propose a correntropy induced loss-based kernelized higher-order FIS (C-KHFIS), which is an extension of TSK fuzzy systems [17], [18]. The correntropy induced loss (C-loss) function [35] was a loss function that could improve robustness against noise for classifier.

The contributions of this work are as follows:

(1) We use the fuzzy rule-based kernel to build a higher-order fuzzy inference systems, which is a kernelized model.

(2) C-loss function is employed to improve the generalization ability of fuzzy systems.

(3) An effective iterative algorithm is proposed to optimize our fuzzy systems.

This work is organized as follows: In section II, we introduce first-order and high-order FIS models. In section III, we propose a C-loss based higher-order fuzzy inference systems. In section IV, we introduce the feature extraction of DNA sequences. In section V, we test our method on several benchmark data sets. Finally, the conclusion and future work are given in section VI.

## II. Related Work

FIS is a nonlinear system, which is described by multiple sets of if-then fuzzy rules. Each subsystem (corresponding to each rule) is a local approximation to the target problem. Finally, multiple subsystems are combined to jointly approximate the objective function. By the fuzzy C-means (FCM) algorithm, the subsystems of each fuzzy set obtain the distribution of local samples. Compared with the traditional neural network based on the backpropagation algorithm, FIS can reduce the computational complexity of the model via local approximation.

### A. First-order fuzzy inference systems

Suppose there is a training set $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N] \in \mathbf{R}^{d\times N}$ with $N$ samples and $d$ dimensions, where $\mathbf{x}_i = (x_{i1}, x_{i2}, ..., x_{id})^T \in \mathbf{R}^{d\times 1}$. In first-order fuzzy systems, the $k-$th fuzzy rule $R_k$ can be represented as:

$$\begin{aligned}&\text{If } x_{i1}\ is\ A_1^k \wedge x_{i2}\ is\ A_2^k \wedge ... \wedge x_{id}\ is\ A_d^k,\\ &\text{Than } f^k(\mathbf{x}_i) = p_0^k + p_1^k x_{i1} + p_2^k x_{i2} + ... + p_d^k x_{id},\\ &k = 1,2,...,M,\end{aligned} \tag{1}$$

where $M$ denotes the number of fuzzy rules. $A_j^k$ is the $k-$th fuzzy set for the $j-$th input feature $x_{ij}$, $\wedge$ represents a fuzzy conjunction, and $f^k(\mathbf{x}_i)$ denotes the defuzzification function for local output under $k-$th fuzzy set. In fact, $f^k(\mathbf{x}_i)$ is a first-order polynomial. When $f^k(\mathbf{x}_i)$ is nonlinear function, the above fuzzy systems belong to higher order fuzzy inference systems. The decision formula of first-order FIS (1-FIS) can be represented as:

$$\begin{aligned}y(\mathbf{x}_i) &= \sum_{k=1}^{M} \frac{\mu^k(\mathbf{x}_i)}{\sum_{k'=1}^{M}\mu^{k'}(\mathbf{x}_i)} f^k(\mathbf{x}_i)\\ &= \sum_{k=1}^{M} \widetilde{\mu}^k(\mathbf{x}_i) f^k(\mathbf{x}_i),\end{aligned} \tag{2}$$

where

$$\mu^k(\mathbf{x}_i) = \prod_{j=1}^{d} \mu_{A_j^k}(x_{ij}), \tag{3a}$$

$$\widetilde{\mu}^k(\mathbf{x}_i) = \frac{\mu^k(\mathbf{x}_i)}{\sum_{k'=1}^{M}\mu^{k'}(\mathbf{x}_i)}, \tag{3b}$$

where $\mu_{A_j^k}(x_{ij})$ denotes the fuzzy membership function, which can be calculated by Gaussian function:

$$\mu_{A_j^k}(x_{ij}) = exp(\frac{-(x_{ij}-c_j^k)}{2\sigma_j^k}), \tag{4}$$

where $\sigma_j^k$ and $c_j^k$ are the center and variance of $k-$th fuzzy set in $j$-th dimension and $\sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)\neq 0$. The models constructed from Gaussian membership function can approximate nonlinear continuous systems with arbitrary precision, and this type of membership function has been widely used in the field of fuzzy control.

For 1-FIS, the parameters of $\sigma_j^k$ and $c_j^k$ are in the if-parts. $\mathbf{p}^k=[p_0^k,p_1^k,...,p_d^k]^T\in\mathbf{R}^{(1+d)\times 1}$ are the parameters of then-parts. The learning of the above two kinds of parameters is implemented independently. For if-parts, FCM algorithm [36] is utilized to estimate $\sigma_j^k$ and $c_j^k$:

$$c_j^k=\sum_{i=1}^{N}\mu_{ik}x_{ij}\sum_{i=1}^{N}\mu_{ik}, \tag{5a}$$

$$\sigma_j^k=\frac{1}{2}\sum_{i=1}^{N}\mu_{ik}(x_{ij}-c_j^k)^2\sum_{i=1}^{N}\mu_{ik}, \tag{5b}$$

where $\mu_{ik}$ is the membership value of sample $\mathbf{x}_i$ belonging to cluster $k$. After if-parts learning, the parameters of of $\sigma_j^k$ and $c_j^k$ are determined. Next, the parameters of then-parts can be obtained by least squares method. Let the output of $k-$th fuzzy rule be:

$$\widetilde{\mathbf{x}}_i{}^k=\widetilde{\mu}^k(\mathbf{x}_i)\mathbf{x}_e\in\mathbf{R}^{(1+d)\times 1}, \tag{6a}$$

$$\mathbf{x}_e=(1,(\mathbf{x}_i)^T)^T\in\mathbf{R}^{(1+d)\times 1}. \tag{6b}$$

Therefore, the output of $M$ fuzzy rules can be represented as follows:

$$\mathbf{x}_{gi}=\left((\widetilde{\mathbf{x}}_i{}^1)^T,(\widetilde{\mathbf{x}}_i{}^2)^T,...,(\widetilde{\mathbf{x}}_i{}^M)^T\right)^T\in\mathbf{R}^{[(1+d)*M]\times 1}. \tag{7}$$

For the $k-$th fuzzy rule, the the parameters of then-parts are defined as:

$$\mathbf{p}^k=(p_0^k,p_1^k,...,p_d^k)^T\in\mathbf{R}^{(1+d)\times 1}. \tag{8}$$

The parameters of $M$ then-parts are defined as:

$$\mathbf{p}_g=\left((\mathbf{p}^1)^T,(\mathbf{p}^2)^T,...,(\mathbf{p}^M)^T\right)^T\in\mathbf{R}^{[(1+d)*M]\times 1}. \tag{9}$$

The output of the first-order fuzzy systems can be rewritten as:

$$y(\mathbf{x}_i)=\mathbf{p}_g^T\mathbf{x}_{gi}. \tag{10}$$

Then-parts learning can be regarded as a linear regression problem:

$$J_{1-FIS}(\mathbf{p}_g)=\frac{\lambda}{2}\|\mathbf{p}_g\|_2^2+\frac{1}{2}\sum_{i=1}^{N}\|\mathbf{p}_g^T\mathbf{x}_{gi}-y_i\|_2^2. \tag{11}$$

## B. Higher-order FIS

For the higher-order FIS, the $k-$th fuzzy rule $R_k$ can be represented as:

$$\begin{aligned}&\text{If } x_{i1}\ is\ A_1^k\wedge x_{i2}\ is\ A_2^k\wedge...\wedge x_{id}\ is\ A_d^k,\\&\text{Than } f^k(\mathbf{x}_i)=p_0^k+\sum_{j=1}^{d}p_j^kx_{ij}+\sum_{j,h}^{d}p_{jh}^kx_{ij}x_{ih}+...\\&+\sum_{j_1,j_2,...,j_m=1}^{d}p_{j_1,...,j_m}^kx_{ij_1}x_{ij_2}...x_{ij_m},\\&k=1,2,...,M,\end{aligned} \tag{12}$$

where $m\geq 2$ is the degree of a higher-order polynomial, which means that the functions are nonlinear.

1-FIS, which is based on TSK fuzzy systems, uses multiple linear systems to fit a nonlinear system and a fuzzy algorithm to deconstruct the input variables. Then, the variables are defuzzified through fuzzy calculus inference to generate the equation of the relationship between inputs and outputs. The identification of 4mC site is a complex classification problem. We conduct research on the basis of 1-FIS and propose a C-loss based higher-order fuzzy inference system.

## III. C-LOSS BASED HIGHER-ORDER FUZZY INFERENCE SYSTEMS

### A. Kernelized higher-order fuzzy inference systems

The kernelized higher-order fuzzy inference systems (KHFIS) can actually be regarded as a nonlinear problem, which can be decomposed into $M$ local nonlinear submodels. To obtain the form of the then-part function $f^k(\mathbf{x}_i)$ for the nonlinear submodel, we introduce the mapping function $\phi(\cdot)$ for $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)=K(\mathbf{x}_i,\mathbf{x}_j)$ (kernel), where $K(\mathbf{x}_i,\mathbf{x}_j)$ can be constructed by a linear, radial basis function (RBF), polynomial. The $k$-th fuzzy rule of the KHFIS can be represented as:

$$\begin{aligned}&\text{If } \mathbf{x}_i\in A^k\wedge x_{i2}\ is\ A_2^k\wedge...\wedge x_{id}\ is\ A_d^k,\\&\text{Than } f^k(\mathbf{x}_i)=\widetilde{\mu}^k(\mathbf{x}_i)(\mathbf{p}^k)^T(1,\phi(\mathbf{x}_i)^T)^T,\\&k=1,2,...,M.\end{aligned} \tag{13}$$

Thus, the output of the higher-order FIS is:

$$f(\mathbf{x}_i)=\sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)(\mathbf{p}^k)^T(1,\phi(\mathbf{x}_i)^T)^T. \tag{14}$$

The nonlinear problem of higher-order FIS is decomposed into a linear combination of $M$ local submodels in a high-dimensional space. Suppose the output of the $M$ if-parts is $\Phi_{gi}$:

$$\begin{aligned}\mathbf{\Phi}_{gi}&=\left\{(\psi(\mathbf{x}_i)^1)^T,...,(\psi(\mathbf{x}_i)^k)^T,...,\right.\\&\quad\left.(\psi(\mathbf{x}_i)^M)^T\right\}^T\in\mathbf{R}^{[(1+d')*M]\times 1},\\\psi(\mathbf{x}_i)^k&=\widetilde{\mu}^k(\mathbf{x}_i)((1,\phi(\mathbf{x}_i)^T)^T)\in\mathbf{R}^{(1+d')\times 1},\\&k=1,2,...,M,\end{aligned} \tag{15}$$

where $\psi(\mathbf{x}_i)^k$ is the output of the $k-$th if-parts and $d'$ is the dimension of nonlinear projection. The objective function of KHFIS is:

$$J_{KHFIS}(\mathbf{p}_g) = \frac{\lambda}{2}\|\mathbf{p}_g\|_2^2 + \frac{1}{2}\sum_{i=1}^{N}\|\mathbf{p}_g^T\mathbf{\Phi}_{gi} - y_i\|_2^2, \tag{16}$$

where $\mathbf{p}_g \in \mathbf{R}^{[(1+d')*M]\times 1}$.

We set $y_i - \mathbf{p}_g^T\mathbf{\Phi}_{gi} = \xi_i$ and Eq. (16) can be rewritten as:

$$\begin{aligned} &min J(\mathbf{p}_g, \xi_i) = \frac{1}{2}\mathbf{p}_g^T\mathbf{p}_g + \frac{1}{2\lambda}\sum_{i=1}^{N}\xi_i^2, \\ &s.t.\ y_i - \mathbf{p}_g^T\mathbf{\Phi}_{gi} = \xi_i,\ i = 1, 2, ..., N. \end{aligned} \tag{17}$$

The Lagrangian function of the optimization problem (17) is:

$$\begin{aligned} L(\mathbf{p}_g, \xi_i, \alpha_i) = &\frac{1}{2}\mathbf{p}_g^T\mathbf{p}_g + \frac{1}{2\lambda}\sum_{i=1}^{N}\xi_i^2 \\ &- \sum_{i=1}^{N}\alpha_i(\xi_i - y_i + \mathbf{p}_g^T\mathbf{\Phi}_{gi}), \end{aligned} \tag{18}$$

where $\alpha_i, i = 1, 2, ..., N$ is the Lagrange multiplier. The partial derivatives of $L$ is found by:

$$\begin{cases} \partial L/\partial \mathbf{p}_g = 0 & \Rightarrow\ \mathbf{p}_g = \sum_{i=1}^{N}\alpha_i\mathbf{\Phi}_{gi}, \\ \partial L/\partial \xi_i = 0 & \Rightarrow\ \xi_i = \lambda\alpha_i, \\ \partial L/\partial \alpha_i = 0 & \Rightarrow\ \mathbf{p}_g^T\mathbf{\Phi}_{gi} + \xi_i - y_i = 0. \end{cases} \tag{19}$$

Similar to TSK fuzzy systems, KHFIS uses a two-step learning strategy. First, the FCM algorithm (unsupervised learning stage) is used to initialize the fuzzy subset; then the supervised learning method is employed to train the parameters of the model according to the error. In Eq. (19), the solution process is the supervised learning stage of our algorithm. We can obtain the following linear equation:

$$(\mathbf{\Omega} + \lambda\mathbf{I})\boldsymbol{\alpha} = \mathbf{y}, \tag{20}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_N)^T$ and $\mathbf{y} = (y_1, y_2, ..., y_N)^T$. $\mathbf{\Omega} = \mathbf{\Phi}_g^T\mathbf{\Phi}_g \in \mathbf{R}^{N\times N}$ is a symmetric matrix, which can be called the fuzzy kernel. $\mathbf{\Phi}_g = \{\mathbf{\Phi}_{g1}, ..., \mathbf{\Phi}_{gi}, ..., \mathbf{\Phi}_{gN}\} \in \mathbf{R}^{[(1+d')*M]\times N}$. $\mathbf{\Omega}_{ij}$ can be calculated by:

$$\begin{aligned} \mathbf{\Omega}_{ij} &= \mathbf{\Phi}_{gi}^T\mathbf{\Phi}_{gj} \\ &= \begin{bmatrix} \left(\psi(\mathbf{x}_i)^1\right)^T, \cdots, \left(\psi(\mathbf{x}_i)^M\right)^T \end{bmatrix}_{1\times[(1+d')*M]} \begin{bmatrix} \psi(\mathbf{x}_j)^1 \\ \vdots \\ \psi(\mathbf{x}_j)^M \end{bmatrix}_{[(1+d')*M]\times 1} \\ &= \sum_{k=1}^{M}\left(\psi(\mathbf{x}_i)^k\right)^T\psi(\mathbf{x}_j)^k \\ &= \sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)\widetilde{\mu}^k(\mathbf{x}_j)\left[1, \phi(\mathbf{x}_i)^T\right]_{1\times(1+d')}\begin{bmatrix} 1 \\ \phi(\mathbf{x}_j) \end{bmatrix}_{(1+d')\times 1} \\ &= \sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)\widetilde{\mu}^k(\mathbf{x}_j)(\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) + 1) \\ &= \sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)\widetilde{\mu}^k(\mathbf{x}_j)(K(\mathbf{x}_i, \mathbf{x}_j) + 1), \end{aligned} \tag{21}$$

where $\widetilde{\mu}^k(\mathbf{x}_i)\widetilde{\mu}^k(\mathbf{x}_j)(K(\mathbf{x}_i, \mathbf{x}_j) + 1)$ can be considered as the element of the fuzzy kernel in the $k$-th fuzzy rule.

The kernel can be constructed by the radial basis function (RBF):

$$K_{RBF}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2), \tag{22}$$

where $\gamma$ denotes the kernel parameters. By a high-dimensional dot product calculation, the RBF kernel function can project samples from a low-dimensional space to a high-dimensional space. The value of the RBF function is a good similarity measure representation, and the range is between 0 and 1. The value increases with decreasing Euclidean distance.

For a test sample $\mathbf{x}_t \in \mathbf{R}^{d\times 1}$, the final output of KHFIS is:

$$\begin{aligned} y(\mathbf{x}_t) &= \mathbf{p}_g^T\mathbf{\Phi}_{gt} \\ &= \sum_{i=1}^{N}\alpha_i\begin{bmatrix} \left(\psi(\mathbf{x}_i)^1\right)^T, \cdots, \left(\psi(\mathbf{x}_i)^M\right)^T \end{bmatrix}_{1\times[(1+d')*M]} \begin{bmatrix} \psi(\mathbf{x}_t)^1 \\ \vdots \\ \psi(\mathbf{x}_t)^M \end{bmatrix}_{[(1+d')*M]\times 1} \\ &= \sum_{i=1}^{N}\alpha_i\sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)\widetilde{\mu}^k(\mathbf{x}_t)(K(\mathbf{x}_i, \mathbf{x}_t) + 1). \end{aligned} \tag{23}$$

For binary classification ($y \in \{+1, -1\}$), Eq. (23) can also be represented as:

$$y(\mathbf{x}_t) = sign\left[\sum_{i=1}^{N}\alpha_i\sum_{k=1}^{M}\widetilde{\mu}^k(\mathbf{x}_i)\widetilde{\mu}^k(\mathbf{x}_t)(K(\mathbf{x}_i, \mathbf{x}_t) + 1)\right]. \tag{24}$$

The process of KHFIS is shown in Algorithm 1 and Fig. 1.

**Algorithm 1** Algorithm of KHFIS

```
Require: The training labels $\mathbf{y} \in \mathbf{R}^{N\times 1}$, features $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N] \in \mathbf{R}^{d\times N}$ and testing sample $\mathbf{x}_t \in \mathbf{R}^{d\times 1}$;
    The parameters of regularization coefficient $\lambda$, number of fuzzy rules $M$;
Ensure: The prediction of $y(\mathbf{x}_t)$;
1: Using FCM to calculate the parameters of if-parts;
2: Estimating $\widetilde{\mu}^k(\mathbf{x}_i), i = 1, 2, ..., N, k = 1, 2, ..., M$ by Eq. (3a) and (3b);
3: Computing $\mathbf{\Omega}$ by Eq. (22) and (21);
4: Estimating $\boldsymbol{\alpha}$ via Eq. (20).
5: Predicting $y(\mathbf{x}_t)$ by Eq. (23);
```

## B. Correntropy induced loss based KHFIS

The C-loss function [35] can be expressed as:

$$l_C(y_i, f(\mathbf{x}_i)) = 1 - exp\left(-\frac{(y_i - f(\mathbf{x}_i))^2}{2\rho^2}\right), \tag{25}$$

where $\rho$ denotes the bandwidth. The C-loss is differentiable and smooth. Fig. 2 shows the square loss and C-loss function under different widths $\rho$. Obviously, the C-loss function can effectively reduce the influence of large errors on the model. The square loss increases as the error increases (the negative error decreases), and the value of the loss function increases

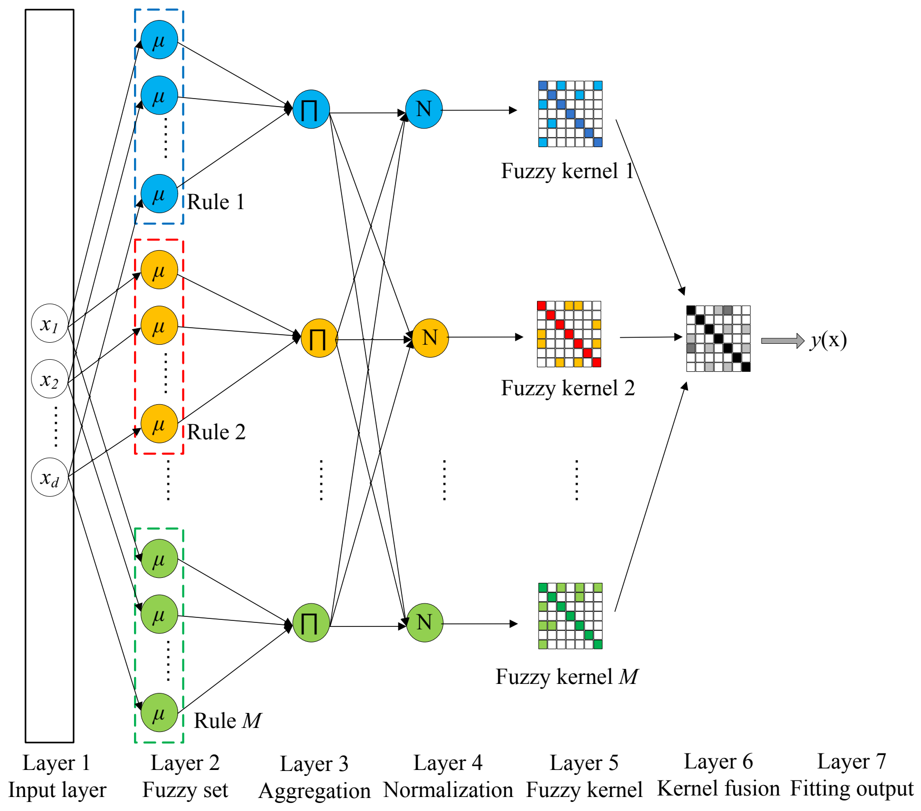


Fig. 1: Schematic of kernelized higher-order fuzzy inference systems.

quickly. In contrast, the incremental slope of the C-loss function is not as steep. We can adjust the width $\rho$ to adapt the sensitivity to outliers.

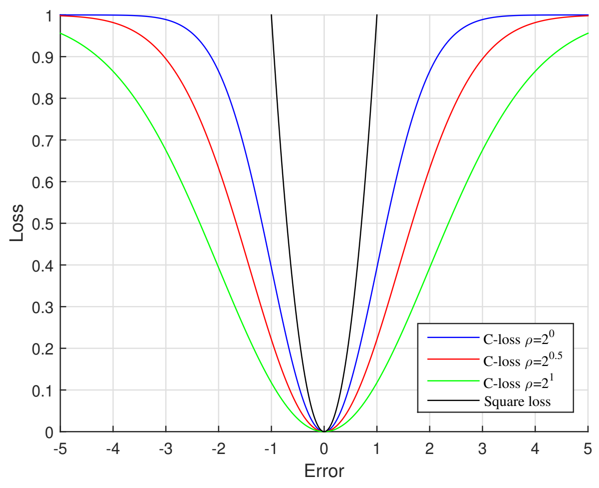


Fig. 2: C-loss with different widths.

We replace the square loss function with the C-loss function, and the mathematical model of the C-loss based KHFIS (C-KHFIS) is defined as:

$$
\begin{aligned}
minJ(\mathbf{p}_g, \xi_i) = & \frac{1}{2}\mathbf{p}_g^T\mathbf{p}_g \\
& + \frac{1}{2\lambda}\sum_{i=1}^{N}\left(1 - exp\left(-\frac{\xi_i^2}{2\rho^2}\right)\right), \\
& s.t.\ y_i - \mathbf{p}_g^T\mathbf{\Phi}_{gi} = \xi_i,\ i = 1, 2, ..., N.
\end{aligned}
\tag{26}
$$

However, Eq. (26) is nonconvex and cannot be solved directly. The half-quadratic (HQ) optimization algorithm [37] can be used to solve the above problem. By introducing an auxiliary variable, we first define a convex function:

$$g(\nu) = -\nu\ log(-\nu) + \nu, \tag{27}$$

where $\nu < 0$. The conjugate function of $g(\nu)$ is:

$$g^*(\tau) = \sup_{\nu} g'(\nu), \tag{28}$$

where

$$g'(\nu) = \tau\nu - g(\nu) = \tau\nu + \nu\ log(-\nu) - \nu, \tag{29}$$

When $g'(\nu)$ is a nonconvex function, let $\frac{dg'(\nu)}{d\nu} = 0$:

$$\tau + log(-\nu) = 0\ \Rightarrow\ \nu = -exp(-\tau) < 0. \tag{30}$$

When formula (30) is combined into formula (28):

$$g^*(\tau) = exp(-\tau). \tag{31}$$

Letting $\tau = \frac{\xi_i^2}{2\rho^2}$, we can obtain:

$$g^*\left(\frac{\xi_i^2}{2\rho^2}\right)=\sup_{\nu}\left\{\frac{\xi_i^2}{2\rho^2}\nu+\nu\ log(-\nu)-\nu\right\} \\ =exp\left(-\frac{\xi_i^2}{2\rho^2}\right), \tag{32}$$

and the supremum is reached at $\nu=-exp\left(-\frac{\xi_i^2}{2\rho^2}\right)$.

From Eq. (26) and (32), we obtain:

$$\begin{aligned} &min\ J(\mathbf{p}_g,\xi_i,\nu_i)\\ &=\frac{\lambda}{2}\mathbf{p}_g^T\mathbf{p}_g\\ &+\sum_{i=1}^{N}\left(1-\sup_{\nu_i}\left\{exp\left(-\frac{\xi_i^2}{2\rho^2}\right)\nu_i-g(\nu_i)\right\}\right),\\ &s.t.\ y_i-\mathbf{p}_g^T\mathbf{\Phi}_{gi}=\xi_i,\ i=1,2,...,N,\end{aligned} \tag{33}$$

where $N$ samples for $\boldsymbol{\nu}=(\nu_1,\nu_2,...,\nu_N)^T\in\mathbf{R}^{N\times 1}$. Eq. (33) can also be simplified as:

$$\begin{aligned} &min\ J(\mathbf{p}_g,\xi_i,\nu_i)\\ &=\frac{\lambda}{2}\mathbf{p}_g^T\mathbf{p}_g\\ &+\sup_{\boldsymbol{\nu}}\left\{\sum_{i=1}^{N}\left(-\frac{\xi_i^2}{2\rho^2}\nu_i+g(\nu_i)\right)\right\},\\ &s.t.\ y_i-\mathbf{p}_g^T\mathbf{\Phi}_{gi}=\xi_i,\ i=1,2,...,N.\end{aligned} \tag{34}$$

We use an alternating algorithm to solve Eq. (34). Fixing $\mathbf{p}_g^{(n)}$ and $\boldsymbol{\xi}^{(n)}$ to optimize $\boldsymbol{\nu}^{(n+1)}$, Eq. (34) becomes:

$$min\ J(\boldsymbol{\nu}^{(n+1)})=\sum_{i=1}^{N}\left(-\frac{(\xi_i^{(n)})^2}{2\rho^2}\nu_i^{(n+1)}+g(\nu_i^{(n+1)})\right), \tag{35}$$

where $n$ is the $n-$th iteration, and $\xi_i^{(n)}=y_i-(\mathbf{p}_g^{(n)})^T\mathbf{\Phi}_{gi},\ i=1,2,...,N$. According to Eq. (29), the closed-form solutions of Eq. (35) are:

$$\nu_i^{(n+1)}=-exp\left(-\frac{(\xi_i^{(n)})^2}{2\rho^2}\right)<0,\ i=1,2,...,N. \tag{36}$$

Fixing $\boldsymbol{\nu}^{(n+1)}$ optimizes $\mathbf{p}_g^{(n+1)}$ and $\boldsymbol{\xi}^{(n+1)}$ by solving the following formula:

$$\begin{aligned} &min\ J(\mathbf{p}_g^{(n+1)},\boldsymbol{\xi}^{(n+1)})\\ &=\frac{\lambda}{2}(\mathbf{p}_g^{(n+1)})^T\mathbf{p}_g^{(n+1)}\\ &+\sum_{i=1}^{N}\left(-\frac{(\xi_i^{(n+1)})^2}{2\rho^2}\nu_i^{(n+1)}\right)\\ &s.t.\ y_i-\mathbf{p}_g^T\mathbf{\Phi}_{gi}=\xi_i,\ i=1,2,...,N.\end{aligned} \tag{37}$$

According to Eq. (37), the Lagrangian function is:

$$\begin{aligned} &L(\mathbf{p}_g^{(n+1)},\boldsymbol{\xi}^{(n+1)},\boldsymbol{\alpha}^{(n+1)})\\ &=\frac{\lambda}{2}(\mathbf{p}_g^{(n+1)})^T\mathbf{p}_g^{(n+1)}\\ &+\sum_{i=1}^{N}\left(-\frac{(\xi_i^{(n+1)})^2}{2\rho^2}\nu_i^{(n+1)}\right)\\ &-\sum_{i=1}^{N}\alpha_i^{(n+1)}\left(\xi_i^{(n+1)}-y_i+(\mathbf{p}_g^{(n+1)})^T\mathbf{\Phi}_{gi}\right).\end{aligned} \tag{38}$$

We set the diagonal matrix $\mathbf{V}^{(n+1)}=diag(-\nu_1^{(n+1)},-\nu_2^{(n+1)},...,-\nu_N^{(n+1)})\in\mathbf{R}^{N\times N}$, and $\mathbf{y}=(y_1,y_2,...,y_N)^T$, $\boldsymbol{\xi}^{(n+1)}=(\xi_1^{(n+1)},\xi_2^{(n+1)},...,\xi_N^{(n+1)})^T\in\mathbf{R}^{N\times 1}$. Then, Eq. (38) can be simplified as:

$$\begin{aligned} &L(\mathbf{p}_g^{(n+1)},\boldsymbol{\xi}^{(n+1)},\boldsymbol{\alpha}^{(n+1)})\\ &=\frac{\lambda}{2}(\mathbf{p}_g^{(n+1)})^T\mathbf{p}_g^{(n+1)}\\ &+\frac{1}{2\rho^2}(\boldsymbol{\xi}^{(n+1)})^T\mathbf{V}^{(n+1)}\boldsymbol{\xi}^{(n+1)}\\ &+(\boldsymbol{\alpha}^{(n+1)})^T\left(\mathbf{y}-\left((\mathbf{p}_g^{(n+1)})^T\mathbf{\Phi}_g\right)^T-\boldsymbol{\xi}^{(n+1)}\right).\end{aligned} \tag{39}$$

The partial derivatives of $L$ is found by::

$$\begin{cases} \frac{\partial L}{\boldsymbol{\xi}^{(n+1)}}=0 & \Rightarrow \boldsymbol{\xi}^{(n+1)}=\rho^2(\mathbf{V}^{(n+1)})^{-1}\boldsymbol{\alpha}^{(n+1)},\\ \frac{\partial L}{\partial\mathbf{p}_g^{(n+1)}}=0 & \Rightarrow \mathbf{p}_g^{(n+1)}=\frac{1}{\lambda}\mathbf{\Phi}_g\boldsymbol{\alpha}^{(n+1)},\\ \frac{\partial L}{\boldsymbol{\alpha}^{(n+1)}}=0 & \Rightarrow \mathbf{y}-\left((\mathbf{p}_g^{(n+1)})^T\mathbf{\Phi}_g\right)^T=\boldsymbol{\xi}^{(n+1)}.\end{cases} \tag{40}$$

Similar to KHFIS, Eq. (40) is the solution of C-KHFIS and belongs to the supervised learning stage of our algorithm, which uses the error to learn parameters. We can obtain $\boldsymbol{\alpha}^{(n+1)}$ and $\mathbf{p}_g^{(n+1)}$:

$$\begin{aligned}\boldsymbol{\alpha}^{(n+1)}&=\left(\frac{1}{\lambda}(\mathbf{\Phi}_g)^T\mathbf{\Phi}_g+\rho^2(\mathbf{V}^{(n+1)})^{-1}\right)^{-1}\mathbf{y}\\ &=\left(\frac{1}{\lambda}\mathbf{\Omega}+\rho^2(\mathbf{V}^{(n+1)})^{-1}\right)^{-1}\mathbf{y}.\end{aligned} \tag{41}$$

$$\mathbf{p}_g^{(n+1)}=\frac{1}{\lambda}\mathbf{\Phi}_g\left(\frac{1}{\lambda}\mathbf{\Omega}+\rho^2(\mathbf{V}^{(n+1)})^{-1}\right)^{-1}\mathbf{y}. \tag{42}$$

The process of C-KHFIS is listed in Algorithm 2. Theorem 1 indicates that Algorithm 2 converges.

*Theorem 1:* The values of Eq. (33) monotonically decrease in each iteration until convergence.

*Proof 1:* Suppose $J(\mathbf{p}_g^{(n)},\boldsymbol{\xi}^{(n)},\boldsymbol{\nu}^{(n)})$ is the value of the objective function Eq. (33) in the $n-$th iteration. In the $(n+1)-$th iteration, $\mathbf{p}_g^{(n)}$ is fixed, and the subproblem Eq. (35) is solved to obtain the optimal $\boldsymbol{\nu}^{(n+1)}$. Since Eq. (35) is convex, then

$$J(\mathbf{p}_g^{(n)},\boldsymbol{\xi}^{(n)},\boldsymbol{\nu}^{(n+1)})\leq J(\mathbf{p}_g^{(n)},\boldsymbol{\xi}^{(n)},\boldsymbol{\nu}^{(n)}). \tag{43}$$

Fixing $\boldsymbol{\nu}^{(n+1)}$ and solving Eq. (37) via Eq. (42), we can achieve the optimal $\mathbf{p}_g^{(n+1)}$ in the $(n+1)-$th iteration. Eq. (37) is convex problem, so we have

**Algorithm 2** Algorithm of C-KHFIS

```
Require: The training labels $\mathbf{y} \in \mathbf{R}^{N\times 1}$, features $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_i, ..., \mathbf{x}_N] \in \mathbf{R}^{d\times N}$ and testing sample $\mathbf{x}_t \in \mathbf{R}^{d\times 1}$; The parameters of regularization coefficient $\lambda$, number of fuzzy rules $M$ and width $\rho$;
Ensure: The prediction of $y(\mathbf{x}_t)$;
 1: Using FCM to calculate the parameters of if-parts;
 2: Estimating $\widetilde{\mu}^k(\mathbf{x}_i), i = 1, 2, ..., N, k = 1, 2, ..., M$ by Eq. (3a) and (3b);
 3: Computing $\mathbf{\Omega}$ by Eq. (22) and (21);
 4: Randomly initializing $\boldsymbol{\alpha}^{(1)}$ ;
 5: for $n = 1 \to n_{max}$ do
 6:     Calculating $\boldsymbol{\xi}^{(n)}$ by $\boldsymbol{\xi}^{(n)} = \mathbf{\Omega}\boldsymbol{\alpha}^{(n)} - \mathbf{y}$;
 7:     Estimating $\nu_i^{(n+1)}, i = 1, 2, ..., N$ by Eq. (36);
 8:     Constructing $\mathbf{V}^{(n+1)} = diag(-\nu_1^{(n+1)}, -\nu_2^{(n+1)}, ..., -\nu_N^{(n+1)})$;
 9:     Calculating $\boldsymbol{\alpha}^{(n+1)} = \left(\frac{1}{\lambda}\mathbf{\Omega} + \rho^2(\mathbf{V}^{(n+1)})^{-1}\right)^{-1}\mathbf{y}$;
10: end for
11: Predicting $y(\mathbf{x}_t)$ by Eq. (23);
```

$$J(\mathbf{p}_g^{(n+1)}, \boldsymbol{\xi}^{(n+1)}, \boldsymbol{\nu}^{(n+1)}) \leq J(\mathbf{p}_g^{(n)}, \boldsymbol{\xi}^{(n)}, \boldsymbol{\nu}^{(n+1)}). \quad (44)$$

Finally, we combine the above results and obtain the following:

$$J(\mathbf{p}_g^{(n+1)}, \boldsymbol{\xi}^{(n+1)}, \boldsymbol{\nu}^{(n+1)}) \leq J(\mathbf{p}_g^{(n)}, \boldsymbol{\xi}^{(n)}, \boldsymbol{\nu}^{(n)}). \quad (45)$$

### C. Robustness analysis

To verify the robustness of C-KHFIS, we employ two experiments, which include classification and regression. The noise samples will affect the construction of the model and lead to poor prediction performance. In Fig. 3(a), we randomly generate two classes (blue and red points) of data under a Gaussian distribution. Each class contains 300 samples. C-KHFIS, KHFIS and 1-FIS have similar classification decision boundaries and separate the two classes easily. In Fig. 3(b), 50 noise points are added to the data set. The 50 noise samples original belong to class 2. However, these points are regarded as class 1. With the addition of noise, the decision boundaries of the three models all change. The changes in KHFIS and 1-FIS are obvious. Due to the suppression of the robust loss function, the decision boundary of C-KHFIS is almost unaffected. In Fig. 4(a), there is no noise in the data set, which is generated by the sinc function. Each model has a good fitting effect. In Fig. 4(b), 30 random noise points are added to the original data set. The performance of each method is affected. Due to the kernel method (nonlinear), C-KHFIS and KHFIS are less affected than the 1-FIS method. Compared with KHFIS, C-KHFIS has a better fitting effect.

## IV. The feature extraction of DNA sequences

In this work, we utilize position-specific trinucleotide sequence propensity (PSTNP) [6] to represent the features of DNA sequences. PSTNP is the extended version of pseudo K-tuple nucleotide composition (PseKNC) [38], [39], [40]. The DNA sequence generally includes four characters: 'A', 'C', 'G', and 'T'. PSTNP generally uses trinucleotide (TriN) to locally represent DNA sequences. The trinucleotide can be expressed as:

$$\begin{cases} TriN_1 & =' AAA'; \\ TriN_2 & =' AAC'; \\ TriN_3 & =' AAG'; \\ & ...; \\ TriN_{64} & =' TTT'; \end{cases} \quad (46)$$

To represent a DNA sequence containing 4mC sites, a PSTNSP profile is defined as:

$$\begin{aligned} C_{i,j} &= F^+(TriN_i|j) - F^-(TriN_i|j), \\ & i = 1, 2, ..., 64; \ j = 1, 2, ..., 39, \end{aligned} \quad (47)$$

where $i$ is the type of trinucleotide and $j$ denotes the position on the 4mC fragment. $F^+(TriN_i|j)$ and $F^-(TriN_i|j)$ are the frequencies of the $i-$th trinucleotide of the $j-$th position for positive and negative samples, respectively. The length of the 4mC fragment was 41 in He's study [6].

## V. Experiments and Results

### A. Data sets of 4mC

In Chen's work [5], six types of data sets were collected. There were Escherichia coli (E.coli), Caenorhabditis elegans (C.elegans), Geoalkalibacter subterraneus (G.subterraneus), Geobacter pickeringii (G.pickeringi), Arabidopsis thaliana (A.thaliana) and Drosophila melanogaster (D.melanogaster). The sizes of the six data sets are listed in Table I. Readers can refer to Chen's work [5] for the construction process.

TABLE I: The size of the 4mC data sets .

| Species | Negative | Positive |
|---|---|---|
| A.thaliana | 1,978 | 1,978 |
| C.elegans | 1,554 | 1,554 |
| D.melanogaster | 1,769 | 1,769 |
| E.coli | 388 | 388 |
| G.pickeringi | 569 | 569 |
| G.subterraneus | 906 | 906 |

### B. Measurement of performance

Matthew's correlation coefficient (MCC), sensitivity (SN), specificity (SP) and accuracy (ACC) are employed to evaluate the models. They are:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}, \quad (48a)$$

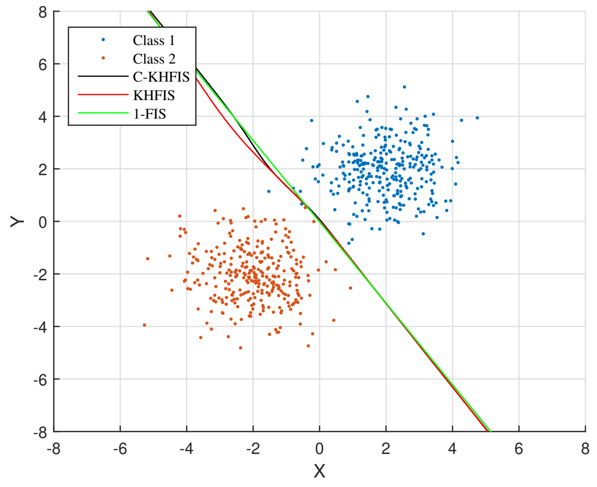
$$SN = \frac{TP}{FN + TP}, \quad (48b)$$

$$Spec = \frac{TN}{FP + TN}, \quad (48c)$$

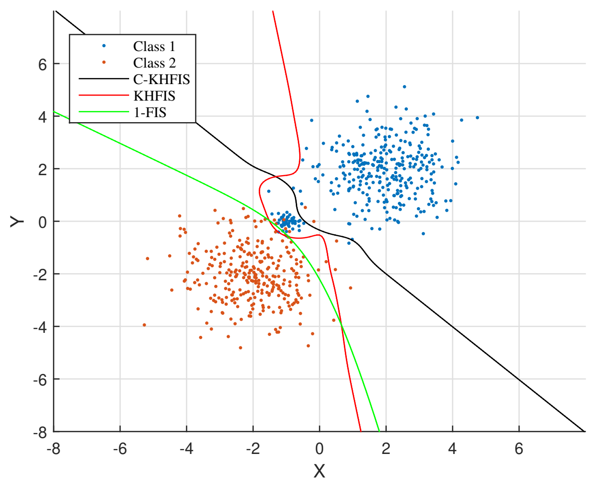
$$ACC = \frac{TN + TP}{TN + FN + TP + FP}, \quad (48d)$$

$$(48e)$$

, where $FP$, $FN$, $TN$ and $TP$ are the numbers of false positives, false negatives, true negatives and true positives, respectively. Ten-fold cross-validation (10-CV) is utilized to verify the performance of classifiers.

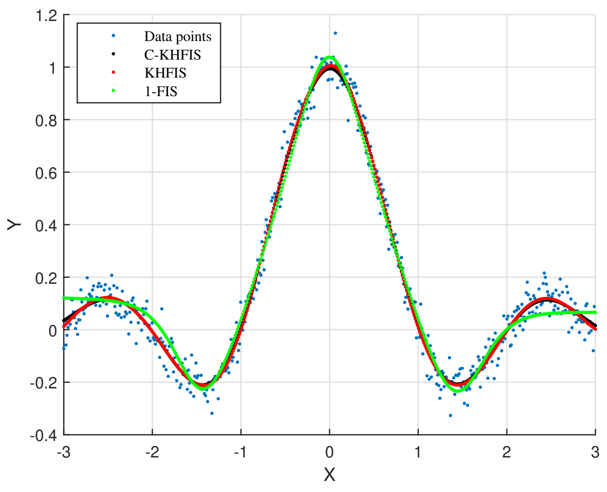


(a) Classifier decision boundaries without noise

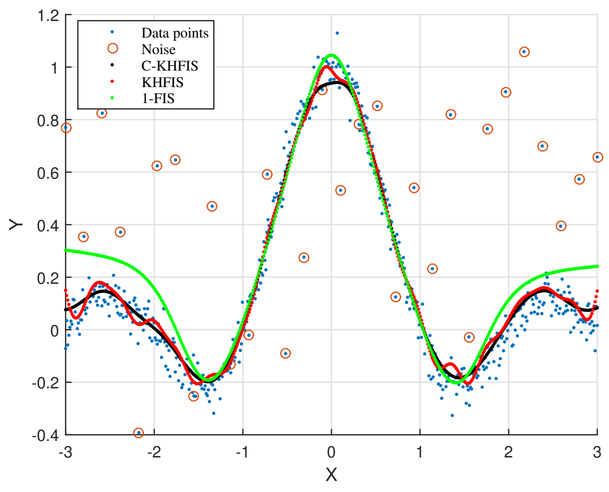


(b) Classifier decision boundaries with 50 noise

Fig. 3: The decision boundaries of different models.



(a) Sinc function fitting without noise



(b) Sinc function fitting with 30 noise

Fig. 4: The sinc function fitting curves of different models.

### C. Convergence on 4mC data sets

The convergence of C-KHFIS is also verified by means of experimental simulation. In Fig. 5, we calculate the objective value of C-KHFIS in each iteration. On all data sets, the C-KHFIS model can converge after $4$ iterations. Therefore, our optimization algorithm is effective and the convergence speed of the C-KHFIS is fast.

### D. Comparison of FIS methods

To show the robustness of C-KHFIS, we evaluate the performance of 1-FIS, KHFIS and C-KHFIS. The results of the comparison are shown in Table II. In all data sets, the kernel-based models (with RBF kernel) have higher accuracy. Compared with 1-FIS, KHFIS has improved ACC by $0.75\%$, $3.19\%$, $1.13\%$, $2.52\%$, $3.19\%$ and $1.8\%$ on the six data sets, respectively. In Fig. 6, the area under the receiver operating characteristic (AUC) curves also show the classification performance of the models. On the six data sets, the C-KHFIS achieves the best AUCs of $0.8944$, $0.9392$, $0.9394$, $0.9891$, $0.9621$ and $0.9512$, respectively.

### E. Comparison with existing predictors for 4mC

Our method is compared with common 4mC prediction model. These methods include 4mCPred [6], Meta-4mCpred[11], iDNA4mC [5], 4mcPred-SVM [7], DeepTorrent [15] and 4mCNN [13]. The classifiers of the above methods mainly consist of SVM and convolutional neural network (CNN). It can be seen from Table III that our method obtains the best prediction accuracy on 6 data sets. Compared with the second-best method (DeepTorrent [15]), C-KHFIS improves

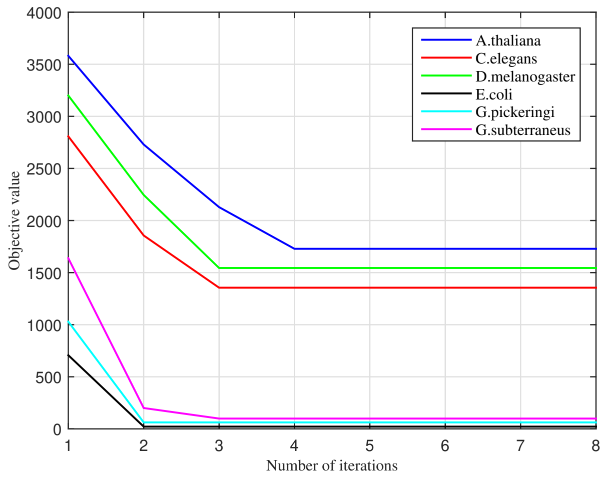


Fig. 5: The convergence curves of C-KHFIS on the 4mC data sets.

TABLE II: Comparison of the prediction performance between different FISs on six data sets (under 10-CV).

| Species | Method | ACC | SN | SP | MCC |
|---|---|---|---|---|---|
| A.thaliana | 1-FIS | 0.7951 | **0.8270** | 0.7630 | 0.5909 |
| | KHFIS | 0.8026 | 0.7897 | 0.8155 | 0.6055 |
| | C-KHFIS | **0.8270** | 0.8181 | **0.8356** | **0.6537** |
| C.elegans | 1-FIS | 0.8239 | 0.8283 | 0.8211 | 0.6493 |
| | KHFIS | 0.8558 | **0.8651** | 0.8517 | 0.7128 |
| | C-KHFIS | **0.8692** | 0.8632 | **0.8769** | **0.7398** |
| D.melanogaster | 1-FIS | 0.8390 | 0.8585 | 0.8191 | 0.6784 |
| | KHFIS | 0.8503 | 0.8407 | 0.8593 | 0.7011 |
| | C-KHFIS | **0.8712** | **0.8726** | **0.8692** | **0.7421** |
| E.coli | 1-FIS | 0.9218 | 0.9405 | 0.8991 | 0.8441 |
| | KHFIS | 0.9470 | 0.9510 | 0.9390 | 0.8941 |
| | C-KHFIS | **0.9529** | **0.9616** | **0.9410** | **0.9056** |
| G.pickeringi | 1-FIS | 0.8545 | 0.8795 | 0.8308 | 0.7104 |
| | KHFIS | 0.8864 | 0.8818 | 0.8923 | 0.7732 |
| | C-KHFIS | **0.9039** | **0.9032** | **0.9053** | **0.8083** |
| G.subterraneus | 1-FIS | 0.8438 | 0.8619 | 0.8261 | 0.6894 |
| | KHFIS | 0.8618 | 0.8608 | 0.8639 | 0.7238 |
| | C-KHFIS | **0.8903** | **0.8866** | **0.8944** | **0.7812** |

ACC by 2.4%, 1.1%, 1%, 8%, 1% and 1%, respectively, on six data sets. DeepTorrent [15] was built based on a deep learning model, which was good at feature representation learning. However, the recognition rate of positive samples is much lower than that of negative samples on multiple data sets. As a result, the value of SP is much higher than that of other machine learning methods. Deep learning requires more samples to train the parameters of the network. When the number of samples is small and the selection of initialization parameters is unreasonable, the prediction results will be biased. Moreover, the loss function of the model is important. The C loss function is a bounded, nonconvex, smooth loss function. C-KHFIS is a C loss-based neuro-fuzzy systems that can effectively reduce the impact of outliers on classification. The gap between SN and SP is not very large.

TABLE III: Comparison between different methods on six data sets (under 10-CV).

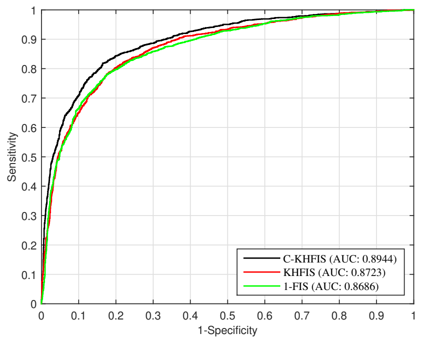
| Species | Method | ACC | SN | SP | MCC |
|---|---|---|---|---|---|
| A.thaliana | Meta-4mCpred[11] | 0.792 | 0.761 | 0.822 | 0.584 |
| | 4mCPred [6] | 0.768 | 0.755 | 0.780 | 0.536 |
| | iDNA4mC [5] | 0.760 | 0.757 | 0.762 | 0.519 |
| | 4mCNN [13] | 0.797 | 0.804 | 0.792 | 0.623 |
| | 4mcPred-SVM [7] | 0.787 | 0.778 | 0.796 | 0.573 |
| | DeepTorrent [15] | 0.803 | 0.703 | **0.903** | 0.620 |
| | C-KHFIS | **0.827** | **0.818** | 0.836 | **0.654** |
| C.elegans | Meta-4mCpred[11] | 0.826 | 0.840 | 0.812 | 0.652 |
| | 4mCPred [6] | 0.826 | 0.825 | 0.826 | 0.652 |
| | iDNA4mC [5] | 0.786 | 0.797 | 0.775 | 0.572 |
| | 4mCNN [13] | 0.842 | **0.895** | 0.825 | 0.694 |
| | 4mcPred-SVM [7] | 0.815 | 0.824 | 0.807 | 0.631 |
| | DeepTorrent [15] | 0.858 | 0.810 | **0.906** | 0.719 |
| | C-KHFIS | **0.869** | 0.863 | 0.877 | **0.740** |
| D.melanogaster | Meta-4mCpred[11] | 0.842 | 0.831 | 0.854 | 0.685 |
| | 4mCPred [6] | 0.822 | 0.824 | 0.821 | 0.646 |
| | iDNA4mC [5] | 0.812 | 0.833 | 0.791 | 0.625 |
| | 4mCNN [13] | 0.854 | 0.864 | 0.854 | 0.687 |
| | 4mcPred-SVM [7] | 0.830 | 0.838 | 0.822 | 0.661 |
| | DeepTorrent [15] | 0.861 | 0.834 | **0.889** | 0.724 |
| | C-KHFIS | **0.871** | **0.873** | 0.869 | **0.742** |
| E.coli | Meta-4mCpred[11] | 0.848 | 0.869 | 0.827 | 0.697 |
| | 4mCPred [6] | 0.826 | 0.819 | 0.832 | 0.655 |
| | iDNA4mC [5] | 0.799 | 0.820 | 0.778 | 0.598 |
| | 4mCNN [13] | 0.859 | 0.881 | 0.789 | 0.688 |
| | 4mcPred-SVM [7] | 0.833 | 0.858 | 0.807 | 0.666 |
| | DeepTorrent [15] | 0.873 | 0.891 | 0.855 | 0.747 |
| | C-KHFIS | **0.953** | **0.962** | **0.941** | **0.906** |
| G.pickeringi | Meta-4mCpred[11] | 0.891 | 0.884 | 0.898 | 0.782 |
| | 4mCPred [6] | 0.830 | 0.850 | 0.810 | 0.668 |
| | iDNA4mC [5] | 0.831 | 0.824 | 0.838 | 0.663 |
| | 4mCNN [13] | 0.872 | 0.858 | 0.893 | 0.750 |
| | 4mcPred-SVM [7] | 0.860 | 0.863 | 0.858 | 0.721 |
| | DeepTorrent [15] | 0.894 | 0.831 | **0.957** | 0.795 |
| | C-KHFIS | **0.904** | **0.903** | 0.905 | **0.808** |
| G.subterraneus | Meta-4mCpred[11] | 0.855 | 0.856 | 0.854 | 0.711 |
| | 4mCPred [6] | 0.828 | 0.818 | 0.837 | 0.662 |
| | iDNA4mC [5] | 0.815 | 0.822 | 0.808 | 0.630 |
| | 4mCNN [13] | 0.860 | 0.852 | 0.843 | 0.704 |
| | 4mcPred-SVM [7] | 0.837 | 0.840 | 0.837 | 0.674 |
| | DeepTorrent [15] | 0.880 | 0.813 | **0.948** | 0.768 |
| | C-KHFIS | **0.890** | **0.887** | 0.894 | **0.781** |

## *F. Comparison on UCI data sets*

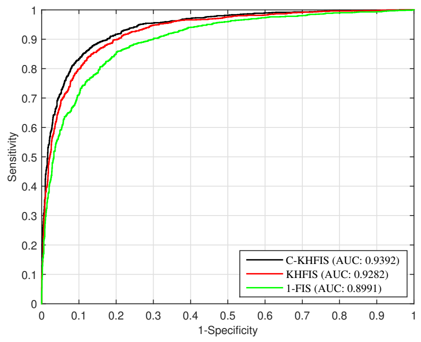
We further utilize eight data sets to evaluate KHFIS and C-KHFIS from the UCI Machine Learning Repository [41]. KNN, standard SVM [42], Kernel sparse representation-based classifier (Kernel SRC) [43] and 1-FIS are performed on these data via 5-CV. The results of the comparison are shown in Table IV. C-KHFIS achieves the highest values of accuracy on Australian (0.8757), ionosphere (0.9601), breast (0.9801), blood (0.8002), hearts (0.8519), diabetes (0.7834), and sonar (0.9150) data sets. In addition, Kernel SRC has the best accuracy (0.7740) on German Credit. C-KHFIS is also competent in other fields.

## VI. Conclusion and Future work

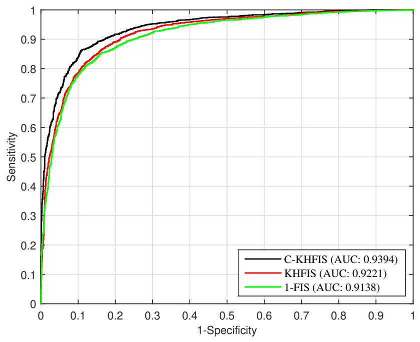
In this study, we employ the position-specific trinucleotide sequence propensity method to extract key information from DNA sequences, propose the C-KHFIS model to construct a classifier, and achieve competitive results. PSTNP can estimate



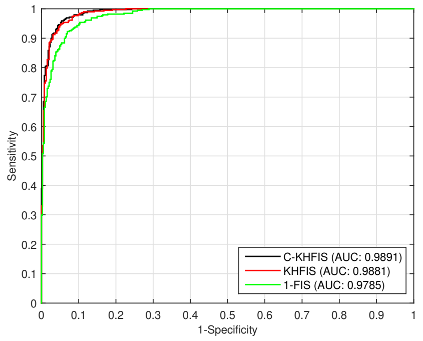
(a) ROCs on A.thaliana



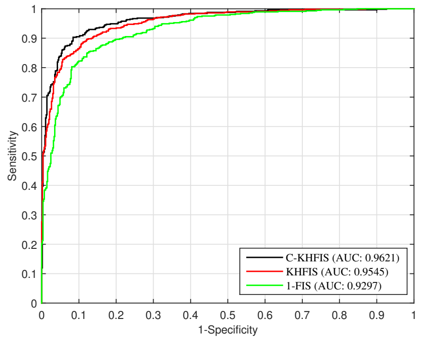
(b) ROCs on C.elegans



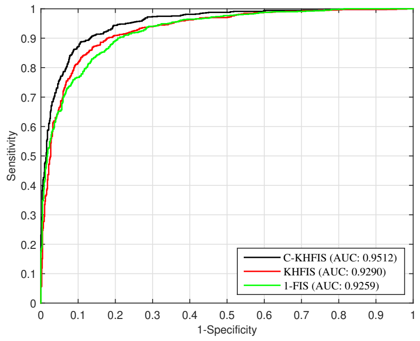
(c) ROCs on D.melanogaster



(d) ROCs on E.coli



(e) ROCs on G.pickeringi



(f) ROCs on G.subterraneus

Fig. 6: The ROCs of different models.

TABLE IV: The accuracy of different classifiers on UCI data sets.

| Data set | KNN | RF | SVM | Kernel SRC | 1-FIS | KHFIS | C-KHFIS |
|---|---|---|---|---|---|---|---|
| Australian | 0.8565 | 0.8638 | 0.8623 | 0.8304 | 0.8609 | 0.8638 | **0.8757** |
| Blood | 0.7888 | 0.7687 | 0.7861 | 0.7928 | 0.7686 | 0.7861 | **0.8002** |
| Breast Cancer Wisconsin (Original) | 0.9722 | 0.9751 | 0.9736 | 0.9634 | 0.9327 | 0.9678 | **0.9801** |
| Diabetes | 0.7552 | 0.7591 | 0.7773 | 0.7785 | 0.7760 | 0.7708 | **0.7834** |
| German Credit | 0.7330 | 0.7690 | 0.7630 | **0.7740** | 0.7650 | 0.7660 | 0.7730 |
| Hearts | 0.8222 | 0.8370 | 0.8370 | 0.8037 | 0.8370 | 0.8407 | **0.8519** |
| Ionosphere | 0.8547 | 0.9430 | 0.9544 | 0.8746 | 0.8832 | 0.9459 | **0.9601** |
| Sonar | 0.8510 | 0.9002 | 0.8702 | 0.8942 | 0.8221 | 0.9038 | **0.9150** |

the composition of each nucleotide at each position of DNA sequences. In the data sets of 4mC sites, there are always some noise samples (outliers) that will affect the hyperplane of classification. C-KHFIS is a C loss-based neuro-fuzzy systems that can effectively reduce the impact of outliers on classification. In Tables III and IV, it can be seen that the C loss function plays a good role in improving the performance of 4mC prediction. We first employ a kernel trick to solve higher-order fuzzy inference systems and propose kernelized higher-order fuzzy inference systems (KHFIS), which have a good ability to solve nonlinear fitting. Then, the extended KHFIS, which is called the correntropy induced loss based KHFIS (C-KHFIS), is developed to reduce the influence of noise samples on the model. From the results in Fig. 3 and Fig. 4, it can be found that C-loss plays a good role in reducing the influence of noise. The experimental results of 4mC show that C-KHFIS achieves the best ACC on A.thaliana (0.827), C.elegans (0.869), D.melanogaster (0.871), E.coli (0.953), G.pickeringi (0.904), and G.subterraneus (0.890). Moreover, C-KHFIS has good results on eight UCI data sets. The introduction of the kernel and C-loss method has greatly improved 1-FIS.

The 4mC site prediction is an important research direction in DNA sequence analysis. The accuracy of biological data affects the processing of the data. Therefore, high-quality 4mC site samples are the key to building high-precision prediction models. The size of the training set is another factor in developing new predictors. Therefore, fuzzy systems (C-KHFIS) are very suitable for solving the problem of small samples and noisy samples. In addition, the introduction of multimodal DNA information is also a method to improve the prediction performance of 4mC sites.

In fact, a variety of information sources and features can be used to describe an object. To further improve the recognition accuracy of 4mC, more features and information can be introduced. Future work can consider multi-kernel learning [44], [45], [46] and multi-view learning [32], [47] to effectively fuse features and further improve the prediction performance of the FIS model.

## References

[1] B. Manavalan, M. Hasan, S. Basith, V. Gosu, T. Shin, and G. Lee, "Empirical comparison and analysis of web-based dna n4-methylcytosine site prediction tools," *Mol Ther Nucleic Acids*, vol. 22, pp. 406–420, 2020.

[2] M. Hasan, W. Shoombuatong, H. Kurata, and B. Manavalan, "Critical evaluation of web-based dna n6-methyladenine site prediction tools," *Briefings in Functional Genomics*, vol. 20, no. 4, pp. 258–272, 2021.

[3] M. Hasan, S. Basith, M. Khatun, G. Lee, B. Manavalan, and H. Kurata, "Meta-i6ma: an interspecies predictor for identifying dna n6-methyladenine sites of plant genomes by exploiting informative features in an integrative machine-learning framework," *Briefings in Bioinformatics*, vol. 22, no. 3, p. bbaa202, 2021.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[5] W. Chen, H. Yang, P. Feng, H. Ding, and H. Lin, "iDNA4mC: identifying DNA N4-methylcytosine sites based on nucleotide chemical properties," *Bioinformatics*, vol. 33, no. 22, pp. 3518–3523, 07 2017. [Online]. Available: https://doi.org/10.1093/bioinformatics/btx479

[6] W. He, C. Jia, and Q. Zou, "4mcpred: machine learning methods for dna n4-methylcytosine sites prediction." *Bioinformatics*, vol. 35, no. 4, pp. 593–601, 2019.

[7] L. Wei, S. Luan, L. A. E. Nagai, R. Su, and Q. Zou, "Exploring sequence-based features for the improved prediction of DNA N4-methylcytosine sites in multiple species," *Bioinformatics*, vol. 35, no. 8, pp. 1326–1333, 09 2018. [Online]. Available: https://doi.org/10.1093/bioinformatics/bty824

[8] M. M. Hasan, B. Manavalan, M. S. Khatun, and H. Kurata, "i4mc-rose, a bioinformatics tool for the identification of dna n4-methylcytosine sites in the rosaceae genome," *International Journal of Biological Macromolecules*, vol. 157, pp. 752–758, 2020.

[9] M. Hasan, B. Manavalan, W. Shoombuatong, M. Khatun, and H. Kurata, "i4mc-mouse: Improved identification of dna n4-methylcytosine sites in the mouse genome using multiple encoding schemes," *Computational and Structural Biotechnology Journal*, vol. 18, pp. 906–912, 2020.

[10] H. Lv, F.-Y. Dao, D. Zhang, Z.-X. Guan, H. Yang, W. Su, M.-L. Liu, H. Ding, W. Chen, and H. Lin, "idna-ms: an integrated computational tool for detecting dna modification sites in multiple genomes," *iScience*, vol. 23, p. 100991, 2020.

[11] B. Manavalan, S. Basith, T. H. Shin, L. Wei, and G. Lee, "Meta-4mcpred: A sequence-based meta-predictor for accurate dna 4mc site prediction using effective feature representation," *Mol Ther Nucleic Acids*, vol. 16, pp. 733–744, 2019.

[12] Q. Tang, J. Kang, J. Yuan, H. Tang, X. Li, H. Lin, J. Huang, and W. Chen, "Dna4mc-lip: a linear integration method to identify n4-methylcytosine site in multiple species," *Bioinformatics*, vol. 36, pp. 733–744, 2020.

[13] J. Khanal, I. Nazari, H. Tayara, and K. T. Chong, "4mccnn: Identification of n4-methylcytosine sites in prokaryotes using convolutional neural network," *IEEE Access*, vol. 7, pp. 145 455–145 461, 2019.

[14] A. Wahab, O. Mahmoudi, J. Kim, and K. T. Chong, "Dnc4mc-deep: Identification and analysis of dna n4-methylcytosine sites based on different encoding schemes by using deep learning," *Cells*, vol. 9, no. 8, p. 1756, 2020.

[15] Q. Liu, J. Chen, Y. Wang, S. Li, C. Jia, J. Song, and F. Li, "DeepTorrent: a deep learning-based approach for predicting DNA N4-methylcytosine sites," *Briefings in Bioinformatics*, 07 2020, bbaa124. [Online]. Available: https://doi.org/10.1093/bib/bbaa124

[16] Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Transactions on Computers*, vol. C-26, no. 12, pp. 1182–1191, 1977.

[17] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man Cybern*, vol. SMC–15, no. 1, p. 116–132, 1985.

[18] C.-J. Lin, "An efficient immune-based symbiotic particle swarm optimization learning algorithm for tsk-type neuro-fuzzy networks design," *Fuzzy Sets and Systems*, vol. 159, no. 21, pp. 2890–2909, 2008.

[19] M. Azeem, M. Hanmandlu, and N. Ahmad, "Generalization of adaptive neuro-fuzzy inference systems," *IEEE Transactions on Neural Networks*, vol. 11, no. 6, pp. 1332–1346, 2000.

[20] 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.

[21] J.-H. Chiang and P.-Y. Hao, "Support vector learning mechanism for fuzzy rule-based modeling: a new approach," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 1, pp. 1–12, 2004.

[22] P. Xu, Z. Deng, C. Cui, T. Zhang, K.-S. Choi, S. Gu, J. Wang, and S. Wang, "Concise fuzzy system modeling integrating soft subspace clustering and sparse learning," *IEEE Transactions on Fuzzy Systems*, vol. 27, no. 11, pp. 2176–2189, 2019.

[23] Y. W. Kerk, K. M. Tay, and C. P. Lim, "Monotone fuzzy rule interpolation for practical modelling of the zero-order tsk fuzzy inference system," *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2021.

[24] Z. Deng, Y. Jiang, K.-S. Choi, F.-L. Chung, and S. Wang, "Knowledge-leverage-based tsk fuzzy system modeling," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 8, pp. 1200–1212, 2013.

[25] X. Gu, F.-L. Chung, and S. Wang, "Bayesian takagi-sugeno-kang fuzzy classifier," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1655–1671, 2017.

[26] B. Rezaee and M. F. Zarandi, "Data-driven fuzzy modeling for takagi-sugeno-kang fuzzy system," *Information Sciences*, vol. 180, no. 2, pp. 241–255, 2010.

[27] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in takagi–sugeno fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1795–1807, 2017.

[28] Z. Deng, Y. Cao, Q. Lou, K.-S. Choi, and S. Wang, "Monotonic relation-constrained takagi-sugeno-kang fuzzy system," *Information Sciences*, vol. 582, pp. 243–257, 2022.

[29] H. Yu, J. Lu, and G. Zhang, "Topology learning-based fuzzy random neural network for streaming data regression," *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2020.

[30] D. Wu and J. M. Mendel, "Patch learning," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 9, pp. 1996–2008, 2020.

[31] D. Wu, Y. Yuan, J. Huang, and Y. Tan, "Optimize tsk fuzzy systems for regression problems: Minibatch gradient descent with regularization, droprule, and adabound (mbgd-rda)," *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 5, pp. 1003–1015, 2020.

[32] Y. Jiang, Z. Deng, F.-L. Chung, G. Wang, P. Qian, K.-S. Choi, and S. Wang, "Recognition of epileptic eeg signals using a novel multiview tsk fuzzy system," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 1, pp. 3–20, 2017.

[33] Y. Jiang, D. Wu, Z. Deng, P. Qian, J. Wang, G. Wang, F.-L. Chung, K.-S. Choi, and S. Wang, "Seizure classification from eeg signals using transfer learning, semi-supervised learning and tsk fuzzy system," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 12, pp. 2270–2284, 2017.

[34] K. Wiktorowicz and T. Krzeszowski, "Training high-order takagi-sugeno fuzzy systems using batch least squares and particle swarm optimization," *International Journal of Fuzzy Systems*, vol. 22, p. 22–34, 2020.

[35] A. Singh, R. Pokharel, and J. Principe, "The c-loss function for pattern classification," *Pattern Recognition*, vol. 47, no. 1, pp. 441–453, 2014.

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

[37] Y. He, F. Wang, Y. Li, J. Qin, and B. Chen, "Robust matrix completion via maximum correntropy criterion and half-quadratic optimization," *IEEE Transactions on Signal Processing*, vol. 68, pp. 181–195, 2020.

[38] W. Chen, H. Lin, and K.-C. Chou, "Pseudo nucleotide composition or pseknc: an effective formulation for analyzing genomic sequences." *Molecular BioSystems*, vol. 11, no. 10, pp. 2620–2634, 2015.

[39] H. Lin, E.-Z. Deng, H. Ding, W. Chen, and K.-C. Chou, "ipro54-pseknc: a sequence-based predictor for identifying sigma-54 promoters in prokaryote with pseudo k-tuple nucleotide composition." *Molecular BioSystems*, vol. 42, no. 21, pp. 12 961–12 972, 2014.

[40] H. Lin, Z.-Y. Liang, H. Tang, and W. Chen, "Identifying sigma70 promoters with novel pseudo nucleotide composition." *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 16, no. 4, pp. 1316–1321, 2019.

[41] D. Dua and C. Graff, "Uci machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[42] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[43] L. Zhang, W. Zhou, P. Chang, J. Liu, Z. Yan, T. Wang, and F. Li, "Kernel sparse representation-based classifier," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1684–1695, 2012.

[44] Y. Ding, J. Tang, and G. Fei, "Identification of drug-side effect association via multiple information integration with centered kernel alignment," *Neurocomputing*, vol. 325, pp. 211–224, 2019.

[45] Y. Ding, J. Tang, and F. Guo, "Human protein subcellular localization identification via fuzzy model on kernelized neighborhood representation," *Applied Soft Computing*, vol. 96, p. 106596, 2020.

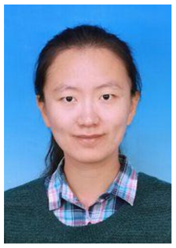
[46] H. Yang, Y. Ding, J. Tang, and F. Guo, "Drug-disease associations prediction via multiple kernel-based dual graph regularized least squares," *Applied Soft Computing*, vol. 112, p. 107811, 2021.

[47] Y. Ding, J. Tang, and F. Guo, "Identification of drug-target interactions via multi-view graph regularized link propagation model," *Neurocomputing*, vol. 461, pp. 618–631, 2021.

**Yijie Ding** received his Ph.D. degree from the School of Computer Science and Technology at Tianjin University in 2018. Currently, he is an associate professor in the Yangtze Delta Region Institute (Quzhou), University of Electronic Science and Technology of China. His research interests include bioinformatics and machine learning. Several related works have been published by Briefings in Bioinformatics, IEEE/ACM TCBB, IEEE JBHI, Information Sciences, Knowledge-Based Systems, Applied Soft Computing and Neurocomputing.

**Prayag Tiwari** (Member, IEEE) received his Ph.D degree from the University of Padova, Italy. He is currently working as a Postdoctoral Researcher at the Aalto University, Finland. Previously, he was working as a Marie Curie Researcher at the University of Padova, Italy. He also worked as a research assistant at the NUST "MISiS", Moscow, Russia. He has several publications in top journals and conferences including, Neural Networks, Information Fusion, IPM, IJCV, IEEE TNNLS, IEEE TFS, IEEE TII, IEEE JBHI, IEEE IOTJ, IEEE BIBM, ACM TOIT, CIKM, SIGIR, AAAI, etc. His research interests include Machine Learning, Deep Learning, Quantum Machine Learning, Information Retrieval, Healthcare, and IoT.

**Quan Zou** is a professor of Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China. He received his PH.D. from Harbin Institute of Technology, P.R.China in 2009, and worked at Xiamen University and Tianjin University from 2009 to 2018. His research is in the areas of bioinformatics, machine learning and parallel computing. Now he is putting the focus on protein classification, genome assembly, annotation and functional analysis from the next generation sequencing data with parallel computing methods. Several related works have been published by Briefings in Bioinformatics, Bioinformatics, PLOS Computational Biology and IEEE/ACM Transactions on Computational Biology and Bioinformatics. Google scholar showed that his more than 100 papers have been cited more than 4000 times. He is the Editor-in-Chief of Current Bioinformatics. He is also a reviewer for many impacted journals and NSFC(National Natural Science Foundation of China).



**Fei Guo** received the bachelor degree and the PhD degree from Shandong University, in 2007 and 2012. She was a postdoctoral fellow at City University of Hong Kong between 2012 and 2013. She is currently a professor in the School of Computer Science and Engineering, Central South University. Her research interests include bioinformatics and computational biology.

**Prof. Hari Mohan Pandey** (Senior Member, IEEE) received the B.Tech. degree from Uttar Pradesh Technical University, India, the M.Tech. degree from the Narsee Monjee Institute of Management Studies, India, and the Ph.D. degree computer science and engineering from the Amity University, India. He worked as a Postdoctoral Research Fellow with the Middlesex University, London, U.K. He also worked on a European Commission project – Dream4car under H2020. He is a Senior Lecturer with the Department of Computer Science, Edge Hill University, Lancashire, U.K. He is 966 specialized in computer science and engineering and his research area includes, artificial intelligence, soft computing, natural language processing, language acquisition, machine learning, and deep learning.