Neural Computing and Applications
https://doi.org/10.1007/s00521-018-3346-z

ORIGINAL ARTICLE

Protein fold recognition using Deep Kernelized Extreme Learning Machine and linear discriminant analysis

Wisam Ibrahim[1] · Mohammad Saniee Abadeh[1]

Received: 24 December 2016 / Accepted: 5 January 2018
© The Natural Computing Applications Forum 2018

Abstract
Protein fold recognition is considered as an essential step in determining the tertiary structure of proteins in bioinformatics. The most complex challenge in the protein folding problem is the high dimensionality of feature vectors and the diversity of the protein fold classes. In this paper, two frameworks are proposed to solve this problem. The main components in the first two-level framework are Deep Kernelized Extreme Learning Machine (DKELM) and linear discriminant analysis. Second framework consists of three levels. In the first level, the dataset is initialized to be used in the next level. In the second level, OVADKELM and OVODKELM are independently employed to extract four and six new features, respectively, which are added into the basic datasets in the third level. DKELM is applied in the third level as a final classifier to classify the instances into folds. The proposed frameworks are implemented on DD and TG datasets which are considered as SCOP datasets. The experimental results indicate that proposed methods could improve the classification accuracy in both datasets.

Keywords Protein fold recognition · Deep learning · Extreme learning machine · Linear discriminant analysis

1 Introduction

Proteins are macromolecules which are built of small blocks called *amino acids*. There are about 20 different amino acids connected together in a special order to create unlimited sequences. One of the fundamental problems related to proteins is the fold recognition. Protein fold recognition is obtaining the tertiary structure of the proteins from the amino acid sequences without relying on the sequence similarities [1]. Protein tertiary structure prediction is very useful in many areas of biomedicine such as approximate family assignments and precise drug screening [2], it also helps to understand the functions of protein [3].

Many traditional methods have been used in tertiary structure prediction such as X-ray-crystallography and Nuclear Magnetic Resonance (NMR). However, these methods are expensive and time-consuming. Therefore, the computational methods have been implemented. The computational methods are faster, cheaper and more efficient [4]. Some of these computational methods such as Classification Trees (CTs), Multilayer Perceptron (MLP), Probabilistic Neural Networks (PNN), Support Vector Machines (SVM) and Nearest Neighbor (NN) have been used as independent or fusion classifiers to recognize the protein folds.

In this paper, two frameworks are proposed to recognize the protein folds. The first framework contains two classifiers which are employed in two sequential levels. In the first level, a deep learning classifier (Deep Kernelized Extreme Learning Machine DKELM) classifies the instances of the structural classes into four classes (all α, all β, α/β and $\alpha + \beta$) to extract four new features. Then, the extracted features are added into the train and test datasets which are used in the second level. The second classifier (linear discriminant analysis LDA) is employed in the second level to predict the protein folds. The main component of the second proposed framework is Deep Kernelized Extreme Learning Machine (DKELM). In the first level of this framework, the dataset is initialized to be used in the next level. In the second level, One-Versus-All

✉ Mohammad Saniee Abadeh
saniee@modares.ac.ir

[1] Faculty of Electrical and Computer Engineering, Tarbiat Modares University, Tehran, Iran

Published online: 15 January 2018

DKELM (OVADKELM) and One-Versus-One DKELM (OVODKELM) are independently employed on the dataset with structural classes (all α, all β, α/β and $\alpha + \beta$) to extract four and six new features, respectively, which are added into the original dataset in the third level. DKELM is applied in the third level as a final classifier to classify the proteins into folds.

The rest of this paper is organized as follows: in the next section, the related works about the protein structure prediction are briefly introduced. Section 3 describes the datasets and feature vectors which are used in this paper. In Sect. 4, the customized methods are explained. The proposed methods are discussed in Sect. 5. The experimental results are reported in Sect. 6. Final section offers the conclusions and future works.

2 Related works

The machine learning algorithms are carried out as computational methods which are used in the protein structure prediction. Ding and Dubchak [1] used SVM and Multilayer Feed-Forward Neural Network with unique One-Versus-Others and All-Versus-All methods and applied them on Structural Classification of Protein (SCOP) datasets. Then, Pal and Chakraborty [5] employed SVM and Radial Basis Function (RBF) networks on seven datasets. Five of the datasets were generated with some new features and two of them from SCOP datasets. A two-level framework is proposed by Cheng and Baldi [6] to predict the protein folds. In the first level, five categories of pairwise features are extracted and in the second level, SVM is applied to predict the folds from these features. Also, Qu et al. [7] proposed a subsystem of two levels called multi-model Back Propagation Neural Network (MMBP) and combined it with multi-model SVM-MMS to make a system with three stages to predict the protein secondary structure. Abbasi et al. [8] proposed a framework included three main methods, Fuzzy Resource Allocating Network (FRAN) is applied on four feature vectors of SCOP, and RBF networks optimized by Particle Swarm Optimization (PSO) are employed on one feature vector of SCOP, whereas k-NN is applied on the rest feature vector.

The classifiers could be combined in a fusion system to improve the efficiency that is achieved by a single classifier, because each classifier can represent different useful aspects of the problem, while other classifiers cannot [9]. Chmielnicki and Stapor [10] combined two classifiers to demonstrate that the ensemble system in protein classification problem achieves accuracy better than the two classifiers separately. Therefore, SVM is employed as discriminative approach and Regularized Discriminant Analysis (RDA) as a generative approach to take advantages of both approaches. Chen et al. [11] presented two fusion systems of three classifiers including k-NN, Class Center and Nearest Neighbor (CCNN), and PNN using the datasets in SCOP. The first system combines homogeneous classifiers and votes the results based on weights of the classifiers. The second system combines heterogeneous classifiers, and the best results are involved in the voting system. PSO is employed to optimize the weights of the classifiers. Then, Hashemi et al. [12] applied two fusion methods Bayesian fusion and voting System for two classifiers MLP and RBF networks separately. The Bayesian fusion works better than voting System with both classifiers. Nanni [13] applied three ensemble methods Best subspace BS, Random subspace RS and Bagging BA for two classifiers Fisher Linear Classifier FLC and K-Local Hyperplane Nearest Neighbor (HKNN) separately. Two datasets are used from SCOP. Jazebi et al. [9] used weighted voting and Ordered Weighted Averaging (OWA) systems as fusion methods to combine six PNN classifiers which are optimized by PSO. The systems are applied on the six feature vectors of SCOP. OWA has achieved accuracy better than the weighted voting system.

To find more solutions for protein fold recognition (PFR) problem, researchers have proposed various methods to extract feature vectors from protein sequences. These methods are named descriptors or extraction techniques. Sharma et al. [14] have proposed a novel extraction technique named PSSM Bigram to extract feature vector from amino acids. Instead of representing the given protein by its original primary sequence, they have used Position Specific Scoring Matrices (PSSM) of protein sequences to avoid zero components in the resulting Bigram feature vector. Paliwal et al. [15] have proposed a trigram extraction technique based on three dimensional structure of the protein sequence. They have used PSSM linear probabilities of a given protein sequence to compute the probabilities of individual trigrams to form 3-dimensional probability matrix. The resulting feature vector consists of $20 \times 20 \times 20 = 8000$ features. Instead of using PSSM, Lyons et al. [16] have used Hidden Markov Model (HMM) profile to develop Bigram and Trigram extraction techniques. HMM profile is calculated by a profile–profile sequence alignment technique, named HHblits. The HMM profile has been shown to be a more effective approach for remote homology detection compared to PSSM. However, Lyons et al. [17] also applied HMM–HMM alignment of protein sequence from HHblits to extract profile HMM (PHMM) matrix. Then, they computed the distance between respective PHMM matrices to find alignment path using dynamic programming. This distance has been used to recognize the protein folds, if the distance matrix between two proteins is low, they belong to the same fold otherwise they are not. In addition to PSSM, Paliwal et al.

[18] have used Secondary Structure Prediction Matrix (SSPM) to find the probabilities of amino acid pairs (AAP). The information from the PSSM matrix and SSPM have been combined to extract relevant and useful knowledge for protein fold recognition.

Huang et al. [19], and Aram and Charkari [20] have proposed two-level hierarchical frameworks using the structural classes in SCOP datasets (all α, all β, α/β, and $\alpha + \beta$) in the first level, while in the second level the protein samples are classified in 27 folds. Huang et al. [19] used five independent classifiers of SVM and MLP. One classifier is applied in the first level to classify the protein features into four classes, and four classifiers are employed in the next level to classify the features resulting from the previous level into 27 folds. The classifiers in the two levels work independently; therefore, the system does not contain any voting mechanisms. However, this allows the incorrect classification in first level affects badly on the results of next level. While, Aram and Charkari [20] applied Rotation Forest (RF), SVM and MLP as three classifiers in the first level and used a voting system to exclude the weak results of the classification in first level and extract a useful new feature which is added into the datasets in next level. Then, another classifier is employed to achieve the final results.

3 Datasets and feature vectors

To be comparable with previous research works, we test our methods on the widely used datasets which are introduced by Ding and Dubchack (DD) [1] and Taguchi and Gromiha (TG) [21].

3.1 DD dataset

The training data in DD dataset contain 313 sequences whose similarity less than 35%, while the test data consist of 385 sequences whose similarity less than 40%. This dataset contains most populated 27 folds which represent all major structural classes (α, β, α/β, $\alpha + \beta$). The folds' names with their indexes and corresponding number of proteins in train and test datasets are described in Table 1.

3.2 DD feature vectors

Dubchak et al. [22] applied three descriptors to extract six feature vectors from the protein sequences. The descriptors are composition, transition and distribution. The extracted feature vectors are amino acids composition (C), predicted secondary structure (S), hydrophobicity (H), polarity (P), normalized van der Waals volume (V) and polarizability (Z). Feature vector C consists of 20 features while each vector of the remaining ones consists of 21 features. The independent and combined feature vectors and the number of their features are shown in Table 2.

3.3 TG dataset

In addition to DD dataset, we have tested our methods on TG dataset which is introduced by Taguchi and Gromiha [21]. TG dataset contains 30 folds represent all major structural classes (α, β, α/β, $\alpha + \beta$). The folds' names and corresponding number of proteins in TG dataset are described in Table 3. There are 1612 amino acid sequence in TG dataset. To create feature vectors from these sequences, we have used Bigram descriptor which has been used in [14]. In the following, we will explain Bigram descriptor.

Let P be the matrix representing PSSM of a given protein. The matrix P will have L rows and 20 columns (where L is the length of the primary sequence). Its element at ith-row and jth-column is denoted by $P_{i,j}$ which can be interpreted as the relative probability of jth amino acid at the ith location of the primary protein sequence ($\sum_{j=1}^{20} P_{i\cdot j} = 1,\ i = 1, 2, \ldots, L$). The frequency of occurrence of transition from mth amino acid to nth amino acid is computed as follows in Eq. 1:

$$B_{mn} = \sum_{i=1}^{L-1} P_{i\cdot m} P_{i+1\cdot n} : \quad 1 \le m \le 20, \quad 1 \le n \le 20 \tag{1}$$

This equation gives 400 frequencies of occurrences B_{mn}; $m = 1, \ldots, 20$; $n = 1, \ldots, 20$ for 400 Bigram transitions. We call the matrix B as the Bigram occurrence matrix, and its 400 elements define the Bigram feature vector F:

$$F = [B_{1.1} \cdot B_{1.2} \ldots B_{1.20} \cdot B_{2.1} \cdot B_{2.2} \ldots B_{2.20} \ldots B_{20.1} \cdot B_{20.2} \ldots B_{20.20}]^T \tag{2}$$

where the superscript T indicates the transpose of the vector. The method of feature vector extraction is shown in Fig. 1.

Since in the computation of feature vector F all the information of PSSM probability has been used, intuitively F contains more useful information for protein fold recognition task than computing Bigram directly from the protein sequence.

4 Customized methods for protein fold recognition

In this paper, two methods are employed as main components in the proposed frameworks, Deep Kernelized Extreme Learning Machine (DKELM) and linear

Table 1 Proteins' folds and structural classes in Ding and Dubchak DD dataset

Fold	Index	#Train data	#Test data
α			
Globin-like	1	13	6
Cytochrome c	3	7	9
DNA-binding 3-helical bundle	4	12	20
4-Helical up-and-down bundle	7	7	8
4-Helical cytokines	9	9	9
Alpha; EF hand	11	7	9
β			
Immunoglobulin-like β-sandwich	20	30	44
Cupredoxins	23	9	12
Viral coat and capsid proteins	26	16	13
ConA-like lectins/glucanases	30	7	6
SH3-like barrel	31	8	8
OB-fold	32	13	19
Trefoil	33	8	4
Trypsin-like serine proteases	35	9	4
Lipocalins	39	9	7
α/β			
(TIM)-barrel	46	29	48
FAD(also NAD)-binding motif	47	11	12
Flavodoxin-like	48	11	13
NAD(P)-binding Rossmann-fold	51	13	27
P-loop containing nucleotide	54	10	12
Thioredoxin-like	57	9	8
Ribonuclease H-like motif	59	10	14
Hydrolases	62	11	7
Periplasmic binding protein-like	69	11	4
$\alpha + \beta$			
β-grasp	72	7	8
Ferredoxin-like	87	13	27
Small inhibitors, toxins, lectins	110	14	27
Total		313	385

discriminant analysis (LDA). In the following, we will present a brief review about them.

4.1 Extreme learning machine (ELM)

Extreme learning machine (ELM) is an advanced Single Layer Forward Network (SLNF) algorithm proposed by Huang et al. [23]. Unlike the traditional SLFNs, ELM algorithm randomly chooses the weights between the input layer and hidden layer then computes the output weights based on the least-squares solutions [24]. The basic ELM network consists of input, hidden and output layers. The neurons in these layers are connected with each other in a way shown in Fig. 2.

ELM can be explained as follows.

Given a training set contains N pairs of tuples $(x_i,\ t_i)$, where $x_i \in R^n$ is the input instance, and $t_i \in R^m$ is the class label corresponding to the input x_i, ELM equation can be summarized as follows:

$$\boldsymbol{H\beta} = \boldsymbol{T} \tag{3}$$

H is the hidden layer output matrix, written as follows:

$$\begin{aligned} \boldsymbol{H} &= \begin{bmatrix} \boldsymbol{h}(\boldsymbol{x}_1) \\ \vdots \\ h(\boldsymbol{x_n}) \end{bmatrix}_{\boldsymbol{n\times L}} \\ &= \begin{bmatrix} \boldsymbol{G}(\boldsymbol{a}_1,\boldsymbol{b}_1,\boldsymbol{x}_1) & \cdots & \boldsymbol{G}(\boldsymbol{a_L},\boldsymbol{b_L},\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \boldsymbol{G}(\boldsymbol{a}_1,\boldsymbol{b}_1,\boldsymbol{x_n}) & \cdots & \boldsymbol{G}(\boldsymbol{a_L},\boldsymbol{b_L},\boldsymbol{x_n}) \end{bmatrix}_{\boldsymbol{n\times L}} \end{aligned} \tag{4}$$

Table 2 Independent and combined feature vectors in Ding and Dubchak DD dataset

Symbol	Parameter	#features
C	Composition	20
S	Predicted secondary structure	21
H	Hydrophobicity	21
P	Polarity	21
V	Normalized van der Waals volume	21
Z	Polarizability	21
CS	Combination of C + S	41
CSH	Combination of C + S+H	62
CSHP	Combination of C + S+H + P	83
CSHPV	Combination of C + S+H + P+V	104
CSHPVZ	Combination of C + S+H + P+V + Z	125

where $G(a_i, b_i, x_j)$ is the activation function of the ith hidden node, and b_i is the threshold of the ith hidden neuron. $\boldsymbol{\beta}$ is the weight vector connecting the ith hidden neuron and the output neurons, written as:

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1^T \\ \vdots \\ \boldsymbol{\beta}_L^T \end{bmatrix}_{L \times m} \tag{5}$$

is the corresponding class label vector, written as:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t}_1^T \\ \vdots \\ \boldsymbol{t}_n^T \end{bmatrix}_{n \times m} \tag{6}$$

According to Eq. (3) the smallest norm least-squares solution $\hat{\beta}$ of this linear system is:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^{\dagger}\boldsymbol{T} \tag{7}$$

where $\boldsymbol{H}^{\dagger}$ is the Moore–Penrose generalized inverse of matrix $\boldsymbol{H}$ [25].

$\boldsymbol{H}^{\dagger}$ can be written as follows:

$$\boldsymbol{H}^{\dagger} = \boldsymbol{H}^{\boldsymbol{T}}\left(\frac{1}{\boldsymbol{C}} + \boldsymbol{H}\boldsymbol{H}^{\boldsymbol{T}}\right)^{-1} \tag{8}$$

In Eq. (8) C is the user specified parameter named regulation coefficient that used in ELM.

The main steps of ELM can be summarized in Algorithm 1:

Algorithm 1. Basic ELM

Given a training set $N = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, \ldots, N\}$, activation function $G(x)$, and hidden neuron number L

(1) Randomly determine hidden node parameters matrix $W \in R^{n \times m}$

(2) Calculate the hidden layer output matrix $H=G(W,X)$;

(3) Find the smallest norm least-squares solution $\hat{\beta} = H^{\dagger} T$

(4) Calculate the output decision function: $f(x) = h(x)\hat{\beta} = h(x)H^{\dagger} T$

Using the least-square solution, ELM can achieve minimum training error, smallest norm of weights and best generalization performance. Comparing with the traditional SLFNs, ELM runs faster and provides better generalization performance [23].

In this paper, we have kernelized the basic ELM and stacked multiple layers of Kernelized ELM to propose a deep model such as Deep representations via ELM (DrELM) which was presented by Yu et al. [26].

4.2 Linear discriminant analysis (LDA)

Linear discriminant analysis is a supervised method used as dimensionality reduction technique to reduce the problem dimensions with keeping the largest possible amount of class discriminatory information [27].

LDA has been suggested based on Fisher linear discriminant that seeks to find the best discriminant projection of the samples [28]. Considering that the distance between the projected means is not a sufficient measure in discriminant, Fisher takes into account the deviation within the classes. Therefore, he proposed a measure called Fisher criterion $J(w)$ that is the ratio of between-class scatter matrix (S_B) and within-class scatter matrix (S_W). The discriminant will be better and more efficient by maximizing *between-class* distance and minimizing *within-class* distance. In the proposed method, LDA will be applied as a single classifier to classify the samples in folds as final classification outputs.

5 Proposed methods

In this section, the proposed solutions for protein fold recognition problem are presented. We have proposed two frameworks for this problem. The first framework uses two methods, Deep Kernelized Extreme Learning Machine (DKELM) and linear discriminant analysis (LDA), in two levels. The second framework consists of three levels and applies DKELM in two levels to recognize the folds of the proteins. In the following, the two proposed frameworks are discussed:

Table 3 Proteins' folds Taguchi and Gromiha TG dataset

ID	Fold	Fold description	Number
All-α			
1	a.3	Cytochrome *C*	25
2	a.4	DNA/RNA binding 3-helical bundle	103
3	a.24	Four helical up-and-down bundle	26
4	a.39	EF hand-like fold	25
5	a.60	SAM domain-like	26
6	a.118	α-α Superhelix	47
All-*β*			
7	b.1	Immunoglobulin-like β-sandwich	173
8	b.2	Toxin/transcription factors/cytochrome f	28
9	b.6	Cupredoxin-like	30
10	b.18	Galactose-binding domain-like	25
11	b.29	Concanavalin A-like lectins/glucanases	26
12	b.34	SH3-like barrel	42
13	b.40	OB-fold	78
14	b.82	Double-stranded α-helix	34
15	b.121	Nucleoplasmin-like	42
α/*β*			
16	c.1	TIM barrel	145
17	c.2	NAD(P)-binding Rossmann-fold domains	77
18	c.3	FAD/NAD(P)-binding domain	31
19	c.23	Flavodoxin-like	55
20	c.26	Adenine nucleotide a hydrolase-like	34
21	c.37	P-loop containing nucleoside triphosphate hydrolases	95
22	c.47	Thioredoxin fold	32
23	c.55	Ribonuclease H-like motif	49
24	c.66	S-Adenosyl-L-methionine-dependent methyltransferases	34
25	c.69	α/β-Hydrolases	37
α + *β*			
26	d.15	β-Grasp, ubiquitin-like	42
27	d.17	Cystatin-like	25
28	d.58	Ferredoxin-like	118
29	d.3	Knottins	80
30	d.41	Rubredoxin-like	28

5.1 DKELM-LDA

As mentioned in Sect. 2, some researchers such as [19, 20] have exploited the structural information of proteins to improve the classification rate. It is proven that the use of structural information is very effective in improvement of the performance of protein fold detection. Accordingly, we have also used structural information in our methods, but in the more effective way. First, let us review the two mentioned methods and their drawbacks.

As mentioned, Aram and Charkari [20] and Huang et al. [19] proposed two-level hierarchical approaches, Two-Layer Classification Framework (TLCF) and Hierarchical Learning Architecture (HLA), respectively. They have used the structural classes in SCOP datasets in the first level, while in the second level the protein samples are classified in 27 folds. In HLA, objects that are classified incorrectly in the first level cannot be recovered in the next level. This weakness of HLA has been partially covered in TLCF. In TLCF, the structural classes of proteins are classified using fusion of three different classifiers in the first level [20]. Then, the classification results of first level directly added into datasets for training in the second level. It should be noted that results of the first level in TLCF are based on Boolean logic, means that only one of the structural classes is considered as a final result for each proteins. Such Boolean assignment leads to performance reduction of classification models in the case of noisy data

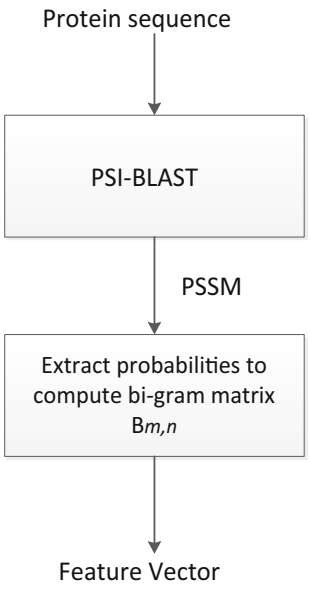

Fig. 1 The method of feature vector extraction using PSSM and Bigram

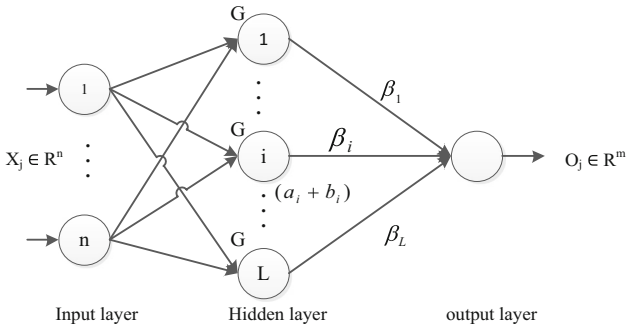

Fig. 2 Basic ELM

(such as protein data). To resolve this problem in the proposed method DKELM-LDA, the rate of belonging to each of the classes is considered. The framework of proposed method DKELM-LDA to predict the fold of an unknown sample is shown in Fig. 3.

As seen in Fig. 3, such as TLCF and HLA, a two-level architecture is used for protein folding. In the first level a deep learning classifier named DKELM (Deep Kernelized Extreme Learning Machine) classifies the instances of the structural classes (α, β, α/β, $\alpha + \beta$) into four classes to extract four new features. The four extracted features are then added into original datasets that are used in the second level. In the second level, LDA is employed to predict the protein folds.

The main difference between the proposed method and TLCF is on how to make the results of the first level. In TLCF only one feature has been extracted from first level, but in proposed method we extract four new features according to four structural classes. In fact, new features represent the outputs of DKELM to predict the structural classes of proteins.

In the first level of proposed architecture, we have proposed a stacked model consisting of multiple layers of Kernelized ELM named Deep Kernelized ELM. The outputs of each layer in DKELM are used as inputs of the next layer KELM. The classification results of KELM are computed by kernel based ELM with RBF kernel, and then a random projection of the classification results integrated into the original features to generate the next layer input. The pseudocode in Algorithm 2 presents DKELM.

Algorithm 2. Deep Kernelized ELM

Input: The training set $X = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1, 2, \ldots, N\}$
The number of layers (model depth k)
The weight parameter α
Output: The prediction function of DKELM

(1) Initial $X_1 = X$;
(2) For i from 1 to k
(3) Apply KELM for X_i to Compute the classification result O_i
(4) Randomly generate projection weight $W_i \in R^{n \times m}$
(5) Normalizing the W_i between [0,1]
(6) Compute the training data for the next layer by $X_{i+1}=X_i+ \alpha O_i W_i$
(7) End
(8) Compute the output function $f_k(X)$ by Eq. (7)

As shown in Algorithm 2, we have used KELM as a base building block. If the feature mapping $h(x)$ in ELM (mentioned in Sect. 4.1) is unknown, by replacing HH^T with kernel matrix Ω_{ELM}, output function of Kernel ELM can be written by

$$f(X) = [k(x, x_1), \ldots, k(x, x_n)](1/C + \mathbf{\Omega}_{ELM})^{-1} T \tag{9}$$

where

$$\Omega_{ELMi,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) \tag{10}$$

In the proposed method $k(\mathbf{x}_i, \mathbf{x}_j)$ is the RBF kernel function that is as follows:

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\Upsilon ||\boldsymbol{x}_i - \boldsymbol{x}_j||^2) \tag{11}$$

where $\| x_i - x_j \|$ is Euclidean distance between the two feature vectors and Υ is kernel parameter that can be any constant number. Note that unlike the ELM, Kernelized ELM takes no consideration to the feature mapping function and the number of hidden layer nodes. It concerns only the kernel functions and the training data [29].

In the proposed method DKELM-LDA, the outputs of first level are calculated by output function $f(x)$. The outputs of the first level are added as new features into the original datasets in the second level, and this new dataset is then delivered to LDA classifier for predicting the folding classes.

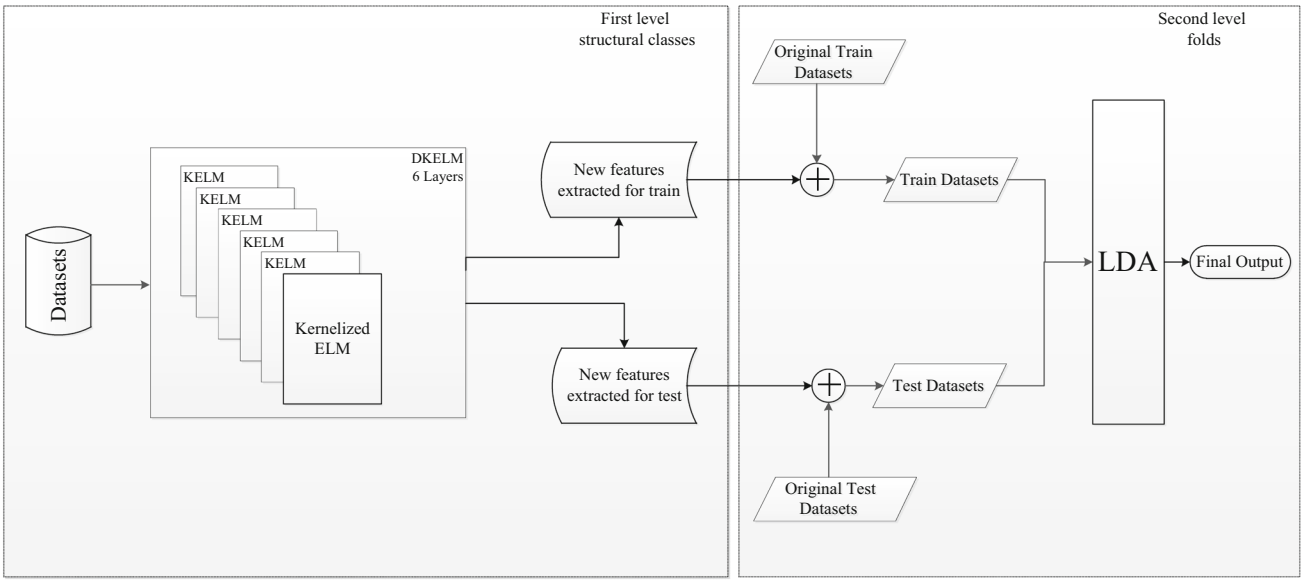

Fig. 3 The proposed framework DKELM-LDA

5.2 OVAOVO-DKELM

As seen in Fig. 4, a three-level architecture is used for protein fold recognition. In the first level, the dataset is initialized to be applicable in the next level. The number of classes in the second level (structural classes) is four classes $k = 4$, (α, β, α/β, $\alpha + \beta$). For this reason, the number of classifiers in the One-Versus-All DKELM (OVADKELM) model equals to $r = k = 4$. But the number of classifiers for the One-Versus-One DKELM (OVODKELM) model equals to $r = k\,(k-1)/2 = 6$. Data should also be prepared to be used by OVODKELM and OVADKELM. The classifiers of the OVODKELM model are learned with two classes (l, m), instances of class l are labeled with $+1$, but instances of class m are labeled with (0). Remaining instances are ignored.

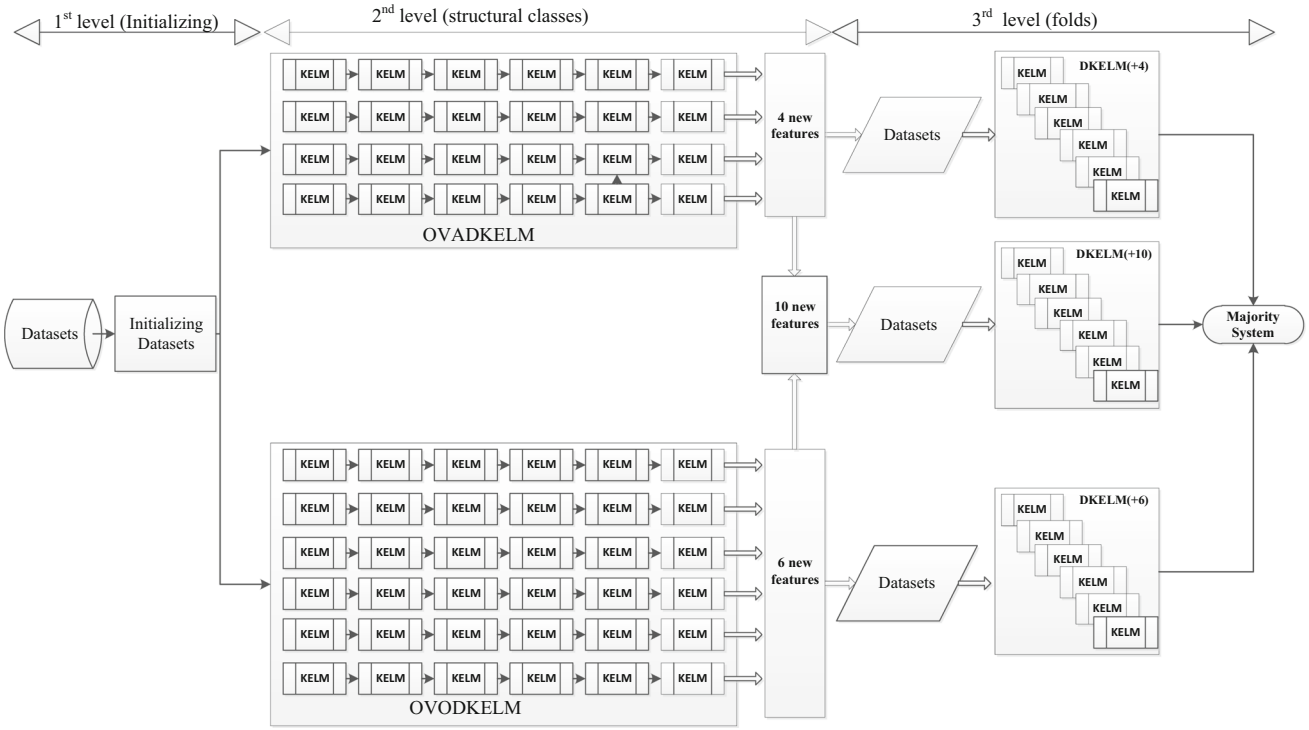

Fig. 4 The proposed framework OVAOVO-DKELM

However, the classifiers in the OVADKELM model are learned with a class (l), instances of class l are labeled with + 1, but other instances are labeled with (0). In the second level, the deep learning classifier DKELM (Deep Kernelized Extreme Learning Machine) classifies the instances of the structural classes according to four classes to extract new features. The four classifiers of OVADKELM model extract four new features, and the six classifiers of OVODKELM model extract six features. The extracted features are then added into the original datasets that are used in the third level. In the third level, three DKELM classifiers are employed to predict the protein folds. The first DKELM classifier is applied on the dataset after adding the four features which are extracted by OVADKELM model, so it will be named DKELM(+ 4) classifier. The second classifier is employed on the dataset after adding the six features which are extracted by OVODKELM model, so it will be named DKELM(+ 6) classifier. The third classifier is used to classify the dataset after adding all features which are extracted by both models (OVADKELM, OVODKELM), i.e., (4 + 6 = 10 features), so it will be named DKELM(+ 10) classifier. Finally, the majority system selects the best results of the three classifiers [DKELM(+ 4), DKELM(+ 6), DKELM(+ 10)].

6 Experimental results

As evaluation criteria, we have used the total accuracy, sensitivity, specificity and Matthew's correlation coefficient (MCC) to evaluate the performance of proposed methods. Sensitivity shows the ratio of correctly classified samples to the total number of test samples for each class which are classified as correct samples [30]. Sensitivity is defined as the following equation:

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

Specificity measures the ratio of correctly rejected samples to the total number of rejected test samples [30]. It is defined as follows:

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{13}$$

Mathew's correlation coefficient (MCC) reflects the stability of the prediction method [31]. MCC is calculated as follows:

$$\text{MCC} = \frac{(\text{TP} \times \text{TN}) - (\text{FP} \times \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{14}$$

TP, TN, FP and FN represent true positive, true negative, false positive and false negative, respectively.

Total accuracy is the popular metric to measure the protein fold recognizer's performance. This metric can be explained as follows.

Suppose that a trained classifier F is applied to classify test samples $S = s_1, \ldots, s_N$ into classes $C = c_1,\ldots,c_L$. If the total number of test samples is N and the total number of correctly classified test samples is M, then the total accuracy of the classifier F is defined as:

$$\begin{aligned}\text{Total Accuracy}(F) &= \frac{M}{N} \times 100 \\ &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \times 100\end{aligned} \tag{15}$$

As shown in Table 4, we have obtained the best accuracy for all independent feature vectors of DD datasets except for H and P feature vectors. In addition, we have combined the feature sets in five datasets CS, CSH, CSHP, CSHPV, CSHPVZ (shown in Table 2) and applied our methods on them. As it is clear from Table 5, our results are the best results for all combined feature sets. The experimental results demonstrate that our methods can improve the prediction accuracy by 1.3–5.4% for independent feature sets and by 3.3–8.8% for combined feature sets. It is necessary to mention that the most important challenge in Ding and Dubchak DD datasets is the low number of training instances and the large number of classes. This challenge makes the accuracy improvement in protein fold recognition problem a very complicated task.

As seen in Figs. 5, 6, 7 and 8 for datasets DD, proposed frameworks have obtained specificity between 98 and 98.45%, while the sensitivity is between 44 and 55.7%. Finally, the MCC value which was obtained is between 43 and 54.7%.

To evaluate the proposed frameworks on TG dataset, we have used the k-cross-validation method as shown in Table 6. By comparing our methods with Bigram [15] (the same feature extractor that has been used in this paper) and other methods on TG dataset, the classifier DKELM(+ 4) has obtained the best accuracy for $k = 5$, while classifier DKELM(+ 6) has obtained the best accuracy for $k = 8$. But for other validations $k = 6, 7, 9, 10$, the best accuracy was obtained by classifier DKELM(+ 10) after adding the ten new features in the third level of the proposed framework OVAOVO-DKELM. However, both the proposed frameworks DKELM-LDA and OVAOVO-DKELM could obtain the accuracy better than Bigram [15], Trigram [15] and Alignment method [30] for all the validations.

As seen in Figs. 9, 10, 11 and 12 for dataset TG, proposed methods have obtained specificity between 98.9 and 99.2%, while the sensitivity is between 62 and 69%. Finally, the MCC value which was obtained is between 67 and 71.5%.

Table 4 Comparison between proposed methods and other related studies on six independent feature sets of Ding and Dubchak DD dataset

Method	Features					
	C	*S*	*H*	*P*	*V*	*Z*
uOvO-SVM [1]	49.40	–	–	–	–	–
AvA-SVM [1]	44.90	–	–	–	–	–
CCNN[a] [11]	42.08	35.84	32.21	27.01	33.77	29.87
HLA[b]–RBFN [19]	44.90	–	–	–	–	–
MOF[c] [32]	44.50	38.50	37.50	35.50	36.50	35.00
Fusion method [9]	49.10	39.70	35.50	36.80	36.00	33.70
Taxonometric approach [33]	45.71	39.22	35.32	33.77	34.81	32.73
Hyperfold approach [34]	–	44.10	36.03	35.51	31.59	27.15
FRAN[d] [8]	51.43	34.55	40.00	38.70	41.04	38.96
TLCF [20]	58.18	50.90	**51.42**	**51.42**	51.42	51.16
DKELM-LDA (proposed method)	61.30	53.76	48.57	48.57	52.72	55.58
DKELM(+ 4) (proposed method)	61.66	54.03	51.17	50.65	**56.64**	55.68
DKELM(+ 6) (proposed method)	**62.42**	53.51	50.39	50.91	55.59	**56.62**
DKELM(+ 10) (proposed method)	62.15	**54.64**	51.34	50.93	55.82	56.12
Majority Results of OVAOVO-DKELM	**62.42**	**54.64**	51.34	50.93	**56.64**	**56.62**

Best result(s) in each column are in bold

[a]Class Center and Nearest Neighbor

[b]Hierarchical Learning Architecture

[c]Multi-Objective Feature Analysis

[d]Fuzzy Resource Allocating Network

Table 5 Comparison between proposed methods and other related studies on five combined feature sets of Ding and Dubchak DD dataset

Method	Features				
	CS	CSH	CSHP	CSHPV	CSHPVZ
uOvO-SVM [1]	48.6	51.1	49.4	50.9	49.6
AvA-SVM [1]	52.1	56.0	56.5	55.5	53.9
OvO-NN [1]	36.8	40.6	41.1	41.2	41.8
OvO-SVM [1]	43.2	45.2	43.2	44.8	44.9
CCNN [11]	45.19	48.31	48.57	47.79	48.05
HLA–RBFN [19]	53.8	53.3	54.3	55.3	56.4
HLA–MLP [19]	48.6	47.5	43.2	43.6	44.7
HLA-GRNN[a] [19]	–	–	–	–	45.2
HLA-SVM [19]	–	–	–	–	53.2
MOF [32]	–	–	–	–	55.0
DKELM-LDA (proposed method)	60.51	60	60.26	58.96	59.70
DKELM(+ 4) (proposed method)	61.34	61.33	**61.72**	60.57	60.27
DKELM(+ 6) (proposed method)	62.37	62.37	60.93	**61.09**	**60.62**
DKELM(+ 10) (proposed method)	**62.66**	**62.89**	61.07	60.95	60.50
Majority results of OVAOVO-DKELM	**62.66**	**62.89**	**61.72**	**61.09**	**60.62**

Best result(s) in each column are in bold

[a]General Regression Neural Network

7 Conclusions and future works

In this paper, we have proposed DKELM-LDA and OVAOVO-DKELM as classification frameworks to recognize protein folds. The proposed methods have been applied on Ding and Dubchak DD and Taguchi and Gromiha TG datasets. In the proposed framework DKELM-LDA, four features have been extracted in the first level. These new features have been used by LDA to recognize the protein folds in the second level. Also in the proposed

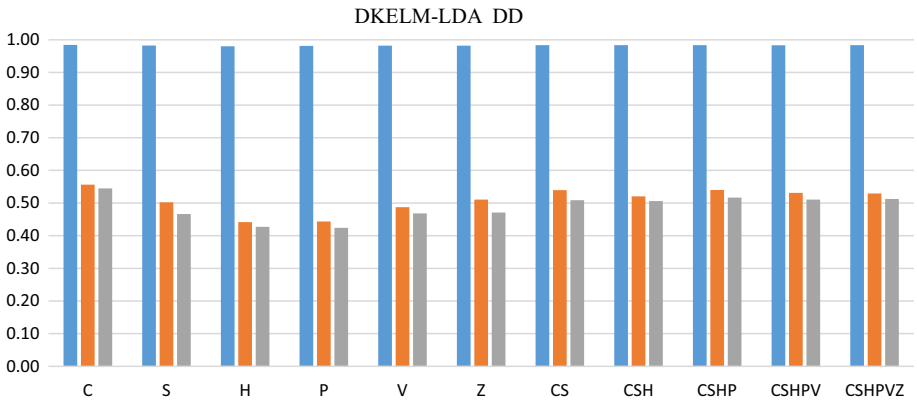

Fig. 5 Specificity, sensitivity and MCC for DKELM-LDA on DD

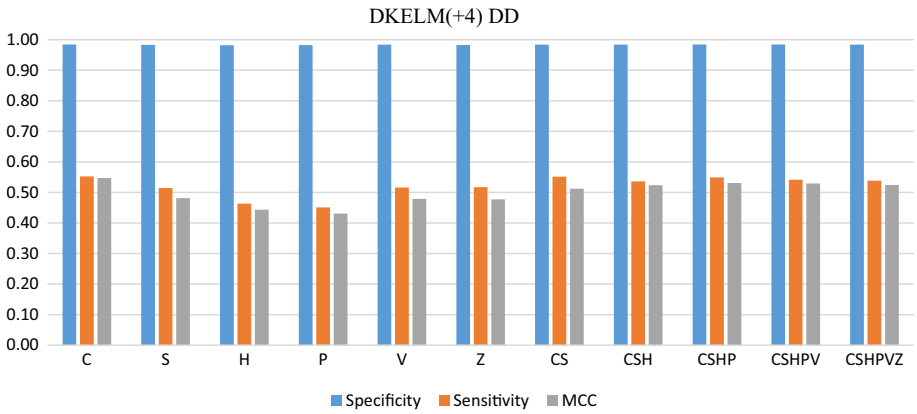

Fig. 6 Specificity, sensitivity and MCC for DKELM(+ 4) on DD

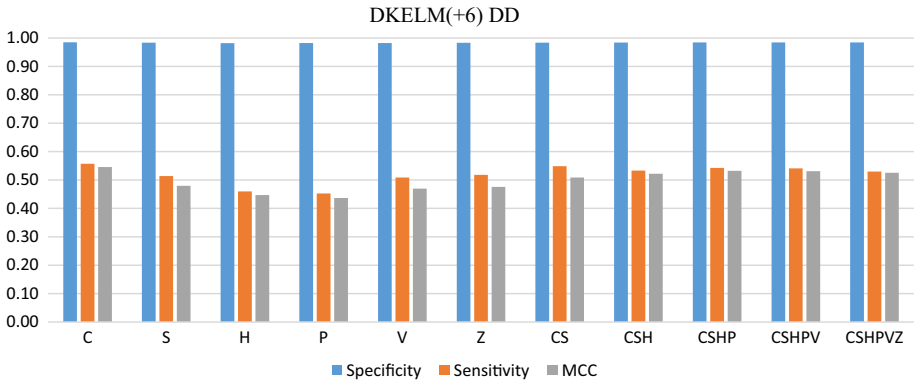

Fig. 7 Specificity, sensitivity and MCC for DKELM(+ 6) on DD

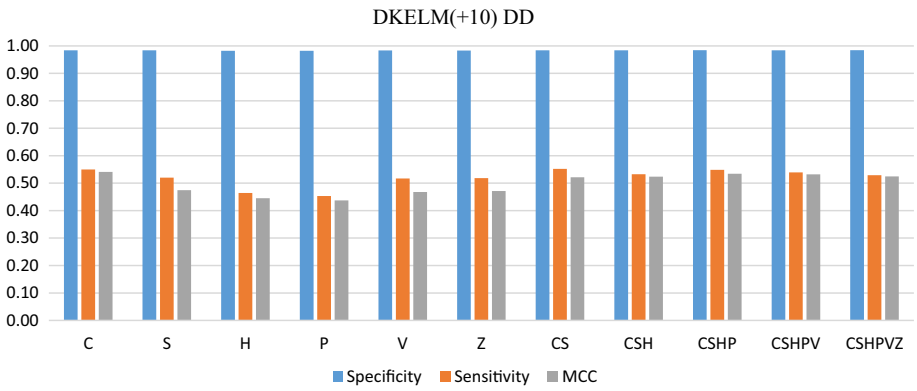

Fig. 8 Specificity, sensitivity and MCC for DKELM(+ 10) on DD

Table 6 Comparison between proposed methods and other related studies on Taguchi and Gromiha TG dataset

	$K = 5$	$K = 6$	$K = 7$	$K = 8$	$K = 9$	$K = 10$
PF1 [35]	38.1	38.4	38.6	38.7	38.8	38.8
PF2 [35]	38.0	38.4	38.5	38.6	38.7	38.8
PF [36]	42.3	42.6	42.7	43.0	43.0	43.1
O [21]	35.8	36.1	36.2	36.1	36.3	36.3
AAC [37]	31.5	31.5	31.7	31.8	31.9	32.0
ACC [38]	64.9	65.4	65.9	66.2	66.4	66.4
Bigram [15]	67.1	67.5	67.6	67.8	68.1	68.1
Trigram [15]	71.4	71.7	72.3	73.3	72.4	72.5
Alignment method [30]	72.0	72.7	73.0	73.5	73.6	74.0
DKELM-LDA (proposed method)	73.08	73.32	73.76	74.12	74.26	74.65
DKELM(+ 4) (proposed method)	74.33	74.35	74.68	74.76	75.32	75.45
DKELM(+ 6) (proposed method)	73.87	74.44	74.71	74.88	75.08	75.62
DKELM(+ 10) (proposed method)	74.25	74.57	74.82	74.85	75.37	75.78
Majority Results of OVAOVO-DKELM	74.33	74.57	74.82	74.88	75.37	75.78

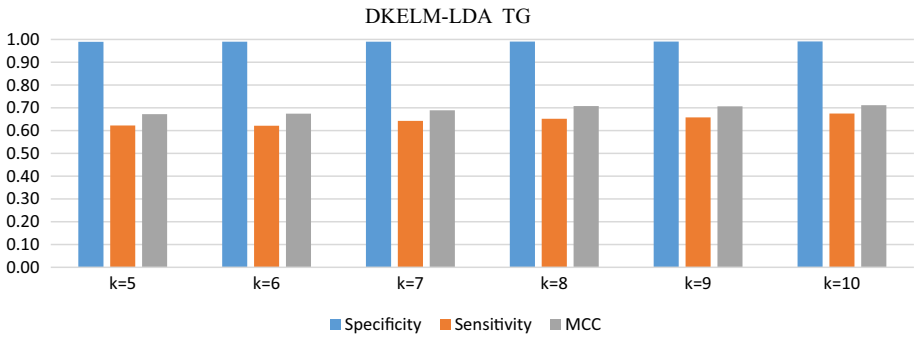

Fig. 9 Specificity, sensitivity and MCC for DKELM-LDA on TG

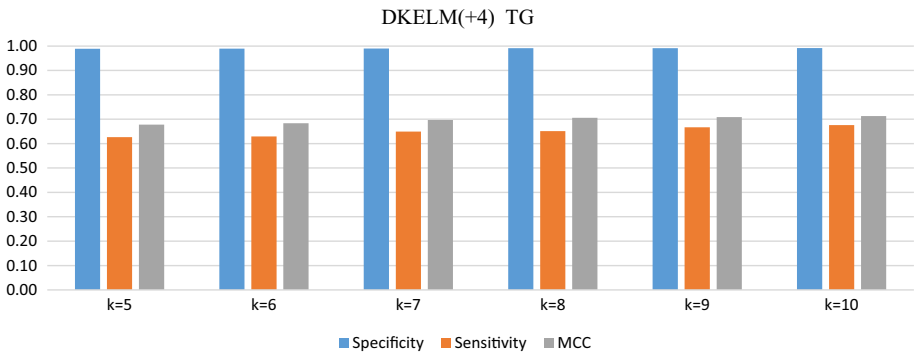

Fig. 10 Specificity, sensitivity and MCC for DKELM(+ 4) on TG

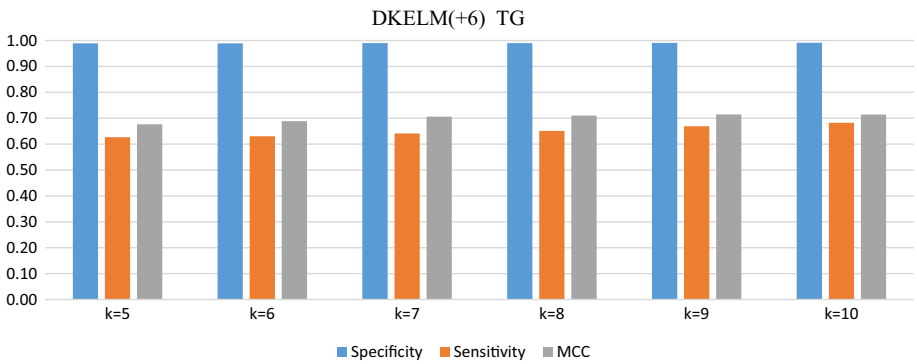

Fig. 11 Specificity, sensitivity and MCC for DKELM(+ 6) on TG

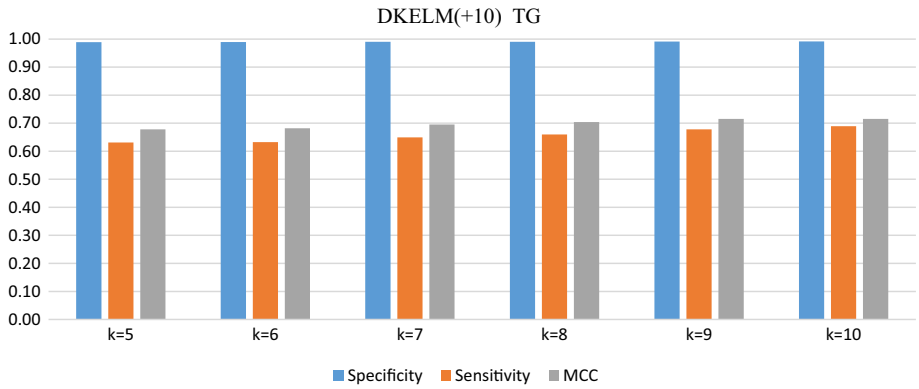

Fig. 12 Specificity, sensitivity and MCC for DKELM(+ 10) on TG

framework OVAOVO-DKELM, four features have been extracted by OVADKELM model and six features have been extracted by OVODKELM model in the second level. These new extracted features have been added to the original datasets in the third level and used by DKELM classifiers to predict the folds of the input proteins.

The experimental results show that the proposed frameworks have obtained best accuracy for most of feature vectors of DD dataset and all validations in TG dataset. The results have been obtained with MATLAB 2013 running on a standard desktop with a CPU core i7 3.40 GHz and RAM 8 GB.

In the proposed frameworks, we have used two levels of protein classification (structural classes and folds). We have employed DKELM to extract new features from protein structural classes (α, β, α/β, $\alpha + \beta$). These extracted features have been added to the original datasets in the folds' level of proposed frameworks to recognize the protein folds. In our future works, we plan applying DKELM in the both levels (structural classes and folds) to extract new features to predict the family and superfamily levels of proteins.

Compliance with ethical standards

Conflict of interest The authors declare that there is not any conflict of interest in this manuscript.

References

1. Ding CHQ, Dubchak I (2001) Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics 17(4):349–358
2. Zhang Y (2009) Protein structure prediction: When is it useful? Curr Opin Struct Biol 19(2):145–155
3. Shenoy SR, Jayaram B (2010) Proteins: sequence to structure and function-current status. Curr Protein Pept Sci 11(7):498–514
4. Valavanis IK, Spyrou GM, Nikita KS (2010) A comparative study of multi-classification methods for protein fold recognition. Int J Comput Intell Bioinf Syst Biol 1(3):332–346
5. Pal NR, Chakraborty D (2003) Some new features for protein fold prediction. In: Artificial neural networks and neural information processing—ICANN/ICONIP 2003. Springer, pp 1176–1183
6. Cheng J, Baldi P (2006) A machine learning information retrieval approach to protein fold recognition. Bioinformatics 22(12):1456–1463
7. Qu W et al (2011) Improving protein secondary structure prediction using a multi-modal BP method. Comput Biol Med 41(10):946–959
8. Abbasi E, Ghatee M, Shiri ME (2013) FRAN and RBF-PSO as two components of a hyper framework to recognize protein folds. Comput Biol Med 43(9):1182–1191
9. Jazebi S, Tohidi A, Rahgozar M (2009) Application of classifier fusion for protein fold recognition. In: Sixth international conference on FSKD'09, vol 7, 2009 Aug 14. IEEE, pp 171–175
10. Chmielnicki W, Sta K (2012) A hybrid discriminative/generative approach to protein fold recognition. Neurocomputing 75(1):194–198
11. Chen Y et al (2008) Ensemble voting system for multiclass protein fold recognition. Int J Pattern Recognit Artif Intell 22(04):747–763
12. Hashemi HB, Shakery A, Naeini MP (2009) Protein fold pattern recognition using Bayesian ensemble of RBF neural networks. In: International conference of soft computing and pattern recognition, SOCPAR'09, 2009 Dec 4. IEEE, pp. 436–441
13. Nanni L (2006) Ensemble of classifiers for protein fold recognition. Neurocomputing 69(7):850–853
14. Sharma A et al (2013) A feature extraction technique using bi-gram probabilities of position specific scoring matrix for protein fold recognition. J Theor Biol 320:41–46
15. Paliwal KK et al (2014) A tri-gram based feature extraction technique using linear probabilities of position specific scoring matrix for protein fold recognition. IEEE Trans Nanobiosci 13(1):44–50
16. Lyons J et al (2015) Advancing the accuracy of protein fold recognition by utilizing profiles from hidden Markov models. IEEE Trans Nanobiosci 14(7):761–772
17. Lyons J et al (2016) Protein fold recognition using HMM–HMM alignment and dynamic programming. J Theor Biol 393:67–74
18. Paliwal KK et al (2014) Improving protein fold recognition using the amalgamation of evolutionary-based and structural based information. BMC Bioinformatics 15(16):S12
19. Huang C-D, Lin C-T, Pal NR (2003) Hierarchical learning architecture with automatic feature selection for multiclass protein fold classification. IEEE Trans Nanobiosci 2(4):221–232
20. Aram RZ, Charkari NM (2015) A two-layer classification framework for protein fold recognition. J Theor Biol 365:32–39

21. Taguchi Y, Gromiha MM (2007) Application of amino acid occurrence for discriminating different folding types of globular proteins. BMC Bioinformatics 8(1):404
22. Dubchak I et al (1995) Prediction of protein folding class using global description of amino acid sequence. Proc Natl Acad Sci 92(19):8700–8704
23. Huang G-B, Zhu Q-Y, Siew C-K (2004) Extreme learning machine: a new learning scheme of feedforward neural networks. In: Proceedings of 2004 IEEE international joint conference on neural networks, vol 2, 2004 Jul 25. IEEE, pp 985–990
24. He Y-L, Geng Z-Q, Zhu Q-X (2015) Data driven soft sensor development for complex chemical processes using extreme learning machine. Chem Eng Res Des 102:1–11
25. Serre D (2002) Matrices: theory and applications. Springer, New York
26. Yu W et al (2015) Learning deep representations via extreme learning machines. Neurocomputing 149:308–315
27. Ghassabeh YA, Rudzicz F, Moghaddam HA (2015) Fast incremental LDA feature extraction. Pattern Recogn 48(6):1999–2012
28. Fisher RA (1936) The use of multiple measurements in taxonomic problems. Ann Eugen 7(2):179–188
29. Bi X et al (2015) Distributed extreme learning machine with kernels based on mapreduce. Neurocomputing 149:456–463
30. Lyons J et al (2014) Protein fold recognition by alignment of amino acid residues using kernelized dynamic time warping. J Theor Biol 354:137–145
31. Rahimi M, Bakhtiarizadeh MR, Mohammadi-Sangcheshmeh A (2017) OOgenesis_Pred: a sequence-based method for predicting oogenesis proteins by six different modes of Chou's pseudo amino acid composition. J Theor Biol 414:128–136
32. Shi SY, Suganthan PN, Deb K (2004) Multiclass protein fold recognition using multiobjective evolutionary algorithms. In: Proceedings of the 2004 IEEE symposium on computational intelligence in bioinformatics and computational biology, 2004. CIBCB'04. IEEE
33. Leon F, Aignatoaiei BI, Zaharia MH (2009) Performance analysis of algorithms for protein structure classification. In: 20th international workshop on database and expert systems application, DEXA'09, 2009 Aug 31. IEEE, pp 203–207
34. Kavousi K et al (2012) Evidence theoretic protein fold classification based on the concept of hyperfold. Math Biosci 240(2):148–160
35. Ghanty P, Pal NR (2009) Prediction of protein folds: extraction of new features, dimensionality reduction, and fusion of heterogeneous classifiers. IEEE Trans Nanobiosci 8(1):100–110
36. Yang T et al (2011) Margin-based ensemble classifier for protein fold recognition. Expert Syst Appl 38(10):12348–12355
37. Huang JT, Tian J (2006) Amino acid sequence predicts folding rate for middle-size two-state proteins. Proteins Struct Funct Bioinf 63(3):551–554
38. Dong Q, Zhou S, Guan J (2009) A new taxonomy-based protein fold recognition approach based on autocross-covariance transformation. Bioinformatics 25(20):2655–2662