Le Zhang,
Advanced Digital Sciences Center, Singapore

Ponnuthurai Nagaratnam Suganthan,
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Research Frontier

Benchmarking Ensemble Classifiers with Novel Co-Trained Kernel Ridge Regression and Random Vector Functional Link Ensembles

Abstract

Studies in machine learning have shown promising classification performance of ensemble methods employing "perturb and combine" strategies. In particular, the classical random forest algorithm performs the best among 179 classifiers on 121 UCI datasets from different domains. Motivated by this observation, we extend our previous work on oblique decision tree ensemble. We also propose an efficient co-trained kernel ridge regression method. In addition, a random vector functional link network ensemble is also introduced. Our experiments show that our two oblique decision tree ensemble variants and the co-trained kernel ridge regression ensemble are the top three ranked methods among the 183 classifiers. The proposed random vector functional link network ensemble also outperforms all neural network based methods used in the experiments.

I. Introduction

Ensemble methods benefit from the "perturb and combine" strategy. The rationale of the ensemble methodology in machine learning is to build a classifier or a regressor by integrating multiple base models. Practitioners from various disciplines use ensemble methods because they usually perform better than the individual base models [1], [2].

Dietterich [3] presented three fundamental reasons for the success of ensemble methods: statistical, computational, and representational reasons. From the statistical point of view, learning can be posed as searching a space $\mathcal{H}$ of hypotheses to identify the best hypothesis in the space for data fitting. However, it may become statistically problematic when the number of available training data is limited compared to the size of the hypothesis space. Without sufficient data, a learning algorithm may end up with many different hypotheses in $\mathcal{H}$ all of which fit the training data well. In this case, an ensemble approach can combine their responses to reduce the risk of choosing a single "wrong" classifier. From the computational point of view, learning algorithms such as training of neural networks and induction of DTs work by performing some form of local search which may suffer from local optima problems. Ensemble methods that perform local searches from many starting points may find a better approximation to the true unknown function than any of the individual classifiers. Another reason is representational. It is possible that the true function $\mathbf{g}$ (which is defined in Eqn. (1)) may not be well represented by any of the hypotheses in $\mathcal{H}$. It may be possible to expand the space of representable functions and representational ability by forming weighted sums of hypotheses drawn from $\mathcal{H}$. In addition, bias-variance decomposition [4] and strength-correlation [5] also explain why ensemble methods work. Due to these reasons, for decades researchers developed numerous ensemble methods to achieve better performance than a single model.

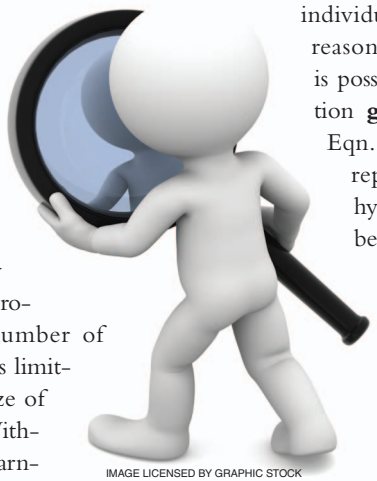

IMAGE LICENSED BY GRAPHIC STOCK

Along this line of research, two approaches are perceived as "classic". The first method is bootstrap aggregation (bagging) [6]. It works by training

Digital Object Identifier 10.1109/MCI.2017.2742867
Date of publication: 12 October 2017

Corresponding author: Ponnuthurai Nagaratnam Suganthan (Email: epnsugan@ntu.edu.sg).

independent classifiers from a set of instances taken with replacement from the training instances. The classifiers' outputs are combined by majority voting. Bagging is suitable for unstable models where small perturbations of the training set may cause significant performance variations. In this case, bagging can lead to better performance by significantly reducing the variance of such unstable models. The second method is boosting [7]. Different from bagging, individual classifiers in boosting must be trained in a sequential manner, where the current classifier being trained will focus on the training instances which cannot be classified correctly by previous ensemble members. The weight of each instance in the training set will be updated in each training iteration. The "difficult" instances will be more likely to be sampled than the "easy" ones.

Random forest (RaF) [5] combines the concepts of bagging and random subspace. In RaF, each DT is trained on a bootstrap version of the training set and each node of the DT is trained on a randomly selected subset of the original features. RaF has achieved state-of-the-art performance on a recent benchmarking study [8]. In [8], the authors evaluated 179 classifiers arising from 17 families of classification methods on 121 datasets, which represent the whole UCI database (excluding a few large-scale problems) and other real-world problems. In addition to RaF, other methods which are within the 20 best performing classifiers are support vector machine (SVM) variants, kernel ridge regression (KRR), DT variants, neural network variants and boosting methods.

KRR was originally proposed for regression and has shown similar classification performance to more sophisticated models such as SVM. Kernel classifiers are widely regarded as stable classifiers but perform poorly under a typical ensemble framework [9]. Some attempts have been made on multiple kernel learning (MKL) [2], [10]. MKL combines an ensemble of different kernels which correspond to either using different notions of similarity or using information coming from multiple sources such as different representations or different feature subsets. These trained multiple kernels are combined using a classification method such as SVM. Efforts in this domain have been devoted to optimizing the combination parameters of different kernels, but how to design different features or kernels to solve general classification problems is still an open question.

Generally speaking, the complexity of KRR is $O(N^3)$, where N is the number of data samples. Thus, KRR based ensemble methods with hundreds of base models may be very time-consuming. This also motivates us to investigate ensemble KRR with a smaller number of base models. In this work, we propose an efficient co-trained KRR method which trains just two KRR models jointly.

As an unstable learner, neural network is also widely used as a base model in ensemble learning. Conventional neural networks employ back-propagation learning to tune their parameters. This is shown to be time-consuming and suffer from local minimum problems. Random vector functional link network (RVFL) remedies this by randomly initializing the parameters of the hidden neurons and optimizing the remaining ones. Though RVFL has been proposed for a long time, the classification ability of its ensemble version has been under-researched. In this paper, we fill this research gap by proposing a simple RVFL ensemble.

The winner of [8] is orthogonal RaF because the hyperplane in each node of the DTs is obtained by searching for a good single feature resulting in only axis parallel hyperplanes. A better alternative is to generate the hyperplane at each node based on a linear combination of several features to better capture the geometric structure of the data samples. The resulting hyperplanes are usually not orthogonal to the feature axes. The resulting RaFs are termed as oblique RaFs. Fig. 1 depicts a toy example of the decision boundary of both orthogonal and oblique DTs. Several studies have shown the superiority of oblique RaF over orthogonal RaF [11]–[13]. However, all these oblique RaFs were evaluated on a limited number of datasets and compared with a limited number of methods. In this study, we extend our previous work on oblique DT ensemble [12] and evaluate them under the same framework of [8]. As a result, our oblique RaF achieves the first Friedman rank. Hence, we posit this oblique RaF can be a new baseline of ensemble classification methods.

The following sections of this paper are organized as follows: Notations and background material are presented in Section II. We briefly review the 20 best

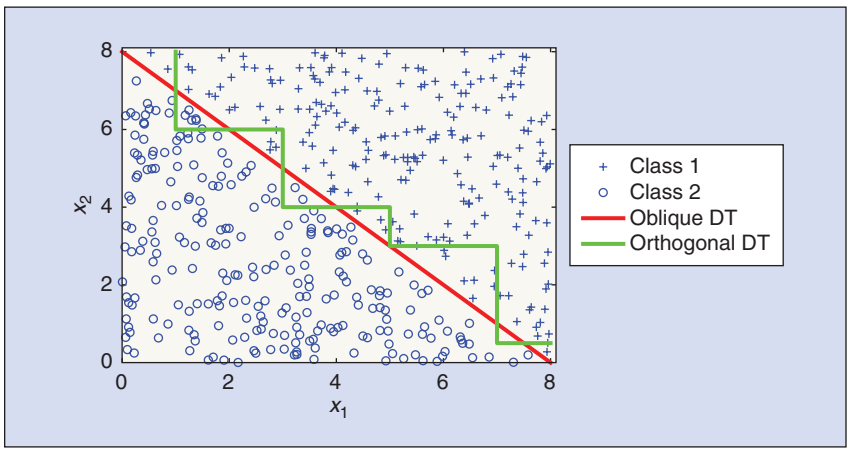

FIGURE 1 A toy example of classification boundaries generated by orthogonal and oblique DTs. Orthogonal models select a single feature at each node to conduct a split. This results in a piece-wise axis orthogonal hyperplane (in green color). Oblique models, on the other hand, can use more than one feature at each node and thus results in an oblique hyperplane (in red color) that classifies the data better.

performing methods of [8] in Section III. In Section IV, we elaborate the oblique DT ensemble. Details of the proposed KRR and RVFL ensembles are presented in Section V and Section VI, respectively. Experimental details are summarized in Table VI in Section VII. Conclusions are presented in Section VIII. Interested readers are referred to the supplementary files which can be found from the authors' homepage (http://www.ntu.edu.sg/home/epnsugan/) for detailed results of the ensemble methods on 121 datasets.

II. Notations and Background Material

All vectors in this paper are column vectors unless transposed to a row vector by a superscript $\top$. Each data sample is represented as an M dimensional vector: $\mathbf{x}_i \in \mathcal{X} \subseteq IR^M, i \in \{1,\ldots,N\}$. Let $\mathbf{X} \in IR^{N\times M}$ be an $N\times M$ matrix obtained by stacking N samples in X, *i.e.*, $\mathbf{X} = [\mathbf{x}_1,\ldots,\mathbf{x}_N]^\top$. Let $\mathbf{Y} \in \{\mathbf{y}_1,\mathbf{y}_2,\ldots,\mathbf{y}_N\}^\top$ be a vector obtained by stacking the labels of samples in X, where $\mathbf{y}_i \in \mathcal{Y}$ takes a value from the set of class labels $\{\omega_1,\omega_2,\ldots,\omega_L\}$. We consider an L-class classification problem. We denote a generic data sample as $\mathbf{x}$ by ignoring the subscript for convenience, and use $\mathbf{x}_\diamond$, with $\diamond$ denoting the placeholder for the index wherever necessary. We use $\mathbf{x}_{\diamond\bullet}$ to indicate the feature subset indexed by $\bullet$ of the $\diamond$th sample.

We consider classification as the task of identifying the discrete category of a new observation by learning on a set of training data to get a function $\mathbf{g}$:

$$\mathbf{y}_i = \mathbf{g}(\mathbf{x}_i, \Theta), \mathbf{y}_i \in \mathbf{Y}, i = 1,2,\ldots,N, \quad (1)$$

where $\mathbf{x}_i$ is the ith observation as a feature vector, $\mathbf{y}_i$ is the category that this observation belongs to, $\mathbf{g}$ is the trained classification function, and Θ is the classification function's parameter set.

Ensemble classification can be considered as learning an ensemble mapping function $\mathcal{G}$ composed of $\mathcal{K}$ base classifiers $\mathbf{g}_k(\mathbf{x}, \Theta), k \in \{1,\ldots,\mathcal{K}\}$. More specifically:

TABLE 1 Nomenclature.

ABBREVIATION	DEFINITION
DT	DECISION TREE
RaF	RANDOM FOREST
RoF	ROTATION FOREST
RRoF	RANDOM ROTATION FOREST
MPSVM	MULTISURFACE PROXIMAL SUPPORT VECTOR MACHINE
MPRaF	MPSVM BASED RANDOM FOREST WITH AXIS-PARALLEL REGULARIZATION
MPRRoF	MPSVM BASED RANDOM ROTATION FOREST WITH AXIS-PARALLEL REGULARIZATION
RVFL	RANDOM VECTOR FUNCTIONAL LINK NETWORK
KRR	KERNEL RIDGE REGRESSION
Co-KRR	CO-TRAINED KERNEL RIDGE REGRESSION
Co-KRR-max	MAX RESULT OF THE CoKRR
Co-KRR-avg	AVERAGE RESULT OF THE CoKRR
2KRR-avg	AVERAGE RESULT OF TWO INDEPENDENT KRR
2KRR-max	MAX RESULT OF TWO INDEPENDENT KRR
KRR-O	KRR TRAINED ON ORIGINAL FEATURE
KRR-C	KRR TRAINED ON COMBINED FEATURE
KRR-T	KRR TRAINED ON TRANSFORMED FEATURE
MKL	MULTIPLE KERNEL LEARNING
NCL	NEGATIVE CORRELATION LEARNING
NN	NEURAL NETWORK

$$\mathcal{G}(\mathbf{x}_i) = \sum_{k=1}^{\mathcal{K}} \alpha_k \mathbf{g}_k(\mathbf{x}_i, \Theta_k), \quad (2)$$

where α_k and Θ_k denote the weights and parameters for the kth base classifier $\mathbf{g}_k$, respectively. α_k may be learned by another model.

III. Details of Best Performing Classification Methods

In this section, we present a brief review of the best performing methods in [8] which will be compared with the methods developed in this study. We have summarized them in Table II. They can be roughly classified into the following families[1]:

- ❏ Random forest.
- ❏ Support vector machine.
- ❏ Kernel ridge regression.
- ❏ Boosting.
- ❏ Neural networks.
- ❏ Rotation forest.

Due to page limit, the details of the methods in Table II are summarized in Appendix A. Moreover, we only provide detailed information about RaF, KRR and Neural Networks in this section in order to relate the three ensemble methods included in this study. For SVM, Boosting and Rotation Forest, we just briefly point out the rationale behind them. Interested readers are referred to the references therein.

A. Random Forest (RaF)

RaFs [5] are collections of classifiers $\mathcal{G} = \{\mathbf{g}_k\}_{k=1}^{\mathcal{K}}$ combined using "bagging" [6]. Each base DT $\mathbf{g}_k$ in an RaF is trained on a boot-strapped version of the training set using a randomly selected subset of features to search for an axis orthogonal decision hyperplane at each node. The predictions $\mathbf{g}_k(\mathbf{x})$ from a set of $\mathcal{K}$ trees are combined to make a single output. Choice of the combination is problem specific and depends on Y. Common choices include majority voting

[1]In the original paper [8], the 6th ranked method is termed as "elm_kernel_m". However, based on the reviewers' comments, we learn that this method is identical to the KRR according to [14], [15], and Section 12.3.7 in [16]. Hence, we follow their advices and name this method as "Kernel Ridge Regression" to acknowledge the original contributors.

TABLE 2 20 best performing methods in [8]. Detailed definition of each method is presented in Appendix A.

RANK	METHOD	MEAN ACCURACY	FAMILY
1	parRF_t	82.0	RaF
2	rf_t	82.3	RaF
3	svm_C	81.8	SVM
4	svmPoly_t	81.2	SVM
5	rforest_R	81.9	RaF
6	KRR[1]	82.0	KRR
7	svmRadialCost_t	81.4	SVM
8	svmRadial_t	81.0	SVM
9	C5.0_t	80.6	BOOSTING
10	avNNet_t	79.4	NN
11	nnet_t	79.5	NN
12	pcaNNet_t	78.7	NN
13	B_LibSVM_w	80.8	SVM
14	mlp_t	80.3	NN
15	RotationForest_w	80.6	RaF
16	RRF_t	80.9	RaF
17	RRFglobal_t	80.7	RaF
18	MAB_LibSVM_w	80.6	SVM
19	LibSVM_w	79.9	SVM
20	adaboost_R	79.1	BOOSTING

for classification and averaging for regression, although more sophisticated aggregation models may be employed such as training another model to aggregate the outputs [17].

A DT $\mathbf{g}_k(\mathbf{x})$ classifies a sample $\mathbf{x} \in \mathcal{X}$ by routing it from the root to a leaf node, which provides a label for the instance. Specifically, each internal node j in the tree is associated with a binary split function:

$$f_{kj}(\mathbf{x}, \theta) \in \{0, +1\}, \tag{3}$$

where θ is the parameter of the split function. Samples are sent to the right child node if $f_{kj}(\mathbf{x}) = 1$ and to the left if $f_{kj}(\mathbf{x}) = 0$ with the process terminating at a leaf node. The output of the tree for an input $\mathbf{x}$ is the prediction stored at the leaf reached by $\mathbf{x}$, which may be a target label $\mathbf{y} \in \mathcal{Y}$ or a distribution over the labels $\mathcal{Y}$.

Although the split function $f_{kj}(\mathbf{x}, \theta)$ in Eqn. (3) may be complex, a common choice is a "stump" parameterized by $\theta = \{m, \tau\}$, where τ is a threshold on $\mathbf{x}_{\diamond m}$, the mth dimension of the sample $\mathbf{x}_\diamond$. Specifically, $f_{kj}(\mathbf{x}_\diamond, \theta) = I[\mathbf{x}_{\diamond m} < \tau]$, where $I[x]$ is the indicator function ($I[x] = 1$, if and only if $x = 1$ and 0 otherwise). This kind of DT is called an orthogonal (also uni-variate or axis-parallel) DT and the resulting RaF is termed as orthogonal RaF. A commonly used criterion for optimizing θ is information gain or Gini-impurity [18], which aims to make the data samples falling in the children nodes more "pure" to facilitate the next splitting, if necessary. For example, the Gini-impurity is given by:

$$\begin{aligned}\text{Gini}(j) = &\frac{n_j^l}{n_j}\left[1 - \sum_{i=1}^{L}\left(\frac{n_{jw_i}^l}{n_j^l}\right)^2\right] \\ &+ \frac{n_j^r}{n_j}\left[1 - \sum_{i=1}^{L}\left(\frac{n_{jw_i}^r}{n_j^r}\right)^2\right],\end{aligned} \tag{4}$$

where n_j is the number of data samples reaching the jth node, n_j^l and n_j^r are the number of data samples that reach the left and right child nodes of the current node respectively, $n_{jw_i}^l$ and $n_{jw_i}^r$ represent the number of samples of class w_i in the left and right nodes respectively, and L is the number of classes.

Conventional RaF methods typically choose an axis-orthogonal hyperplane at every node, resulting in an ensemble decision surface that is piece wise axis-orthogonal, even when there is little evidence for this in the data, as demonstrated in Fig. 1 and Fig. 3(a)-3(e). As a result, the decision boundary turns out to be "stage-like" and each hyperplane is orthogonal to one feature axis. All RaF methods in [8] are orthogonal RaFs.

B. Support Vector Machine

Here we consider the binary case where $\gamma_i \in \{-1, +1\}$ and multiclass SVMs can be obtained by one-vs-one or one-vs-all strategy. An SVM finds discriminant function with maximum margin in the feature space induced by the mapping function $\phi : R^M \to R^S$. The discriminant function is defined as follows:

$$\mathbf{g}(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b, \tag{5}$$

where $\mathbf{w}$ and constant b are the parameters of the hyperplane of the SVM.

The above function can be obtained by solving the following quadratic optimization problem:

$$\begin{aligned}&\min \ \frac{1}{2}(\|\mathbf{w}\|_2)^2 + C\sum_{i=1}^{N}\xi_i, \\ &\text{with respect to } \mathbf{w} \in R^S, \xi \in R_+, b \in R \\ &\text{subject to } \mathbf{y}_i(\langle \mathbf{w}^\top \phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i, \forall i,\end{aligned} \tag{6}$$

where C is a predefined positive trade-off parameter between model simplicity and generalization ability. ξ_i is a slack variable.

Based on the Representer theorem [19], the Lagrangian dual function of the problem in Eqn. (6) is as follows:

$$\begin{aligned}\max \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j\mathbf{y}_i\mathbf{y}_j \\ \times \underbrace{\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle}_{k(\mathbf{x}_i, \mathbf{x}_j)},\end{aligned} \tag{7}$$

where $k : R^D \times R^D \to R$ is the kernel function and α is the dual variable. One can simply employ the commonly used kernel functions such as Gaussian kernel, RBF kernel, and linear kernel (no feature

mapping is employed) without explicitly knowing the form of the feature transformation function $\phi(\mathbf{x})$.

Solving Eqn. (7), we can get $\mathbf{w} = \Sigma_{\mathbf{i}=1}^{\mathbf{N}} \alpha_{\mathbf{i}} \phi(\mathbf{x_i})$. Hence, the discriminant function can be written as:

$$\mathbf{g}(\mathbf{x}_i) = \sum_{j=1}^{N} \alpha_j \mathbf{y}_j k(\mathbf{x}_j, \mathbf{x}_i) + b. \quad (8)$$

C. Kernel Ridge Regression

KRR [14], [15] (and Section 12.3.7 in [16]) extends the ridge regression to non-linear cases via the kernel trick. It was originally proposed to solve the regression problem and is shown to achieve similar performance to more sophisticated models such as the support vector regression. However, the classification ability of the KRR has been under-researched [15]. In this work, we conduct a comprehensive study on KRR for classification and propose a novel KRR ensemble method.

A typical linear regression problem can be formulated as:

$$\min_{\mathbf{w}} \sum_i (\mathbf{w}^\top \mathbf{x}_i - \mathbf{y}_i)^2 + C \left\| \mathbf{w} \right\|_2^2, \quad (9)$$

where parameter C is a user-defined regularization parameter that controls the model complexity. This problem can lead to an elegant closed-form solution:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X} + C\mathbf{I})^{-1} \mathbf{X}^\top \mathbf{Y}, \quad (10)$$

where the data matrix $\mathbf{X}$ has one sample per row $\mathbf{x}_i$, and each element of the vector $\mathbf{Y}$ is the output target $\mathbf{y}_i$ of $\mathbf{x}_i$. $\mathbf{I}$ is an identity matrix.

KRR extends the linear regression nonlinearly through the kernel trick. Based on the Representer theorem [19], the solution of $\mathbf{w}$ can be formulated as a linear combination of the samples in the feature space $\phi(\mathbf{x})$ as: $\mathbf{w} = \Sigma_i \alpha_i \phi(\mathbf{x}_i)$. The KRR problem is then formulated as:

$$\min_{\alpha} \left\| \mathbf{Y} - \mathbf{K}\alpha \right\|^2 + C\alpha^\top \mathbf{K}\alpha. \quad (11)$$

Similarly, the solution is given in a closed-form manner as:

$$\alpha = (\mathbf{K} + C\mathbf{I})^{-1} \mathbf{Y}. \quad (12)$$

In the same way, by applying the kernel trick, the kernel matrix $\mathbf{K}$ can be obtained by $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$.

A classification problem can be posed as a regression problem by defining the output target $\mathbf{Y}$ with $0-1$ encoding [15]. More specifically, if there are L classes with N samples, the output $\mathbf{Y}$ should be an $N \times L$ matrix which can be generated by the following equation:

$$\mathbf{Y}_{ij} = \begin{cases} 1 & \text{If } i\text{th sample belongs to the } j\text{th class} \\ 0 & \text{Otherwise} \end{cases}. \quad (13)$$

D. Feedforward Neural Network

Single hidden layer feedforward neural network (SLFN) gains its popularity in classification amongst the family of feedforward neural networks because of its global approximation ability. Fig. 2 demonstrates the basic structure of an SLFN which consists of three layers: input layer, hidden layer and output layer. Denote the input data samples as $\mathbf{X}$ and their corresponding output classes as $\mathbf{Y}$. The input features are firstly linearly scaled by the weights $\mathbf{a}$ between the input and hidden layer. After that, a nonlinear activation function $\Phi_{\mathbf{h}}$ is applied to the transformed features to get the features in the hidden layer.

$$\mathbf{H} = \Phi_{\mathbf{h}}(\mathbf{Xa} + \mathbf{b}). \quad (14)$$

The bias vector $\mathbf{b}$ in Eqn. (14) can be omitted by augmenting the input as $\mathbf{x} = [\mathbf{x}^\top, \mathbf{1}]^\top$ and Eqn. (14) can be simplified as:

$$\mathbf{H} = \Phi_{\mathbf{h}}(\mathbf{Xa}). \quad (15)$$

The features $\mathbf{H}$ are forwarded to the output layer. The output layer computes the loss of the data samples by comparing the output $\tilde{\mathbf{Y}}$ from the SLFN and the ground truth label $\mathbf{Y}$:

$$\tilde{\mathbf{Y}} = \Phi_{\mathbf{o}}(\mathbf{H}\beta), \delta = \mathbf{l}(\mathbf{Y}, \tilde{\mathbf{Y}}). \quad (16)$$

Usually the output $\tilde{\mathbf{Y}}$ is obtained by transforming the features $\mathbf{H}$ from the hidden layer directly without any nonlinear activation function. In this case, $\Phi_{\mathbf{o}}(\mathbf{H}\beta) = \mathbf{H}\beta$. Moreover, the choices of loss function can be problem dependent and common choices of loss function are hinge loss, mean square loss, logistic loss and so on.

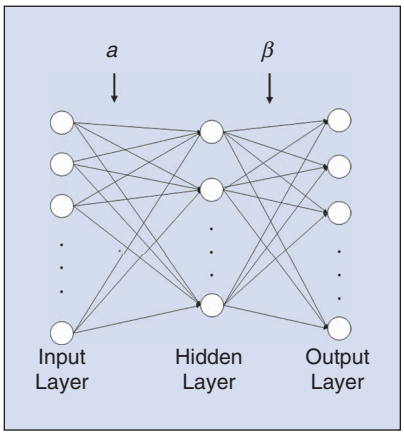

FIGURE 2 The structure of SLFN.

E. Boosting

Most boosting algorithms work by iteratively training unstable classifiers with respect to a distribution and adding them to a final stable classifier. Boosting methods operate in a "divide and conquer" manner. When a new base classifier is generated, misclassified examples gain weight while correctly-classified examples lose weight. Thus, future unstable learners focus more on the examples that previous models misclassified. Due to page limit, we refer the interested readers to [7] for more information.

F. Rotation Forest

Rotation Forest (RoF) is also a well-known DT ensemble. It constructs "high strength" and "low correlation" classifiers [20]. RoF differs from RaF mainly in two aspects: firstly, RoF applies feature extraction based on a rotation matrix for each DT. Secondly, all features are used as candidate features when searching for the "optimal feature" for a hyperplane in RoF while random subspace is used in RaF.

IV. Oblique DT Ensemble

Orthogonal DT learning algorithms work in a greedy top down fashion. For each node, it evaluates the score function exhaustively for each candidate feature and all possible thresholds for that

feature and chooses a threshold with the best value for a criterion such as Eqn. (4). The computational complexity of these methods are, therefore, at least $O(M \cdot N\log(N))$, where M and N are the dimensionality of the input feature vector and the number of the data samples, respectively. An oblique DT usually employs a non-orthogonal hyperplane at each node for data splitting. More specifically, it tests a linear combination of the attributes at each internal node which can be formulated as:

$$f_{kj}(x_\diamond) = \begin{cases} 1 & \text{if } \sum_{m=1}^{M} \mathbf{a_m}^\top \mathbf{x}_{\diamond \mathbf{m}} < \tau \\ 0 & \text{otherwise} \end{cases}, \quad (17)$$

where $\mathbf{a_m}$ and τ stand for the coefficients and the threshold for this split function.

As aforementioned, exhaustive search for an optimal oblique hyperplane turns out to be computationally infeasible for large datasets. In [12], the authors proposed to use multisurface proximal support vector to better capture the geometric structure of the data samples efficiently. The multisurface proximal SVM (MPSVM) was originally proposed for binary classification problems. Let matrix $\mathbf{A}$ with size of $N_1 \times M$ represents the samples of class 1 and matrix $\mathbf{B}$ with size $N_2 \times M$ represents the samples of class 2. Then the MPSVM seeks two planes in R^M:

$$\begin{aligned} \mathbf{X}\mathbf{W_1} - \mathbf{b_1} &= \mathbf{0}, \\ \mathbf{X}\mathbf{W_2} - \mathbf{b_2} &= \mathbf{0}, \end{aligned} \quad (18)$$

where each element of the vector $\mathbf{b_1}$ is equal to b_1 and the same definition can be applied to vector $\mathbf{b_2}$. The first plane $(\mathbf{W_1}, b_1)$ is closer to the samples of class 1 and further from the samples of class 2, while the second plane $(\mathbf{W_2}, b_2)$ is closer to the samples of class 2 and further from the samples of class 1. To obtain the first plane of Eqn. (18), the authors propose to minimize the sum of squares of two-norm distances between each of the samples of class 1 to the plane divided by the squares two-norm distance between each of the samples of class 2 to the plane. Hence, the first plane can be obtained by solving the following optimization problem:

$$\min_{(\mathbf{W},b)\neq 0} \frac{\left\|\mathbf{A}\mathbf{W} - \mathbf{e}b\right\|^2 \Big/ \left\|\begin{matrix}\mathbf{W}\\ b\end{matrix}\right\|^2}{\left\|\mathbf{B}\mathbf{W} - \mathbf{e}b\right\|^2 \Big/ \left\|\begin{matrix}\mathbf{W}\\ b\end{matrix}\right\|^2}, \quad (19)$$

and similarly, the second plane can be obtained by solving:

$$\min_{(\mathbf{W},b)\neq 0} \frac{\left\|\mathbf{B}\mathbf{W} - \mathbf{e}b\right\|^2 \Big/ \left\|\begin{matrix}\mathbf{W}\\ b\end{matrix}\right\|^2}{\left\|\mathbf{A}\mathbf{W} - \mathbf{e}b\right\|^2 \Big/ \left\|\begin{matrix}\mathbf{W}\\ b\end{matrix}\right\|^2}, \quad (20)$$

where $\|\|$ denotes the two-norm and $\mathbf{e}$ is a vector of ones with the same dimension as $\mathbf{AW}$ and $\mathbf{BW}$. By defining:

$$\begin{aligned} \mathbf{G} &= [\mathbf{A} - \mathbf{e}]'[\mathbf{A} - \mathbf{e}], \\ \mathbf{J} &= [\mathbf{B} - \mathbf{e}]'[\mathbf{B} - \mathbf{e}], \\ \mathbf{z} &= [\mathbf{W} \quad \mathbf{b}]', \end{aligned} \quad (21)$$

the optimization problem of Eqn. (19) becomes:

$$\min_{\mathbf{z}\neq 0} \frac{\mathbf{z}'\mathbf{G}\mathbf{z}}{\mathbf{z}'\mathbf{J}\mathbf{z}}. \quad (22)$$

Similarly, the optimization problem of Eqn. (20) becomes:

$$\min_{\mathbf{z}\neq 0} \frac{\mathbf{z}'\mathbf{J}\mathbf{z}}{\mathbf{z}'\mathbf{G}\mathbf{z}}. \quad (23)$$

Thus, the two clustering hyperplanes can be found by the eigenvectors corresponding to the smallest eigenvalues of the following two generalized eigenvalue problems:

$$\begin{aligned} \mathbf{G}\mathbf{z} &= \lambda\mathbf{J}\mathbf{z}, \mathbf{z} \neq \mathbf{0}, \\ \mathbf{J}\mathbf{z} &= \gamma\mathbf{G}\mathbf{z}, \mathbf{z} \neq \mathbf{0}. \end{aligned} \quad (24)$$

Due to the way they are defined, these clustering hyperplanes capture the geometric property that is useful for discriminating the classes. Hence, a hyperplane that passes in between them could be good for splitting the feature space. For multiclass classification problems, the data samples are divided into two hyper-classes by their pair-wise Bhattacharyya distance which measures the similarity between two discrete or continuous probability distributions, which is considered as a good measure for class separability between two normal classes w_1 and w_2 with distributions of $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$, respectively. μ stands for the mean and Σ is the vriance. It can be defined as:

$$\begin{aligned} B(w_1, w_2) = \frac{1}{8}(\mu_2 - \mu_1)^T & \left(\frac{\Sigma_1 + \Sigma_2}{2}\right)^{-1}(\mu_2 - \mu_1) \\ & + \frac{1}{2}\ln\frac{\left|(\Sigma_1 + \Sigma_2)/2\right|}{\sqrt{\left|\Sigma_1\right|\left|\Sigma_2\right|}}. \end{aligned} \quad (25)$$

As the tree grows, the data samples reaching a particular node will be reduced. In this case, the problem in Eqn. (24) will be ill-posed. The authors in [12] proposed to use Tikhonov Regularization or switch back to axis-parallel split when matrix $\mathbf{G}$ or $\mathbf{J}$ in Eqn. (24) becomes rank deficient. They reported superior performance of the proposed MPSVM based DT ensemble against the conventional RaF and RRoF on a limited number of datasets. In this paper, we choose MPSVM based RaF and RRoF with axis-parallel regularization from [12] and name them as MPRaF and MPRRoF, respectively. As a toy example, Fig. 3 illustrates that the proposed oblique DT ensemble methods can better capture the geometrical structure of the data than a single DT and orthogonal RaF.

V. A Diversified Random Vector Functional Link Network Ensemble

Neural networks are also unstable classifiers. Typically, training algorithms of neural networks minimize a particular loss function. Conventional training algorithms usually optimize the parameters based on the derivatives of the loss function. However, much of the power of neural network comes from the nonlinear function in the hidden units, which is used to obtain the nonlinear mapping between the input space and the output space. In this case, the learning problems turn out to be nonlinear optimization problems which are usually solved iteratively. In this case, the error signal has to be propagated backwards to serve as a guidance for tuning the parameters [21]. Those methods are reported to

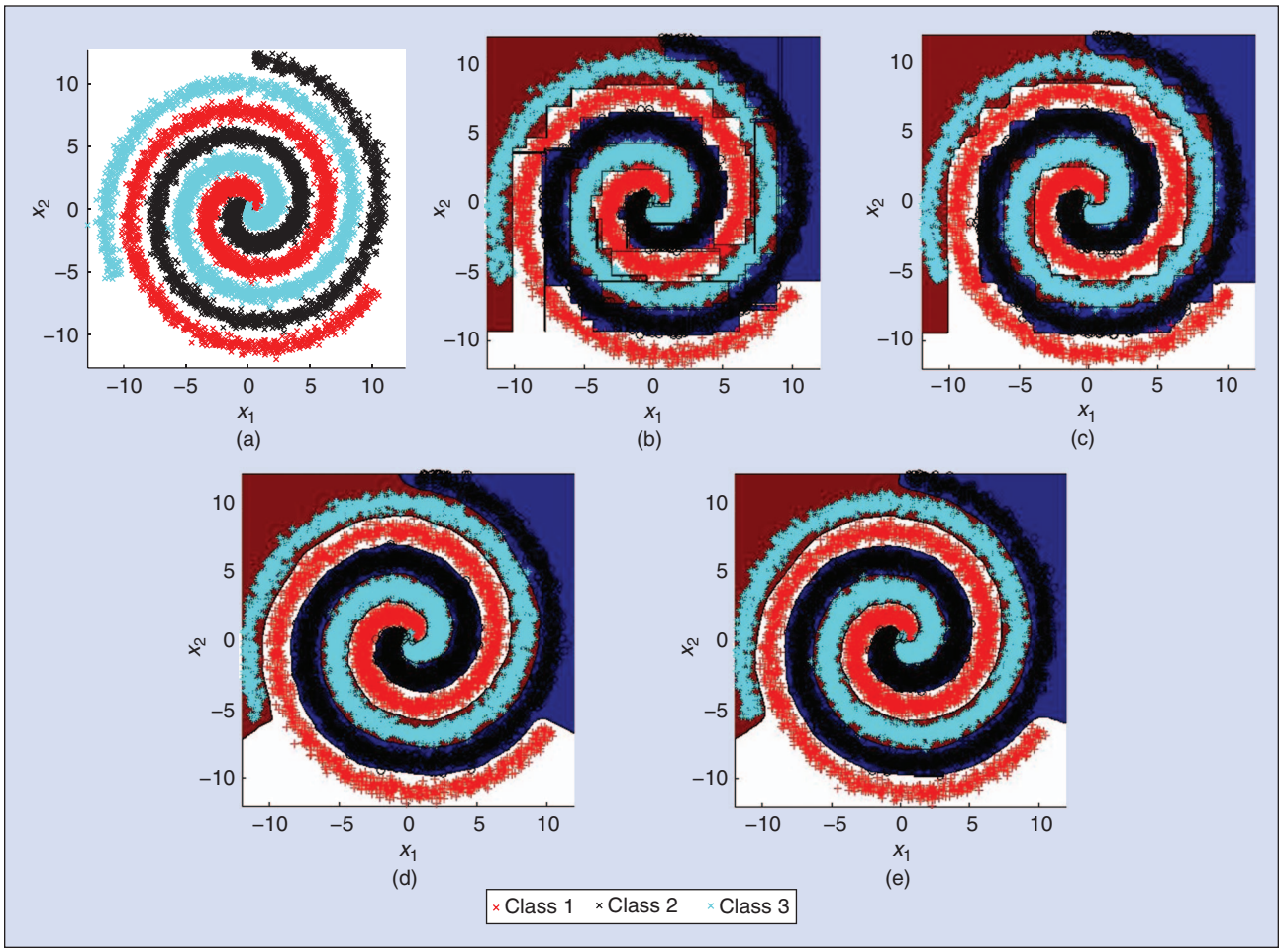

FIGURE 3 Decision boundaries of the spiral dataset. (a) 2D spirals dataset, (b) Decision Boundary of Single DT, (c) Decision Boundary of RaF, (d) Decision Boundary of MPRaF and (e) Decision Boundary of MPRRoF.

be slow [22] and may not converge to a single global minimum because there exist many local minima [23], [24]. In addition, the learned neural network can be unstable in the real-world noisy situations when there are a limited number of data samples.

An alternative of the above methods is the randomization based methods [25] with closed form solution. One pioneering work on training neural network with randomization can be found in [26]. The RVFL [26] neural network model can be regarded as a semi-random realization of the functional link neural networks, which is demonstrated in Fig. 4.

The rationale behind this architecture is to improve the generalization ability of the network with some enhanced features which can be achieved by applying nonlinear transformation on the dot product of original features and

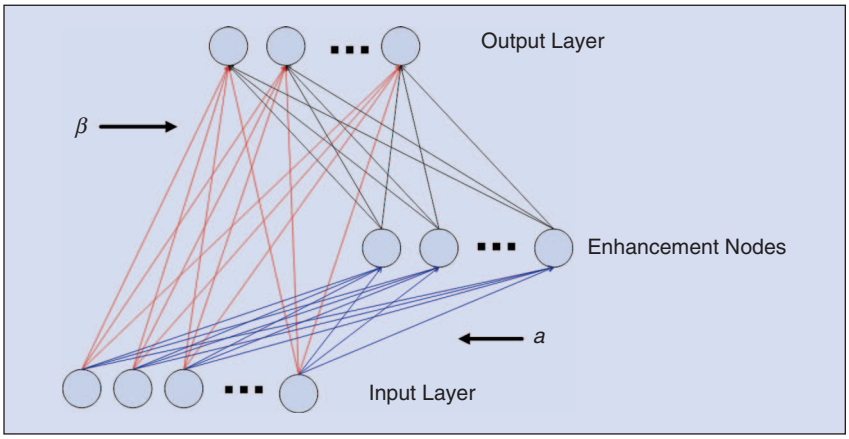

FIGURE 4 The structure of RVFL. The input features are firstly transformed into the enhanced features by the enhancement nodes where parameters within them are randomly generated. All the original and enhanced features are connected to the output neurons.

weights between input and the hidden layers. The weights a_{ij} from the input to the enhancement nodes are randomly generated such that the activation function $\Phi_h(\mathbf{a_j^\top x} + \mathbf{b_j})$ is not saturated most of the time. For RVFL, only the output weights β_j need to be optimized. Suppose that the input data has M features

TABLE 3 Parameter settings for base RVFL models.

PARAMETERS	TUNED RESULT	RANDOMIZED PARAMETER
HIDDEN NEURON NUMBERS	$\mathcal{N}$	$ROUND((1+\delta)\times\mathcal{N})$
REGULARIZATION PARAMETER	C	$(1+\delta)\times C$
SCALE PARAMETER	$\mathcal{S}$	$(1+\delta)\times\mathcal{S}$

The second column represents the best value for each parameter for each dataset. The third column stands for the parameter for each RVFL model. δ is a random variable drawn from [0, 1] with uniform distribution.

and there are S enhancement neurons, there are in total $M+S$ inputs to each output node. Learning is achieved by minimizing the following expression when the mean square loss function is employed:

$$E=\sum_{i=1}^{N}\left\|\mathbf{y}_i-\mathbf{d}_i\beta\right\|_2^2, \tag{26}$$

where β consists of the weight values β_j, $j=1,2,\ldots,M+S$, and d denotes the concatenation of the input and enhancement features. There are N input samples in total. E is quadratic with respect to each $M+S$ dimensional vector β_j, indicating that the unique minimum can be found in no more than $M+S$ iterations of the learning procedure in methods such as the conjugate gradient method. For simplicity, let $o_p=\sum\beta_j\mathbf{x}_{pj}$ be the output of the pth data, then the changes in the weight are set to be $\Delta\beta_{pj}=\eta(\gamma_p-o_p)\mathbf{x}_{pj}$ when using a steepest descent approach. In this case, for the $(k+1)$th iteration, the weights are updated as $\beta_j(k+1)=\beta_j(k)+\Sigma_p\Delta\beta_{pj}$. The learning procedure iterates until a stopping criterion is met. An alternative solution with a single learning step is achieved by the Moore-Penrose pseudo-inverse [27], [28]. In this case, the weight β can be computed by $\beta=\mathbf{d}^{+}\mathbf{Y}$, where + represents the Moore-Penrose pseudo inverse. An alternative to Moore-Penrose pseudo inverse is the regularized least square with an extra regularization parameter C, $\beta=(\mathbf{d}^{\top}\mathbf{d}+C\mathbf{I})^{-1}\mathbf{d}^{\top}\mathbf{Y}$, which is equivalent to having another penalty on the L2 norm of β and is reported to perform better than the Moore-Penrose pseudo inverse in most cases [29].

In [29], the authors conducted a comprehensive evaluation on the variants of RVFL for classification, which showed that the direct links from the input layer to the output layer usually help to improve the performance. A similar conclusion can be found in context of time series forecasting [30][2] and computer vision [31]. In this work, we incorporate ensemble learning to improve the performance of a single RVFL. Following the pipeline in [8], we generate 500 basic RVFL models. We first train a single RVFL which tunes all the parameters on the training data. Then we generate the parameters for each base model by adding a small noise to the tuned parameters to generate different RVFL base models. We choose the RVFL with direct link and with bias as recommended in [29]. The details of the parameter settings are presented in Table III.

In [32], the authors proposed a regression ensemble based on negative correlation learning (NCL) [33]. The basis functions of the base RVFL models are generated randomly and a penalty term is integrated to encourage diversities among the ensemble models so that base RVFL models are uncorrelated with each other. The parameters of each RVFL network can be determined by solving a linear equation. However, this method can only be applied to small ensemble sizes $\mathcal{K}$ since it needs to have a matrix inversion of size $(\mathcal{K}\times S)\times(\mathcal{K}\times S)$, where $\mathcal{K}$ and $\mathbf{S}$ denote the ensemble size and the number of enhancement neurons in a base RVFL model, respectively.

Different from [32], our simple RVFL ensemble is scalable to both large ensemble size and a large number of enhancement neurons. Most importantly, it can naturally fit into a parallel computing environment because the learning of each base model is decoupled. The proposed method strikes a good trade-off between the "strength and correlation". On the one hand, each base model is stable because the parameters are within a small neighbourhood of the optimal ones. On the other hand, each base model is not correlated because of assigning each base model slightly different parameters.

[2]As direct link emulates time-delayed inputs as in the finite impulse response (FIR) filter, removing the direct links degrades the performance of RVFL in a statistically significant manner.

VI. Co-Trained Kernel Ridge Regression Ensemble

The kernel matrix in Eqn. (11) plays a key role in KRR. Actually different kernels may correspond to different notions of similarity, and using a specific kernel may lead to a risk of bias and result in inferior classification result. One may use an ensemble of kernels to tackle this problem. However, the time complexity of the solution is around $O(N^3)$, where N is the number of data samples. When N is large, too many kernels may become computationally prohibitive. In this work, we propose to train a couple of KRR models jointly. We show that the proposed method, named as Co-KRR, performs competitively against the state-of-the-art RaF classifier.

More specifically, with two kernels, the proposed method solves the following problem:

$$E(\alpha)=\min_{\alpha}\sum_{i=1}^{2}\left(\left\|\mathbf{Y}-\mathbf{K}_i\alpha_i\right\|^2+C_1\alpha_i^T\mathbf{K}_i\alpha_i\right)+C_2\sum_{i,j=1}^{2}\left\|\mathbf{K}_i\alpha_i-\mathbf{K}_j\alpha_j\right\|^2. \tag{27}$$

The first part in Eqn. (27) forces each individual model to be as accurate as possible with respect to the desired output $\mathbf{y}$. The second part $(\alpha_i^T\mathbf{K}_i\alpha_i)$ denotes the Tikhonov regularization in a reproducing kernel Hilbert Space. The third part regularizes each model in a pair-wise manner which weights the influence of pairwise disagreements. The objective function in Eqn. (27) allows high performance to be achieved since it is usually unlikely for compatible and accurate classifiers trained on independent kernels/features to agree on an incorrect label.

When $C_2 = 0$, the proposed method is equivalent to two independent KRR methods. The proposed method can also be regarded as a purely supervised version of the co-regularized multi-view learning [34].

It is straight-forward to obtain the solution of Eqn. (27) by setting the differentiation of E with respect to α_i to be 0.

Let

$$\mathbf{G}_i = (2C_2 + 1)\mathbf{K}_i\mathbf{K}_i + C_1\mathbf{K}_i, \quad (28)$$

we can obtain the solution for Eqn. (27)

$$\begin{aligned}\alpha_1 &= (\mathbf{G}_1 - 4C_2^2\mathbf{K}_1\mathbf{K}_2\mathbf{G}_2^{-1}\mathbf{K}_2\mathbf{K}_1)^{-1} \\ &\quad \times (\mathbf{K}_1\mathbf{Y} + 2C_2\mathbf{K}_1\mathbf{K}_2\mathbf{G}_2^{-1}\mathbf{K}_2\mathbf{Y}), \\ \alpha_2 &= (\mathbf{G}_2 - 4C_2^2\mathbf{K}_2\mathbf{K}_1\mathbf{G}_1^{-1}\mathbf{K}_1\mathbf{K}_2)^{-1} \\ &\quad \times (\mathbf{K}_2\mathbf{Y} + 2C_2\mathbf{K}_2\mathbf{K}_1\mathbf{G}_1^{-1}\mathbf{K}_1\mathbf{Y}).\end{aligned} \quad (29)$$

Let $\nu = 2C_2 + 1$ and notice

$$\mathbf{G}_i^{-1} = (\nu\mathbf{K}_i + C_1\mathbf{I})^{-1} \times \mathbf{K}_i^{-1}. \quad (30)$$

Substituting Eqn. (28) into Eqn. (29), we have the following closed-form solution:

$$\begin{aligned}\alpha_1 &= ((\nu\mathbf{K}_1 + C_1\mathbf{I}) \\ &\quad - 4C_2^2\mathbf{K}_2(\nu\mathbf{K}_2 + C_1\mathbf{I})^{-1}\mathbf{K}_1)^{-1} \\ &\quad \times (\mathbf{Y} + 2C_2\mathbf{K}_2(\nu\mathbf{K}_2 + C_1\mathbf{I})^{-1}\mathbf{Y}), \\ \alpha_2 &= ((\nu\mathbf{K}_2 + C_1\mathbf{I}) \\ &\quad - 4C_2^2\mathbf{K}_1(\nu\mathbf{K}_1 + C_1\mathbf{I})^{-1}\mathbf{K}_2)^{-1} \\ &\quad \times (\mathbf{Y} + 2C_2\mathbf{K}_1(\nu\mathbf{K}_1 + C_1\mathbf{I})^{-1}\mathbf{Y}).\end{aligned} \quad (31)$$

The proposed method is contradictory to the existing NCL [33] framework in which each base model is forced to be diverse in order to decorrelate the prediction error of unstable learners. However, NCL was originally proposed to regularize neural network ensembles. Without negative correlation constraints, all base unstable neural network models may converge to the same local minimal solution. However, the proposed method can be regarded as a positive correlation learning method. It is effective because here each kernel is constructed with either different source of features or different kernel functions. It is unlikely for the inherently diversified kernels to have the same "wrong" solutions for a particular data sample considering that KRR is a stable learner [9]. The positive correlation constraint can force the output of one "poor" model to be as accurate as the other "good" model in the ensemble. We also conducted experiments with NCL and found that the results were worse than the proposed method here, as we expected.

Our method here trains two KRRs jointly. Though it can also be extended to any large number of kernel matrices, it will become computationally prohibitive. We have presented the extension in Appendix B. More importantly, it has been found empirically that two kernels can usually lead to satisfactory results and outperform all the methods in [8]. However, one can simply treat co-trained KRRs as simple base models and apply any existing ensemble learning strategy for a larger number of co-trained kernels.

In recent years, MKL has become a hot topic in machine learning and computational intelligence [10]. MKL combines an ensemble of different kernels which may correspond to using different notions of similarity or may be using information coming from multiple sources, such as different representations or different feature subsets. Unlike a typical ensemble framework, most MKL based methods attempt to optimize the combination parameters for each kernel. In our case, we can simply aggregate the results for each KRR by averaging or choosing the maximum response.

An alternative of the above methods is the randomization based method [25] with closed form solution. One pioneering work on training neural network with randomization can be found in [26]. The RVFL [26] neural network model can be regarded as a semi-random realization of the functional link neural networks, which is demonstrated in Fig. 4.

VII. Experiments

All the 121 datasets are from the UCI repository [35]. Details of each dataset are presented in Appendix C. We normalize each feature by removing the mean and dividing the resulting feature values by its variance, as done in [8]. One training set and one test set (each with 50% of the available patterns) are generated from each dataset, where randomized stratified sampling is employed to make sure that the training set and test set have the same number of patterns in each class. Parameter tuning is performed on this couple of sets, which favors those parameters with the best performance on the test set [8]. Then, with the selected values for these parameters, a 4-fold cross validation is performed on the whole dataset [8]. However, for some datasets, where the training-testing partitions are already available (such as annealing and audiology-std, among others), the classifier is trained and tested on the respective datasets. In this case, the test result is calculated on the test set. All the data partitions are the same as in [8].

A. Oblique DT Ensemble

We choose MPRaF-P from [12] as a variant of RaF since it has fewer parameters to tune than MPRaF-T. For RoF, we choose MPRRoF-P for the same reason. For simplicity, we rename them as MPRaF and MPRoF in this work, respectively. For RaF, we only tune the number of features randomly selected at each node, as done in [8]. We do not tune any parameter for RoF and they are set to the default values as done in [12]. The detailed results of oblique DT ensemble are presented in Appendix D. The results of "parRaF_t" are copied from the best performing classifier in [8].

Our proposed oblique DT forest performs better than original RaF in

TABLE 4 Statistical comparisons of the 20 best performing methods.

	MPRaF	CoKRR-max	MPRRoF	parRaF_t	RaF_t	svm_C	svmPoly_t	KRR^3	RaForest_R	svmRadialCost_t	RVFL ensemble	svmRadial_t	C5.0_t	avNNet_t	nnet_t	BG_LibSVM_w	pcaNNet_t	mlp_t	RotationForest_w	RRaF_t
MPRaF							+		+		+	+	+	+	+	+	+	+	+	+
CoKRR-max							+		+	+	+	+		+	+	+	+	+	+	+
MPRRoF											+		+	+	+	+	+	+	+	+
parRaF_t													+		+		+	+	+	+
RaF_t													+		+		+	+	+	+
svm_C											+					+		+	+	+
svmPoly_t	–	–															+		+	
KRR^3															+		+	+	+	+
RaForest_R	–	–																+	+	+
svmRadialCost_t		–															+		+	
RVFL ensemble	–	–	–			–											+			+
svmRadial_t	–	–																	+	
C5.0_t	–		–	–	–															
avNNet_t	–	–	–																	
nnet_t	–	–	–	–	–			–												
BG_LibSVM_w	–	–	–			–														
pcaNNet_t	–	–	–	–	–		–	–		–	–									
mlp_t	–	–	–	–	–	–		–	–											
RotationForest_w	–	–	–	–	–	–	–	–	–	–		–								
RRaF_t	–	–	–	–	–	–		–	–		–									

+ means that the method in the corresponding row is statistically significantly better than the method in the corresponding column. Similarly, – indicates that the method in the corresponding row is statistically significantly worse than the method in the corresponding column. Note in the original paper [8], the 6th rank method is termed as "elm_kernel_m". However, based on the reviewers' comments, we learn that this method is identical to the KRR according to [14], [15], and Section 12.3.7 in [16].

most cases, as expected from our previous study that used only about 40 datasets [12]. Note that the original RoF achieves only the 15th ranking in [8], which indicates a significant performance gap compared with RaF. Interestingly, here we find that MPSVM based RoF outperforms the original RaF.

B. RVFL Ensemble

The parameter settings for RVFL ensemble are summarized in Table III. We follow the same pipeline as done in [8] and the results are summarized in Appendix E.

According to the work in [36], if there are only two methods and N_d datasets, the method which wins at least $N_d/2 + \sqrt{N_d}$ times is considered as statistically significantly better than the other. In this case, we can see RVFL ensemble wins 79 ($> 60 + 11 = 71$) times. Hence we consider the proposed diversified RVFL ensemble is statistically significantly better than a single RVFL model. We also compare the proposed method with a typical bagging RVFL ensemble, where each RVFL model uses the optimized parameter and was trained on a bootstraped version of the training set. Experimental results show that the proposed ensemble method win 75 times and lose 34 times among the 121 datasets. Our results are in line with the ensemble learning theory which advocates for model diversity [2], [37]. In particular, by varying the parameters of each base model, we are able to manipulate the architectures for different RVFL models and finally manipulate the set of *hypotheses accessible to a learner* [37]. This is also in line with a commonly used strategy in neural network ensemble, where one can choose a different structure and a starting point for each back-prop based SLFN.

C. Co-trained KRR Ensemble

Following the same pipeline in [8], the regularization parameter C in Eqn. (9) is set to be 2^{λ} and λ is set to be -14:5. We use Gaussian kernel for all cases and the kernel parameter is set to be 2^{g} and g is set to be -16:8. For the Co-KRR, we firstly generate one extra feature set by a linear transformation of the original features. More specifically, we generate a random projection matrix $\mathbf{W}$, where each entry is sampled from normal distribution $\mathcal{N}(0,1)$. The size of $\mathbf{W}$ is $M \times 2M$, where M is the dimensionality of the original feature set. We

TABLE 5 Friedman rank of different KRR methods.

METHOD	RANK
Co-KRR-max	3.6364
Co-KRR-avg	3.6942
2KRR-max	3.7479
2KRR-avg	3.7893
KRR-O	3.8554
KRR-C	4.0744
KRR-T	5.2025

tune these two base models separately within the same pipeline. After that, we fix these parameters as in Eqn. (27) and further tune the parameter C_2. The range of C_2 is set to be exactly the same as C_1.

The output of the two models can be obtained by either averaging the results or choosing the one with the maximal response. In Table XI in Appendix F, the second and third columns denote the average result and the one with maximal response of the Co-KRR, respectively. The fourth column, denoted as "2KRR-avg", represents the average results of the two independent KRR. 2KRR-avg can be regarded as setting the parameter C_2 in Eqn. (27) to be 0. In this case, two KRR models are trained independently with their own features. Similarly, the "2KRR-max" means the maximal result of the two KRR models. The last three columns represent the single KRR models. "KRR-O" and "KRR-T" stand for the single KRR model trained on the original features and the transformed features, respectively. In order to show the advantage of the proposed method, we also train a KRR model on the concatenation of the original and transformed feature sets, which is named as "KRR-C" in Table XI in Appendix F. We summarize the Friedman rank of each method in Table V. We can see that for both Co-KRR and 2KRR methods, selecting the one with maximal response leads to a better performance. Co-KRR methods outperform 2KRR methods, indicating that the effectiveness of the proposed positive correlation constraint in Eqn. (27). Moreover, using the same statistical test, we found that the KRR-T is statistically significantly worse than all other methods.

TABLE 6 20 best performing out of the 183 methods. "*" indicates the methods introduced in this work. We follow the names in [8] and the other results are copied from [8].

METHOD	RANK	FRIEDMAN RANK	MEAN ACCURACY
MPRaF*	1	33.10	82.05
CoKRR-max*	2	34.13	81.79
MPRRoF*	3	34.56	82.35
parRaF_t	4	34.97	81.96
RaF_t	5	35.16	82.30
svm_C	6	36.78	81.77
svmPoly_t	7	40.36	80.31
KRR[3]	8	41.65	81.52
RaForest_R	9	41.67	81.94
svmRadialCost_t	10	42.47	80.54
RVFL ensemble*	11	44.27	80.94
svmRadial_t	12	44.78	80.17
C5.0_t	13	45.22	80.56
avNNet_t	14	46.54	79.37
nnet_t	15	47.99	79.52
BG_LibSVM_w	16	49.44	80.77
pcaNNet_t	17	49.61	78.65
mlp_t	18	49.82	80.25
RotationForest_w	19	50.20	80.61
RRaF_t	20	52.64	80.92

We also conducted an extra experiment to compare the proposed Co-KRR with MKL. Most MKLs employ SVM as their base model and thus do not scale well for a larger number of kernels. In our experiment, we set the number of kernels to be six. We use the ultra-fast implementation of sparse MKL method in [38] as it is reported to be more accurate than other MKL methods. Moreover, it can lead to a sparse solution which can eliminate some poorly designed kernels. We firstly optimize the kernel parameters for an SVM with the same pipeline in [8]. Then we follow the approach in [39] to assign different kernel parameters to each base SVM by slightly perturbing the optimized parameters. Specifically, we assign each SVM with the kernel parameter $\delta \times g$, where δ is a uniform variable in $(0, 1)$ and g is the optimal parameter for a Gaussian kernel. We also manually set δ to be 1 for one of the SVMs in MKL to make sure at least one SVM is optimized. We use the suggested values in [38] for other parameters. For "miniboone" dataset we fail to get results for MKL because of high computational complexity. Finally, Co-KRR with only two kernels outperforms MKL on 104 out of the 120 datasets, demonstrating superior performance of the co-KRR over MKL.

D. Overall Comparisons Among Best Performing Methods

In this section, we compare the performance of MPRaF, MPRRoF, RVFL ensemble and Co-KRR-max, together with the 179 classifiers in [8][3] Thus, in total there are 183 classifiers and 121 datasets. For clarity, we present the overall 20 best performing methods in Table VI.

In Table VI, MPRaF achieves the 1st rank and MPRRoF achieves the 3rd rank. The Co-trained KRR method achieves the second best performance while KRR achieves the 8th rank.

[3]See footnote 1.

Although the *No-Free-Lunch* theorem indicates that no classifier can always be the best, in the experiments, our proposed methods can achieve the best performance on almost all datasets. The RVFL classifier, with the Friedman rank of 44.27, achieves the 11th rank. The porposed RVFL method outperforms all other neural network methods in [8], indicating the feasibility of the randomized neural network ensembles.

E. Analysis

According to [36], sign test [40] is employed to test the statistical significance between each pair of classifiers among the 20 best performing methods. If two algorithms are equivalent as assumed under the null-hypothesis, each should win on approximately $N_d/2$ out of N_d datasets. The number of wins is distributed according to the binomial distribution. For a greater number of datasets, the number of wins under the null-hypothesis is distributed according to $\mathcal{N}(N_d/2, \sqrt{N_d}/2)$, which allows the use of z-test: if the number of wins is at least $N_d/2 + 1.96\sqrt{N_d}/2$ (or, for a quick rule of a thumb, $N_d/2 + \sqrt{N_d}$), the algorithm is significantly better with $p < 0.05$. In this case, if one algorithm wins more than 71.28 times on the 121 datasets, it is considered as statistically significantly better than the other one. The statistical results are presented in Table IV. The empty entry indicates that no statistically significant difference is observed. + means the method in the corresponding row is statistically significantly better than the method in the corresponding column. Similarly, – indicates the method in the corresponding row is statistically significantly worse than the method in the corresponding column. We can see that among the six best performing classifiers, no statistical significance is observed. MPRaF is statistically significantly better than the methods in the rank range of 11-20. The results of CoKRR-max and MPRRoF-P are almost the same.

VIII. Conclusion

This work added value to the existing ensemble classification literature in a significant manner. We proposed an RVFL ensemble and a co-trained KRR ensemble. We also benchmarked the detailed results of our previous work [12] on 121 datasets and compared it with additional state-of-the-art methods in [8]. All these ensemble methods (oblique DT ensemble, Co-trained KRR and RVFL ensemble) performed competitively against the state-of-the-art classification methods. Thus, we anticipate that these ensemble methods could be considered as new baselines for the practitioners owing to their superior performances and the availability of the source codes from authors' homepages.

Appendix

Contents of the appendix are available from "Publications" at http://www.ntu.edu.sg/home/epnsugan/ and https://sites.google.com/site/zhangleuestc/home.

References

[1] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*. New York: Springer, 2000, pp. 1–15.

[2] Y. Ren, L. Zhang, and P. N. Suganthan, "Ensemble classification and regression-recent developments, applications and future directions [review article]," *IEEE Computational Intell. Mag.*, vol. 11, no. 1, pp. 41–53, Feb. 2016.

[3] T. G. Dietterich, "Ensemble learning," *The Handbook of Brain Theory and Neural Networks*. 2002, vol. 2, pp. 110–125.

[4] P. Domingos, "A unified bias-variance decomposition," in *Proc. Int. Conf. Machine Learning*, 2000, pp. 231–238.

[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Jan. 2001.

[6] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996.

[7] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Machine Learning*, 1996, vol. 96, pp. 148–156.

[8] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3133–3181, Oct. 2014.

[9] G. Valentini and T. G. Dietterich, "Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods," *J. Mach. Learn. Res.*, vol. 5, pp. 725–775, July 2004.

[10] M. Gönen and E. Alpaydin, "Multiple kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 2211–2268, Jul., 2011.

[11] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht, "On oblique random forests," in *Proc. Machine Learning and Knowledge Discovery in Databases Conf.*, Sept. 2011, pp. 453–469.

[12] L. Zhang and P. N. Suganthan, "Oblique decision tree ensemble via multisurface proximal support vector machine," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2165–2176, Oct. 2015.

[13] L. Zhang and P. N. Suganthan, "Random forests with ensemble of feature spaces," *Pattern Recogn.*, vol. 47, no. 10, pp. 3429–3437, Oct. 2014.

[14] C. Saunders, A. Gammerman, and V. Vovk, "Ridge regression learning algorithm in dual variables," in *Proc. 15th Int. Conf. Machine Learning*, 1998, pp. 515–521.

[15] S. An, W. Liu, and S. Venkatesh, "Face recognition using kernel ridge regression," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007, pp. 1–7.

[16] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning. Springer Series in Statistics*. Berlin, Germany: Springer, 2001, vol. 1.

[17] S. Ren, X. Cao, Y. Wei, and J. Sun, "Global refinement of random forest," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2015, pp. 723–730.

[18] L. Breiman, *Classification and Regression Trees*. London, U.K.: Chapman & Hall, 1984.

[19] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.

[20] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 10, pp. 1619–1630, Oct. 2006.

[21] S. Haykin and N. Network, "Neural networks: A comprehensive foundation," *Neural Netw.*, vol. 2, no. 2004, Dec. 2004.

[22] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Netw.*, vol. 1, no. 4, pp. 295–307, 1988.

[23] Y.-H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Boston, MA: Addison-Wesley Longman, 1989.

[24] M. Gori and A. Tesi, "On the problem of local minima in backpropagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 1, pp. 76–86, Jan. 1992.

[25] L. Zhang and P. Suganthan, "A survey of randomized algorithms for training neural networks," *Inform. Sci.*, vol. 364, pp. 146–155, Oct. 2016.

[26] Y.-H. Pao and Y. Takefji, "Functional-link net computing: theory, system architecture, and functionalities," *IEEE Comput. J.*, vol. 25, no. 5, pp. 76–79, May. 1992.

[27] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Trans. Neural Netw.*, vol. 6, no. 6, pp. 1320–1329, Nov. 1995.

[28] Y.-H. Pao and S. M. Phillips, "The functional link net and learning optimal control," *Neurocomputing*, vol. 9, no. 2, pp. 149–164, Oct. 1995.

[29] L. Zhang and P. Suganthan, "A comprehensive evaluation of random vector functional link networks," *Inform. Sci.*, vol. 367, pp. 1094–1105, Nov. 2016.

[30] Y. Ren, P. Suganthan, N. Srikanth, and G. Amaratunga, "Random vector functional link network for short-term electricity load demand forecasting," *Inform. Sci.*, vol. 367, pp. 1078–1093, Nov. 2016.

[31] L. Zhang and P. N. Suganthan, "Visual tracking with convolutional random vector functional link network," *IEEE Trans. Cybernetics*, to be published.

[32] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Inform. Sci.*, vol. 264, pp. 104–117, Apr. 2014.

[33] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Netw.*, vol. 12, no. 10, pp. 1399–1404, Dec. 1999.

[34] V. Sindhwani, P. Niyogi, and M. Belkin, "A co-regularization approach to semi-supervised learning with multiple views," in *Proc. ICML Workshop on Learning with Multiple Views*, 2005, pp. 74–79.

[35] M. Lichman. (2013). UCI machine learning repository. [Online]. Available: http://archive.ics.uci.edu/ml

[36] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[37] G. Brown, J. Wyatt, R. Harris, and X. Yao, "Diversity creation methods: a survey and categorisation," *Inform. Fusion*, vol. 6, no. 1, pp. 5–20, Mar. 2005.

[38] F. Orabona and L. Jie, "Ultra-fast optimization algorithm for sparse multi kernel learning," in *Proc. 28th Int. Conf. Machine Learning*, 2011, pp. 249–256.

[39] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[40] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. Boca Raton, FL: CRC, 2003.