EVALUATIONS OF RANDOM VECTOR FUNCTIONAL LINK AND KERNEL RIDGE REGRESSION

Kong Lingdong

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Email: KONG0129@e.ntu.edu.sg, **Matri. Number:** G1902089A

ABSTRACT:

This programming assignment presents the evaluations and comparisons of algorithms learnt during course EE6227: Genetic Algorithms & Machine Learning. Specifically, we will discuss the random vector functional link (RVFL) neural network [1], deep RVFL (dRVFL) [2], ensemble deep RVFL (edRVFL) [2], and kernel ridge regression (KRR) [3]. The basic concepts of these algorithms are discussed in detail. Furthermore, the performances of the RVFL model and the edRVFL model (10 layers) with ReLU and sigmoid activation functions and KRR model with RBF kernel are evaluated on 10 UCI datasets. Experimental results verify the effectiveness of these algorithms for solving real-world classification problems.

KEYWORDS: Random Vector Functional Link, Kernel Ridge Regression, Classification.

I. RANDOM VECTOR FUNCTIONAL LINK

The random vector functional link (RVFL) network [1], [4] is a special type of neural networks whose weights of the hidden layers are randomly generated. In this section, we will review the basic concept of RVFL, as well as it variants, i.e., the deep RVFL (dRVFL) and the ensemble deep RVFL (edRVFL).

A. Overview of RVFL

RVFL has a very simple structure (see Figure. 1) and an identical neural expression with the multi-layer perceptron, which can be summarized as follows:

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \beta \cdot g(\boldsymbol{W}\boldsymbol{x} - \boldsymbol{b})$$

where $f(x)$ denotes the output of the neural network, $g(\cdot)$ denotes a non-linear activation function, β denotes the weights of the direct links (from input nodes to output nodes), $\boldsymbol{W} =$

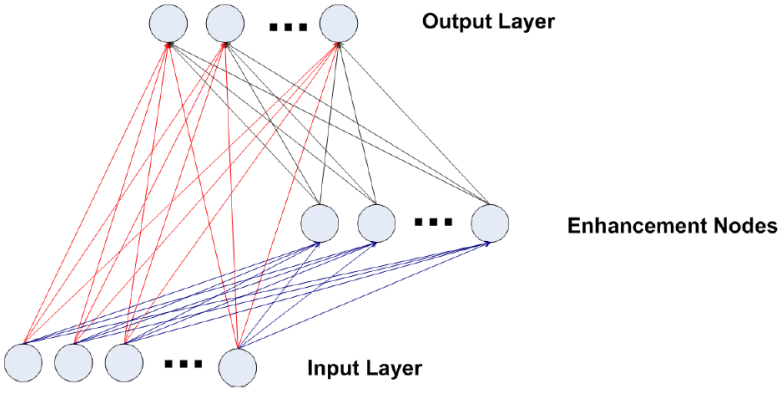

Figure. 1 Neural Architecture of RVFL [5]

$[w_1, w_2, \dots, w_m]$ denotes the weights of the hidden neurons (from input nodes to enhancement nodes), $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$ denotes the input samples, $\boldsymbol{b} = [b_1, b_2, \dots, b_n]$ denotes the biases. The difference between RVFL and multi-layer perceptron is that all the parameters (i.e., weights β, $\boldsymbol{W}$, and biases $\boldsymbol{b}$) of the multi-layer perceptron are to be learned by a training process (e.g., the stochastics gradient descent) while that of RVFL are randomly and independently generated in advance within suitable ranges and remain constrained [6]. Therefore, an error function E which is quadratic can be defined as follows:

$$E(\beta) = E(\hat{\beta}) + \frac{1}{2}(\beta - \hat{\beta})^{\mathrm{T}}\boldsymbol{H}(\beta - \hat{\beta})$$

$$\nabla E(\beta) = \boldsymbol{H}(\beta - \hat{\beta})$$

where $\boldsymbol{H}$ denotes the semi-positive-definite Hessian matrix. If $\boldsymbol{H}$ is strictly positive-definite, i.e., if there exists a well-defined global minimum $\hat{\beta}$, then it can be found based on the following Newton's rule [6]:

$$\hat{\beta} = \beta - \boldsymbol{H}^{-1}\nabla E(\beta) = -\boldsymbol{H}^{-1}\nabla E(0)$$

In [1], the learning of RVFL is defined by $o_i = \beta * d_i, i = 1, 2, \dots, n$, where n denotes the number of input sample, o_i denotes the output of the combined features. This formulates the following optimization problem:

$$\min_{w} ||\boldsymbol{D}\beta - \boldsymbol{Y}||^2 + \lambda||\beta||^2$$

where λ denotes the regularization parameter. Matrix $\boldsymbol{D}$ combines the outputs of the hidden layer with input samples, i.e.,

$$\boldsymbol{D} = \begin{bmatrix} h_1(x_1) & \cdots & h_k(x_1) & x_{11} & \cdots & x_{1m} \\ h_1(x_2) & \cdots & h_k(x_2) & x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ h_1(x_n) & \cdots & h_k(x_n) & x_{n1} & \cdots & x_{nm} \end{bmatrix} = \begin{bmatrix} d(x_1) \\ d(x_2) \\ \vdots \\ d(x_n) \end{bmatrix}$$

The solutions of the above quadratic problem can be summarized as follows [1]:

- In prime space (the number of training samples > the number of total features):

$$\beta = (\lambda \boldsymbol{I} + \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{Y}$$

- In dual space (the number of training samples < the number of total features):

$$\beta = \boldsymbol{D}^{\mathrm{T}}(\lambda \boldsymbol{I} + \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})^{-1}\boldsymbol{Y}$$

According to [7], RVFL is a universal approximator for continuous function on bounded finite dimensional sets, which is with rate of approximation error converges to zero in order $O(\frac{C}{\sqrt{n}})$, where n denotes the number of basic functions and C is independent of n.

B. Evaluation Protocol

The performance of RVFL algorithms is evaluated by Friedman Ranking [8], which can be summarized as follows:

1. Rank the algorithms for each dataset separately. The best performing algorithm will get the rank of #1, the second best will get the rank of #2, etc. In case of ties, average ranks are assigned.
2. Let r_{ij} denote the rank of the j-th algorithm of k algorithms evaluated on the i-th dataset of N datasets. The Friedman test compares the average ranks of algorithms as follows:

$$R_j = \sum_i r_{ij}$$

3. When k and N are large, we can get the following Friedman statistics [8]:

$$X_F = \frac{12N}{k(k+1)}\left(\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right)$$

4. With X_F, the F-score which is distributed based on F-distribution with $(k-1)$ and $(k-1)(N-1)$ degrees-of-freedom can be computed as follows:

$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2}$$

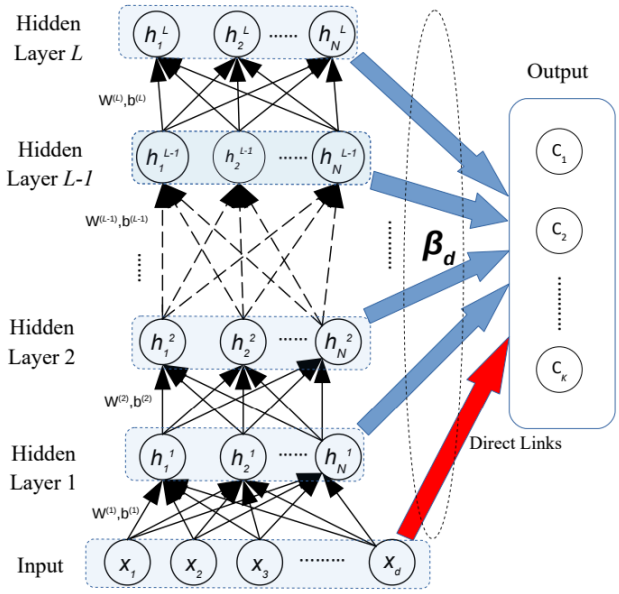

Figure. 2 Neural Architecture of dRVFL [9]

5. If the critical value q_α is based on the Studentized range statistics, then the corresponding average ranks of two algorithms differ by at least the critical difference is:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

C. Deep RVFL (dRVFL)

In Refs. [2] and [9], the deep version of RVFL (termed deep RVFL of dRVFL) is build based on stacking several hidden layers of RVFL network together. The neural network structure of dRVFL is shown in Figure. 2. The output of the first hidden layer of dRVFL is defined as follows:

$$\boldsymbol{H}^{(1)} = g(\boldsymbol{X}\boldsymbol{W}^{(1)})$$

And for every layer > 1, the outputs are:

$$\boldsymbol{H}^{(L)} = g(\boldsymbol{H}^{(L-1)}\boldsymbol{W}^{(L)})$$

where $\boldsymbol{W}^{(1)}$ and $\boldsymbol{W}^{(L)}$ are weights between the input layer and the inter hidden layers, which are randomly generated within suitable ranges and kept fixed during training [2]. $g(\cdot)$ denotes the non-linear activation function. The final input of the output layer is constructed by concatenating the input samples and the outputs of hidden layers, i.e.,

$$\boldsymbol{D} = \left[\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}, \ldots, \boldsymbol{H}^{(L-1)}, \boldsymbol{H}^{(L)}, X\right]$$

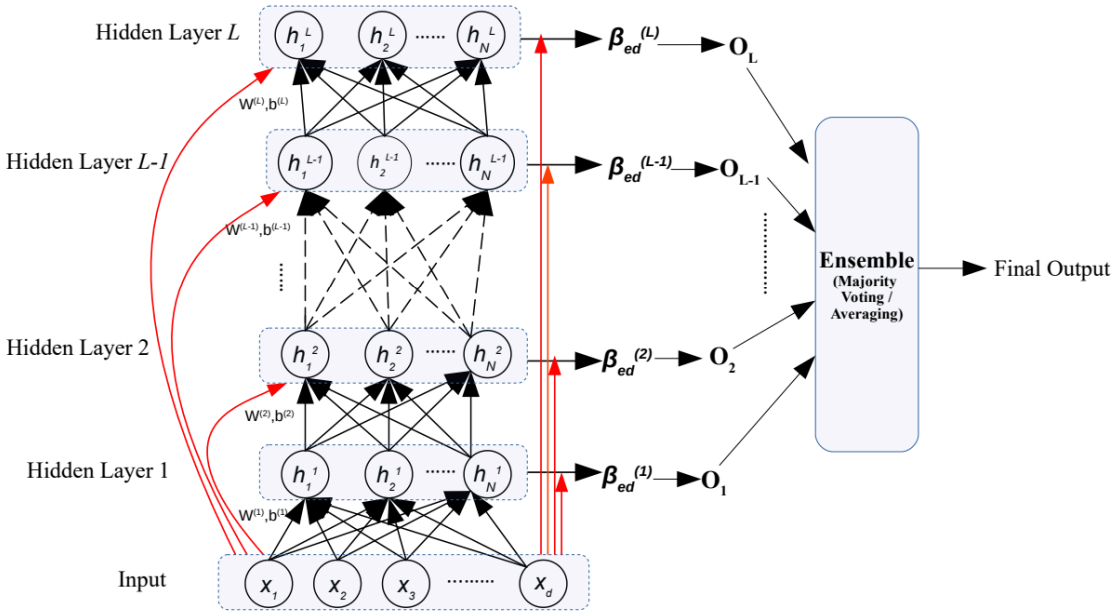

Figure. 3 Neural Architecture of edRVFL [9]

The overall output of the dRVFL is:

$$\boldsymbol{Y} = \beta_d \boldsymbol{D}$$

Similar as RVFL [1], weight β_d can be solved using the primal/dual solutions:

- In prime space (the number of training samples > the number of total features):

$$\beta_d = (\lambda \boldsymbol{I} + \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{Y}$$

- In dual space (the number of training samples < the number of total features):

$$\beta_d = \boldsymbol{D}^{\mathrm{T}}(\lambda \boldsymbol{I} + \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})^{-1}\boldsymbol{Y}$$

D. Ensemble Deep RVFL (edRVFL)

According to [2], the ensemble deep RVFL (edRVFL) serves the following purposes:

1. Instead of using features concatenated from all hidden layers, edRVFL employs rich intermediate features for making the final decision.
2. The ensemble is obtained by a single dRVFL structure with a training cost slightly higher than that of a single dRVFL, but cheaper than training several independent models of dRVFL.
3. Similar as the dRVFL, the edRVFL framework is generic and ant RVFL variant can be used with it.

The network structure of the edRVFL is shown in Figure. 3. Specifically, the input to each

hidden layer is the non-linear transformed features from the proceeding layer as in dRVFL along with the original input features (i.e., the direct links) as in standard RVFL [2]. The direct links act as a regularization for the randomization. The input of the first hidden layer is then defined as follows:

$$\boldsymbol{H}^{(1)} = g(\boldsymbol{X}\boldsymbol{W}^{(1)})$$

And for every layer > 1, the outputs are:

$$\boldsymbol{H}^{(L)} = g(\boldsymbol{H}^{(L-1)}\boldsymbol{W}^{(L)})$$

Similar as RVFL [1] and dRVFL [2], weight β_{ed} of the edRVFL can be solved using the primal/dual solutions:

- In prime space (the number of training samples > the number of total features):

$$\beta_{ed} = (\lambda \boldsymbol{I} + \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})^{-1}\boldsymbol{D}^{\mathrm{T}}\boldsymbol{Y}$$

- In dual space (the number of training samples < the number of total features):

$$\beta_{ed} = \boldsymbol{D}^{\mathrm{T}}(\lambda \boldsymbol{I} + \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D})^{-1}\boldsymbol{Y}$$

II. Kernel Ridge Regression

The section reviews the basic concept of the kernel ridge regression (KRR) [10], which is a special case of the support vector regression.

A. Linear Ridge Regression

The expression of linear ridge regression can be summarized as follows:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^{\mathrm{T}}x_i)^2 + \lambda||\boldsymbol{w}||^2$$

where $x_i = [x_1, x_2, \dots, x_n]^{\mathrm{T}}$ denotes the input samples, $y_i = [y_1, y_2, \dots, y_n]^{\mathrm{T}}$ denotes the label vector, $\boldsymbol{w}$ denotes the weight vector, and λ denotes the regularization parameter. According to [3], the solution of linear ridge regression is:

$$\sum_i (y_i - \boldsymbol{w}^{\mathrm{T}}x_i)x_i = \lambda \boldsymbol{w}$$

$$\boldsymbol{w} = \left(\lambda \boldsymbol{I} + \sum_i x_i x_i^{\mathrm{T}}\right)^{-1}\left(\sum_j y_j x_j\right)$$

B. Kernel Ridge Regression

Considering the following kernel trick [3]:

$$x_i : \varphi(x_i)$$

The solution of the kernel ridge regression can be obtained as follows:

$$\boldsymbol{w} = \sum_i a_i \varphi(x_i)$$

According to [3], the quadratic minimization objective with kernel trick is formulated as follows:

$$\min_a ||\boldsymbol{Y} - \boldsymbol{K}a||^2 + \lambda a^{\mathrm{T}} \boldsymbol{K} a$$

And its solution is just:

$$a = (\lambda \boldsymbol{I} + \boldsymbol{K})^{-1} \boldsymbol{Y}$$

The kernel trick is commonly applied to linear methods to have non-linear characteristics.

III. EXPERIMENTS

In this section, we evaluate the performances of the RVFL and edRVFL (both with ReLU and sigmoid activation functions, respectively) and the kernel ridge regression with RBF kernel for solving real-world classification problems based on 10 UCI open-source datasets. The configurations of these datasets are shown in Table. I.

Table. 1 Dataset Configurations

Dataset	# of Patterns	# of Features	# of Classes
car	1728	6	4
contrac	1473	9	3
hill-valley	606	100	2
led-display	1000	7	10
plant-margin	1600	64	100
plant-shape	1600	64	100
plant-texture	1600	64	100
semeion	1593	256	10
statlog-credit	1000	24	2
statlog-image	2310	18	7

A. Evaluations of RVFL

We are now going to evaluate the classification performance of RVFL [1]. The experimental results of RVFL activated by ReLU and sigmoid activation functions are shown in Table. II and Table. III. Compared with these two tables, we can see that their training time and test time are almost the same, while the one with sigmoid activation function achieves better accuracies on most of the datasets than that of with ReLU activation function.

Table. II Evaluations of RVFL with ReLU activation function

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.9472	0.9776	0.0175	0.9509	0.0028	0.9450
contrac	0.5526	0.55637	0.0011	0.5763	0.0014	0.4482
hill-valley	0.7028	0.7647	0.0109	0.7119	0.0025	0.6574
led-display	0.7363	0.7625	0.0015	0.7000	0.0018	0.7175
plant-margin	0.7859	0.9750	0.0140	0.8031	0.0027	0.7828
plant-shape	0.6281	0.9164	0.0186	0.6188	0.0031	0.6195
plant-texture	0.8022	0.9765	0.0170	0.8563	0.0034	0.7912
semeion	0.8799	0.9929	0.0331	0.8997	0.0046	0.8022
statlog-credit	0.7575	0.7788	0.0009	0.7700	0.0009	0.6525
statlog-image	0.9513	0.9762	0.0213	0.9524	0.0024	0.9405

Table. II Evaluations of RVFL with sigmoid activation function

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.9674	0.9913	0.0181	0.9624	0.0024	0.9616
contrac	0.5551	0.6197	0.0164	0.5797	0.0024	0.4643
hill-valley	0.7069	0.7307	0.0182	0.7037	0.0027	0.6801
led-display	0.7325	0.7600	0.0314	0.7150	0.0162	0.7150
plant-margin	0.7680	0.9789	0.0207	0.7813	0.0038	0.7523
plant-shape	0.5875	0.8914	0.0196	0.5531	0.0036	0.5703
plant-texture	0.8030	0.9828	0.0200	0.8750	0.0037	0.8014

semeion	0.8563	0.9914	0.0296	0.9060	0.0043	0.7661
statlog-credit	0.7725	0.8088	0.0018	0.7500	0.0018	0.6525
statlog-image	0.9497	0.9719	0.0194	0.9286	0.0030	0.9313

B. Evaluations of edRVFL

Now, the classification performance of an edRVFL model [2] with 10 hidden layers is evaluated with ReLU and sigmoid activation functions, respectively. The experimental results are shown in Table. IV and Table. V. By stacking hidden layers, the edRVFL achieves better classification performances than single-layer RVFL [1]. However, the training and test time of edRVFL are much greater than that of RVFL.

Table. IV Evaluations of 10-layer edRVFL with ReLU activation function

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.9732	0.9999	0.3655	0.9711	0.0432	0.9732
contrac	0.5551	0.6273	0.1981	0.6068	0.0354	0.4448
hill-valley	0.7121	0.7833	0.0764	0.6996	0.0223	0.6739
led-display	0.7338	0.7575	0.0030	0.7050	0.0023	0.7150
plant-margin	0.8109	0.9961	0.2824	0.8281	0.0390	0.8023
plant-shape	0.6805	0.9930	0.2732	0.6781	0.0365	0.6688
plant-texture	0.8381	0.9992	0.3814	0.8813	0.0583	0.8303
semeion	0.9254	0.9999	0.4935	0.9279	0.0487	0.8611
statlog-credit	0.7725	0.8425	0.0213	0.7800	0.0115	0.5950
statlog-image	0.9681	0.9930	0.4999	0.9632	0.0538	0.9589

Table. V Evaluations of 10-layer edRVFL with sigmoid activation function

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.9862	0.9999	0.3226	0.9855	0.0424	0.9855
contrac	0.5518	0.5985	0.0195	0.5932	0.0136	0.4270
hill-valley	0.7307	0.7864	0.2578	0.7490	0.0347	0.6987

led-display	0.7325	0.7600	0.0314	0.7150	0.0162	0.7150
plant-margin	0.8172	0.9969	0.3314	0.8406	0.0604	0.8039
plant-shape	0.6422	0.9516	0.4188	0.6125	0.0608	0.6023
plant-texture	0.8421	0.9969	0.3286	0.9000	0.0673	0.8358
semeion	0.9173	0.9999	0.4601	0.9404	0.0470	0.8407
statlog-credit	0.7675	0.8150	0.0811	0.7800	0.0209	0.5963
statlog-image	0.9627	0.9805	0.5100	0.9589	0.0619	0.9405

C. Evaluations of Kernel Ridge Regression

Finally, we evaluate the classification performance of the kernel ridge regression (KRR) [3]. The experimental results are shown in Table. VI. Specifically, KRR requires more training time than that of RVFL [1] and edRVFL [2], while its classification accuracies are lower than that of edRVFL.

Table. VI Evaluations of KRR with RBF kernel

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.9595	0.9776	1.0124	0.9740	0.0123	0.9370
contrac	0.5093	0.5533	0.5262	0.5153	0.0085	0.4108
hill-valley	0.5573	0.5897	0.3312	0.5309	0.0074	0.5553
led-display	0.7487	0.7798	0.1283	0.7050	0.0045	0.7038
plant-margin	0.7711	0.8103	0.9442	0.7938	0.0143	0.7289
plant-shape	0.6164	0.7758	0.8930	0.6281	0.0159	0.6078
plant-texture	0.7670	0.9817	1.4853	0.8344	0.0134	0.7615
semeion	0.9027	0.9669	0.6232	0.9279	0.0135	0.8507
statlog-credit	0.7025	0.7878	0.1239	0.7450	0.0044	0.6625
statlog-image	0.9545	0.9845	2.8762	0.9589	0.0194	0.9529

IV. Conclusion

In this assignment, we have discussed the basic concepts of RVFL, dRVFL, edRVFL, and KRR, which are learned throught EE6227 course. The classification performances of these models are evaluated on 10 UCI datasets. Experimental results verify the effectiveness of these algorithms for solving real-world classification problems.

References

[1] P. N. Suganthan, "Random Vector Functioanl Link (RVFL) Neural Networks," *Course Slides of EE6227*, School of EEE, Nanyang Technological University, Singapore, 2020.

[2] P. N. Suganthan, "Deep RVFL and Ensemble Deep RVFL," *Course Slides of EE6227*, School of EEE, Nanyang Technological University, Singapore, 2020.

[3] P. N. Suganthan, "Kernel Ridge Regression," *Course Slides of EE6227*, School of EEE, Nanyang Technological University, Singapore, 2020.

[4] Y.-H. Pao., S. M. Phillips, and D. J. Sobajic, "Neural-Net Computing and the Intelligent Control of Systems," *International Journal of Control*, vol. 56, no. 2, pp. 263-289, 1992.

[5] L. Zhang and P. N. Suganthan, "A Comprehensive Evaluation of Random Vector Functional Link Networks," Information Science, vol. 367, pp. 1094-1105, 2016.

[6] D. Husmeier, "Random Vector Functional Link (RVFL) Networks," *Neural Networks for Conditional Probability Estimation*, Springer, London, pp. 87-97, 1999.

[7] B. Igelnik and Y.-H. Pao, "Stochastic Choice of Basis Functions in Adaptive Function Approximation and the Functional-Link Net," *IEEE Transactions on Neural Network*, vol. 6, pp. 1320-1329, 1995.

[8] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?" *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3133-3181, 2014.

[9] R. Katuwal, P. N. Suganthan, and M. Tanveer, "Random Vector Functional Link Neural Network based on Ensemble Deep Learning," *arXiv preprint*, arXiv:1907.00350, 2019.

[10] S. Bernhard, Z. Luo, and V. Vovk, "Empirical Inference," *Springer Science & Business Media*, 2013.