Evaluations of Random Forest and Random Vector Functional Link Algorithms on UCI data sets

Yuhao Zhou[1a]

***Abstract*—This paper presents the progress of completing the second assignment of EE6227. First, the basic concept of several neural network algorithms would be discussed, including the random forest (RF) algorithm, random vector functional link (RVFL) algorithms, its derivatives deep RVFL (dRVFL), and ensemble deep RVFL (edRVFL) algorithms. Later we would evaluate the performance of these algorithms on 10 selected UCI data sets. Finally, the conclusions would be made based on the effectiveness of the algorithms for solving classification problems.**

Keywords—Neural network. Random forest, Random vector functional link networks

I. INTRODUCTION

In this assignment, we select the random forest algorithm and the random vector functional link algorithm to test several data sets to further strengthen the understanding of their principles as well as their advantages and disadvantages. Besides, the influences of activation function would also be discussed in the later sections.

A. Random Forest Algorithm

Random forest [1] is an extended variant of Bagging [2] based on the decision tree. The decision tree is utilized to construct a classification model using the concept of a tree structure. Each internal node represents an attribute, and each branch of internal nodes represents an output of one decision result, finally, the leaf node characterizes a certain category, thus achieving the purpose of classification. The basic process of constructing a random forest is as follows:

1. Using Bagging to generate *m* training samples for T_i
2. Calculate the optimum split using the *n* randomly chosen features in the training set for T_i
3. Return to Step 2 until reach the termination condition if the final node is pure or if:

   number of samples in that node < *minleaf*

Here T_i represents each random tree in the forest.

Bagging and ensemble learning are two critical concepts in this algorithm. Bagging represents the self-sampling integration. Initially, duplicate certain percentages of original training data and then mix them (usually around 65% of distinct samples and 35% duplicates), then this method divides the mixed training set into *m* new training set with the same number of samples to import as input of decision tree. The concept of ensemble learning is to solve the inherent shortcomings of using only one model to improve the accuracy of the classification. Random forest utilizes many decision trees to integrated as a forest and combine all the independent decisions of each tree to predict the outcome. Usually, the ensemble of classifiers would use majority voting for classification problems and averaging for regression problems.

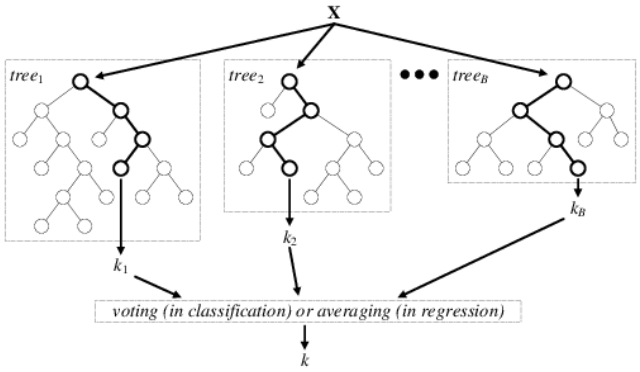

Fig. 1: Structure of Random forest [3]

One commonly used method to optimize the threshold for branching the internal nodes is Gini impurity. In the CART [4] algorithm, Gini impurity refers to the probability that a randomly selected sample would be misclassified in a subset. Gini impurity is the probability of this sample being selected multiplied by the probability of it being misclassified. When all the samples in a node are of one class, the Gini impurity would be 0. The Gini impurity can be calculated as:

$$\text{maximize Gini}_{\text{beforesplit}} - \text{Gini}_{\text{aftersplit}}\text{, where:}$$

$$\text{Gini}(t)_{\text{beforesplit}} = 1 - \sum_{i=1}^{c}\left(\frac{n_{w_i}}{n}\right)^2$$

$$\text{Gini}(t)_{\text{aftersplit}} = \frac{n^l}{n}\left[1 - \sum_{i=1}^{c}\left(\frac{n^l_{w_i}}{n^l}\right)\right] + \frac{n^r}{n}\left[1 - \sum_{i=1}^{c}\left(\frac{n^r_{w_i}}{n^r}\right)\right]$$

here *c* is the total number of classes at the node, *n* is the total samples, *r* and *l* refer to right and left branches.

Within each internal node, there are various methods to find the best hyperplane to find the split, including hill-climbing, simulated annealing, SVM, and genetic algorithm to design a linear classifier.

The main advantages of random forest algorithm are:

[a]Corresponding author: Yuhao Zhou {YZHOU035@e.ntu.edu.sg}
Matriculation Number: G2002124F
Code and data: https://github.com/YuHoChau/NTU_EE6227
The author was with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639789 Singapore.

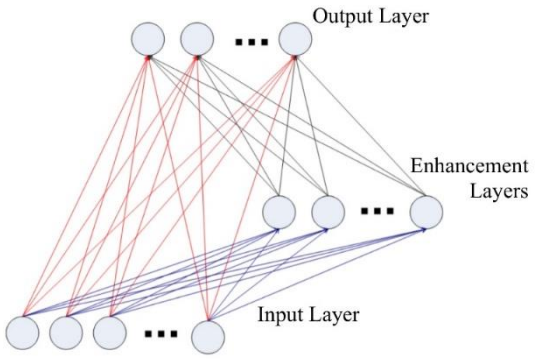

Fig. 2: Structure of RVFL [7]

1. Prevent overfitting of the training data
2. High accuracy among other current algorithms
3. Could be applied to high dimensional input features without features deletion
4. A certain degree of noise resistance
5. Do not need Cross-Validation

The limitations of the random forest algorithm include that when the number of decision trees in a random forest is large, the training time would be long which is not effective.

B. Random Vector Functional Link

Then Random vector functional link (RVFL) network [5] is a neural network that directly connects the input layer with the output layer with randomly generated weights. Besides, there is only one hidden layer (or called enhancement layer) in the structure of RVFL, as shown in Fig. 2.

The randomly generated weights and biases from the input layer to the enhancement layer would be fixed during the training. At the output layer, the input and enhanced features are concatenated. Each feature matrix after the enhancement nodes is as follows:

$$\mathbf{H} = g\left(\mathbf{X}\mathbf{W} + b\right)$$

were $\boldsymbol{g}(\cdot)$ is the activation function, $\mathbf{X} = [x_1, x_2, \ldots, x_n]$ refers to the input training data with n samples and m features, $\mathbf{W} = [w_1, w_2, \ldots, w_n]$ refers the weights for the enhancement nodes, $\boldsymbol{b} = [b_1, b_2, \ldots, b_n]$ is the biases. Note that all the training parameters are randomly generated in the training process within a suitable range and kept fixed. After the enhancement nodes, the expression of H is:

$$\mathrm{H} = \begin{bmatrix} h(x_1) \\ h(x_2) \\ \vdots \\ h(x_n) \end{bmatrix} = \begin{bmatrix} h_1(x_1) & \cdots & h_k(x_1) \\ h_1(x_2) & \cdots & h_k(x_2) \\ \vdots & \ddots & \vdots \\ h_1(x_n) & \cdots & h_k(x_n) \end{bmatrix}$$

The learning of RVFL aims to solve the following problem:

$$\mathrm{o}_i = d_i * \beta,\ i = 1,2, \ldots, n$$

where n refers to the number of training samples, β refers to weights used in the output layer, and o_i denotes the output of combined features. Overall, the RVFL has the following objective function to minimize the errors:

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

were λ refers to the regularization parameter. The matrix D represents the combination of both original input features and the enhanced features and is:

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

If the number of training samples is larger than the total features, in the prime space we could get the following solution:

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

While smaller, in the dual space, the solution is:

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

To evaluate the performance of RVFL, the Friedman Ranking [6] was utilized and can be concluded as follows:

1. Based on the performance of each dataset, rank them as 1^{st} place, 2^{nd} place, …n^{th} place, respectively. If the ranking ties, assign the averaged places
2. r_{ij} refers to the rank of j-th over k algorithms based on the performance on the i-th over N datasets. Then the Friedman Ranking evaluates the average ranking as:

$$R_j = \sum_i r_i^j$$

3. If both k and N are large numbers, the Friedman statistics are as:

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

4. The $\mathrm{F_F}$ is distributed according to F-distribution with k-1 and (k-1)*(N-1) degrees of freedom, where:

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

5. The performance would be considered to be greatly different if the average ranking of two algorithms differs by at least the critical difference at:

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

C. Deep Random Vector Functional Link

Deep random vector functional link (dRVFL) [7] network is a marriage of ensemble learning and deep learning where multiple stacked layers were employed rather than only one enhancement layer compared to the conventional RVFL. The parameters for the hidden layers are randomly generated with a range and then keep fixed. For the output layer, the weights are

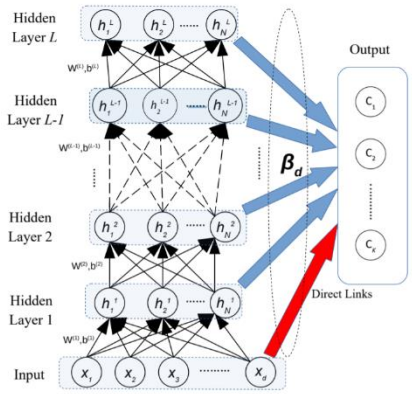

Fig. 3: Structure of dRVFL [7]

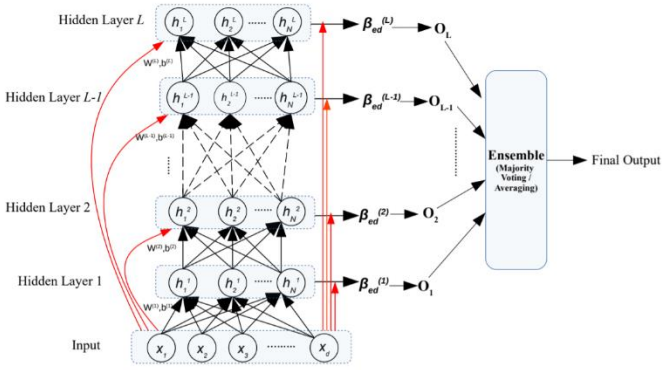

Fig. 4: Structure of edRVFL [7]

calculated like the solution in the conventional RVFL. Figure 3 shows the basic structure of dRVFL.

After the first process of the hidden layer, the output is as follows (here the bias term was omitted for the ease of notation):

$$\mathbf{H}^{(1)} = g\left(\mathbf{X}\mathbf{W}^{(1)}\right)$$

For layers > 1, the outputs are:

$$\mathbf{H}^{(L)} = g\left(\mathbf{H}^{(L-1)}\mathbf{W}^{(L)}\right)$$

where $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(L)}$ are weights randomly generated within a suitable range and kept fixed in the training process. $\boldsymbol{g}(\cdot)$ refers to the activation function. The output layer would have the input $\mathbf{D}$, as:

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

Finally, the overall output of the dRVFL network is:

$$Y = \mathbf{D}\boldsymbol{\beta}_d$$

The solution for the β_d is the same as RVFL that mentioned in the last section. If the number of training samples is larger than the total features, in the prime space we could get the following solution:

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

While smaller, in the dual space, the solution is:

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

D. Ensemble Deep Random Vector Functional Link

The differences between dRVFL and ensemble deep random vector functional link (edRVFL) network are as follow:

1. edRVFL utilizes ensemble module in the output layer (majority voting for classification and averaging for regression problem). The final output of β_d is decomposed into multiple β_{ed}, and each is independently computed.
2. Each hidden layer is directly connected with the input layer. This direct link regularizes for randomization.

Fig. 4 shows the structure of edRVFL neural network. Mathematically, the output (represented by $\mathbf{O}_1$, …, $\mathbf{O}_L$) of the input layer into the first hidden layer is:

$$\mathbf{H}^{(1)} = g\left(\mathbf{X}\mathbf{W}^{(1)}\right)$$

For layers > 1, the outputs are:

$$\mathbf{H}^{(L)} = g\left(\left[\mathbf{H}^{(L-1)}\mathbf{X}\right]\mathbf{W}^{(L)}\right)$$

The weight β_{ed} of edRVFL can be calculated using a similar solution as RVFL/dRVFL. If the number of training samples is larger than the total features, in the prime space we could get the following solution:

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

While smaller, in the dual space, the solution is:

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

Rather than training various models independently as in conventional ensembles, the edRVFL only requires training the network once. However, this algorithm still has certain limitations, including that edRVFL failed to perform convolutions.

Note that in the training of RVFL and its derived algorithms, the K-fold cross-validation should be utilized to verify if the prediction is correct. This cross-validation method divides the sample set *S* into K parts, using (K-1) of them respectively as training set and the remaining one as cross-validation set. Finally, take the average error to evaluate the model.

Based on the above algorithms including RF, RVFL, and its derivatives, in the next section, the experiments would be conducted to evaluate their performance.

II. Experiment

We selected 10 UCI data sets to evaluate the overall performance of the four algorithms discussed in the last section. For the experiment of RVFL, dRVFL, and edRVFL networks, several activation functions would be compared.

Table 1 shows the dataset configuration.

TABLE I: 10 SELECTED UCI DATA SETS

Number	Datasets	Patterns	Features	Classes
1	car	1728	6	4
2	high-valley	606	100	2
3	plant-margin	1600	64	100
4	plant-shape	1600	64	100
5	plant-texture	1600	64	100
6	ozone	2536	72	2
7	yeast	1483	8	10
8	contrac	1473	9	3
9	molec-biol-splice	3190	60	3
10	statlog-german-credit	1000	24	2

TABLE II: EXPERIMENT ON RANDOM FOREST

Dataset	best_acc	train_time	test_acc	test_time	val_acc
car	0.96382	0.145506	0.959538	0.011087	0.962363
high-valley	0.573802	3.47E+02	0.514403	0.269354	0.558315
plant-margin	0.767188	119.3544879	0.825	0.404478	0.7625
plant-shape	0.590625	301.3464841	0.546875	0.257725	0.590625
plant-texture	0.746652	173.9828014	0.809375	0.42185	0.736503
ozone	0.970414	53.9353899	0.978346	0.147	0.966963
yeast	0.565307	1.8424875	0.531987	0.041191	0.565307
contrac	0.5	0.8221511	0.542373	0.021424	0.485582
molec-biol-splice	0.943966	10.5558339	0.948276	0.161207	0.9279
statlog-german-credit	0.7675	2.2874387	0.75	0.028687	0.75625

TABLE III: EXPERIMENT ON RVFL WITH RELU ACTIVATION

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.92776	0.99711	0.0031517	0.913893	0.005942	0.797547
high-valley	0.662637	0.703704	0.0008702	0.703704	0.0013951	0.539139
plant-margin	0.6125	1	0.0021491	0.521094	0.005051	0.41875
plant-shape	0.45625	0.9875	0.0023876	0.447656	0.005425	0.265625
plant-texture	0.640625	1	0.0024633	0.611415	0.007182	0.5375
ozone	0.982283	0.982283	0.0005697	0.968442	0.00124	0.641732
yeast	0.622568	0.808081	0.00083	0.538332	0.002085	0.444685
contrac	0.525639	0.562712	0.0005531	0.521222	0.0008271	0.352601
molec-biol-splice	0.766372	0.915361	0.0037928	0.766066	0.008891	0.565704
statlog-german-credit	0.765	0.85	0.0005615	0.7325	0.001021	0.675

TABLE IV: EXPERIMENT ON DRVFL WITH RELU ACTIVATION

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.921979	1	0.0212417	0.909551	0.076011	0.913292
high-valley	0.662773	0.917695	0.0214641	0.917695	0.03658	0.539071
plant-margin	0.5	1	0.0299609	0.521094	0.109953	0.58125
plant-shape	0.628125	1	0.0127532	0.451563	0.046348	0.459375
plant-texture	0.665625	1	0.0193506	0.581704	0.073818	0.61875
ozone	0.984252	1	0.0103903	0.967456	0.020795	0.978346
yeast	0.589189	0.976431	0.0051058	0.516428	0.012442	0.541892
contrac	0.525731	0.6	0.0006914	0.509338	0.0006534	0.447566
molec-biol-splice	0.75227	0.838558	0.0012827	0.774687	0.00599	0.662982
statlog-german-credit	0.77	0.995	0.003511	0.70375	0.009246	0.72

TABLE V: EXPERIMENT ON EDRVFL WITH RELU ACTIVATION

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.933607	1	0.0265299	0.89508	0.062464	0.907511
high-valley	0.707855	0.942387	0.021862	0.691434	0.063303	0.629645
plant-margin	0.65	1	0.0267311	0.584375	0.073519	0.609375
plant-shape	0.509375	0.990625	0.027992	0.990625	0.067395	0.44375
plant-texture	0.665625	1	0.0195769	0.626271	0.044094	0.640625
ozone	0.982283	0.982283	0.0055986	0.968442	0.015349	0.970472
yeast	0.626216	0.888889	0.0166884	0.569503	0.043269	0.528649
contrac	0.518928	0.59661	0.0035989	0.537351	0.006351	0.437107
molec-biol-splice	0.764829	0.934169	0.0128406	0.795063	0.035707	0.66762
statlog-german-credit	0.78	0.88	0.0082342	0.7225	0.011287	0.68

TABLE VI: EXPERIMENT ON RVFL WITH SIGMOID ACTIVATION

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.930634	0.968208	0.00119	0.913169	0.003328	0.875835
high-valley	0.679167	0.703704	8.01E-04	0.604747	0.001375	0.568306
plant-margin	0.5875	1	0.001788	0.519531	0.005087	0.38125
plant-shape	0.3875	0.99375	0.008963	0.390625	0.005827	0.228125
plant-texture	0.646875	1	0.002242	0.620016	0.006967	0.534375
ozone	0.982283	0.982283	7.38E-04	0.968442	0.001278	0.785433
yeast	0.609279	0.757576	1.12E-03	0.556024	0.003022	0.521667
contrac	0.539245	0.60339	1.34E-03	0.513582	4.13E-03	0.372871
molec-biol-splice	0.766421	0.9279	0.004891	0.77116	0.008429	0.54376
statlog-german-credit	0.79	0.89	1.24E-03	0.73125	0.003022	0.605

TABLE VII: EXPERIMENT ON DRVFL WITH SIGMOID ACTIVATION

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.939388	0.99711	0.020262	0.921129	0.075976	0.898991
high-valley	0.691325	0.773663	9.43E-04	0.650155	9.29E-04	0.60485
plant-margin	0.609375	1	0.02298	0.529688	0.073158	0.5875
plant-shape	0.496875	1	0.023241	0.467188	0.081448	0.403125
plant-texture	0.65	1	0.017575	0.602033	0.068319	0.63125
ozone	0.982283	0.982283	0.001245	0.968442	0.004057	0.980315
yeast	0.642883	0.774411	0.017421	0.566133	0.066371	0.481216
contrac	0.518928	0.640678	1.97E-03	0.516978	4.57E-03	0.393512
molec-biol-splice	0.771138	0.9279	0.033915	0.773511	0.12726	0.633274
statlog-german-credit	0.78	0.885	0.008938	0.7425	0.035334	0.705

TABLE VIII: EXPERIMENT ON EDRVFL WITH SIGMOID ACTIVATION

Dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
car	0.939354	0.976879	0.031428	0.916787	0.080354	0.881616
high-valley	0.703893	0.72428	0.017773	0.614035	0.043027	0.609221
plant-margin	0.625	1	0.024491	0.584375	0.072781	0.590625
plant-shape	0.44375	0.99375	0.033472	0.442188	0.084688	0.240625
plant-texture	0.65625	1	0.028465	0.639562	0.070967	0.653125
ozone	0.982283	0.982283	0.007167	0.968442	0.01528	0.956693
yeast	0.622748	0.811448	0.027056	0.553496	0.071941	0.480991
contrac	0.535913	0.59661	0.006247	0.510187	0.011761	0.396797
molec-biol-splice	0.777398	0.984326	0.091588	0.794279	0.105108	0.72102
statlog-german-credit	0.775	0.865	0.013927	0.7325	0.039077	0.675

A. Experiment on Random Forest

The first algorithm to evaluate is the random forest. The settings of the experiment are:

Number of trees	m_try/m	*minleaf*
500	round(sqrt(size(trainX,2)))	1

The result of the evaluation is shown in Table II. The result showed that the training of the random forest algorithm would take a long time (some even last for hours), this is because the training data has a large number of features.

B. Experiment on RVFL, dRVFL, edRVFL (Relu activation)

Then the evaluation of RVFL neural network and its derivatives were conducted using the relu activation function.

The settings of the experiment on RVFL are:

Number of layers	Number of neurons	Regularization parameter	Scaling parameter	Activation
1	100	0.1	1	Relu

The settings of the experiment on dRVFL and edRVFL are:

Number of layers	Number of neurons	Regularization parameter	Scaling parameter	Activation
10	100	0.1	1	Relu

The result of the experiment is shown in Table III, IV, and V, respectively.

C. Experiment on RVFL, dRVFL, edRVFL (Sigmoid activation)

Utilizing the same parameter listed in the last section, we change the activation function of the experiment from relu to sigmoid to evaluate the influences of activation function on the overall performance of the algorithms. The second activation function chosen is sigmoid. The final results of the performance of RVFL, dRVFL, and edRVFL are shown in Table VI, VII, VIII, respectively.

III. Discussion

Based on the experiment result, we compared the performance of selected algorithms on their training and testing accuracy.

Most of the testing accuracy of using random forest is much better than the RVFL series. Nevertheless, the training time is signification longer.

Compare the performance within the RVFL category, it showed that edRVFL performs higher accuracy on most of the training sets than RVFL and dRVFL. However, the training and test time is much longer, as well.

Using different activation functions, the result also showed that the performance is very close and the sigmoid activation function performs slightly better.

IV. Conclusion

In this assignment, the basic concepts of several algorithms including RVFL, dRVFL, edRVFL, and random forest were discussed. Next, the performances of these algorithms were evaluated on 10 selected UCI datasets. Finally, by analyzing the experiment result, the experiment verified the effectiveness of these algorithms.

References

[1] Breiman L. Random forests. Machine learning. 2001 Oct;45(1):5-32.

[2] Breiman L. Bagging predictors. Machine learning. 1996 Aug;24(2):123-40.

[3] Verikas A, Vaiciukynas E, Gelzinis A, Parker J, Olsson MC. Electromyographic patterns during golf swing: Activation sequence profiling and prediction of shot effectiveness. Sensors. 2016 Apr;16(4):592.

[4] Breiman, L., J. Friedman, C. J. Sto and R. A. Olshen. (1984). Classification and Regression Trees. Chapman & Hall/CRC, Boca Raton, FL

[5] Y. H. Pao, Y. Takefuji, Functional-link net computing: theory, system architecture, and functionalities, IEEE Computer 25 (5) (1992) 76–79.

[6] 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.

[7] Katuwal R, Suganthan PN, Tanveer M. Random vector functional link neural network based ensemble deep learning. arXiv preprint arXiv:1907.00350. 2019 Jun 30.