Using Random Vector Function Link and Kernel Ridge Regression to Build Classification

SUN YIHAN*

[1]*School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore*

SUNY0045@e.ntu.edu.sg Matric Number:G2001779A

Abstract: Through the introduction of random vector functional link, deep random vector functional link and KRR algorithm, this paper explains the basic principle and definition of the algorithm. Adjusting the parameters and activation functions of different algorithms can optimize the performance of these algorithms and the datasets can be more accurately identified and classified after training.

Keywords: RVFL, deep RVFL, KRR

1. Introduction

1) Random vector function link

a. The basis of random vector function link:

The RVFL network is based on the FL (Functional link) network. The RVFL algorithm can solve the problem of the coexistence of linear and non-linear relationships between sample data. The basic framework of the RVFL network is shown in Figure 2.1: The inputs of the output layer in the RVFL network contains the nonlinear transformation feature H and the original input feature X from the hidden layer. If d is the input data feature and N is the number of hidden nodes, then the input of each output node is d+N and outputs of the output layer includes that calculated by the input nodes as well as the hidden nodes which also includes input. The output of the node since the hidden layer parameters are random, they are generated during the training phase and remain unchanged. Only the output weight βneeds to be calculated. This network structure introduces direct connections between the input and output layers to prevent network overfitting. The difference between this algorithm and the traditional feedforward neural network is that the algorithm does not need to set the network training parameters in advance, making the algorithm use process simple, and the connection weight calculation between the RVFL input layer and the hidden layer does not require too much manual intervention. The algorithm is fast to train because of the simple network structure and can perform good classification of linear and nonlinear data that may exist in the data stream.

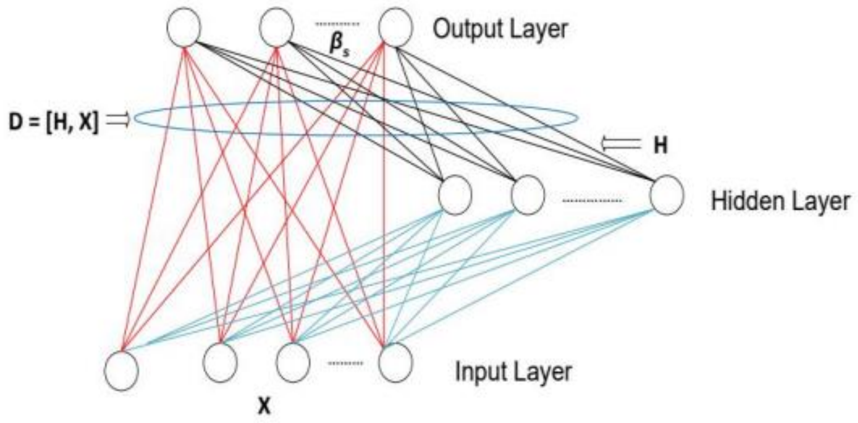

Fig 2.1 RVFL network structure

The RVFL can be expressed as:

$$f(x) = \sum_{i=1}^{n} \beta * g(Wx - b)$$

$f(x)$ means outputs of the neural network, the parameter β presents the weight of the direct links between different nodes, $g(x)$ means the activation function which can be nonlinear, $W = [w_1, w_2, \dots, w_n]$ is a matrix describing the weight of hidden layers, $X = [x_1, x_2, \dots, x_n]$ is a matrix which means the inputs from the input layer, $b = [b_1, b_2, \dots, b_n]$is also a matrix to describe the biases. It is obviously that the parameters of RVFL are random and learned by the training process and original inputs. The error function of RVFL can be described as:

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

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

The matrix H presents the enhancement layer, we want to minimize the error by using the formula. It is evident that a suitable parameter β can reduce the error, therefore the value of the parameter can be calculated by the formula :

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

The result can be optimization as following:

$$\min||D\beta - Y||^2 = E(\hat{\beta}) + \lambda||\beta||^T$$

In above formula, λ is the regularization parameter. D is the final output of the system and D can be presented as following:

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

When the number of training samples is higher than that of total features, it indicates the solution is in prime space:

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

When the number of training samples is less than that of total features, it indicates the solution is in dual space：

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

As we mentioned, λ is the regularization parameter, which is aim to declare the irrelevant items. When the λ tend to be 0, the matrix of D, Y are closer to the target. By using the algorithm RVFL can make an estimation for continuous function whose boundary is limited and the results of error can be converged to order zero, which can be presented as O($\frac{C}{sqrt(n)}$), the function of C is independent of n.

b. Deep random vector function link(dRVFL):

Combine the RVFL network with the deep learning framework of the random neural network to establish a stacked layer model deep random vector function link (dRVFL). The hidden layer of the model generates parameters randomly in the feasible region, and keeps the parameters fixed in each calculation to obtain the output.

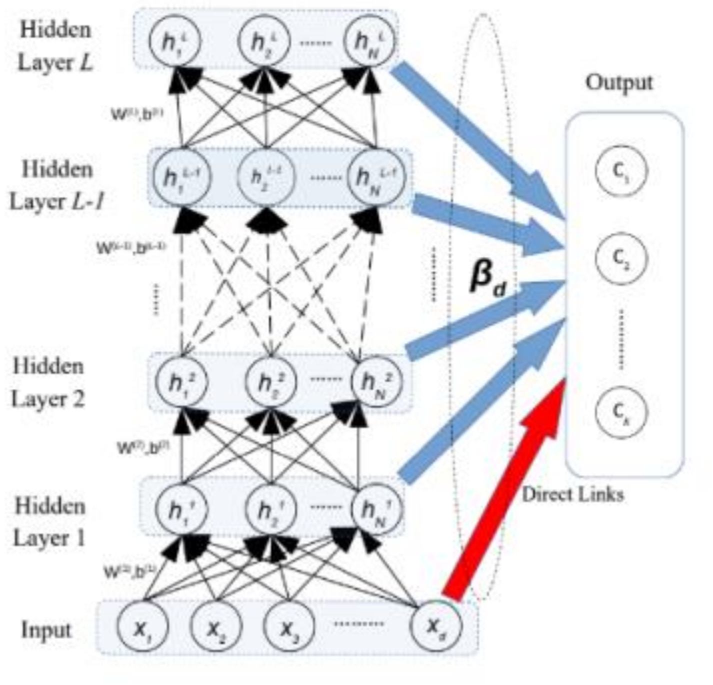

Fig. 2.2 Structure of dRVFL

The structure of dRVFL is shown in fig.2.2. Under this model, the output of the first layer of the hidden layer is:

$$H^{(1)} = g(XW^{(1)})$$

After the output of the first layer is generated, the output of each layer when the number of layer is over 1:

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

For the above two expressions, the parameters$W^{(1)}$, $W^{(L)}$are the weights of the input layer and the hidden layer of the Lth layer. Their values are randomly generated and used in a single It remains unchanged during the calculation. Same as RVFL g(x) means the activation function which can be nonlinear, the input of the output layer is composed of the original input layer and the output of the last hidden layer. The specific expression is：

$$D = [H^{(1)}, H^{(2)}, \dots, H^{(L-1)}, H^{(L)}, X]$$

For dRVFL, the final output of the model is:

$$Y = \beta_d * D$$

dRVFL and RVFL models are optimized in the same way, when the number of training samples is higher than that of total features, it indicates the solution is in prime space:

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

When the number of training samples is less than that of total features, it indicates the solution is in dual space：

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

c. Ensemble deep random vector function link(edRVFL):

The edRVFL model is proposed to achieve the following three purposes and has the following three attributes:

1) It employs intermediate features to make the final decision separately, this property can avoid to use features from intermediate layer as well as redundant link.

2) The model edRVFL is obtained through the dRVFL model, which has a higher cost during training than a single dRVFL, but is more effective and has a lower cost than multiple dRVFL models.

3) edRVFL model is similar as dRVFL model, so it is genetic and multiple variant also can be used in edRVFL.

The structure of edRVFL is shown in fig.2.3. The inputs of hidden layers are the features transformed from previous one, which are similar as the one in dRVFL. When the model try to link different layer, the direct link is likely to standard RVFL and it act as a regularization.

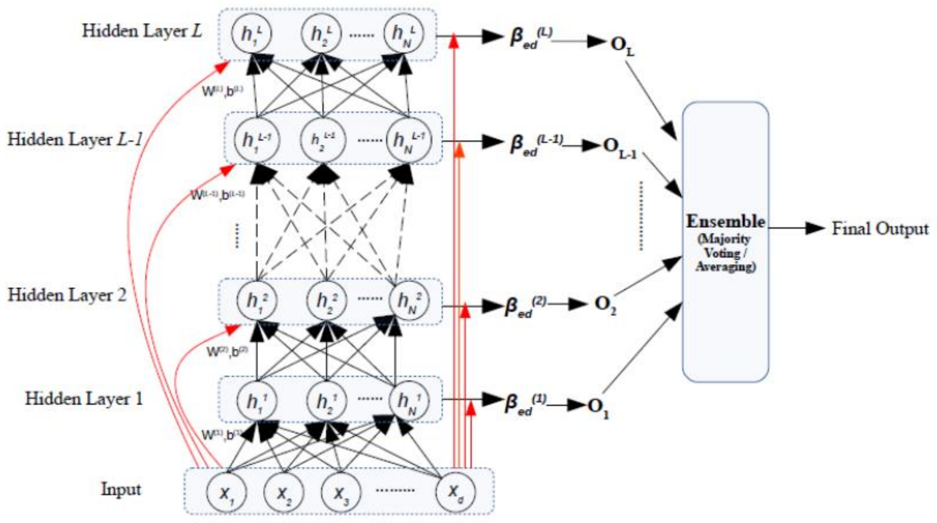

Fig. 2.3 Structure of edRVFL

It is obviously that the inputs of the first hidden layer can be expressed as:

$$H^{(1)} = g(XW^{(1)})$$

After the output of the first layer is generated, the output of each layer when the number of layer is over 1:

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

As we mentioned before, the model is come from dRVFL, so the solution is also the same with dRVFL, when the number of training samples is higher than that of total features, it indicates the solution is in prime space:

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

When the number of training samples is less than that of total features, it indicates the solution is in dual space：

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

2) Kernel Ridge Regression

a. Ridge regression

Ridge regression is a kind of regression method, also called weight decay in machine learning. Ridge regression mainly solves the following two problems: one is the regression when the number of predictor variables exceeds the number of observed variables, and the other is the multicollinearity between the data sets, that is, the correlation between the predictor variables.

The matrix form of regression analysis is as follows:

$$y = \sum_{j=1}^{p} \beta_j * x_j + \beta_0$$

Where x is the predictor variable, y is the observed variable, $\beta_0$and β_j are the required parameters, and β_0 is also called bias. The above objective function is solved by the least square method. In the ridge regression method, the objective function is to minimize the sum of the objective value and the penalty term. Which can be expressed as:

$$\beta^{bridge} = argmin\beta\{\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_i * \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2\}$$

Ridge regression is a least squares regression with a two-norm penalty. This kind of estimation objective of ridge regression is called shrinkage estimator.

b. Linear ridge regression

The linear ridge regression can be expressed as:

$$\frac{1}{n}\sum_{j=1}^{p}(y_j - w^T * x_i)^2 - + ||\lambda||^2$$

The input of the structure is $x_i = [x_1, x_2, \dots, x_n]^T$, the label vector is $y_i = [y_1, y_2, \dots, y_n]$, w is the weight vector and λ means the regularization parameter. As the evolution of linear ridge regression, the expression of this regression is shown as:

$$\sum_{i}^{p}\left(y_j - w^T * x_i\right) * x_i = \lambda w$$

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

c. Kernel ridge regression

The features of vectors are replaced as:

$$x_i : \varphi(x_i)$$

When the dimensions are higher than the data cases, the solution can be expressed as the following no matter the dimension of the space as well as the number of cases:

$$w = \sum_i a_i * \varphi(x_i)$$

When we try to minimize the object with kernel trick, the expression can he be transformed as:

$$\min_{a} ||Y - Ka||^2 + \lambda a^T K a$$

This method is usually used in linear methods and the solution can act as non-linear characteristics:

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

2. Experiment

This article uses RVFL, edRVFL as well as KRR methods to come up with the solutions for real problems, these problems come from ten different open-source datasets, and details are showed in table1.

Table1. details of dataset

Dataset	classes	features	patterns
Bank	2	16	3316
Car	4	6	1728
Contrac	3	9	1473
hill-valley	2	100	606
led-display	10	7	1000
Ozone	2	72	2028
Semeion	10	256	1593
plant-margin	100	64	1280
plant-shape	100	64	1280
plant-texture	100	64	1280

1） By the usage of RVFL:

Calculate the classification ability of edRVFL through the training data set and obtain relevant parameters and training results through experiments and training. In this experiment, we use ReLU and sigmoid activation functions to obtain the corresponding data. I set the paraments as the table 2.2 below:

Table 2.2 the parameters setting in edRVFL

Num. layer	Num. neuron	Regular.para	Scaling.param	activation
10	100	0.1	1	Relu/Sigmoid

The results of the experiments are shown in following table 2.3 and 2.4

Table 2.3 the result using ReLU

dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
bank	0.8952	0.9004	0.0304	0.8996	0.0053	0.8930
car	0.9466	0.9773	0.0174	0.9503	0.0025	0.9452
contrac	0.5520	0.5564	0.0011	0.5863	0.0012	0.4483
hill-valley	0.7022	0.7637	0.0105	0.7099	0.0017	0.6572
led-display	0.7363	0.7552	0.0017	0.7102	0.0020	0.7174
ozone	0.9673	0.9770	0.0603	0.9703	0.0038	0.9665
semeion	0.8789	0.9934	0.0330	0.8992	0.0043	0.8011
plt-margin	0.7859	0.9770	0.0201	0.8033	0.0019	0.7803
plt-shape	0.6682	0.9164	0.0183	0.6184	0.0027	0.6195
plt-texture	0.8020	0.9702	0.0157	0.8553	0.0035	0.7911

Table 2.4 the result using sigmoid

dataset	Best_acc	Train_acc	Train_time	Test_acc	Test_time	Val_acc
bank	0.9002	0.9204	0.0314	0.9196	0.0047	0.9130

car	0.9656	0.9897	0.0162	0.9602	0.0023	0.9610
contract	0.5553	0.6203	0.0157	0.5892	0.0030	0.4683
hill- valley	0.7062	0.7337	0.0115	0.7057	0.0027	0.6872
led-display	0.7352	0.7455	0.0317	0.7132	0.0160	0.7132
ozone	0.9473	0.9470	0.1203	0.9505	0.0138	0.9465
semeion	0.8799	0.9834	0.0294	0.9092	0.0052	0.7613
plt-margin	0.7689	0.9760	0.0207	0.7933	0.0025	0.7603
plt-shape	0.5882	0.8934	0.0175	0.5784	0.0034	0.5895
plt-texture	0.8028	0.9902	0.0217	0.8753	0.0040	0.8011

2） By the usage of edRVFL:

Calculate the classification ability of edRVFL through the training data set and obtain relevant parameters and training results through experiments and training. In this experiment, we use ReLU and sigmoid activation functions to obtain the corresponding data. The following table shows the training of the two activation functions and the test results. By comparing the data, it can be seen that the performance of the two activation functions of ReLU and sigmoid are generally the same, and the accuracy of sigmoid is higher.

When using the activation function, set the network depth to 10 layers and the number of neurons to 100. The data set is divided into four parts. Each part is used as the test set. After the activation function is calculated, the variance and mean of val_trainX and val_trainX are returned. Assign the value of each time to val_acc to find the average of the four accuracy. Compare the best value each time with the current best value. If the current value is better, replace it and modify the parameters for the next cycle. I set the paraments as the table 2.5 below:

Table 2.5 the parameters setting in edRVFL

Num. layer	Num. neuron	Regular.para	Scaling.param	activation
10	100	0.1	1	Relu/Sigmoid

The results of each function using edRVFL with ReLU and sigmoid are expressed as table 2.6 and table 2.7:

Table 2.6 the results of ReLU with ten layers

dataset	best_acc	train_acc	train_time	test_acc	test_time	val_acc
bank	0.8971	0.9204	1.2094	0.9006	0.1343	0.8930
car	0.9732	1	0.8041	0.9711	0.0681	0.9732
contrac	0.5518	0.5985	0.0299	0.5932	0.0236	0.4270
hill-valley	0.7121	0.7833	0.1388	0.6996	0.0328	0.6739
led-display	0.7313	0.7675	0.0369	0.6950	0.0238	0.7188
ozone	0.9689	0.9758	1.1602	0.9823	0.0978	0.9660
semeion	0.9254	1	1.1338	0.9279	0.0828	0.8611
plt-margin	0.8109	0.9953	0.8823	0.8281	0.0375	0.8023
plt-shape	0.6805	0.9922	0.8723	0.6781	0.0343	0.7688
plt-texture	0.8381	0.9999	0.9302	0.8709	0.0572	0.8001

Table 2.7 the results of sigmoid with ten layers

dataset	Best_acc	Train_acc	Train_time	Test_acc	Test_time	Val_acc
bank	0.8991	0.9126	0.5722	0.8972	0.0816	0.8938
car	0.9862	1	0.6073	0.9855	0.0561	0.9855
contrac	0.5551	0.6273	0.2779	0.6068	0.0347	0.4448

hill-valley	0.7307	0.7864	0.4526	0.7490	0.0426	0.6987
led-display	0.7325	0.7600	0.0385	0.7150	0.0197	0.7150
ozone	0.9684	0.9684	0.0501	0.9823	0.0205	0.9645
semeion	0.9137	1	1.1335	0.9404	0.0854	0.8407
plt-margin	0.8172	0.9969	0.5941	0.8406	0.0639	0.8039
plt-shape	0.6422	0.9516	0.5688	0.6125	0.0664	0.6023
plt-texture	0.8421	0.9969	0.5456	0.9000	0.0618	0.8358

3) By the usage of KRR:

Finally, the classification performance of KRR is obtained through experiments, and the experimental results are shown in table 2.8. The classification performance of KRR is obviously better than RVFL and edRVFL.

Table 2.8 the results of sigmoid with ten layers

dataset	Best_acc	Train_acc	Train_time	Test_acc	Test_time	Val_acc
bank	0.8453	0.8832	1.3722	0.8867	0.0216	0.8552
car	0.9532	0.9765	1.0148	0.9736	0.056	0.9855
contract	0.5207	0.5873	0.5379	0.5368	0.0047	0.4248
hill- valley	0.5537	0.5876	0.3518	0.5374	0.0066	0.5557
led-display	0.7403	0.7782	0.1385	0.7049	0.0028	0.7050
ozone	0.9184	0.9284	0.0501	0.9823	0.0205	0.9645
semeion	0.9034	0.9723	0.6231	0.9274	0.0154	0.8407
plt-margin	0.7724	0.8104	0.9941	0.7938	0.0157	0.6978
plt-shape	0.6164	0.7732	0.9230	0.6181	0.0155	0.6017

plt-texture	0.7872	0.9819	1.7233	0.8217	0.0136	0.7658

3. Discussion

By using different active functions in different algorithms, it is obviously to compare the accuracy between them. The performance of edRVFL is better than that in RVFL, it can provide higher accuracy on datasets but because of more layers and the complexity, the training time is higher. The result in edRVFL is better in chosen algorithms and the result is almost similar with little differences.

4. Conclusion：

Through experiments, it is obvious that when different algorithms and activation functions are used on the same data set, the impact on classification performance is different. Through the introduction, definition and calculation of the algorithm principle, the parameters in the algorithm are changed to realize the classification results of different algorithms. optimization. This article mainly uses three algorithm models, RVFL, deepRVFL, and KRR, and two activation functions, sigmoid and ReLU, to compare their results. According to the parameters selected in the article, the deepRVFL algorithm can maximize the classification accuracy. The sigmoid activation function is The performance is better. When the number of hidden layers increases, the algorithm performance can also be improved.

References:

[1] P. N. Suganthan, “Random Vector Functional Link (RVGL) Neural Networks,” Course Slides of EE6227, School of EEE, Nanyang Technological University, Singapore, 2021.

[2] Vapnik V. Statistical Learning Theory[J]. Wiley,1998.

[3] Ikonomovska E, Gama J, Sebastio R, et al. Refression trees from data streams with drift detection[C]. Discovery Science. Springer Berlin Heidelberg, 2004: 286-295.

[4] L. Zhang and P. H. Suganthan, “A Comprehensive Evolution of Random Vector Functional Link Networks,” Information Science, vol.367, pp. 1094-1105,2016

[5] 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.

[6] S.Bernhard, Z.Luo, and V.Vovk,”Empirical Inference,” Springer Science & Business Media, 2013.