Adaptive Ensemble Variants of Random Vector Functional Link Networks

Minghui Hu[1], Qiushi Shi[1], P. N. Suganthan[1](✉), and M. Tanveer[2]

[1] Nanyang Technological University,
50 Nanyang Avenue, Singapore 639798, Singapore
{minghui.hu,epnsugan}@ntu.edu.sg, qiushi001@e.ntu.edu.sg
[2] Indian Institute of Technology Indore, Simrol, Indore 453552, India
mtanveer@iiti.ac.in

Abstract. In this paper, we propose a novel adaptive ensemble variant of random vector functional link (RVFL) networks. Adaptive ensemble RVFL networks assign different weights to the sub-classifiers according to prediction performance of single RVFL network. Generic Adaptive Ensemble RVFL is composed of a series of unrelated, independent weak classifiers. We also employ our adaptive ensemble method to the deep random vector functional link (dRVFL). Each layer in dRVFL can be regarded as a sub-classifier. However, instead of training several models independently, the sub-classifiers of dRVFL can be obtained by training a single network once.

Keywords: Random vector functional link · Ensemble classifiers · Deep neural networks · Adaptive boosting

1 Introduction

In recent years, deep learning methods have become attractive because of their success in multiple areas. Convolutional neural networks (CNN) won lots of competitions with conventional methods in visual tasks, and recurrent neural networks (RNN) based models were good at processing sequential inputs [10]. These methods use gradient-based back-propagation (BP) learning algorithm to train a large number of parameters in their networks. Proposed by Hinton in [13], the main idea of BP algorithm is to tune the weights and biases following the gradient of the loss function. However, due to the large computational cost, some modern-day neural networks may need several weeks to finish the training step [11]. Moreover, overfitting [6] is another serious problem that can cause the model to perform well during the training while achieving poor performance when testing.

Meanwhile, some randomization based neural networks have been proposed to overcome the flaws of the BP-based models [14,15,17]. The weights and biases in these models are randomly generated and kept fixed during the training process. Only the parameters in the output layers are obtained by the close-form

H. Yang et al. (Eds.): ICONIP 2020, CCIS 1333, pp. 30–37, 2020.
https://doi.org/10.1007/978-3-030-63823-8_4

solution [16]. Random vector functional link network (RVFL) is a typical single-hidden-layer randomized neural network [12]. It has a direct link to convey the information from the input layer directly to the output layer. This is useful because the output layer contains both the linear original features and the non-linear transformed features. The newest version of this model was proposed in [8] and called ensemble deep random vector functional link network (edRVFL), the authors convert the single-hidden-layer RVFL to the deep version and employ the idea of ensemble learning to reduce the computational complexity. Similar to the conventional neural networks, the edRVFL network also consists of an input layer, an output layer, and several hidden layers. The hidden weights and biases in this network are randomly generated and do not need to be trained. The uniqueness of this frame is that each layer is treated as an independent classifier, just like a single RVFL network. Eventually, the final output is obtained by fusing all the outputs.

Ensemble learning methods are widely used in classification problems, they combine multiple models for prediction to overcome the weakness of each single learning algorithm [19]. Among them, bagging [1], boosting [4], and stacking [18] are the three most popular and successful methods. Adaboost was originally proposed to improve the performance of the decision trees [3]. This method intends to combine several weak classifiers to obtain a strong classifier. The misclassified samples in the previous classifiers will be given greater importance in the following classifiers. Furthermore, the classifier with higher accuracy will also be assigned a higher weight in the final prediction. In this paper, we introduce two novel methods using Adaboost to generate the ensemble model of RVFL and the deep version of RVFL. We called them adaptive ensemble random vector functional link networks (ada-eRVFL) and adaptive ensemble deep random vector functional link networks (ada-edRVFL). In ada-eRVFL, we treat each single RVFL network as the weak classifier in Adaboost. However, in ada-edRVFL, we treat every single layer as the weak classifier.

2 Related Works

2.1 Random Vector Functional Link Networks

Random vector functional link network (RVFL) is a randomization based single hidden layer neural network proposed by Pao [12]. The basic structure of RVFL is shown in Fig. 1.

Both the linear original features and the non-linearly transformed features are conveyed to the output layer through the direct link and the hidden layer, respectively. Therefore, the output weights β can be learned from the following optimization problem:

$$O_{RVFL} = \min_{\beta} ||D\beta - Y||^2 + \lambda||\beta||^2 \quad (1)$$

where D is the combination of all linear and non-linear features, and Y are the true labels of all the samples. λ denoted as the parameter for controlling

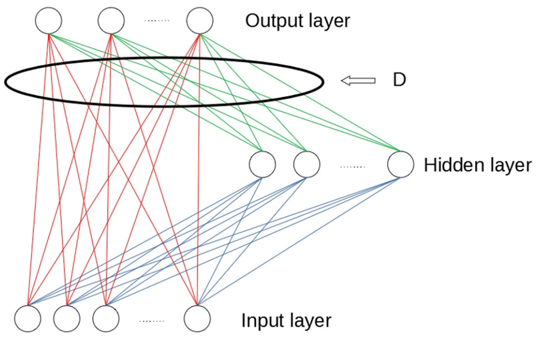

Fig. 1. The structure of RVFL network. The red lines are defined as the direct link which transfer the linear original features to the output layer. (Color figure online)

how much the algorithm cares about the model complexity. This optimization problem can be solved by ridge regression [7], and the solution can be written as follows:

$$PrimalSpace : \beta = (D^T D + \lambda I)^{-1} D^T Y \tag{2}$$

$$DualSpace : \beta = D^T (DD^T + \lambda I)^{-1} Y \tag{3}$$

The computational cost of training the RVFL network is reduced by suitably choosing between the primal or dual solution [15].

2.2 Ensemble Deep Random Vector Functional Link Networks

Inspired by other deep learning models, the authors of [8] proposed a deep version of the basic RVFL network. They also used ensemble learning to improve the performance of this model. The structure of edRVFL is shown in Fig. 2. Let n be the hidden neuron number and l be the hidden layer number. The output of the first hidden layer can be obtained by:

$$H^{(1)} = g(XW^{(1)}), \quad W^{(1)} \in \mathbb{R}^{d \times n} \tag{4}$$

where X denotes the input features, d represents the feature number of the input samples, and $g(\cdot)$ is the non-linear activation function used in each hidden neuron. When the layer number $l > 1$, similar to the RVFL networks, the hidden features in the previous hidden layer as well as the original features in the input layer are concatenated together to generate the next hidden layer. So Eq. 4 becomes, when $l > 1$:

$$H^{(l)} = g([H^{(l-1)} X]W^{(l)}), \quad W^{(l)} \in \mathbb{R}^{(n+d) \times n} \tag{5}$$

edRVFL network treats every hidden layer as a single RVFL classifier too. After getting predictions from all the layers via ridge regression, these outputs will be fused by ensemble methods to reach the final output.

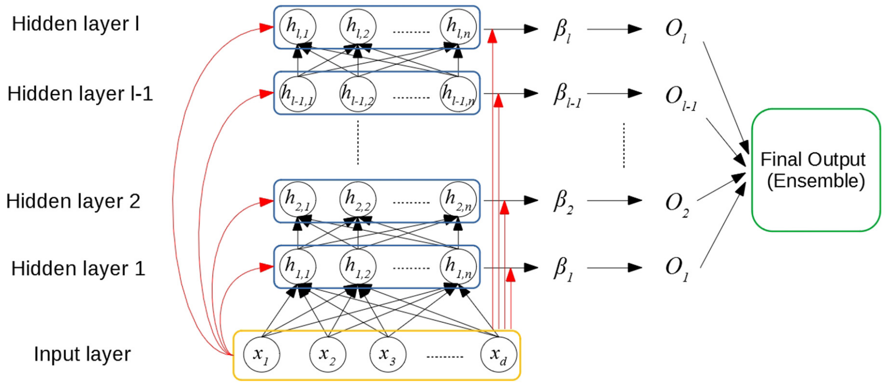

Fig. 2. The structure of edRVFL network. Each single layer is treated as an independent RVFL network. The final output is obtained by combing the predictions from all the classifier.

2.3 AdaBoost

Boosting has been proven to be successful in solving classification problems. It was first introduced by [3], with their algorithm called Adaboost. This method was originally proposed for the two-class classification problem and improving the performance of the decision tree. The main idea of Adaboost is to approximate the Bayes classifier by combining several weak classifiers. Typically, the Adaboost algorithm is an iterative procedure, it starts with using unweighted samples to build the first classifier. During the following steps, the weights of the misclassified samples in the previous classifier will be boosted in the next classifier. That means these samples are given higher importance during the error calculation. After several repetitions, it employs weighted majority voting to combine outputs from every classifier to obtain the final output.

In [5], the authors developed a new algorithm called Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) which directly extended the Adaboost to the multi-class case without complicating it into multiple two-class problems.

3 Method

In this section, we proposed the adaptive ensemble random vector functional link networks (ada-eRVFL). ada-eRVFL is inspired by the adaptive boosting method. For a data set $\boldsymbol{x}$, the weights α are assigned to the sub-classifiers according to the error function:

$$err^{(m)} = \sum_{i=1}^{n} \omega_i \mathbb{1}(c_i \neq R^{(m)}(x_i)) / \sum_{i=1}^{n} \omega_i \tag{6}$$

where R represents the single RVFL classifier and ω is the sample weight described as the follows:

$$\omega_i \leftarrow \omega_i \cdot exp\left(a^{(m)} \cdot \mathbb{1}\left(c_i \neq R^{(m)}(x_i)\right)\right) \tag{7}$$

It is worth noting that the sample weights only contribute in computing the error function of the sub-classifier. The training phase of the weak RVFL-classifiers only utilizes the original samples. The weak RVFL classifiers for ensemble are individual and independent. The mis-classified samples would be assigned a larger weight.

Algorithm 1: ada-eRVFL

```
Input: A set of training data $\boldsymbol{x} \in \{x_1, x_2, \ldots, x_n\}$.

Initialize the sample weights $w_i$ and the classifier weight $a_i$;
for $m \leftarrow 0$ to $M$ do
    Train an RVFL classifier $R^{(m)}(\boldsymbol{x})$ with the raw training data $\boldsymbol{x}$;
    Compute the error of the classifier $R^{(m)}$;
        $$err^{(m)} = \sum_{i=1}^{n} \omega_i \mathbb{1}(c_i \neq R^{(m)}(x_i)) / \sum_{i=1}^{n} \omega_i \quad (8)$$
    if $1 - err^{(m)} > 1/K$ then
        Compute the weight of the classifier;
            $$a^{(m)} = log\frac{1 - err^{(m)}}{err^{(m)}} + log(K-1) \quad (9)$$
        Set the sample weights as follows:
            $$\omega_i \leftarrow \omega_i \cdot exp\left(a^{(m)} \cdot \mathbb{1}\left(c_i \neq R^{(m)}(x_i)\right)\right) \quad (10)$$
    else
        The classifier weight is 1 ;
        The sample weight are $1/N$ ;
    end
end
Output: The ensemble classifier $C$
    $$C(\boldsymbol{x}) = argmax \sum_{i=1}^{n} a^{(m)} \cdot \mathbb{1}(R^{(m)}(\boldsymbol{x}_i) = k) \quad (11)$$
```

Another proposed variant is adaptive ensemble deep random vector functional link networks (ada-edRVFL). The deep RVFL network consists of a few hidden layers instead of single hidden layer in RVFL. Each hidden layer can

```
Algorithm 2: ada-edRVFL
Input: A set of training data $\boldsymbol{x} \in \{x_1, x_2, \ldots, x_n\}$.

Initialize the sample weights $w_i$ and the classifier weight $a_i$;
Train an deep RVFL classifier $D(\boldsymbol{x})$ with the training data $\boldsymbol{x}$ with $L$ hidden layers.;
for $l \leftarrow 0$ to $L$ do
    Compute the error of the sub classifier $R^{(l)}$;
        $$err^{(m)} = \sum_{i=1}^{n} \omega_i \mathbb{1}(c_i \neq R^{(l)}(\boldsymbol{x}_i)) / \sum_{i=1}^{n} \omega_i \quad (12)$$
    if $1 - err^{(l)} > 1/K$ then
        Compute the weight of the classifier;
            $$a^{(l)} = log \frac{1 - err^{(m)}}{err^{(m)}} + log(K - 1) \quad (13)$$
        Set the sample weights as follows:
            $$\omega_i \leftarrow \omega_i \cdot exp\left(a^{(l)} \cdot \mathbb{1}\left(c_i \neq R^{(l)}(\boldsymbol{x}_i)\right)\right) \quad (14)$$
    else
        The classifier weight is 1 ;
        The sample weight are $1/L$ ;
    end
end
Output: The ensemble classifier $C$
    $$D(\boldsymbol{x}) = argmax \sum_{i=1}^{n} a^{(l)} \cdot \mathbb{1}(R^{(l)}(\boldsymbol{x}_i) = k) \quad (15)$$
```

constitute a sub-classifier with the input layer, output layer and the direct link between them.

For both proposed ideas, we need to make sure the classifier weights α are positive, thus it is required that $1 - err^{(m)} > 1/K$.

4 Experiments

The experiments are performed on 9 classification datasets selected from UCI Machine Learning Repository [2], including both binary and multiple classification problems. Sample volume and feature dimensions are coverage from hundred to thousand. The details of the dataset are stated below. In the interests of a fair comparison, we use the exact same validation and test subsets and the same data pre-processing method as in [9].

For the RVFL based methods, the regularization parameter λ is set as $\frac{1}{C}$ where C is chosen from range $2^x\{x = -6, -4, \ldots, 10, 12\}$. Based on the size of the dataset, the hidden neuron number these methods can be tuned from $\{2^2, 2^3, \ldots, 2^{10}, 2^{11}\}$.

For the dRVFL based methods, the regularization parameter λ is set as $\frac{1}{C}$ where C is chosen from range $2^x\{x = -6, -4, \ldots, 10, 12\}$. Based on the size of the dataset, the hidden neuron number these methods can be tuned from $\{2^2, 2^3, \ldots, 2^{10}, 2^{11}\}$. Besides, the maximum number of hidden layers for the edRVFL based methods is set to 32, which is also the number of sub-classifiers in ada-edRVFL.

Table 1. Accuracy (%) and Average rank of variant approaches

Dataset names	Batch norm	ada Boost	Layer norm	High-way	RVFL	ResNet	ada-e RVFL	MSRA init	edRVFL	ada-ed RVFL
Led-display	62.8	67.8	64.8	70.4	71.6	71.6	72.6	72	71.6	**74.3**
Statlog-german-credit	75.2	74.4	74	**77.6**	74.1	77.2	75.7	72.8	75.6	76.69
oocytes4d	80.78	74.51	76.86	71.76	78.43	80	80.98	81.96	83.04	**83.92**
Haberman-survival	73.68	72.37	68.42	64.47	68.42	68.42	72.28	72.37	73.36	**74.01**
Contrac	45.38	51.3	45.92	50.54	50.06	51.36	50.2	51.36	51.36	**53.6**
Yeast	49.06	43.13	60.92	60.65	58.49	54.99	58.82	**61.73**	59.97	61.25
Heart-va	28	29	24	40	32	26	35	26	34.5	**37.5**
Pima	71.88	74.09	69.27	71.88	75.52	71.35	76.22	**76.56**	74.09	76.43
wine-quality-red	54.5	54.44	61	56.25	57.13	61.5	59.44	62.5	65.87	**66.5**
Teaching	50	51.97	**63.16**	52.63	51.61	55.26	52.63	60.53	51.97	55.92
Average accuracy	59.128	59.301	60.835	61.618	61.736	61.768	63.387	63.781	64.136	**66.012**
Average rank	7.55	7.25	7.2	6.05	6.8	5.8	4.6	3.95	4.1	**1.7**

From Table 1, the ada-dRVFL achieves the best performance on all datasets. On average, the accuracy of ada-edRVFL has an improvement of over two percentages. It suggests the proposed adaptive ensemble method can effectively boost the performance of weak classifiers, and this method make single RVFL network competitive with deep RVFL network.

5 Conclusion

We proposed an adaptive ensemble method of random vector functional link networks. Compared to the edRVFL, our framework outperforms the edRVFL's result. The proposed method can be employed to different RVFL based networks. After utilizing the proposed ensemble method, the performance of ensemble RVFL can compete with the deep RVFL networks. Specifically, we test the proposed method on 9 UCI classification task machine learning datasets. The experimental results show that the proposed ensemble variant is both effective and general.

References

1. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
2. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml
3. Freund, Y., Schapire, R.E.: A desicion-theoretic generalization of on-line learning and an application to boosting. In: Vitányi, P. (ed.) EuroCOLT 1995. LNCS, vol. 904, pp. 23–37. Springer, Heidelberg (1995). https://doi.org/10.1007/3-540-59119-2_166
4. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: ICML, vol. 96, pp. 148–156. Citeseer (1996)
5. Hastie, T., Rosset, S., Zhu, J., Zou, H.: Multi-class adaboost. Stat. Interface **2**(3), 349–360 (2009)
6. Hawkins, D.M.: The problem of overfitting. J. Chem. Inf. Comput. Sci. **44**(1), 1–12 (2004)
7. Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970)
8. Katuwal, R., Suganthan, P., Tanveer, M.: Random vector functional link neural network based ensemble deep learning. arXiv preprint arXiv:1907.00350 (2019)
9. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks. In: Advances in Neural Information Processing Systems, pp. 971–980 (2017)
10. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
11. Livni, R., Shalev-Shwartz, S., Shamir, O.: On the computational efficiency of training neural networks. In: Advances in Neural Information Processing Systems, pp. 855–863 (2014)
12. Pao, Y.H., Takefuji, Y.: Functional-link net computing: theory, system architecture, and functionalities. Computer **25**(5), 76–79 (1992)
13. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science (1985)
14. Schmidt, W.F., Kraaijveld, M.A., Duin, R.P., et al.: Feed forward neural networks with random weights. In: International Conference on Pattern Recognition, p. 1. IEEE Computer Society Press (1992)
15. Suganthan, P.N.: On non-iterative learning algorithms with closed-form solution. Appl. Soft Comput. **70**, 1078–1082 (2018)
16. Te Braake, H.A., Van Straten, G.: Random activation weight neural net (RAWN) for fast non-iterative training. Eng. Appl. Artif. Intell. **8**(1), 71–80 (1995)
17. Widrow, B., Greenblatt, A., Kim, Y., Park, D.: The no-prop algorithm: a new learning algorithm for multilayer neural networks. Neural Netw. **37**, 182–188 (2013)
18. Wolpert, D.H.: Stacked generalization. Neural Netw. **5**(2), 241–259 (1992)
19. Zhang, C., Ma, Y.: Ensemble Machine Learning: Methods and Applications. Springer, Heidelberg (2012). https://doi.org/10.1007/978-1-4419-9326-7