ATLANTIS PRESS

Advances in Economics, Business and Management Research, volume 53

7th International Conference on Education and Management (ICEM 2017)

Ensemble Radical Basis Function Neural Networks for Regression Based on Statistical Learning Theory

Liangzhi Gan[1, a], Dawei Jiang[1, b] and Mi He[2,c*]

[1]School of Electrical Engineering and Automation, Jiangsu Normal University, 221116, China

[2]Office of Budget and Finance, Jiangsu Normal University, 221116, China

[a]lzh_box@163.com, [b]dwj_jsnu@163.com, [c]hemi_jsnu@163.com

*Corresponding author: Mi HE, hemi_jsnu@163.com

Keywords: Statistical learning theory; Radical basis function neural networks; VC dimension; Ensemble learning

Abstract. We proposed an algorithm to construct ensemble radical basis function neural networks for regression estimation. Taking full advantage of the characteristic of radial basis function, we calculated groups of approximate basis in Reproducing Kernel Hilbert Space (RKHS). The approximate basis could be used to represent all the samples by the way of linear combination. By this way, the weak learners of radial basis function neural network were built. But it was proved that the weak learners were not accurate enough. In order to get accurate and stable learning machine with better generalization ability, we proposed the Ensemble Radical Basis Function Neural Networks (ERBFNNs). Employing the sinc function, the proposed ERBFNNs have shown exciting outcomes as have come out at the end of the paper.

Introduction

The problem of empirical data modeling is germane to many engineering applications. For this purpose, Broomhead and Lowe developed RBFNNs in 1988 [1].

Consider the standard supervised learning problem. In a regression problem, one is supplied with a set of data points $\boldsymbol{D}=\{(\boldsymbol{x}_1,y_1),\cdots,(\boldsymbol{x}_N,y_N)\}\subset \boldsymbol{R}^{\boldsymbol{d}}\times R$ that are sampled from an unknown function, where $\boldsymbol{x}_i, i=1,\cdots,N$ are input points and $y_i, i=1,\cdots,N,$ the corresponding output points. The goal is to find a fit to the data points such that an approximation to the unknown function is obtained. A RBFNN is an approximation of the form:

$$f(\boldsymbol{x},\boldsymbol{w})=\sum_{i=1}^{N} w_i\varphi_i(\boldsymbol{x})+b, \tag{1}$$

where w_i's are the connecting weight and $\varphi_i(\boldsymbol{x})=\exp(\frac{\|\boldsymbol{x}-c_i\|^2}{\sigma_i^2})$'s are the radical basis functions ($i=1,\cdots,N$), b is the balance parameter or sometimes called a threshold. c_i and σ_i are the i-th center vector and the width parameter. To construct RBF network, the number of the hidden layer must be set, and the centers c_i, the widths σ_i and the weights w_i must be estimated. In RBF typical learning, the network structure will be determined based on prior knowledge or the experiences of experts. The parameters are estimated using either the clustering or the least mean squared method. On the other hand, there are approaches in which the network structure and its parameters are estimated by the evolutionary computation [2,3,4].

In statistical learning theory(STL), some important conclusion has been drawn by Vapnik[5,6] and his cooperators: If there were a "kernel function" $k(\boldsymbol{x}_i,\boldsymbol{x})$ satisfying $k(\boldsymbol{x}_i,\boldsymbol{x})=\varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x})$, we would only need to use $k(\boldsymbol{x}_i,\boldsymbol{x})$ in the training algorithm, and would never need to explicitly even know what $\varphi(x)$ is. One example is :

$$k(\boldsymbol{x}_i, \boldsymbol{x}) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}\|^2 / \sigma^2} \tag{2}$$

In this example, $\varphi(\boldsymbol{x})$ is infinite dimensional [5,6], and we have defined dot product by (2). The problem could be written as:

$$\arg\min_{\boldsymbol{w},b} \Phi(\boldsymbol{w}, b, \boldsymbol{e}) = \sum_{k=1}^{N} e_k^2$$

$$s.t.\ \ y_k w_k k(\boldsymbol{x}, \boldsymbol{x}_k) + b = y_k - e_k, k = 1, \cdots, N \tag{3}$$

Observing equation (1) and problem (3), we can find that: selecting (2) as the kernel function, equation (1) is the solution to problem (3).

Ensemble Radical Basis Function Neural Networks

With the help of STL, we construct the RBFNN. So some important results are listed as follow[5,6].

Lemma 1 The VC dimension of a set of functions that are linear in the parameters

$$f(\boldsymbol{z}, \boldsymbol{\alpha}) = \sum_{i=1}^{n-1} \alpha_i \phi_i(\boldsymbol{z}) + \alpha_0 \tag{4}$$

equals n, the numbers of parameters of a set of functions.

Lemma 2 With probability $1-\eta$ simultaneously for all functions in a set of real-valued bounded functions $Q(\boldsymbol{z}, \boldsymbol{w}), \boldsymbol{w} \in \boldsymbol{\Lambda}$, the inequality

$$R(\boldsymbol{w}) \le R_{emp}(\boldsymbol{w}) + (B - A)\sqrt{\varepsilon(l)} \tag{5}$$

is valid, where

$$\varepsilon(l) = \frac{H_{ann}^{\Lambda,\mathrm{B}}(2l) - \ln \eta / 4}{l} + \frac{1}{l}$$

where $R(\boldsymbol{w})$ is the expected risk, $R_{emp}(\boldsymbol{w})$ is the empirical risk, and, $H_{ann}^{\Lambda,\mathrm{B}}(2l)$, the annealed entropy.

From lemma 2 we can draw a conclusion:

$$R(\boldsymbol{w}) \le R_{emp}(\boldsymbol{w}) + (B - A)\sqrt{\frac{h \ln(2l/h) - \ln \eta / 4}{l} + \frac{1}{l}}. \tag{6}$$

And from lemma 1, we can get the VC dimension h of the set of the functions. So inequality (6) could be used to estate the upper bound of the expected risk.

Construct weak RBFNN learner Consider the standard supervised learning problem for RBFNNs. Weights, center vectors and the width parameters must be estimated. The number of the hidden layer must be set. All this questions seem difficult. But it is easy in STL opinion. We can find some samples to represent all the samples by the following method. A learning program is given training examples of the form $\{(x_1, y_1), \cdots, (x_N, y_N)\}$ for some unknown function y=f(x). Mapping the input data into feature space and denoted as $\{\varphi(x_k)\}_{k=1}^{N}$, the following steps are used to find the center vectors, which we called **Vectors Selecting Method**.

1) Found two new sets $\boldsymbol{X_h} = \{\boldsymbol{\varphi}(\boldsymbol{x_1})\}$, $\boldsymbol{X_A} = \boldsymbol{\phi}$;
2) To every $\varphi(\boldsymbol{x_k})$, $k = 2, \cdots, N$, get value of

$$\min \ (\varphi(\boldsymbol{x_k}) - \sum_{\varphi(\boldsymbol{x_i}) \in \boldsymbol{X_l}} \lambda_i \varphi(\boldsymbol{x_i}))^T (\varphi(\boldsymbol{x_k}) - \sum_{\varphi(\boldsymbol{x_i}) \in \boldsymbol{X_l}} \lambda_i \varphi(\boldsymbol{x_i}));$$

3) If the value is less than ε (a very small positive number that approximate to zero), $\varphi(x_k)$ is added to set X_A; else $\boldsymbol{\varphi}(x_k)$ is added to set X_h;
4) Go back to step 2 until all the vectors are checked.

Once the center vectors setted, the number of the hidden layer is settled. The set $X=\{x_1,\cdots,x_N\}$ is denoted as X' in feature space, that is $X'=[\varphi(x_1),\cdots,\varphi(x_N)]$. The above process split X' into two parts: X_h and X_A. All the elements in X_h are linearly independent and the elements in X_A can be linearly approximated by X_h. The elements in set X_h are different from each other, so we can look it as a row vector: $X_h=[\varphi(x_1),\cdots\varphi(x_h)]$.

Since we have known that $w=\sum_{k=1}^{N}\alpha_k y_k\varphi(x_k)$ and the elements in X_A can be approximately represented by X_h, so w could be write as $w\approx\sum_{\varphi(x_i)\in X_h}\theta_i\varphi(x_i)\approx X_h\theta^T$, we could rewrite Problem (3) as:

$$\min_{\theta,b}\Phi(\theta,b,e)=\sum_{k=1}^{h}e_k^2,$$

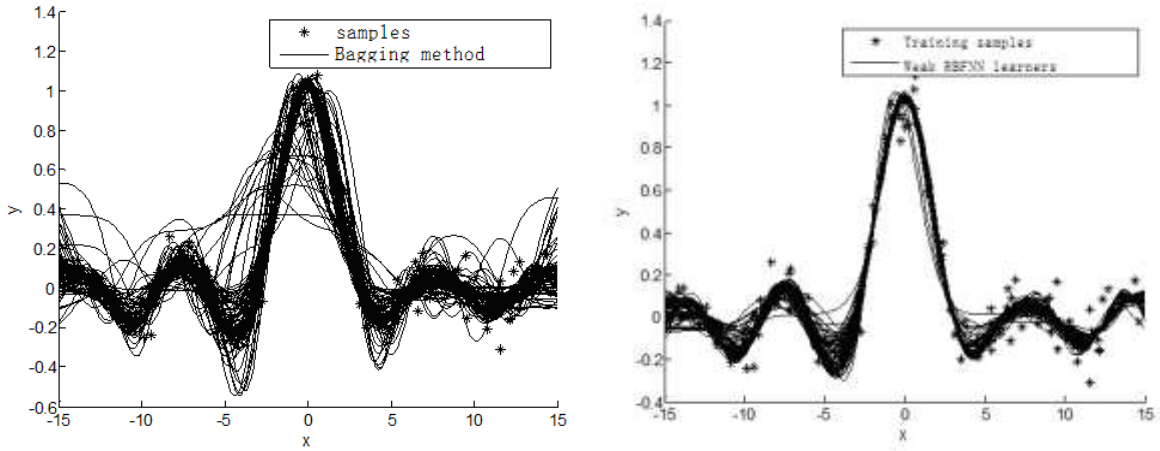
$$s.t.\ (X_h\theta)^T\varphi(x_k)+b=y_k-e_k,k=1,\cdots,h. \tag{7}$$

Where $\theta=[\theta_1,\cdots,\theta_h]$, h is the number of elements in set X_h. Substitute $e_k(k=1,\cdots,h)$ in equations (7) for $e_k(k=1,\cdots,h)$ in Eq. 7, we get

$$f(\boldsymbol{x},\boldsymbol{w})=\sum_{i=1}^{h}\theta_i\exp(\frac{\|\boldsymbol{x}_i-\boldsymbol{x}\|^2}{\sigma^2})+b \tag{8}$$

Dimension of equations set (8) is $h+1$. It is completely controlled by the number of elements in set X_h. It is constructed by partial of the training data $\{(x_1, y_1), \cdots, (x_N, y_N)\}$. So it is a weak learner from the opinion of ensemble learning[7,8]. This is called **a weak RBFNN learner**.

The aim of machine learning is to make a good predictor. This means finding a machine learning method that generates a predictor based on a set of examples, where the predictor is a good approximation of the function that generated the examples. It is well known that **weak RBFNN learner**s are not stable[7,8] and ensemble learning can improve the stability by aggregating unstable weak learners. The **Vectors Selecting Method** can be repeated many times and we can get enough **weak RBFNN learner**s. Aggregating these **weak RBFNN learner**s, a good approximation of the function is generated[9,10,11].



a) Bagging leaners b) Weak RBFNN learners

Figure 1. Comparison of Bagging learners and weak RBFNN learners.

Experiments

In order to achieve a high degree of comparability of our results, we have selected frequently used tasks for evaluation of learning algorithms. In particular, we have used a well-known sinc function approximation. For simulated test function $y = \frac{\sin x}{x} + e$ we generated 400 observations as a training dataset. With this dataset, we generated two kinds of weak learners: the weak RBFNN learners (which are shown in part b of figure.1) and the bagging learners(which are shown in part a of figure.1). It's easy to find that the weak RBFNN learners are much closer to the simulated test function and the ensemble RBFNNs can approximate it more accurate and fast.

Summary

This paper presents a new algorithm that addresses the problem of calculating the number and locations of the hidden node centers, in the process of training an RBF network. The ensemble learning algorithm improved RBFNNs for regression. How to design and combine weak learners is one of the core questions for ensemble learning. We suggest the sparse RBF networks as the weak learners. Using the approximation regression, we compared the performance to other procedures that are widely used. Our main results are concluded as: Firstly, the proposed RBFNNs are effective. Fig.1 illustrates that RBFNNs are much better than bagging predictors in regression. Secondly, RBFNNs are more robust. RBFNNs can select parameters in more widely ranges. We didn't give a new method to aggregate the weak learners; this will be our task in the future.

References

[1] D. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems. 2 (1988) 321–355.
[2] Z.Q. Zhao, D.S. Huang. A mended hybrid learning algorithm for radial basis function neural networks to improve generalization capability. Appl Math Model 2007;31:1271–81.
[3] J.X. Peng, K. Li, D.S. Huang. A hybrid forward algorithm for RBF neural network construction. IEEE Trans Neural Networks 2006;17(6):1439–51.
[4] S. N.Qasem, S. M. Shamsuddin, (2011). Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis.Applied Soft Computing, 2011; 11(1), 1427-1438.
[5] V. Vapnik, Statistical learning theory. New York: John Wiley and Sons, 1998.
[6] C.J.C Burges. A Tutorial on Support Vector Machines for Pattern Recognition[J]. Data Mining & Knowledge Discovery, 1998, 2(2):121-167.
[7] S.Oh, M.S.Lee , B.T.Zhang. Ensemble learning with active example selection for imbalanced biomedical data classification[J]. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 2011, 8(2): 316-325.
[8] S. Tang , Y.T. Zheng , Y. Wang , et al. Sparse ensemble learning for concept detection[J]. Multi-media, IEEE Transactions on, 2012, 14(1): 43-54.
[9] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting. in: Proceedings of the Second European Conference on Computational Learning Theory, Springer, Berlin, 1995, pp. 23-37.
[10] D.P. Solomatine, D.L. Shrestha. AdaBoost, RT: a boosting algorithm for regression problems[C]//Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on. IEEE, 2004, 2: 1163-1168.
[11] R. Polikar. Ensemble Machine Learning, first ed. Springer , New York, 2012.