Soft Meta-Cognitive Neural Network for Classification Problems

Maedeh Kafiyan
Faculty of engineering
Ferdowsi University Of Mashhad
Mashhad, Iran
Email: maedeh.kafiyansafari@mail.um.ac.ir

Modjtaba Rouhani
Faculty of engineering
Ferdowsi University Of Mashhad
Mashhad, Iran
Email: rouhani@um.ac.ir

***Abstract*—Classification problems in a sequential framework is considered as an important field in pattern recognition. One of the main concerns in these types of problems is overtraining. In metacognitive neural networks (McNN), overtraining could be avoided by using the confidence of classifier (CoC) measure, which assigns a value between [0,1] to class label. In this paper, a more accurate measure of CoC for McNN is presented. In addition, the hinge loss function which has no particular probabilistic interpretation is replaced by cross-entropy loss, which the output layer of the Soft meta-cognitive neural network (SMcNN) is a SoftMax layer. The classification performance is improved by applying the proposed SMcNN to well-known UCI datasets and comparing the results to McNN, SVM, and some other classifiers.**

***Index Terms*—Meta-cognitive, Cross-entropy, SoftMax, Classification, Neural network**

I. Introduction

Radial Basis function neural network (RBFNN) which has been proposed in the late 1996s [1] are considered to be one of the most widely used neural network methods. RBFNN has been broadly used in classification [2], regression, signal processing [3] and time series problems [4], [5]. This interest is due to the several advantages of RBFNN, such as their universal approximation [6], the simplicity of architecture and its good speed. In recent years, a series of new RBFNN approaches have used theoretical foundations of learning in human in their learning algorithms. Recent studies in human learning claim that, when students apply self-regulation in learning process using meta-cognition, the results could be more reliable [7]. Planning, choosing learning strategies and observing their progress are in one model to precisely control the learning process by the learner. Metacognition in human-body is a way to address the three main questions about learning: what-to-learn, when-to-learn, and how-to-learn [7]. thus, extending a Meta-cognitive Neural Network is essential to reply to these main questions. In learning process by using self-regulation and Nelson and Narence memory, two components are defined as cognitive and meta-cognitive, which the cognitive component is responsible for learning and sending the monitoring signal to the meta-cognitive component, and the meta-cognitive component selects the best possible strategy for cognitive component from received data and in the end, sends the results to the cognitive component by means of control signals. These questions in learning machines could be replied by applying one of these four strategies: sample deleting, weights updating, adding new neurons to the hidden layer and adding sample to the reserved list [8].

The basic concept of using meta-cognitive learning in RBF neural networks was initiated in [9], called Self-Adaptive Resource Allocation Network (SRAN). SRAN transfers the idea of meta-memory model Nelson and Narens and the initial meta-cognitive learning into the machine learning area. This idea has been extended to four different branches: Meta-cognitive neural network, Meta-cognitive neuro-fuzzy inference system, Meta-cognitive interval type-2 neuro-fuzzy system and Meta-cognitive extreme learning machine. so the first meta-cognitive learning algorithm is SRAN. In[8] inspired by meta-cognitive learning principal in human, second Meta-cognitive neural network for classification problems was proposed, called (McNN). This neural network used Hing loss function as an error and Extended Kalman filter to find the optimal weight. because Extended Kalman filter is complicated in [10], the projection learning algorithm was proposed called (PBL-MCRBFN). In [10] Sateesh find a close-form to calculate the optimal weight of McNN. In [11] and [12], Meta-cognitive extreme learning machine (McELM) is proposed for classification and regression problems. these two learning approaches decrease run time of McNN. Meta-cognitive neuro-fuzzy inference system called McFIS, proposed in [13] is for classification problem. All previous works were not robust against outliers, so, in[14] Meta-cognitive interval type-2 neuro-fuzzy inference system (McIT2FIS) is proposed for the regression problem. Both McFIS and McIT2FIS using Extended Kalman filter. Moreover, Pratama in [15] proposed a new recurrent neuro-fuzzy algorithm called rClass. This method tries to overcome the uncertainty problem and lack of robustness against destitution changes. Another algorithm present in the literature of Meta-cognitive learning is (RIVMcSFNN). In [16] recurrent interval-valued Metacognitive scaffolding fuzzy neural network is proposed, RIVMcSFNN which combines meta-cognitive and scaffolding theory to rich better performance.

Motivated by the advantages of SoftMax activation function in multiclass classification, in this paper, Soft Meta-cognitive Neural Network (SMcNN) is presented which uses SoftMax

978-1-5386-9569-2/18/$31.00 ©2018 IEEE

function as the confidence of classifier (CoC) measure and as the activation function of the output layer. In addition to this, due to its well-known probabilistic interpretation, cross-entropy loss function is implemented instead of hinge loss. Since error function and activation function of the last layer is changed, all the formulations of McNN are updated. The error function, the predicted class label, a new confidence of classifier measure obtained from SoftMax output, and class-wise significance are all used as knowledge measure of data. A gaussian kernel is employed to calculate class-wise significance measure, and the best strategy for each data sample is chosen by metacognitive component based on this information [8].

This paper is divided into four sections. The second section II explains the structure of meta-cognitive neural networks and describes the SMcNN as well. In the third section III the performance of SMcNN is compared to the other meta-cognitive algorithms. The performance of SMcNN classifier is evaluated using UCI classification datasets. The results indicate that the mean performance of SMcNN over 100 runs provides better accuracies as compared to previous meta-cognitive neural networks. Finally, we conclude the paper in Section IV.

II. Proposed Soft Meta-Cognitive network for classification problems

In this section, we introduce the proposed learning algorithm and the main idea of SMcNN classifier. This section begins with motivation and defining the classification problem. Next, we present the Soft Meta-cognitive Neural Network (SMcNN) architecture. In the end, we describe the learning algorithm.

A. Motivation

One of the biggest concerns in classification problems is overtraining, In previous meta-cognitive learning algorithms this problem handle by using the CoC measure, but the value assigns to this measure is not accurate enough. It calculated as follow

$$CoC = \frac{\min\left(1, \max\left(-1, y_j^t\right)\right) + 1}{2} \tag{1}$$

Where the y_j^t is the output of jth neuron in output layer which j in this formula is equal to the class label of tth input sample($j = c^t$). when the correct class is not detected, the output of this formula is not a very appropriate value, because the summation of neural network output is not equal to one. Assume we have 3 classes and, output is $[1.25, 0.5, 1.5]$, and the correct class label is one, but network prediction is 3, the output of CoC is 1 which does not give precise information about the confidence of classifier. So we proposed using SoftMax function as the activation function of the output layer and also as the CoC measure. Also in order to achieve higher accuracy we replace hinge loss function with cross-entropy error function.

B. Problem definition

Let us assume that data are arriving in a sequential order $\left\{\left(X^1, c^1\right), .., \left(X^t, c^t\right), ...\right\}$,where a learner attempts to resolve an online prediction task by learning and updating a model for future data at each step. Each $X^t \in \mathbb{R}^m$ is the tth input vector and the $c^t \in (1, n)$ is it's class label, where n and m represent the number of different classes and dimension of input vector respectively. Labels are encoded as the binary vector of length n, ($\mathbf{y}^t = \left[y_1^t, ...y_j^t, ...y_n^t\right]$) as follows :

$$y_j^t = \{ \begin{array}{ll} 1, & j = c^t \\ 0, & otherwise \end{array} \quad j = 1, 2..., n \tag{2}$$

At first, there is zero neuron in the hidden layer, then in each step, the strategy will be changed according to the data and network for a better approximation. More details about this are provided in the next section.

C. SMcNN architecture

Similar to the McRBFN, the proposed method consists of two components, a cognitive component, and a meta-cognitive component [10]. the cognitive component of SMcNN is a three-layered RBF Network with a SoftMax and Gaussian activation function in the output and hidden layer respectively. The meta-cognitive component must control the cognitive component, and therefore maintains a dynamic model of the cognitive component in order to specify control strategies [8], [10].

We assume the SMcNN adds K neurons to the hidden layer considering $t-1$ training data samples. When new training data arrives at the SMcNN, the meta-cognitive component determines the knowledge of data due to the estimated class label, error, Confidence of classifier and Class-wise significance. Using this information, the meta-cognitive component identifies the best learning strategy for present data. Figure 1 presents details about SMcNN architecture. The cognitive and meta-cognitive components are discussed in the next section [10].

1) Cognitive Component: Same as the architecture of standard radial basis function (RBF) networks, the proposed method consists of three layers: (1) an input layer using m nodes, (2) a nonlinear hidden layer with Gaussian activation function, and (3) an output layer using n nodes with SoftMax activation function. we assume that meta-cognitive component builds K hidden nodes from data 1 to $t-1$. the output of SMcNN for tth data is

$$z_j^t = \frac{exp(\widehat{y}_j^t)}{\sum_{i=1}^{n} exp(\widehat{y}_i^t)}, \quad j = 1, 2, ..., n \tag{3}$$

Where the z_j^t is the output network using SoftMax activation function which predicts the probability distribution over class j given the input data, and $\widehat{y}_j^t$ is the value entered to jth neuron before applying the SoftMax activation function. The $\widehat{y}_j^t$ is calculated by the following formula

$$\widehat{y}_j^t = \sum_{k=1}^{K} w_{jk} \phi_k(x^t), \quad j = 1, 2, ..., n \tag{4}$$

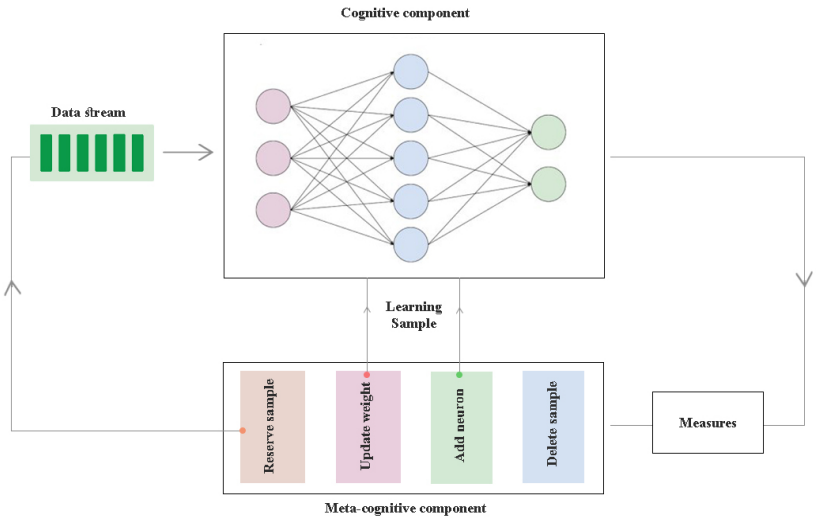

Fig. 1: SMcNN architecture

Where w_{ij} is the weight connecting the jth output neuron to kth hidden neuron and also $\phi_k(x^t)$ is the output of k hidden node with respect to the input x^t Which is calculated with

$$\phi_k(x^t) = exp(-\frac{\parallel x^t - \mu_k^l \parallel^2}{(\sigma_k^l)^2}) \tag{5}$$

Where μ_k^l is the center of kth hidden node for corresponding class label l. Unlike previous methods, the SoftMax activation function is used for the output layer in SMcNN.

2) Meta-Cognitive Component: Meta-cognitive neural networks use self-regulatory learning mechanism [10]. Based on a measuring predicted class label($\widehat{c}^t$), maximum error ($\widehat{E}^t$),confidence of classifier (z^t) and class-wise significance (ψ_c) this mechanism specifies what, when and How to learn data sample [8]. When new data (tth) is entered into the network, the amount of knowledge available in the data is calculated using the mentioned measures. The meta-cognitive component must specify, the best strategy for presented data based on the amount of its knowledge. More details on these strategies are given below.

Delete Strategy: This Strategy helps to overcome the over-fitting problem by deleting data, the network already learned.

Add Neuron: This strategy helps to learn data by adding new neuron to the hidden layer.

Update Weights: The important part of Learning in each neural network is finding the optimal weights. This strategy helps to learn data by updating network weights.

Reserve List: Sample adds to the end of data stream. Using this strategy, samples with the most information are learned first and other samples which add to the reserved list can be used to tune the parameters in the future.

The knowledge of the sample is measured by four measures as follow: The predicted class label for the presented sample is defined as

$$\widehat{c}^t = \max z_j^t, \quad j = 1, 2, ..., n \tag{6}$$

We assume cross-entropy function as error function. Cross-entropy in this problem is expressed as

$$error^t = -\sum_{j=1}^{n} c_j^t \log z_j^t \tag{7}$$

if we take the gradient of the error function with respect to one of the $\mathbf{w}_j$ vector, we achieve formula (8) and we use this formula as our optimal error [17].

$$e_j^t = y_j^t - \widehat{z}_j^t \quad j = 1, 2, ..., n \tag{8}$$

we get the error by subtracting the true value of each output node from the predicted output, and Maximum error for data tth is calculated by maximizing the error vector e_j^t as follow

$$E^t = \max e_j^t \quad j = 1, 2, ..., n \tag{9}$$

Taking advantage of SoftMax activation function, we define confidence of classifier. The class-wise significance which is a direct measure of novelty in the current sample [18] defined as the average distance of the current sample belongs to the class c from significantly contributing hidden neurons with the same class in a hyper-dimensional feature space [8]. Class-wise significance can be obtained using Eq (10).

$$\psi_c = \frac{1}{K^c} \sum_{K=1}^{K^c} h(x^t, \mu_k^c) \tag{10}$$

we assume that K^c is the number of contributing hidden neurons with the same class as x^t and h is the Gaussian kernel function. μ_k^c refers to the mean of kth hidden neuron, where c stands for hidden neuron with class label c. The smaller ψ_c gets (near to zero), the more novelty data has [10]

3) SMcNN learning algorithm: The meta-cognitive component based on the knowledge it maintains from measures, should find the best learning strategy for the current sample which represents the principles of self-regulated in human learning (what to learn, when to learn and how to learn)[10]. When the new sample arrives, one of the following strategy selected by SMcNN.

Delete Strategy: This strategy is used when the conditions (11) are satisfied.

$$if \quad \hat{c}^t == c^t \quad and \quad z^t < \beta_d \tag{11}$$

If the predicted class label of current data is as same as the class label and the output of SMcNN is lower than the deleted threshold (β_d), data have the same information as network and can delete without learning. This strategy helps SCmNN to avoid overtraining. The deleted threshold should select between [0.85,0.95].

Add New Neuron: When predicted class label is different from the true label for tth train data, or maximum error (E^t) is more than add threshold and also class-wise significance is lower than meta-cognitive knowledge measurement threshold, it means data has new information to learn [10]. The condition for adding the new neuron is expressible as follow

$$if \quad \left(\hat{c}^t \neq c^t \quad or \quad E^t \geq \beta_a\right) \quad and \quad \psi_c\left(x^t\right) \leq \beta_c \tag{12}$$

Where β_c is meta-cognitive novelty threshold, if ψ_c is smaller than the novelty threshold, it shows data novelty is high and different from network information. The range of novelty threshold can choose in the interval [0.3,0.7]. β_a is the self-adaptive add threshold. The initial value of this β_a can be selected between [0.6,0.8]. After adding new neuron to the hidden layer the add threshold updates as follow

$$\beta_a = \delta\beta_a + (1-\delta)E^t \tag{13}$$

where δ is self-adaptation slope. It must choose close to one. Value of β_a is increasing in each step, Due to this, the chance of adding a new neuron at the beginning is much more than the end of data stream. When new neuron is added to the hidden layer, SMcNN must specify the mean, variance and weights of the new neuron. The $K+1$ Neuron may be far away from the nearest neuron in the same classes and overlap with other classes, so it's critical how to set the variance and the mean of the new neuron. Based on the distance of presented data with inter(Nrl)/intra(Nrs) class nearest neuron Sateesh Babu In [8] has considered different modes to determine the conditions of the overlapping with other class. He determines four different states based on a Euclidian distance between current data and the mean of the inter-class nearest neuron (d_l) and intra-class nearest neuron (d_s). We assume μ_{K+1} ,σ_{K+1},and $\mathbf{W}_{K+1}$ are the mean, variance and weights of newly added neuron respectively.

No-overlapping with any classes states: when ($d_s \gg \sigma_{nrs}$ And $d_l \gg \sigma_(nrl)$) is satisfied, the current data does not overlap with any of inter/intra class nearest neuron. The mean, variance and weights of new neuron set as follow

$$\mu_{K+1}^c = x^t, \quad \sigma_{K+1}^c = \rho\sqrt{x^t x^t} \tag{14}$$

where ρ is positive value between [0.5,1].

No-overlapping states: when the ratio of intra/inter-class is less than one, current data do not overlap with other classes. New neuron variance and mean can be determined as

$$\mu_{K+1}^c = x^t, \quad \sigma_{K+1}^c = \rho \parallel x^t - \mu_{Nrs}^c \parallel \tag{15}$$

Minimum overlapping with the inter-class: when the ratio of intra/inter-class is between [1,1.5], current data is closer to the inter-class nearest neuron than to the other, so the parameters set as

$$\mu_{K+1}^c = x^t + \zeta\left(\mu_{Nrs}^c - \mu_{Nrl}^l\right),\ \sigma_{K+1}^c = \rho \parallel \mu_{K+1}^c - \mu_{Nrs}^c \parallel \tag{16}$$

where ζ is the center shift factor and has a fixed value, set to 0.1. it specifies how much new center must shift from the current data location.

Notable overlapping with the inter-class: when intra/inter-class ratio is more than 1.5, current data has notable overlapping with other classes. in this state, mean of new neuron must shift away from Nrl and define as follow

$$\mu_{K+1}^c = x^t - \zeta\left(\mu_{Nrl}^l - x^t\right)\sigma_{K+1}^c = \rho \parallel \mu_{K+1}^c - \mu_{Nrl}^l \parallel \tag{17}$$

These conditions help SMcNN to reduce the misclassifications error. Now we need to define the weight for the new hidden node. The weight is calculated as

$$\mathbf{w}_{K+1} = \frac{\mathbf{e}}{\phi_{K+1}\left(x^t\right)} \tag{18}$$

Update parameter: This strategy is used when condition (19) is satisfied.

$$\hat{c^t} = c^t \quad and \quad E^t \geq \beta_u \tag{19}$$

Where β_u is the self adaptive update threshold. Same as β_a, update threshold, is adapted by using (13) formula. The parameter vector α is define as $\alpha = [\mathbf{w}_1, ...\mathbf{w}_K]^T$. We assume weight matrix as follow

$$W = \begin{bmatrix} w_{11} & w_{12} & ... & w_{1K} \\ w_{21} & w_{22} & ... & w_{2K} \\ . & & & \\ . & & & \\ . & & & \\ w_{n1} & w_{n2} & ... & w_{nK} \end{bmatrix} \tag{20}$$

Each element of α is built by column of weight matrix [10]. In SMcNN we use Extended Kalman filter (EKF) to update weights.

$$\alpha_{new} = \alpha_{old} + \mathbf{Ge} \tag{21}$$

Where $\mathbf{G} \in \mathbb{R}^{q\times n}$ is Kalman gain matrix. We consider $q = K \times n$, and Kalman gain can compute as

$$\mathbf{G} = \mathbf{PB}\left[\mathbf{R} + \mathbf{B}^T\mathbf{PB}\right]^T \tag{22}$$

where $\mathbf{R}$ is $n \times n$ matrix and defines as $\mathbf{R} = r_0\mathbf{I}_{n\times n}$. $\mathbf{R}$ is the variance of the measurement noise. $\mathbf{P} \in \mathbb{R}^{q\times q}$ is the error covariance matrix and given by

$$\mathbf{P}_{new} = \left[\mathbf{I}_{q\times q} - \mathbf{GB}^T\mathbf{P}_{old} + s_0\mathbf{I}_{q\times q}\right] \tag{23}$$

Where s_0 is calls process noise that helps the algorithm to avoid from falling to local minima. $\mathbf{B}$ is the partial derivative of each output with respect to elements of α and is obtained,

$$\mathbf{B} = \left[\mathbf{M}\left(\mathbf{w}_1\right), \mathbf{M}\left(\mathbf{w}_2\right), ..., \mathbf{M}\left(\mathbf{w}_K\right)\right]^T \tag{24}$$

where $\mathbf{M}\left(\mathbf{w_i}\right)$ calculated as

$$\begin{bmatrix} \left(\frac{(L-L_1)\phi_i e^{\hat{y_1}}}{L^2}\right) & \left(\frac{-L_1\phi_i e^{\hat{y_2}}}{L^2}\right) & \cdots & \left(\frac{-L_1\phi_i e^{\hat{y_n}}}{L^2}\right) \\ \left(\frac{-L_2\phi_i e^{\hat{y_1}}}{L^2}\right) & \left(\frac{(L-L_2)\phi_i e^{\hat{y_2}}}{L^2}\right) & \cdots & \left(\frac{-L_2\phi_i e^{\hat{y_n}}}{L^2}\right) \\ \vdots & & & \\ \left(\frac{-L_n\phi_i e^{\hat{y_1}}}{L^2}\right) & \left(\frac{-L_n\phi_i e^{\hat{y_2}}}{L^2}\right) & \cdots & \left(\frac{(L-L_n)\phi_i e^{\hat{y_n}}}{L^2}\right) \end{bmatrix} \tag{25}$$

where L and L_j is equal to Eq(26) and Eq(27) respectively.

$$L = \sum_{j=1}^{n} e^{\hat{y}_j} \tag{26}$$

$$L_j = e^{\hat{y}_j} \quad j = 1, ..., n \tag{27}$$

By adding new neuron to the hidden layer we must update $\mathbf{P}$ matrix another time. Dimension of this matrix increases because q is changing thus we need to update $\mathbf{P}$ matrix as bellow

$$\begin{bmatrix} \mathbf{P}_{q\times q} & 0_{q\times n} \\ 0_{n\times q} & p_0\mathbf{I}_{n\times n} \end{bmatrix} \tag{28}$$

Where p_0 is the initial estimated uncertainty and I is the identity matrix.

Reserve Sample: If conditions for the other strategy are not satisfied, current data add to the reserved list to tune parameters in the future. When there is no data in the stream or the size of reserved list is not changed, training process will exit.

III. Experimental data

This section presents the results of the experiments performed. We did experiments on popular UCI classification data sets and we evaluate the effectiveness of SMcNN in terms of classification accuracy. Details about selected data sets are presented in table 1.

The measures used in evaluation is the overall classification accuracy which is obtained as follow

$$Acc = \left(\frac{DataCorrectlyClassified}{TotalNumberofData}\right) \times 100 \tag{29}$$

In the papers presented in [8], [10], no information is provided on the exact value of parameters for each dataset. Moreover, due to the sequential stream of data, the accuracy values are different during several performances, and in [8], [10] there is no explanation on how to consider the accuracy (taking into account the maximum accuracy or average over multiple performances). We have executed the algorithm 100 times and the average accuracy in these 100 runs has been higher than previous algorithms McNN and PBL-McRBFN. Table 2 shows the results of test data accuracy on these networks.

The results for both binary and the multi-category classification problems indicate that there is a significant increase in accuracy in all selected data sets except Image segmentation. As shown in table 2, not only using SoftMax function as the confidence of classifier measure improve general accuracy for multi-classification problem, it also increases the accuracy for binary classification problems.

Moreover, we compare our algorithm with well-known classifiers SVM, and ELM. Table 3 shows our test data accuracy for SVM and ELM, on some of UCI datasets which refer in table 1.

Based on result maintain from table 3 we can note that SMcNN performs much better in binary classification compared to SVM and ELM classifier.

TABLE I: Details about selected data sets

Data sets	Classes	Training Sample	Testing Sample
Wine	3	60	118
IRIS	3	45	105
Image-segmentation	7	210	2100
Liver disorders	2	200	145
Ionosphere	2	100	251

TABLE II: Accuracy comparison of SMcNN, McNN and PBL-McRBFN

Data sets	**SMcNN**	**McNN**	**PBL-McRBFN**
	Test	**Test**	**Test**
Wine	**91.47**	85.91	86.50
IRIS	**88.34**	80.30	75.6
Image segmentation	76.14	70.10	**87.3**
Liver disorders	**79.93**	72.34	75.50
Ionosphere	**93.61**	89.60	90.40

TABLE III: Accuracy comparison of SMcNN, ELM and SVM

Data sets	SMcNN	ELM	SVM
	Test	Test	Test
Wine	91.47	98.04	**98.04**
IRIS	88.34	96.19	**96.19**
Image segmentation	76.14	**90.67**	90.62
Liver disorders	**79.93**	72.41	70.21
Ionosphere	**93.61**	87.52	88.51

Despite the result that Sateesh and Suresh [10] found out, Meta-cognitive neural network classifier gives better accuracy compared to the SVM and ELM classifier. we found that Soft meta-cognitive neural network classifier is more accurate than batch SVM and ELM in binary classification problems.

IV. Conclusion

This paper presents Soft meta-cognitive neural network for classification problems in sequential stream of data mode. This algorithm is based on the cognitive foundations of learning in human and consists of two cognitive and meta-cognitive components. The meta-cognitive component is of particular importance because it is effective in choosing the best strategy, deleting sample, updating weights, adding new hidden neuron and reserving sample, for each input data. Based on these strategies we can specify what, how and when to learn efficiently. We used SoftMax function as a neural network output activation function and confidence of classifier. Using SoftMax function in addition to ease of differentiation and being in the range $(0-1)$, it is a suitable choice for representing the probability of each class. Finally, we evaluate our proposed SMcNN in compare to the best known classifier on the UCI classification repository. The results of our experiment indicate that SMcNN has higher accuracy in binary and multi-category classification in compare to SVM, ELM, and McNN, PBL-McRBFNN respectively.

References

[1] M. J. Orr *et al.*, "Introduction to radial basis function networks," 1996.

[2] S. Albrecht, J. Busch, M. Kloppenburg, F. Metze, and P. Tavan, "Generalized radial basis function networks for classification and novelty detection: self-organization of optimal bayesian decision," *Neural Networks*, vol. 13, no. 10, pp. 1075–1093, 2000.

[3] M. B. Li, G. B. Huang, P. Saratchandran, and N. Sundararajan, "Performance evaluation of GAP-RBF network in channel equalization," *Neural Processing Letters*, vol. 22, pp. 223–233, 2005.

[4] C. Harpham and C. Dawson, "The effect of different basis functions on a radial basis function network for time series prediction: A comparative study," *Neurocomputing*, vol. 69, no. 16-18, pp. 2161–2170, oct 2006.

[5] C.-C. Lee, Y.-C. Chiang, C.-Y. Shih, and C.-L. Tsai, "Noisy time series prediction using m-estimator based robust radial basis function neural networks with growing and pruning techniques," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4717–4724, 2009.

[6] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural computation*, vol. 3, no. 2, pp. 246–257, 1991.

[7] J. D. Vermunt, "Metacognitive, cognitive and affective aspects of learning styles and strategies: A phenomenographic analysis," *Higher Education*, vol. 31, no. 1, pp. 25–50, 1996.

[8] G. Sateesh Babu and S. Suresh, "Meta-cognitive Neural Network for classification problems in a sequential learning framework," *Neurocomputing*, vol. 81, pp. 86–96, 2012.

[9] S. Suresh, K. Dong, and H. J. Kim, "A sequential learning algorithm for self-adaptive resource allocation network classifier," *Neurocomputing*, vol. 73, pp. 3012–3019, 2010.

[10] G. S. Babu and S. Suresh, "Meta-cognitive RBF Network and its Projection Based Learning algorithm for classification problems," *Applied Soft Computing Journal*, vol. 13, pp. 654–666, 2013.

[11] R. Savitha, S. Suresh, and H. J. Kim, "A Meta-Cognitive Learning Algorithm for an Extreme Learning Machine Classifier," *Cognitive Computation*, vol. 6, pp. 253–263, 2014.

[12] Y. Zhang and M. J. Er, "Sequential active learning using meta-cognitive extreme learning machine," *Neurocomputing*, vol. 173, pp. 835–844, 2016.

[13] K. Subramanian, S. Member, S. Suresh, and S. Member, "A Metacognitive Neuro-Fuzzy Inference System (McFIS) for Sequential Classification Problems," *IEEE Transactions on Fuzzy Systems*, vol. 21, pp. 1080–1095, 2013.

[14] A. K. Das, K. Subramanian, and S. Sundaram, "An Evolving Interval Type-2 Neurofuzzy Inference System and Its Metacognitive Sequential Learning Algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 23, pp. 2080–2093, 2015.

[15] M. Pratama, S. G. Anavatti, J. Lu, and S. Member, "Recurrent Classifier based on An Incremental Meta- Cognitive-based Scaffolding Algorithm," *IEEE TRANSACTIONS ON FUZZY SYSTEMS*, vol. 6706, pp. 2048–2066, 2015.

[16] M. Pratama, E. Lughofer, M. J. Er, S. Anavatti, and C. P. Lim, "Data driven modelling based on Recurrent Interval-Valued Metacognitive Scaffolding Fuzzy Neural Network," *Neurocomputing*, vol. 262, pp. 4–27, 2017.

[17] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006. [Online]. Available: http://research.microsoft.com/en-us/um/people/cmbishop/prml/

[18] H. Hoffmann, "Kernel PCA for novelty detection," *Pattern Recognition*, vol. 40, pp. 863–874, 2007.