Classification of Escherichia Coli Bacteria using Meta-Cognitive Neural Network

Padma.S
Asst Professor Dept of Computer Science
K.S.Rangasamy College of Arts and Science
Tiruchengode, India
padmacde@gmail.com

Arun Kumar.M
Asst Professor, Dept of ECE
Bannari Amman Institute of Technology
Sathyamangalam, India

Abstract— The classification of data using machine learning involves different challenging tasks which depend on the learning method, selection of neurons, selection of dataset, selection of algorithm etc. This paper deals with classifying Escherichia Coli bacteria proteins from the amino acid sequence using Meta-cognitive learning. Different machine learning techniques are implied on the imbalanced dataset of E-Coli with 10 cross-fold validations to prove the performance of Meta-cognitive Neural Network (McNN) .McNN is capable of learning what –to-learn, when-to-learn and how-to-learn. Extreme Learning Machine (ELM) a batch learning algorithm, Self – adaptive Resource Allocation Network (SRAN) a sequential learning algorithm and Meta- cognitive Neural Network (McNN) which employs meta-cognition in sequential learning are applied for experimental study. This paper shows that classification of McNN performs well with respect to other machine learning algorithms.

Keywords- ELM, SRAN, McNN, Meta-Cognitive

I. INTRODUCTION

Data mining deals with two different types of learning supervised and unsupervised learning. Supervised learning is involved for classification as it is used to model different input and output relationships. The learning methods can be given to any kind of data. Likewise people, machine can also be trained for different learning methods. Learning may be rather sequential or batch. In sequential learning structure the samples that require training arrives one-by-one and after learning they are discarded which results in less memory and computational time. In addition the sequential algorithms automatically determine the minimal architecture that can accurately approximate the true decision function described by a stream of training samples. Radial basis function networks have been extensively used in a sequential learning framework due to its universal approximation ability and simplicity of architecture. [5]

A well known paradigm in fast learning neural network is extreme learning machine (ELM) [10]. It is a batch learning algorithm for a single-hidden layer feed forward neural network.

ELM chooses input weights randomly and analytically determines the output weights using minimum norm least-squares. Three activation functions are used in ELM namely Unipolar, Bipolar and Gaussian. The efficiency of this machine learning algorithm changes from one execution to the other, which can be averaged to a better efficiency. In case of sparse and imbalance data sets, the random selection of input weights in the ELM and its variants affects the performance significantly [2,11,12,25].

Another Sequential learning algorithm is Self-adaptive Resource Allocation Network (SRAN). [9] in which significant samples are selected using misclassification errors and hinge loss function. It has been shown in [9,13] that the selection of appropriate samples by removing repetitive samples helps in achieving better generalization performance. The sample neurons are reserved for future use. Pruning strategy is executed to improve the performance of classification.

Recent studies in human learning suggested that the learning process is effective when the learners adopt self-regulation in learning process using meta-cognition. The term meta-cognition is defined as ‘one’s knowledge concerning one’s own cognitive processes or anything related to them’. In particular the learner should control the learning process, by planning and selecting learning strategies and monitor their progress by analyzing the effectiveness of the proposed learning strategies. When necessary, these strategies should be adapted appropriately. Meta- cognition present in human-being provides a means to address what-to-learn, when-to-learn and how-to-learn, i.e., the ability to identify the specific piece of required knowledge, judge when to start and stop learning by emphasizing best learning strategy. Meta-cognitive Neural Network classifier is capable of deciding what-to-learn, when-to-learn and how-to-learn the decision functions from the training

data[15][16][17][18][19]. Meta-cognitive Neural Network (McNN) classifier which employs human-like meta-cognition to regulate the sequential learning process. [20][21][22][23][24]In this paper section II describes the methods and the dataset, section III discusses the experimental results and section IV concludes the paper.

II. METHODS

A. Extreme Learning Machine

Description of Extreme learning machine (ELM)

For *N* arbitrary distinct samples (x_i, t_i), where $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in R^m$, and $t_i = [t_{i1}, t_{i2}, ..., t_{in}]^T \in R^m$, standard SLFNs with $\tilde{N}$ hidden nodes and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(x_j) = \sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i . x_j + b_i) = o_j, \quad j = 1, 2, ..., N, \tag{1}$$

where $w_i = [w_{i1}, w_{i2}, ..., w_{in}]^T$ is the weight vector connecting the ith hidden node and the input nodes, $\beta_i = [\beta_{i1}, \beta_{i2}, ..., \beta_{in}]^T$ is the weight vector connecting the ith hidden node and the output nodes, and *bi* is the threshold of the *i*th hidden node. $w_i . x_j$ denotes the inner product of **wi** and **xj.** The standard SLFNs with $\tilde{N}$ hidden nodes with activation function $g(x)$ can approximate these *N* samples with zero error means by $\sum_{j=1}^{\tilde{N}} \| o_j - t_j \| = 0,$ i.e., there exist β_i, **w***i* and *bi* such that

$$\sum_{i=1}^{\tilde{N}} \beta_i g_i(w_i . x_j + b_i) = o_j, \quad j = 1, 2, ..., N, \tag{2}$$

The above equations can be rewritten compactly as

$$H\beta = T, \tag{3}$$

Where $H(w_1, ..., w_{\tilde{N}}, b_1, ..., b_{\tilde{N}}, x_1, ..., x_N)$

$$= \begin{bmatrix} g(w_1 . x_1 + b_1) & ... & g(w_{\tilde{N}} . x_1 + b_{\tilde{N}}) \\ . & & . \\ g(w_1 . x_N + b_1) & ... & g(w_{\tilde{N}} . x_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}}, \tag{4}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ . \\ \beta_{\tilde{N}}^T \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad T = \begin{bmatrix} t_1^T \\ . \\ t_N^T \end{bmatrix}_{N \times m} \tag{5}$$

As named in Huang et al. [10,12], **H** is called the hidden layer output matrix of the neural network; the ith column of **H** is the *i*th hidden node output with respect to inputs $x_1, x_2, ..., x_N$. In the case of learning an arbitrary function with zero training error, Baum had presented several constructions of SLFNs with sufficient hidden neurons. However, in practice, the number of hidden neurons required to achieve a proper generalization performance on novel patterns is much less. And the resulting training error might not approach to zero but can be minimized by solving the following equation:

$$\| H(\tilde{w}_1, ..., \tilde{w}_{\tilde{N}}, \tilde{b}_1, ..., \tilde{b}_{\tilde{N}}) \tilde{\beta} - T \| = \min_{w_i b_i \beta} \| H(w_1, ..., w_{\tilde{N}}, b_1, ..., b_{\tilde{N}}) \beta - T \| \tag{6}$$

When H is unknown the gradient-based learning algorithms are generally used to search the minimum of $\|H\beta - T\|$.The above problem is well established and known as a linear system optimization problem. Its unique least-squares solution with minimum norm is given by:

$$\hat{\beta} = H^{\dagger}T \tag{7}$$

where $H^{\dagger}$ is the Moore-Penrose generalized inverse of matrix H.

Algorithm ELM: Given a training set $\aleph = \{(x_i, t_i) \mid x_i \in R^n, t_i \in R^m, i = 1,..,N\}$, activation function g(x),and hidden node number $\tilde{N}$,

Step 1: Randomly assign input weight **w***i* and bias *bi* , $i = 1,..,\tilde{N}$.

*Step*2:Calculate the hidden layer output matrix **H.**

Step 3: Calculate the output weight β

B. Self-Adaptive Resource Allocation Network

Online/sequential learning for a multi-category classification problem can be stated in the following manner. The observation data arrives one-by-one and one at a time. After learning, the sample is discarded from the sequence. Suppose we have the observation data **{(x1,y1),(x2,y2),**... **,(xt,yt),**...}, where $x_t \in \Re^m$ is an m-dimensional features of observation *t* and $y_t \in \Re^m$ is its coded class label. Here, *n* represents the total number of classes. For notational convenience, the subscript *t* is left out in all further discussion. If the feature observation **x** is assigned to the class label c, then cth element of **y** = [y1,... *,yc*,... *,yn]T* is 1 and other elements are -1.

$$y_j = \begin{cases} 1 & if \;\; j == c \\ -1 & otherwise, \qquad j = 1,2,..,n \end{cases} \tag{8}$$

The observation data are random variables and the observation **x** provides some useful information on probability distribution over the observation data to predict the corresponding class label with certain accuracy. Hence, the classification problem is to predict the coded class label **y** of a new observation **x.** This requires us to estimate a functional relationship between the coded class label and feature space from sequential training data. In the SRAN classifier, a radial basis function network is used as a building block. The SRAN network approximates the functional relationship between the feature space and the coded class label.

The output of the SRAN classifier $(\hat{y} = [\hat{y}_1,...,\hat{y}_n]^T)$ with K hidden neurons has the following form:

$$\hat{y}_i = \sum_{j=1}^{K} \alpha_{ij} y_h^j, \quad i = 1,2,..,n \tag{9}$$

$$y_h^j = \exp\left(-\frac{\left\| x - \mu_j^I \right\|^2}{(\sigma_j^I)^2}\right) \tag{10}$$

where μ_j^I is the jth neuron center corresponding to the ith class, σ_j^I is the width of the jth neuron and α_{ij} is the weight connecting the ith output neuron and jth Gaussian neuron.

The predicted class label *cc* for the new training sample is given by

$$\hat{c} = \arg \max_{i \in 1,2,..,n} \hat{y}_i \tag{11}$$

In other sequential learning algorithms, the error **(e**) is usually the difference between the actual output and predicted output $(y - \hat{y})$**,** which is used in the mean square error loss function. For classification problems, the above definition of error restricts the outputs of the neural classifier between 7-1. In [13,14], it is shown that the classifier developed using a hinge loss function can estimate the posterior probability more accurately than the mean square error loss function. Hence, in our formulation, we use a hinge loss function to calculate the error **e** = [e1,e2,... *,en]T* and is given below

$$e_i = \begin{cases} y_i - \hat{y}_i & if \;\; y_i \hat{y}_i < 1 \\ 0 \quad otherwise, & i = 1,2,..,n \end{cases} \tag{12}$$

With this hinge loss function, the network output can grow beyond ± 1 and prevent the saturation problems in the learning process. The truncated outputs of the classifier model approximate the posterior probability accurately. The truncated output is defined as

$$T(\hat{y}_i) = \min(\max(\hat{y}_{i,} - 1), 1), \qquad i = 1,2,...,n \tag{13}$$

Since, the target vectors are coded as - 1 or 1, the posterior probability of observation vector **x** belonging to class *c* is

$$\hat{p}(c \mid x) = \frac{T(\hat{y}) + 1}{2} \tag{14}$$

In the setting of standard online/sequential learning, the training sample arrives one at a time and the network adapts its parameters based on the difference in knowledge between the network and the current sample. When new sample **(x***t)* is arrived to the network, based on the sample error **(e),** the sample is either used for network training (growing/learning) immediately, pushed to the rear end of the stack for learning in future, or deleted from the data set. The detailed description of SRAN is explained in [8].

Algorithm for SRAN

Input : Present the training data one-by-one to the network from data stream.

Output : Decision function that estimates the relationship between feature space and class label.

START

Initialization : Assign the first sample as the first neuron (K = 1).

Start learning for samples *t* = 2,3,...

DO

Compute significance of the sample to the network:

Compute the network output $\hat{y}_t$.

Find the maximum absolute hinge error *E* and predicted class label $\hat{c}$.

Delete Redundant samples:

IF $E \le 0.05$ **THEN**

Delete the sample from the sequence without learning.

ENDIF

IF $c \neq \hat{c} \quad AND \quad E \ge \eta_a$ **THEN**

Add a neuron to the network (K = *K* +1).

Choose the parameters of the network

Update the control adding parameters

ELSEIF $c == \hat{c} \quad AND \quad E \ge \eta_1$ **THEN**

Update the parameters of the network using EKF

Update the control parameters

ELSE

The current sample $\mathbf{x^t}$, $\mathbf{y^t}$ is pushed to the rear end of the sample stack to be used in future.

They can be later used to fine-tune the network parameters.

ENDIF

ENDDO

C. Meta-cognitive Neural Network

In this section, we present the architecture of the Meta cognitive Neural Network (McNN) classifier and its working principles. McNN architecture is developed based on the Nelson and Narens meta-cognition model. The information flow from the cognitive component to meta-cognitive component is considered monitoring, while the information flow in the reverse direction is considered control. McNN has two components namely the

cognitive component and the meta-cognitive component. The cognitive component of McNN is a three layered feed forward radial basis function network with Gaussian activation function in the hidden layer. The meta-cognitive component contains copy of the cognitive component. When a new training sample arrives, the meta-cognitive components of McNN predicts the class label and estimate the knowledge present in the new training sample with respect to the cognitive component. Based on this information, the meta-cognitive component selects a suitable learning strategy, for the current sample. Thereby, addressing the three fundamental issues in learning process: (a) what-to-learn, (b) when-to-learn and (c) how-to-learn [15-24].

a. Cognitive component of McNN

The cognitive component of McNN is a three layered feed forward radial basis function network. The input layer maps all features to the hidden layer without doing any transformation, the hidden layer employs Gaussian activation function and the output layer uses a linear activation function.

Without loss of generality, we assume that the meta-cognitive learning algorithm builds K Gaussian neurons from i-1 training samples. For given training sample X^i , the predicted output $(\hat{Y}^i = [\hat{Y}_1,...,\hat{Y}_j,...,\hat{Y}_n]^T)$ of McNN classifier with K hidden neurons is

$$\hat{Y}_j^i = \alpha_{j0} + \sum_{k=1}^{k} \alpha_{jk}\phi_k(X^i),\ \text{j = 1,2,..,n} \tag{15}$$

where α_{j0} is the bias to the α_{jk} output neuron, $\phi_k(X^i)$ is the weight connecting the kth hidden neuron to the jth output neuron and $\phi_k(X^i)$ is the response of the kth hidden neuron to the input x^i is given by

$$\phi_k(X^i) = \exp\left(-\frac{\| X^i - \mu_k^l \|^2}{(\sigma_k^l)^2}\right) \tag{16}$$

where μ_k^l is the center and σ_k^l is the width of the kth hidden neuron. Here, the superscript l represent the class that hidden neuron belongs to.

b. Meta-cognitive component of McNN

The meta-cognitive component uses estimated class label $\hat{c}$, maximum hinge error (E), posterior probability as confidence measure $(\hat{p}(j \mid x^i))$ and spherical potential based class-wise [20] significance as a measure of knowledge in the new training sample. Using these measures, the meta-cognitive component devices various learning strategies. First, we describe these measures in detail.

Estimated class label ($\hat{c}$): Using the predicted output $(\hat{y}^i)$, the estimated class label ($\hat{c}$) can be obtained as

$$\hat{c} = \arg^{\max}_{j\in 1,2,..,n} \hat{y}_j^i \tag{17}$$

Maximum hinge error (E): The objective of the classifier is to minimize the error between the predicted output $(\hat{y}^i)$ and actual output (y^i). In classification problems, it has been shown that the classifier developed using hinge loss function estimates the posterior probability more accurately than the classifier developed using mean square error function . Hence, in McNN, we use the hinge loss error $(e = [e_1,..,e_j,..,e_n]^T)$ defined as

$$e_j = \begin{cases} 0 & \text{if } \hat{y}_j^i y_j^i > 1, \\ \hat{y}_j^i - y_j^i & \text{otherwise,} \end{cases} \quad j = 1,2,..,n \tag{18}$$

The maximum absolute hinge error (E) is given by

$$E = \max_{j\in 1,2,..,n} \left| e_j \right| \tag{19}$$

Confidence of classifier: The confidence level of classification or predicted posterior probability is given as

$$\hat{p}^{i}(c \mid x^{i}) = \frac{\min(1, \max(-1, \hat{y}_j^{i})) + 1}{2} \quad (20)$$

Class-wise significance (ψ_c) : In general, the input feature (x) is mapped on to a hyper-dimensional spherical feature space S using K Gaussian neurons, i.e., $x \to \phi$. Therefore, all $\phi(x)$ lie on a hyper-dimensional sphere. The knowledge or spherical potential of any sample in original space is expressed as a squared distance from the hyper-dimensional mapping $\mathbb{S}$ centered at ϕ_0 .

In McNN, the center (μ) and width (σ) of the Gaussian neurons describe the feature space $\mathbb{S}$. Let the center of the K-dimensional feature space be $\phi_0 = (\frac{1}{k})\sum_{k=1}^{k}\phi(\mu_k)$. The knowledge present in the new data x^i can be expressed as the potential of the data in the original space, which is squared distance from the K-dimensional feature space to the center ϕ_0 .The potential (ψ) is given as

$$\psi = \left\| \phi(x^i) - \phi_0 \right\|^2 \quad (21)$$

Using the steps , the above equation can be expressed as

$$\psi = \phi(x^i, x^i) - \frac{2}{k}\sum_{k=1}^{k}\phi(x^i, \mu_k^l) + \frac{1}{k^2}\sum_{k,r=1}^{k}\phi(\mu_k^l, \mu_r^l) \quad (22)$$

From, the above equation, we can see that for Gaussian function the first term $(\phi(x^i, x^i))$ and last term $((\frac{1}{k^2})\sum_{k,r=1}^{k}\phi(\mu_k^l, \mu_r^l))$ are constants. Since potential is a measure of novelty, these constants may be discarded and the potential can be reduced to

$$\psi \approx -\frac{2}{k}\sum_{k=1}^{k}\phi(x^i, \mu_k^l) \quad (23)$$

Since we are addressing classification problems, the class-wise distribution plays a vital role and it will influence the performance of the classifier significantly. Hence, we use the measure of the spherical potential of the new training sample x^i belonging to class c with respect to the neurons associated to same class (i.e. l=c). Let K^c be the number of neurons associated with the class c, then class-wise spherical potential or class-wise significance (ψ_c) is defined as

$$\psi_c = \frac{1}{k^c}\sum_{k,r=1}^{K^c}\phi(x^i, \mu_k^c) \quad (24)$$

The spherical potential directly indicates the knowledge contained in the sample, a higher value of spherical potential (close to one) indicates that the sample is similar to the existing knowledge in the cognitive component and a smaller value of spherical potential (close to zero) indicates that the sample is novel.

McNN algorithm can be summarized as below:

1. For each new training sample input (X^i) compute the cognitive component output $(\hat{y})$.

2. The meta-cognitive component finds the estimated class label $(\hat{c})$, maximum hinge error (E), confidence $(\hat{p}(\hat{c} \mid X^i))$ and class wise significance ψ_c measures for the new training sample (X^i).

3. The meta-cognitive component selects one of the following strategies based on the above calculated measures.

(a) Sample delete strategy: If $(\hat{p}(\hat{c} \mid X^i)) \geq \beta_d$ AND $c == \hat{c}$ then delete the sample from the sequence without learning.

(b) Neuron growth strategy: If $\hat{c} \neq c$ AND $\psi_c(X^i) \leq \beta_c$ AND $E \geq \beta_a$, then allocate a new hidden neuron in the cognitive component and its parameters are calculated based on the intra- and inter-class nearest neurons distances . Also, update the self-adaptive meta-cognitive addition threshold and increase the dimensionality of P.

(c) Parameters update strategy: If $c == \hat{c}$ AND $E \geq \beta_u$, then update the cognitive component parameters using EKF. Also, update the self-adaptive meta-cognitive update threshold

(d) Sample reserve strategy: When the new sample does not satisfy deletion, growth and update criterion, then push the sample to the reserve to be used later for learning.

4. The cognitive component executes the above selected strategy.

5. Continue steps 1-4 until there are no more samples in the training data stream or number of samples remains same in the reserve.

D. Dataset

a. E.coli Dataset

Performance of classification may get differ depending on the dataset. If the dataset is imbalanced it results in the performance of the machine learning algorithm used.[2,25].The prokaryotic gram-negative bacterium Escherichie Coli is an important component of the biosphere, it colonizes the lower gut of animals and humans. The Escherichia Coli benchmark dataset has been submitted to the UCI Machine Learning Data Repository. The dataset patterns are characterized by attributes calculated from the amino acid sequences. Protein patterns in the E.coli dataset are classified to eight classes, it is a drastically imbalanced dataset of 336 patterns. Protein patterns in this dataset are organized as follows: 143 patterns of cytoplasm (cp), 77 of inner membrane without signal sequence (im), 52 of periplasm (pp), 35 of inner membrane without uncleavable signal sequence (imU), 20 of outer membrane without lipoprotein (omL), 5 of outer membrane with lipoprotein (omL), 2 of inner membrane without lipoprotein (imL) and 2 patterns of inner membrane with cleavage signal sequence (imS). The class distribution is extremely imbalanced, especially for imL and imS proteins.[1]

III. EXPERIMENTAL RESULTS

The performance evaluation of the ELM , SRAN and McNN classifier using real world classification problems from UCI machine learning repository. The proposed methods have been implemented using MATLAB. The statistical measures could be used to evaluate the performance of the proposed method.

a. Cross Validation

In this study, we used Cross Validation tests to evaluate the classifier robustness, this methodology is most suitable to avoid biased results. Thus, the whole training set was divided into ten mutually exclusive and approximately equal-sized subsets and for each subset used in test, the classifier was trained on the fusion of all the other subsets. So, cross validation was run ten times for each classifier and the average value of the ten-cross validations was calculated to estimate the overall classification accuracy.

b. Performance Measures :

Class-level performance is indicated by the percentage classification which tells us how many samples belonging to a particular class have been correctly classified [9]. The percentage classification η_i for class *ci* is

$$\eta_i = \frac{q_{ii}}{N_i^T} \tag{25}$$

where q_{ii} is the number of correctly classified samples and N_i^T is the number of samples for the class *ci* in the testing data set. The global performance measure the average (η_a) classification efficiency, is defined as

$$\eta_a = \frac{1}{n_c} \sum_{i=1}^{n_c} \eta_i \tag{26}$$

where *nc* is the total number of classes.

TABLE 1. TRAINING AND TESTING EFFICIENCIES OF 10 FOLDS.

Fold No	ELM		SRAN		McNN	
	Tra Eff	*Tes Eff*	*Tra Eff*	*Tes Eff*	*Tra Eff*	*Tes Eff*
1	86	81	90	88	93	96
2	85	84	91	87	93	90
3	84	85	87	85	93	86
4	86	87	91	93	92	93
5	83	84	90	89	93	92
6	86	85	90	85	94	86
7	87	80	91	85	93	85
8	85	83	91	93	91	93
9	84	83	90	90	91	90
10	86	82	90	84	93	85

Table 1. Discusses in detail the average training and testing efficiencies of ten cross fold validation. As already discussed the dataset is divided into ten out of that 9 folds are considered for training and the remaining one fold for testing. Every fold result is compared with ELM, SRAN and McNN and it shows that the performance result of McNN is better when compared to ELM and SRAN.

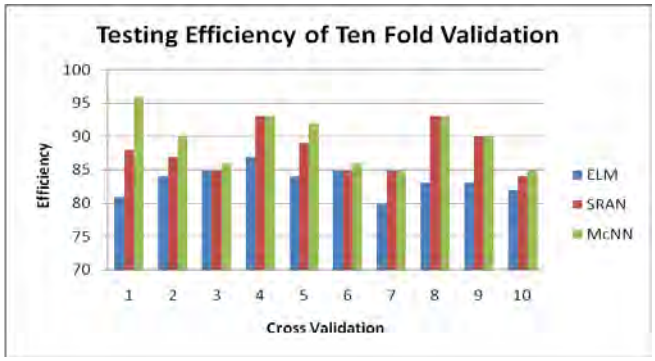

Figure 1 Comparison of testing efficiences with algorithms. x-axis with its fold numbers and testing efficiencies in the y-axis.

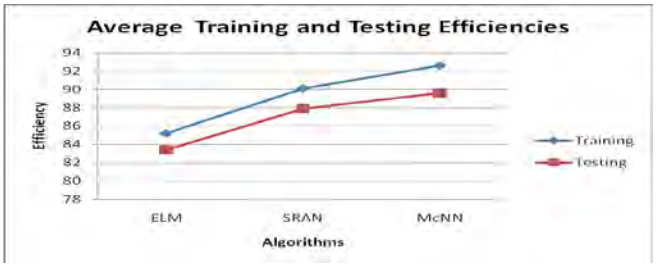

Figure 2 Comparison of average training and testing efficiences with algorithms. x-axis with its effieciency values and algorithms used for experiment in the y-axis.

The above figures fig 1 and fig 2 clearly explains the better performance of McNN. Both training and testing efficiences comparison are shown which describes that McNN performs well when compared to SRAN. SRAN performs well when compared to ELM.

IV. CONCLUSION

This paper concludes that McNN a sequential learning algorithm with the policy of when-to-learn, what-to-learn and how-to-learn performs well in classifying an imbalanced dataset E-Coli. The learning strategy of the classifier reduces the growth of neurons and hence improves the performance of classification which is represents in the experimental results. The ten cross fold validation is used to improve results. Finally when compared to batch learning algorithm ELM and sequential algorithm SRAN, McNN which is also a sequential algorithm performs well.

REFERENCES

[1] Hafida Bouziane, Belhadri Messabih, and Abdallah Chouarfia, "Meta-Learning for Escherichia Coli Bacteria Patterns Classification" Proceedings ICWIT 2012.

[2] Padma, S., Kumar, S.S. , Manavalan, R., "Performance analysis for classification in balanced and unbalanced data set", Industrial and Information Systems (ICIIS), 2011 6th IEEE International Conference on (2011) 300-304.

[3] J.C. Platt, "A resource allocation network for function interpolation", Neural Computing. 3 (2) (1991) 213–225.

[4] L. Yingwei, N. Sundararajan, P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural net-works", Neural Comput. 9 (2) (1997) 461–478.

[5] L. Yan, N. Sundararajan, P. Saratchandran, "Analysis of minimal radial basis function network algorithm for real-time identification of non-linear dynamic systems", IEE Proc. Control Theory Appl. 147 (4) (2000) 476–484.

[6] G.-B. Huang, P. Saratchandran, N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks", IEEE Trans. Syst. Man Cybern. Part B Cybern. 34 (6) (2004) 2284–2292.

[7] N.-Y. Liang, G.-B. Huang, P. Saratchandran, N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks", IEEE Trans. Neural Networks 17 (6) (2006) 1411–1423.

[8] S. Suresh, N. Sundararajan, P. Saratchandran, "A sequential multi-category classifier using radial basis function networks", Neurocomputing 71 (1) (2008) 1345–1358.

[9] S. Suresh, K. Dong, H.J. Kim, "A sequential learning algorithm for self-adaptive resource allocation network classifier", Neurocomputing 73 (16–18) (2010) 3012–3019.

[10] G. Huang, Q. Zhu, C. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks". in: IEEE International Joint Conference on Neural Networks. Proceedings, vol. 2, 2004, pp. 985–990.

[11] S. Suresh, R.V. Babu, H.J. Kim, "No-reference image quality assessment using modified extreme learning machine classifier", Appl. Soft Comput. 9 (2) (2009) 541–552.

[12] G.-B. Huang, D. Wang, Y. Lana, "Extreme learning machines: a survey", Int. J. Mach. Learn. Cybern. 2 (2) (2011) 107–122.

[13] S. Suresh, N. Sundararajan, P. Saratchandran, "Risk sensitive hinge loss functions for sparse multi-category classification problems", Inf. Sci. 178 (12) (2008) 2621-2638.

[14] T. Zhang, "Statistical behavior and consistency of classification methods based on convex risk minimization", Ann. Stat. 32 (1) (2003) 56-85.

[15] R. Savitha, S. Suresh and H. J. Kim, "A Meta-cognitive Learning Algorithm for an Extreme Learning Machine Classifier," Cognitive Computation (Accepted), 2013

[16] K. Subramanian, S. Suresh, N. Sundararajan, "A meta-cognitive neuro-fuzzy inference system (McFIS) for sequential classification problems", IEEE Trans. on Fuzzy Systems, 2013.

[17] G. Sateesh Babu, S. Suresh, "Sequential projection based metacognitive learning in a Radial basis function network for classification problems", IEEE Trans, on Neural Networks and Learning Systems, 24(2), pp. 194-206, 2013.

[18] K. Subraminan, and S. Suresh, "Human action recognition using meta-cognitive neuro-fuzzy inference system", International Journal of Neural Systems, 22(6), 2012.

[19] K. Subraminan, and S. Suresh, "A meta-cognitive sequential learning algorithm for neuro-fuzzy inference system", Applied soft computing, 12(11), 36703-3614, 2012.

[20] G. Sateesh Babu, and S. Suresh," Meta-cognitive RBF networks and Its Projection based Learning Algorithm for Classification problems", Applied Soft Computing, 13(1), pp. 654-666, 2013.

[21] G. Sateesh Babu, and S. Suresh, "Parkinson's Disease Prediction using Gene Expression - A Projection based Learning Metacognitive Neural Classifier Approach", Expert System with Applications, 40(5), pp. 1519-1529, 2012.

[22] R. Savitha, S. Suresh, and N. Sundararajan, "Metacognitive learning algorithm for a fully complex-valued relaxation network", Neural Networks, 32, pp. 309-318, 2012.

[23] G. Sateesh Babu, and S. Suresh, "Meta-cognitive neural network for classification problems in a sequential learning framework", Neurocomputing, 81(1), pp. 86-96, 2012.

[24] R. Savitha, S. Suresh, and N. Sundararajan, "Meta-cognitive learning in a fully complex-valued radial basis function network", Neural Computation, 24(5), pp. 1297-1328, 2012.

[25] S.Padma. and J.Joshua Sam Paul "Novel Methods for Classification Using Machine Learning Concepts", AET_ACS 2013 Delhi, 164-167.