2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Constrained Extreme Learning Machine: a Novel Highly Discriminative Random Feedforward Neural Network

Wentao Zhu, Jun Miao and Laiyun Qing

***Abstract*—In this paper, a novel single hidden layer feedforward neural network, called Constrained Extreme Learning Machine (CELM), is proposed based on Extreme Learning Machine (ELM). In CELM, the connection weights between the input layer and hidden neurons are randomly drawn from a constrained set of difference vectors of between-class samples, rather than an open set of arbitrary vectors. Therefore, the CELM is expected to be more suitable for discriminative tasks, whilst retaining other advantages of ELM. The experimental results are presented to show the high efficiency of the CELM, compared with ELM and some other related learning machines.**

I. Introduction

MANY neural network architectures have been proposed during the past two decades. The feedforward neural networks are the most popular ones studied by researchers. It has been proved that multilayer feedforward networks with non-polynomial activation functions can approximate any continuous function [1]. Single hidden layer feedforward neural networks (SLFNs) were studied extensively because of their relatively fast learning speed and simple neural network structure. It is shown that the SLFNs have the same approximate capabilities as multi-layer feed forward neural networks [2, 3]. Moreover, Tamura et al. [2] showed that the weights from the input layer to the hidden layer of SLFNs can be randomly generated. Huang et al. [4] further proved the above theory rigorously, and proposed a new type of SLFNs called ELM.

ELM can be considered as a linear system after the nonlinear feature mapping of the hidden layer [5]. Therefore, ELM has a closed form of solution due to the simple network structure and randomness of hidden layer parameters. The essence of linear system used by ELM is to minimize the training error and the norm of connection weights from the hidden layer to the output layer at the same time. Hence ELM has a good generalization performance according to feedforward neural network theory [6]. As a consequence, ELM has some desirable features, such as that hidden layer parameters need not be tuned, fast learning speed and good generalization performance. These advantages lead to the popularity of ELM both for researchers and engineers.

However, the random selection of hidden layer parameters makes quite inefficient use of hidden nodes [7]. ELM usually has to randomly generate a great number of hidden nodes to achieve desirable performance. This leads to time consuming in test process, which is not helpful in real applications. Large numbers of hidden nodes also easily lead to over fitting in training. There are mainly three ways to solve the problem:

1. Use online incremental learning methods to add hidden layer nodes dynamically [7, 8]. These methods randomly generate parts or all of the hidden layer nodes, and then select the candidate hidden nodes one by one or chunk by chunk with fixed or varying chunk size. Whether the hidden layer node is added or not is usually depending on the output layer objective function.
2. Use pruning methods to select the candidate hidden layer nodes [9, 10]. These methods start with a large neural network using the traditional ELM, and then apply some metrics, such as statistical criteria and multi-response sparse regression, to rank these hidden nodes. Finally, eliminate those low relevance hidden nodes to form a more compact neural network structure.
3. Use gradient based methods to update the weights from the input layer to the hidden layer in ELM [11]. These methods study the mathematical model of ELM, and then find the derivation of ELM optimal function with respect to hidden parameters. After randomly initialize the weights from the input layer to the hidden layer and use a close-form least square solution to calculate the weights from the hidden layer to the output layer, these methods use the gradient descending method to update the weights from the input layer to the hidden layer in ELM iteratively.

The above methods can overcome the drawbacks of the traditional ELM to some degree. However, they do not solve the problem directly from the essence of hidden nodes. Besides, these methods are somewhat time-consuming.

Actually, the essence of hidden layer functions is to map the data into a feature space, where the output layer can use a classifier to separate the feature-mapped data perfectly. Therefore, the hidden layer should extract discriminative features for data classification tasks.

LDA [12] is probably the most commonly used methods to extract discriminative features. However, LDA has some drawbacks, such as that the number of feature-mapped dimensions is less than the number of classes, "Small Sample Size" (SSS) problem and Gaussian distribution assumption of

W. Zhu and J. Miao are with the Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China (e-mail: wentao.zhu@vipl.ict.ac.cn; jmiao@ict.ac.cn).

L. Qing is with School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: lyqing@ucas.ac.cn).

This work was supported in part by Natural Science Foundation of China (Nos. 61175115 and 61272320) and President Fund of Graduate University of Chinese Academy of Sciences (No. Y35101CY00).

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

equal covariance and different means. Su et al. [13] proposed a projection pursuit based LDA method to overcome these problems. The method showed that the difference vectors of between-class samples have a strong discriminative property for classification tasks, but this method is rather complex with many embedded trivial tricks.

In this work, to balance between the high discriminative feature learning and the simplicity and fast training speed of the ELM, we propose a novel model, called Constrained Extreme Learning Machine (CELM), which utilizes a random subset of difference vectors of between-class samples to replace the completely random connection weights from the input layer to the hidden layer in ELM. Experimental results show that, CELM has better generalization ability than ELM and other related methods. We also compared the CELM algorithm with SVM and ELM related algorithms [14, 15] on CIFAR-10 data set [16]. The results show that the CELM algorithm outperforms these methods.

The remaining part of the paper is organized as follows: in section Ⅱ, we firstly review the traditional ELM algorithm, and then propose the CELM algorithms. Experiments are presented in section Ⅲ. Conclusion and discussion are given in section Ⅳ.

II. CONSTRAINED EXTREME LEARNING MACHINE

In this section, we firstly review the traditional ELM algorithm. Then we introduce the CELM algorithm with the idea of using difference vectors to generate discriminative hidden nodes into the traditional ELM.

A. Review of Extreme Learning Machine

ELM is a type of SLFNs. The hidden layer parameters, i.e., the connection weights from the input layer to the hidden nodes, are randomly generated in ELM. The output layer is a linear system, where the connection weights from the hidden layer to the output layer are learned by computing the Moore-Penrose generalized inverse [5]. The ELM network has extreme fast learning speed due to the simple network structure and its closed form solution. Additionally, the randomness makes ELM not necessarily tune these hidden layer parameters iteratively.

Given the training samples and class labels $\aleph=\{(\mathbf{x}_i,\mathbf{t}_i)|\mathbf{x}_i\in\mathbf{R}^n,\mathbf{t}_i\in\mathbf{R}^m, i=1,\ldots,N\}$, the number of hidden nodes L and activation function $G(\mathbf{a},b,\mathbf{x})$, where $\mathbf{x}\in\mathbf{R}^n$ is the input vector, $\mathbf{a}\in\mathbf{R}^n$ is the associated connection weight vector and $b\in\mathbf{R}$ is the bias, the algorithm of ELM network can be concluded as the following three steps:

Step 1: Assign the parameters $\{(\mathbf{a}_j,b_j)\,|\,j=1,\ldots,L\}$ of hidden nodes with randomly generated values.

Step 2: Calculate the hidden layer output matrix H for all the training samples:

$$\mathbf{H}=\begin{bmatrix}\mathbf{h}(\mathbf{x}_1)\\ \vdots\\ \mathbf{h}(\mathbf{x}_N)\end{bmatrix}=\begin{bmatrix}G(\mathbf{a}_1,b_1,\mathbf{x}_1) & \cdots & G(\mathbf{a}_L,b_L,\mathbf{x}_1)\\ \vdots & \cdots & \vdots\\ G(\mathbf{a}_1,b_1,\mathbf{x}_N) & \cdots & G(\mathbf{a}_L,b_L,\mathbf{x}_N)\end{bmatrix}_{N\times L}$$

, where $G(\mathbf{a}_i,b_i,\mathbf{x}_j)$ is the activation function of ith hidden node for jth sample.

Step 3: Calculate the hidden layer's output connection weights β by solving the least squares problem:

$$\beta=\mathbf{H}^{\dagger}\mathbf{T}$$

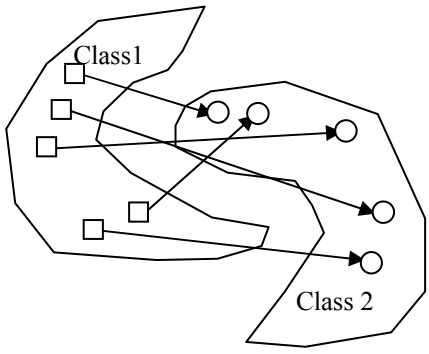
, where $\mathbf{H}^{\dagger}$ is the generalized inverse matrix of the matrix $\mathbf{H}$, and $\mathbf{T}=\begin{bmatrix}\mathbf{t}_1^T\\ \vdots\\ \mathbf{t}_N^T\end{bmatrix}_{N\times m}$.

As analyzed in theory and further verified by the simulation results in [17], ELM for classification tends to achieve better generalization performance than traditional SVM. ELM can also overcome the local minimal problem that BP neural nets faced, due to its convex model structure. The learning speed of ELM is extremely fast at the same time.

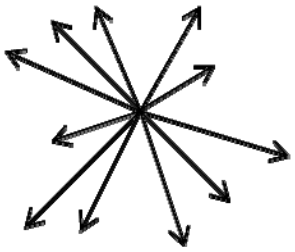
B. Constrained Extreme Learning Machine

The completely random parameters in the hidden layer of ELM do not always represent discriminative features. Such unconstrained random parameters may make ELM has to generate a great number of hidden nodes to meet desirable generalization performance. More hidden nodes mean more processing time, more computational resource and more easily over fitting. These problems in ELM should be solved.

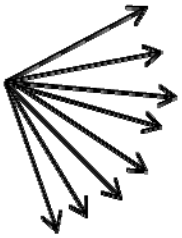
Although the method [13] is rather complex with many embedded trivial tricks, it shows that the difference vectors of between-class samples are effective to classification tasks. Considering the simplicity and the extreme fast learning speed of the ELM, we extend the ELM model to Constrained ELM (CELM) by constraining the weight vector parameters $\{\mathbf{a}_j\,|\,j=1,\ldots,L\}$ of ELM to be randomly drawn from the closed set of difference vectors of between-class samples instead of from the open set of arbitrary vectors to tackle the problem of generation of discriminative hidden nodes. We use a simple case to illustrate the idea of difference vectors of between-class samples.



(a)



(b)



(c)

Fig. 1. Illustration of difference vectors of between-class samples. How to generate the difference vectors is illustrated in (a). The completely random connection weight vectors from the input layer to the hidden layer of ELM are illustrated in (b). The constrained random weight vectors of CELM are illustrated in (c).

The essence of the weight vectors from the input layer to the neurons in the hidden layer is to map the original samples into a discriminative feature space spanned by these vectors, where the samples can be classified. The weight vectors are helpful for classification if the directions of the weight vectors are from class 1 to class 2 or reversely, as illustrated in Fig. 1(a). The blocks in the figure represent the samples in class 1, and the circles represent the samples in class 2. As a comparison, the weight vectors from the input layer to the hidden nodes in ELM are completely random without constraints, as illustrated in Fig. 1(b). It can be inferred the weight vectors which do not follow the direction from class 1 to class 2 are less discriminative for classification tasks. This is the reason why not all the hidden nodes in ELM are efficient or discriminative.

We randomly generate the weight vectors from the input layer to the hidden layer with the differences of between-class samples as illustrated in Fig. 1(a). The difference vectors of between-class samples can map the samples to a higher discriminative feature space than ELM. The weight vectors from the input layer to the hidden layer in CELM are illustrated in Fig. 1(c). The directions of these weight vectors are close to the direction from class 1 to class 2, which are more discriminative for the classification tasks intuitively.

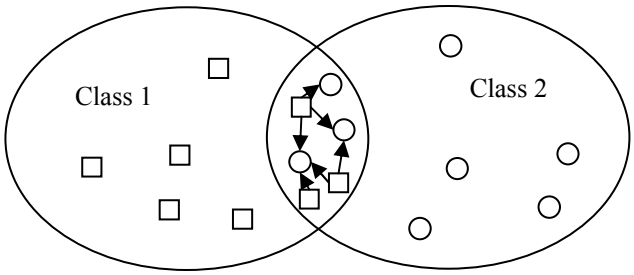


Fig. 2. Illustration of noise difference vectors of between-class samples

Two pre-processing operations are utilized to delete noise and too relevant difference vectors for better generalization of CELM. First, delete the small difference vectors. When the between-class samples are located in the overlapped area of two classes as illustrated in Fig. 2, the deference vectors are small and may contain non-discriminative or noise information. So deleting the small deference vectors is helpful for classification tasks. Second, delete the nearly parallel difference vectors. Nearly parallel vectors are so relevant that they will make the data repeatedly projected to near points in the feature space, which may risk error accumulating [22]. If some vectors are nearly parallel, only one vector is retained.

In addition to these two operations, we normalize the rest difference vectors as the weights from the input layer to the hidden layer. The reason why normalize these weights and how to normalize will be introduced in the following discussion.

In CELM, the prior information of samples' class distribution is utilized to generate the weights from the input layer to the hidden layer. The aim is to split different classes' samples into different areas in the feature space. The ideal case is that, for example, class 1 is mapped into negative semi axis and class 2 is mapped into positive semi axis in the feature space. Hence the bias must be set as the middle point of the two selected samples from the geometric sense intuitively. As a result, the biases to the hidden neurons can be determined by assuming that the samples from one class are mapped to -1 and the samples from another class are mapped 1 respectively. Denote $\mathbf{x}_{c1}$ and $\mathbf{x}_{c2}$ as the samples drawn from two classes. Then the weight vector $\mathbf{w}$ from the input layer to one hidden neuron can be generated with $\alpha(\mathbf{x}_{c2}-\mathbf{x}_{c1})$, where α is the normalized factor. The original data $\mathbf{x}$ is transformed to $\mathbf{x}^T\mathbf{w}+b=\alpha\mathbf{x}^T(\mathbf{x}_{c2}-\mathbf{x}_{c1})+b$ by feature mapping, where b is the bias with respect to the weight vector $\mathbf{w}$ in ELM model. The assumption that $\mathbf{x}_{c1}$ and $\mathbf{x}_{c2}$ are mapped to -1 and 1 can be written as

$$\alpha\mathbf{x}_{c1}^T(\mathbf{x}_{c2}-\mathbf{x}_{c1})+b=-1, \text{ and}$$
$$\alpha\mathbf{x}_{c2}^T(\mathbf{x}_{c2}-\mathbf{x}_{c1})+b=1.$$

We can obtain that the normalization factor $\alpha=\dfrac{2}{\left\|\mathbf{x}_{c2}-\mathbf{x}_{c1}\right\|_{L_2}^2}$

and the corresponding bias $b=\frac{(\mathbf{x}_{c1}+\mathbf{x}_{c2})^T(\mathbf{x}_{c1}-\mathbf{x}_{c2})}{\|\mathbf{x}_{c2}-\mathbf{x}_{c1}\|_{L_2}^2}$ by solving the above two equation constraints.

The commonly used activation function for hidden neurons is sigmoid function $f(x)=\frac{1}{1+e^{-x}}$. The output layer in CELM is a simple linear system as same as that of ELM.

Algorithm 1: Training of the Constrained Extreme Learning Machine (CELM)

Input: the training samples $\aleph=\{(\mathbf{x}_i,\mathbf{t}_i)|\mathbf{x}_i\in\mathbf{R}^n,\mathbf{t}_i\in\mathbf{R}^m, i=1,\ldots,N\}$, the hidden node number L and the activation function $G(\mathbf{w},b,\mathbf{x})$.

Output: the model parameters of CELM, i.e., the weight matrix $\mathbf{W}_{n\times L}$ and the bias vector $\mathbf{b}_{1\times L}$ from the input layer to the hidden layer, the weight matrix $\beta_{L\times m}$ from the hidden layer to the output layer.

1) While the number of chose difference vectors is less than L

a) Randomly draw training samples $\mathbf{x}_{c1}$ and $\mathbf{x}_{c2}$ from any two different classes respectively and generate the difference vector $\mathbf{x}_{c2}-\mathbf{x}_{c1}$;

b) If the norm of vector is small enough, delete it and go to a);

c) If the vector is nearly parallel with the previous generated vectors, delete it and go to a);

d) Normalize the difference vector by $\mathbf{w}=\frac{2(\mathbf{x}_{c2}-\mathbf{x}_{c1})}{\|\mathbf{x}_{c2}-\mathbf{x}_{c1}\|_{L_2}^2}$, and calculate the corresponding bias $b=\frac{(\mathbf{x}_{c1}+\mathbf{x}_{c2})^T(\mathbf{x}_{c1}-\mathbf{x}_{c2})}{\|\mathbf{x}_{c2}-\mathbf{x}_{c1}\|_{L_2}^2}$.

e) Use the vector $\mathbf{w}$ and bias b to construct the weight matrix $\mathbf{W}_{n\times L}$ and bias vector $\mathbf{b}_{1\times L}$.

2) Calculate the hidden layer output matrix $\mathbf{H}$ as

$$\mathbf{H}=\begin{bmatrix}\mathbf{h}(\mathbf{x}_1)\\ \vdots\\ \mathbf{h}(\mathbf{x}_N)\end{bmatrix}=\begin{bmatrix}G(\mathbf{a}_1,b_1,\mathbf{x}_1) & \cdots & G(\mathbf{a}_L,b_L,\mathbf{x}_1)\\ \vdots & \cdots & \vdots\\ G(\mathbf{a}_1,b_1,\mathbf{x}_N) & \cdots & G(\mathbf{a}_L,b_L,\mathbf{x}_N)\end{bmatrix}_{N\times L}.$$

3) Calculate the hidden layer's output weight matrix $\beta_{L\times m}$ by solving the least squares problem:

$$\beta=\mathbf{H}^{\dagger}\mathbf{T}$$

, where $\mathbf{H}^{\dagger}$ is the generalized inverse matrix and $\mathbf{T}=\begin{bmatrix}\mathbf{t}_1^T\\ \vdots\\ \mathbf{t}_N^T\end{bmatrix}$.

From the above discussion, the training algorithm for CELM can be concluded in the Algorithm 1. The essence of CELM is to constrain the hidden neuron's input connection weights to be consistent with the directions from one class to another class. So the random weights are constrained to be chosen from the set that is composed of the difference vectors of between-class samples.

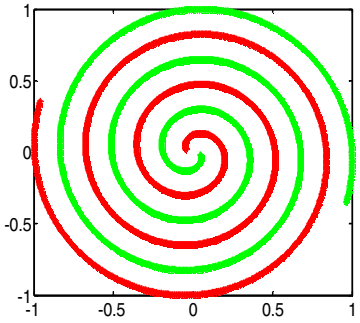
III. Performance Evaluation

In this section, we evaluate the proposed CELM and compare it with some classifiers, such as ELM, SVM and some related deep learning methods, on both synthetic and real-world datasets. Ten round experiments are conducted for each data set. In each experiment, the training set and the test set are randomly generated using the samples from synthetic datasets and UCI database [18]. The samples from UCI database are normalized to be of zero means and unit variances. The performances are recorded with the means and the standard deviations of classification accuracies.

In these experiments, we also compare the CELM with the normalized ELM, which deletes small, nearly parallel weight vectors and normalize them in ELM. The only difference between CELM and the normalized ELM is that, CELM utilizes the difference vectors to construct the hidden nodes. The aim of this comparison is to verify the effect of the difference vectors sufficiently. The code of ELM used in the experiments was downloaded from [4].

A. Experiments on Synthetic Dataset

We first evaluate our CELM algorithm on the synthetic dataset of the spiral data. It is illustrated in Fig. 3. To retain the symmetrical shape of the spiral, we normalize the samples into the range [-1, 1] as same as that in [4]. The total generated number of such spiral data is 5000. The training set contains 4000 samples and the test set contains 1000 samples. The two sets are randomly generated in each one of the total ten round experiments.



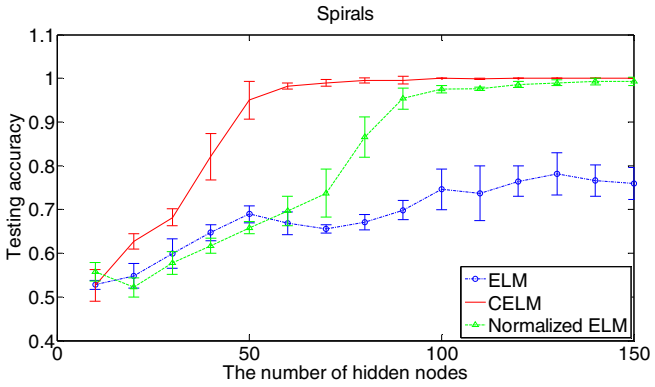
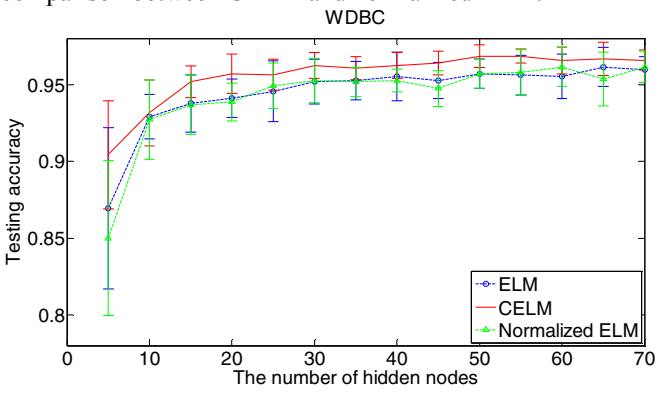
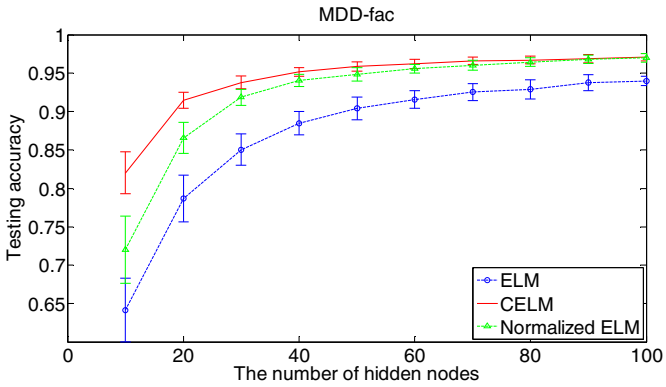


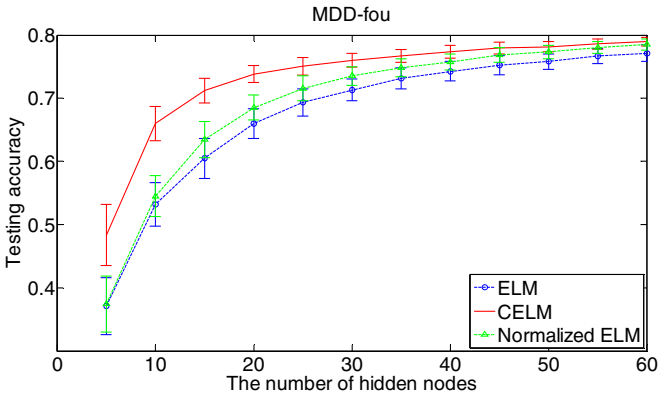
Fig. 3. The spiral synthetic dataset is illustrated in the top side. The performance comparisons are shown in the down side.

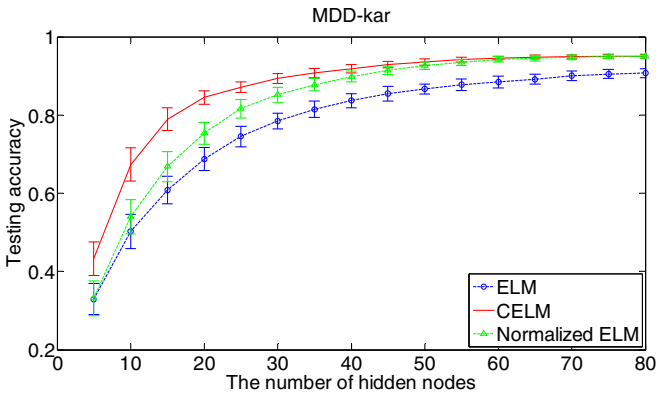
We compare the performances of CELM with ELM and normalized ELM. The number of hidden nodes is selected from 10 to 300 at a step 10. The performances of these models are illustrated in Fig. 3. The solid line represents the performance of CELM. The dotted line with circles represents that of ELM and the dashed line with triangles represents the normalized ELM.

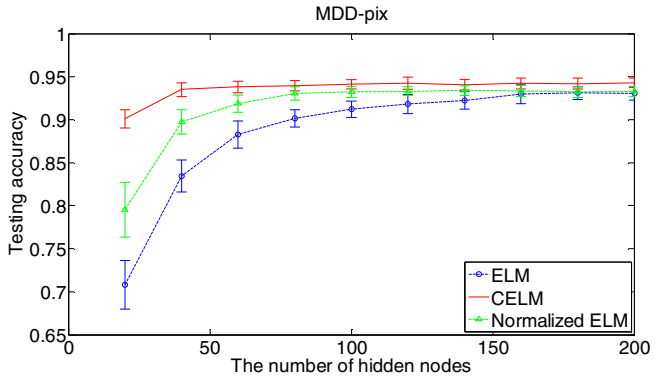
As shown in Fig. 3, CELM has a perfect performance when the number of hidden nodes reaches 50, while normalized ELM needs 100 hidden nodes to reach the same perfect performance. The test accuracy of ELM is less than 0.8 even when the number of hidden nodes is 300. The test accuracies of CELM are above those of normalized ELM and ELM all the time. The result shows that CELM has the better generalization ability than normalized ELM. Therefore, the difference vectors really work from the performance comparison between CELM and normalized ELM.











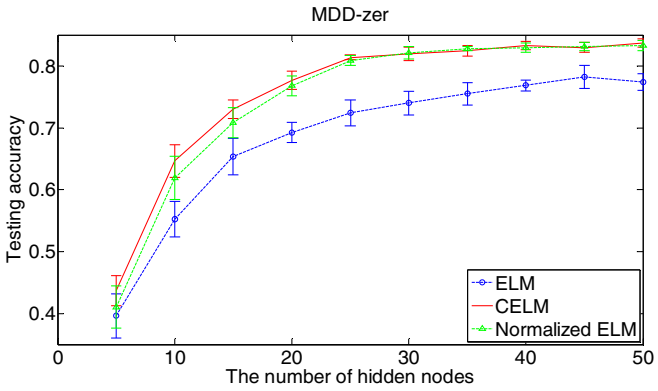


Fig. 4. Experimental results of ELM, CELM and normalized ELM on the six UCI datasets.

B. Experiments on UCI Datasets

Six datasets from UCI database [18], including Wisconsin diagnostic breast cancer dataset (WDBC) and five multi feature digit datasets (MDD-fac, fou, kar, pix, zer), are used for evaluating the proposed CELM. CELM is compared with ELM and normalized ELM. In MDD dataset, five different feature sets are used.

In the first experiment, the number of training samples is 4/5 size of the total samples, and the rest is used as test samples. These samples are randomly divided into training and test sets. The comparison of ELM, CELM and normalized ELM in the first experiment are illustrated in Fig. 4. The performances of these models are displayed sufficiently from the trends of these curves in the figure. It can be seen that the test accuracy curves of CELM are above those of other two methods in all the experiments and so the generalization ability of the proposed CELM is better than other two models on these real world datasets. The samples' distribution prior we introduced makes the efficient use of hidden nodes in CELM and really helps the classification tasks.

In the second experiment, a benchmark performance evaluation is conducted on the six datasets. The size of training set is half of the total number of samples just as other methods usually do on these datasets. The training set and the test set are randomly generated in each of ten rounds of experiments. The mean of test accuracies are recorded in Table I. It can be seen the performances of CELM always outperform those of ELM. The CELM improves the ELM's performance greatly.

C. Experiments on Large Scale Datasets

We also evaluate the CELM and ELM on two large size datasets, i.e., MNIST [19] and CIFAR-10 [16]. The MNIST database of handwritten digits contains a training set of 60,000 samples, and a test set of 10,000 samples. It consists of binary images of ten classes and the size of these digits is 28 $\times$28 pixels. The samples are normalized and input into the CELM and ELM to compare their performances. The CIFAR-10 dataset contains 60,000 color images in 10 classes, with 6000 image per class. The training set and the test set consist of 50,000 images and 10,000 images respectively. In CIFAR-10 dataset, the standard pipeline defined in [20] is used. First, extract dense 6×6 local patches with ZCA whitening and the stride is 1. Second, use threshold coding with α=0.25 to encode. The codebook is trained with OMP-1 [23] and the codebook size is 50 in the experiment. Third, average-pool the features on a 2×2 grid to form the global image representation.

Due to large number of samples, we only use 1/5 instances in CIFAR-10 dataset and 1/10 instances in MNIST dataset for experiments. These instances are uniformly sampled from the two datasets respectively. The performances of CELM, ELM and normalized ELM are illustrated in Fig. 5.

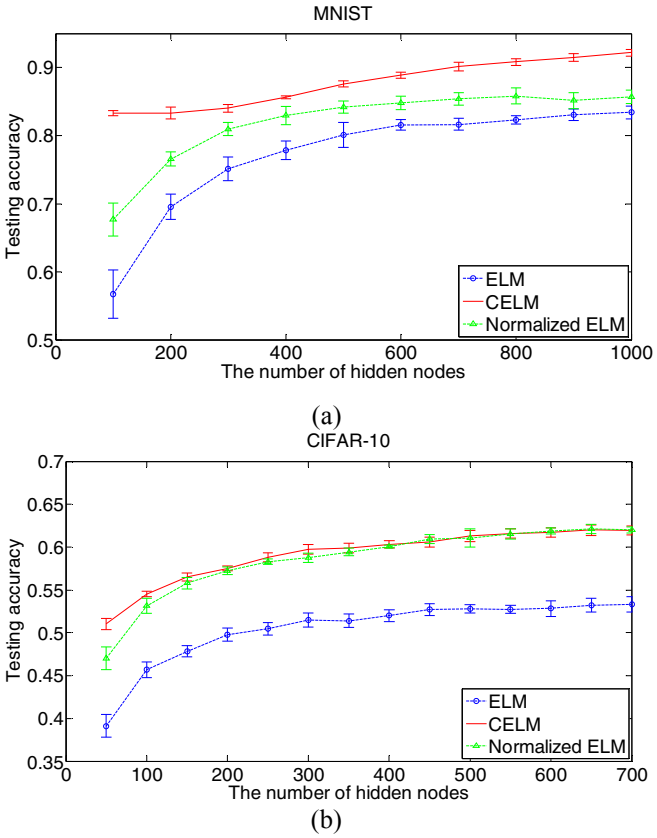


Fig. 5. Experimental results of ELM, CELM and normalized ELM on the subsets of MNIST and CIFAR-10 datasets.

From Fig. 5, it can be learned the performances of CELM are 10 percentages higher than those of ELM averagely on the two datasets. The efficiency of the difference vectors can be evaluated as follows. When the same numbers of vectors are used, the method with higher test accuracy means more efficient. In Fig. 5, the performance of CELM is higher than that of normalized ELM when they are not converged. The gap between the performance of CELM and normalized ELM shows that the difference vector is effective. The curve of CELM's performance is always above those of normalized ELM and ELM in all these datasets, which shows that difference vectors really help the efficient use of hidden nodes.

The CELM is also compared with SVM related methods, e.g., Linear SVM and R^2SVM [14], and deep ELM related method, e.g., DCN [15], on CIFAR-10 dataset. The R^2SVM is

a deep learning model and its building block is a linear SVM model. The outputs of the previous layer are transformed by a random matrix, and then the transformed outputs are added to the original features. The modified features are input into the next layer after transformed by a sigmoid function. The DCN is also a deep learning model, but its building block is an ELM based model, in which parts of the hidden nodes are built with random projection and the other part of hidden nodes are built with RBM weights [15, 24]. Instead of the way that adds the output of previous layer as a bias to the next layer, the outputs of the previous layer are concatenated with the original features sequentially and are input into the next layer in DCN model.

In this experiment, all the 50,000 training samples are used to train the model, and all the 10,000 test samples are used to evaluate the performance. Table Ⅱ shows the performances of CELM, linear-SVM, R^2SVM [14] and DCN [15] methods. Note the performances of linear SVM, R^2SVM and DCN are cited from [14]. And the experimental conditions of CELM, such as the feature exaction and the number of used training and test sets, are the same with [14].

TABLE I
AVERAGE CLASSIFICATION ACCURACIES OF CELM AND ELM

Datasets	CELM	ELM
WDBC	0.972	0.971
MDD-fac	0.982	0.964
MDD-fou	0.837	0.797
MDD-kar	0.972	0.900
MDD-pix	0.977	0.858
MDD-zer	0.844	0.805

TABLE Ⅱ
PERFORMANCE ON CIFAR-10 DATASET

Algorithms	No. of hidden layers /nodes	Test accuracy
Linear SVM	-	0.647
R^2SVM	35 Layers	0.693
DCN	-	0.672
CELM	7000 nodes	0.723

From the Table Ⅱ, the CELM can be found to have the best performance than that of linear-SVM, DCN and R^2SVM. The CELM has the test accuracy of 8 percentages higher than that of linear SVM. In [14], the R^2SVM has 35 layers, and each layer is a linear SVM after sigmoid transformation and random projection. Although R^2SVM and DCN have many layers, the CELM of one layer has the test accuracy of 3 percentages higher than that of these discriminative deep learning methods. Besides, the CELM can train a model on a case of 50,000 training data well, which suggests that the proposed CELM can tackle large scale data effectively.

IV. CONCLUSION AND DISCUSSION

To address the inefficient use of hidden nodes in ELM, this paper proposed a novel learning model, CELM. The CELM constrains its random weights' generation from a smaller space than that of the ELM, i.e., replacing the completely random weight vectors with ones that are randomly drawn from the set of difference vectors of between-classes samples. The main contribution of CELM is that it introduces sample distribution prior into the construction of the hidden layer to make a better discriminative feature mapping. The effective feature mapping greatly contributes the efficient use of hidden nodes in ELM. Extensive comparisons between CELM and some related methods on both synthetic and real-world datasets showed that CELM always has a better performance.

However, the CELM still has some problems that ELM owned. One is that CELM faces over fitting problem when the number of hidden nodes is very large, although CELM improves the efficient use of discriminative hidden nodes. To our relief, the method in [21] can tackle the problem effectively. Incorporating the idea of this method into CELM is expected to improve the performance and robustness of CELM. Another problem is that the generation speed of difference vectors will be much slower when we want to obtain more than 10,000 hidden nodes. This is because the anti-correlation operation involving vector direction comparisons is time consuming. At last, the solving of the weights from the hidden layer to the output layer is time-consuming when the number of hidden nodes is very large. This case is much common in large scale applications. Some gradient based solving methods for linear system can tackle the problem iteratively.

The further research will include the study of invariant feature generating and other measures for the improvement of CELM and the experimental verification on CELM's application to regression problems. The analyses on what kinds of problems that the CELM will work with and such related theories are also expected to be studied in the future.

REFERENCES

[1] M. Leshno, V. Ya. Lin, A. Pinkus and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," Neural Networks, vol. 6(6), pp. 861–867, 1993.

[2] S. Tamura and M. Tateishi, "Capabilities of a four-layered feedforward neural network: four layers versus three," IEEE Trans. Neural Networks, vol. 8(2), pp. 251–255, 1997.

[3] G.-B. Huang and H.A. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," IEEE Trans. Neural Networks, vol. 9(1), pp. 224–229, 1998.

[4] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: Theory and applications," Neurocomputing, vol. 70 (1 – 3), pp. 489 – 501, Dec. 2006, [Code: http://www.ntu.edu.sg/home/egbhuang/elm_random_hidden_nodes.html].

[5] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in Proceedings of International Joint Conference on Neural Networks (IJCNN2004), vol. 2, (Budapest, Hungary), pp. 985–990, 25-29 Jul., 2004.

[6] P.L. Bartlett, "The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network," IEEE Trans. Inf. Theory, vol. 44(2), pp. 525 – 536, 1998.

[7] Q.-Y. Zhu, A.K. Qin, P.N. Suganthan and G.-B. Huang, "Evolutionary extreme learning machine," Pattern Recognition, vol. 38(10), pp. 1759 – 1763, Oct. 2005.

[8] L. Yuan, Y. C. Soh, and G.-B. Huang, "A constructive enhancement for online sequential extreme learning machine," in Proceedings of International Joint Conference on Neural Networks (IJCNN2009), pp. 1708–1713, 14-19 Jun., 2009.
[9] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten and A. Lendasse, A., "OP-ELM: optimally pruned extreme learning machine," Neural Networks, IEEE Transactions on, vol. 21(1), pp. 158-162, 2010.
[10] H. J. Rong, Y. S. Ong, A. H. Tan and Z. Zhu, "A fast pruned-extreme learning machine for classification problem," Neurocomputing, vol. 72(1), pp. 359-366, 2008.
[11] D. Yu and L. Deng, "Efficient and effective algorithms for training single-hidden-layer neural networks," Pattern Recognition Letters, vol. 33(5), pp. 554 – 558, 1 Apr. 2012.
[12] K. P. Murphy, "Machine learning: a probabilistic perspective," The MIT Press, 2012.
[13] Y. Su, S. Shan, X. Chen and W. Gao, "Classifiability-based discriminatory projection pursuit," IEEE Trans. Neural Networks, vol. 22(12), pp. 2050-2061, 2011.
[14] O. Vinyals, Y. Jia, L. Deng and T. Darrell, "Learning with recursive perceptual representations," In Advances in Neural Information Processing Systems, pp. 2834-2842, 2012.
[15] L. Deng and D. Yu, "Deep convex net: A scalable architecture for speech pattern classification," In Proceedings of the Interspeech 2011.
[16] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009, [http://www.cs.toronto.edu/~kriz/ cifar.html].
[17] G.-B. Huang, X. Ding and H. Zhou, "Optimization method based extreme learning machine for classification," Neurocomputing, vol. 74(1), pp. 155-163, 2010.
[18] C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases", 1998, [http://www.ics.uci.edu/~mlearn/MLRepository. html]. Irvine, CA: University of California. Department of Information and Computer Science, 460.
[19] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86(11), pp. 2278-2324, 1998, [Online]. Available: http://yann.lecun.com/exdb/mnist.
[20] A. Coates and A. Ng, "The importance of encoding versus training with sparse coding and vector quantization," In Proceedings of the 28th International Conference on Machine Learning (ICML-11) (pp. 921-928), 2011.
[21] W. Zhu, J. Miao and L. Qing, "Extreme support vector regression," in Proceedings of International Conference on Extreme Learning Machines (ELM2013), Beijing, China, Springer-Verlag, 15-17 Oct. 2013.
[22] C. M. Bishop and N. M. Nasrabadi, N. M., "Pattern recognition and machine learning", New York: springer, 2006.
[23] T. Blumensath and M. E. Davies, "On the difference between orthogonal matching pursuit and orthogonal least squares," unpublished manuscript, 2007, [http://www.see.ed.ac.uk/~tblumens/ papers/BD OMPvsOLS07.pdf].
[24] G. E. Hinton, S. Osindero and Y. W. Teh, "A fast learning algorithm for deep belief nets," Neural computation, vol. 18(7), pp. 1527-1554, 2006.