Engineering Applications of Artificial Intelligence 42 (2015) 57–66

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

A novel algorithm of extended neural networks for image recognition ☆

Kankan Dai, Jianwei Zhao, Feilong Cao *

Department of Applied Mathematics, College of Sciences, China Jiliang University, Hangzhou 310018, Zhejiang Province, PR China

ARTICLE INFO

Article history:
Received 31 October 2014
Received in revised form
17 March 2015
Accepted 17 March 2015
Available online 10 April 2015

Keywords:
Feedforward neural networks
Matrix data
BP algorithm
Image recognition

ABSTRACT

As a class of important classifiers, feedforward neural networks (FNNs) have been used considerably in the study of pattern recognition. Since the inputs to FNNs are usually vectors, and many data are usually presented in the form of matrices, the matrices have to be decomposed into vectors before FNNs are employed. A drawback to this approach is that important information regarding correlations of elements within the original matrices are lost. Unlike traditional vector input based FNNs, a new algorithm of extended FNN with matrix inputs, called two-dimensional back-propagation (2D-BP), is proposed in this paper to classify matrix data directly, which utilizes the technique of incremental gradient descent to fully train the extended FNNs. These kinds of FNNs help to maintain the matrix structure of the 2D input features, which helps with image recognition. Promising experimental results of handwritten digits and face-image classification are provided to demonstrate the effectiveness of the proposed method.

1. Introduction

It is well-known that image recognition is a hot topic in the fields of machine learning and computer vision. In such recognition systems, many data images, such as handwritten digital images, face images, and palm images, are usually presented in the form of matrices. Clearly, determining how to classify these kinds of data is an important topic in pattern recognition.

Traditional image recognition system contains three steps: image pre-processing, feature extraction, and classification. At present, there are various methods for feature extraction, such as principal component analysis (PCA) (Jolliffe, 2002), independent component analysis (ICA) (Hyvärinen and Oja, 2000), and several popular applied classifiers, including K-nearest neighbourhoods (KNNs) (Shakhnarovich et al., 2008), support vector machines (SVMs) (Vapnik, 2000), feedforward neural networks (FNNs) (Hornik et al., 1989), and so on.

Nevertheless, these methods are usually based on vector inputs. Thus, when they are used in image processing, we first have to expand the matrix inputs into vector form. These types of transformation often lead to the loss of important information regarding the original matrix data, and thus affect the recognition process. On the other hand, expanding matrix inputs into vectors usually causes high dimensionality and increases the complexity of the used models.

To solve this problem, two-dimensional (2D) methods that directly operate on matrix data are proposed, for example, the two-dimensional principal component analysis (2DPCA) (Yang et al., 2004; Zhang and Zhou, 2005) and two-dimensional linear discriminant analysis (2DLDA) (Li and Yuan, 2005; Sanguansat et al., 2006; Yang et al., 2010), which were verified useful for extracting effective information about the inner structure of matrix data, as well as reducing the computational complexity of the extraction. It is natural to raise the question: For the existing vector-based classifiers, such as the SVM and FNN methods, can they be extended for matrix input?

Neural networks have played an important role in pattern recognition (Bishop, 1995; Lin et al., 1997; LeCun et al., 1998; Shang et al., 2006). In order to classify matrix data directly, and to preserve the matrix or 2D feature structure effectively, Lu et al. (2014) proposed a novel classifier, the two-dimensional neural network with random weights (2D-NNRW) method, and achieved good performance on face recognition. In fact, it is an extended 2D single hidden layer feed forward neural network (2D-SLFN) model that employs left and right projecting vectors to regress matrix inputs. Additionally, it uses the random idea to train the network, i.e. it randomly sets the left and right projecting vectors and the hidden biases, and then determines the output weights by solving a linear equation system. The results obtained in Lu et al. (2014) show that the use of 2D-SLFN improves face recognition accuracy.

The randomness in the NNRW algorithm can be understood in deeper detail when we consider the function approximation with Monte Carlo (MC) methods. It was shown in Igelnik and Pao (1995) that any continuous function defined on a compact set can be

☆Supported by the National Natural Science Foundation of China (Nos. 61272023 and 91330118) and Zhejiang Provincial Natural Science Foundation of China (No. LY14A010027).

* Corresponding author.
E-mail address: feilongcao@gmail.com (F. Cao).

http://dx.doi.org/10.1016/j.engappai.2015.03.010
0952-1976/

represented by a limit-integral of a multivariate continuous function that is integrated in parameter space. Although using the NNRW algorithm can simplify the learning steps taken by SLFNs, the following issues still remain in both the NNRW and 2D-NNRW methods:

- The number of hidden nodes should be sufficiently large and supervised initialization is needed in order to model and compensate for the system's uncertainties.
- There exists an over-fitting phenomenon caused by many additional hidden nodes in the NNRW method due to the MC approximating approach.
- There is a predictive instability caused by randomly assignment of nonlinear weights, and the way of learning SLFNs with NNRW methods using singular value decomposition (SVD) usually produces large magnitude linear weights, which makes the networks highly sensitive to new data.

To overcome these issues, we have attempted to fully train the network with the aim of implementing incremental gradient based learning for 2D-SLFN. A learning algorithm called two-dimensional back-propagation (2D-BP) is proposed, where a momentum modification is added to improve convergence. A series of comparative studies of handwritten digits and face-image classification were carried out. The results of the testing datasets are promising, and support a positive statement regarding their performances among the 1D-BP and 2D-NNRW methods.

The rest of this paper is organized as follows. FNNs models and their corresponding training algorithms are reviewed in Section 2. A detailed description of our method is given in Section 3. An evaluation of the performance of our algorithm, the handwritten digits and face datasets employed in the experiments, and our results, which include comparisons and discussions, are presented in Section 4. Conclusions are presented in the final section.

2. FNN models and learning

2.1. Single hidden-layer feedforward neural networks

Generally, a SLFN is described as follows:

$$f(x)=\sum_{k=1}^{L}\beta_k g(w_k^{\top}x+b_k)+\alpha, \tag{1}$$

where x is the input pattern vector, $w_k=[w_{k1},w_{k2},\ldots,w_{kd}]^{\top}$, b_k are input layer weights and biases, respectively, and $\beta_k=[\beta_{k1},\beta_{k2},\ldots,\beta_{ko}]^{\top}$ and α are the output layer weights and biases, respectively.

It is true that SLFNs are universal approximators (Hornik et al., 1989; Cybenko, 1989; Barron, 1993), even when the hidden-layer weights and bias are randomly assigned. In the case that weights and bias are randomly assigned, i.e., the input layer weights w_k and biases b_k as defined on a probabilistic space $S_p(\Omega,P)$, where (Ω,P) should be determined in the learning stage, then we just need to tune the linear weights (β,α), and call SLFN as a neural network with random weights (NNRW) (Schmidt et al., 1992). It has been proved that the NNRW's approximation error converges to zero with the order $(C/\sqrt{L})$ (Igelnik and Pao, 1995; Pao et al., 1994), where L is the number of hidden nodes and C is a constant.

Notations

o	the dimensionality of the output
d	the dimensionality of the vector input
m	the first dimensionality of the matrix input
n	the second dimensionality of the matrix input
$\mathbf{u}_k$	the left projection vector on the kth hidden node
$\mathbf{v}_k$	the right projection vector on the kth hidden node
$g(\cdot)$	the active function
L	the number of hidden nodes
$\mathbf{t}^p$	the expected output of the pth pattern
$\mathbf{y}^p$	the actual output on the pth pattern

We have attempted to approximate the function $y=e^x-x\sin(x)\cos(x)$ using SLFNs (i.e. randomly generate 50 points on the interval $(-7,2)$). Both the NNRW algorithm and the general gradient based full network training (BFGS) can achieve sufficient precision (see Fig. 1). However, they are different approaches that result in different problems.

When using the NNRW method, which is motivated by Monte Carlo integration, it only offers a statistical measure of the approximation quality. Its approximation rate is achievable only when the number of hidden nodes in the network is sufficient large (Tyukin and Prokhorov, 2009). After the input layer weights and biases are randomly assigned, we always use the least square method to tune the SLFN, which results in large-magnitude linear

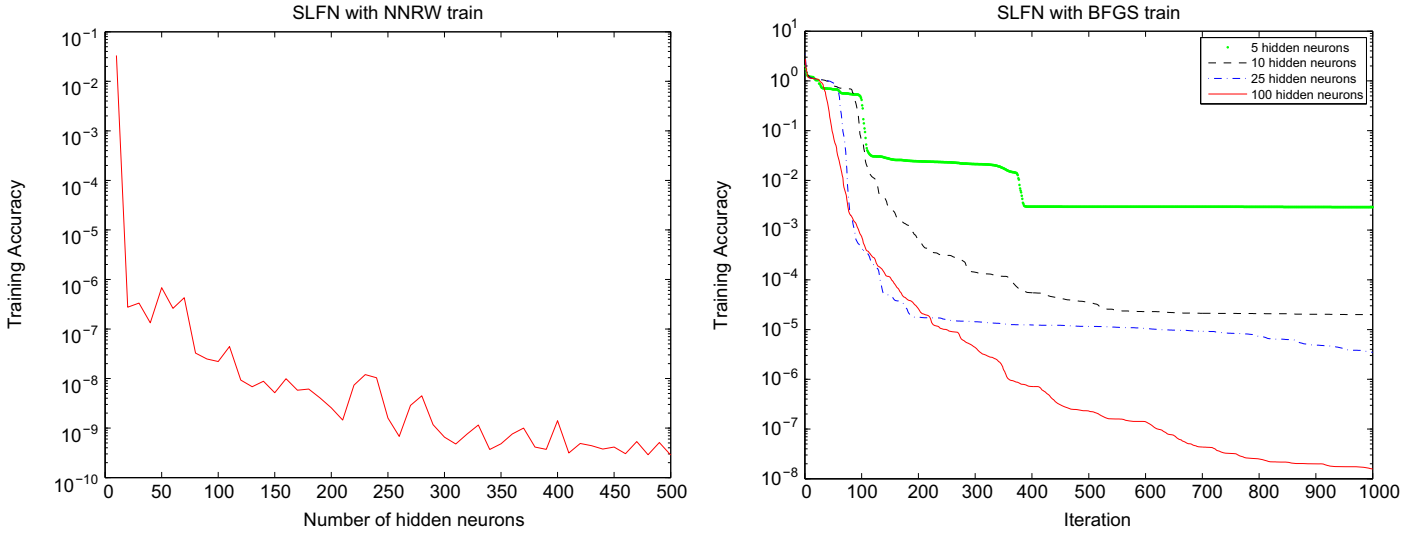

Fig. 1. Neural network training.

weights solutions. Thus the NNRW algorithm suffers from numerical instabilities, and is highly sensitive to new data. Although employing regularization (McLoone and Irwin, 2001) can penalize large magnitude weights, it usually results in under-fitting.

With full network training, sufficient approximation accuracy can be achieved by using fewer hidden nodes, which leads to a more compact network. Indeed, this method is the generally accepted way to train SLFNs, and many optimization methods can be used. These methods include gradient descent (1D-BP), as described in nodesAlgorithm 1 (with momentum modifications), batch gradient descent (BBP), conjugate gradient (CG) (Fletcher and Reeves, 1964), Gauss–Newton, Levenberg–Marquardt (McLoone and Irwin, 1997), Broyden–Fletcher–Golfarb–Shanno (BFGS), memory-less Broyden–Fletcher–Golfarb–Shanno (LBFGS) (Battiti and Masulli, 1990), and so on. Among them, some are batch training methods (off-line) which have faster convergence than incremental training methods (on-line). However, batch training methods, including NNRW algorithms, are subjected to memory restriction. Therefore in this paper, we adopted the latter method. However, batch training

Algorithm 1. 1D-BP.

Require: Training dataset (N instances with vector inputs), learning rate η, number of hidden nodes L, number of iterations s and the momentum γ.

Ensure: Trained 1D-SLFN.

1: Initialize the weights of 1D-SLFN

$$f(x)=\sum_{k=1}^{L}\beta_k g(w_k^{\top}x+b_k)+\alpha.$$

2: **for** $i\leftarrow 1$ to s **do**
3: **for** $p\leftarrow 1$ to N **do**
4: $\Delta\beta(i)\leftarrow\gamma\Delta\beta(i-1)-\eta(1-\gamma)\nabla\beta$
5: $\Delta\mathbf{w}(i)\leftarrow\gamma\Delta\mathbf{w}(i-1)-\eta(1-\gamma)\nabla\mathbf{w}$
6: $\Delta\mathbf{b}(i)\leftarrow\gamma\Delta\mathbf{b}(i-1)-\eta(1-\gamma)\nabla\mathbf{b}$
7: $\Delta\alpha(i)\leftarrow\gamma\Delta\alpha(i-1)-\eta(1-\gamma)\nabla\alpha$
8: $\beta\leftarrow\beta+\Delta\beta(i)$
9: $\mathbf{w}\leftarrow\mathbf{w}+\Delta\mathbf{w}(i)$
10: $\mathbf{b}\leftarrow\mathbf{b}+\Delta\mathbf{b}(i)$
11: $\alpha\leftarrow\alpha+\Delta\alpha(i)$
12: **end for**
13: **end for**
14: **return** 1D-SLFN.

2.2. Extended 2D single-hidden layer feedforward neural networks

Although general SLFNs are universal approximators, there is nothing particular impressive or exciting about that. Because universality means the network can be fit to any training data, it does not mean it can be generalized to make predictions of new data in a reasonable (natural) way.

When we use general 1D-SLFNs to classify matrix data, such as image classification, the procedure of decomposing the matrix data into vectors is no doubt unreasonable because this kind of processing unavoidably destroys structural correlations between elements that may subsequently influence recognition performance. This is why we have been trying to find more suitable neural network models for image recognition. Recently, Lu et al. (2014) designed a new network model, the extended 2D-SLFN, which is described as follows:

$$f(X)=\sum_{k=1}^{L}\beta_k g(\mathbf{u}_k^{\top}X\mathbf{v}_k+b_k)+\alpha, \tag{2}$$

where $\mathbf{u}_k=[u_{k1},u_{k2},\ldots,u_{km}]^{\top}$ and $\mathbf{v}_k=[v_{1k},v_{2k},\ldots,v_{nk}]^{\top}$ are the left and right projection vectors, respectively, X is the input pattern matrix, b_k is the input layer biases, $\beta_k=[\beta_{1k},\beta_{2k},\ldots,\beta_{ok}]^{\top}$ and $\alpha=[\alpha_1,\alpha_2,\ldots,\alpha_o]^{\top}$ are the output layer weights and biases, respectively. In Lu et al. (2014), the NNRW algorithm was extended to a 2D-NNRW algorithm to train the network, which can be described as Algorithm 2 (below), and it was shown that this model is an effective 2D classifier. This idea, to some extent, was inspired by the matrix input-based classifier referred to as multiple rank regression (MRR) (Hou et al., 2013).

Algorithm 2. 2D-NNRW.

Require: Training dataset (N instances with matrix inputs), number of hidden nodes L. (We generally select the sample space $\Omega=(-1,1)$, and set the probability distribution P to be a uniform distribution).

Ensure: Trained 2D-SLFN.

1: Initialize the left, right projection vectors and biases of input layer for the 2D-SLFN

$$f(X)=\sum_{k=1}^{L}\beta_k g(\mathbf{u}_k^{\top}X\mathbf{v}_k+b_k)+\alpha.$$

2: Compute the hidden matrix

$$H=\begin{bmatrix} g(\mathbf{u}_1^{\top}X^{(1)}\mathbf{v}_1+b_1) & \cdots & g(\mathbf{u}_L^{\top}X^{(1)}\mathbf{v}_L+b_L) & 1\\ \vdots & \ddots & \vdots & \vdots\\ g(\mathbf{u}_1^{\top}X^{(N)}\mathbf{v}_1+b_1) & \cdots & g(\mathbf{u}_L^{\top}X^{(N)}\mathbf{v}_L+b_L) & 1\end{bmatrix}_{N\times(L+1)}.$$

3: Calculate the linear weights using least square method

$\omega_{linear}=H^{\dagger}T$,

where $H^{\dagger}$ is the pseudo-inverse of matrix H,
$\omega_{linear}=[\beta_1,\beta_2,\ldots,\beta_L,\alpha]^{\top}$ and $T=[\mathbf{t}^{(1)},\mathbf{t}^{(2)},\ldots,\mathbf{t}^{(N)}]^{\top}$.

4: **return** 2D-SLFN.

3. 2D-BP for single hidden-layer feedforward neural networks

In order to maintain the matrix structure of the image inputs, 2D-SLFN is used in Lu et al. (2014), where the NNRW algorithm is extended to train this network, and the results suggest that 2D-SLFN is superior to general vector based SLFN (1D-SLFN) for image recognition. In this paper, we propose an incremental gradient based learning method for 2D-SLFN (2D-BP). Due to the left and right projecting vectors, the free weights of the 2D-SLFN are less than the general 1D-SLFN with the same number of hidden nodes. Surprisingly, this more compact model still maintains an overall good recognition accuracy rate. Next, we will present our algorithm in detail.

Given a 2D-SLFN and a training dataset of N instances, incremental learning is to minimize error in each training pattern, which is defined as follows:

$$E^{(p)}(\omega)=\frac{1}{2}(\mathbf{y}^{(p)}-\mathbf{t}^{(p)})^{\top}(\mathbf{y}^{(p)}-\mathbf{t}^{(p)})=\frac{1}{2}\sum_{j=1}^{o}(y_j-t_j)^2,\quad p=1\cdots N, \tag{3}$$

where $\omega=(\beta_1,\ldots,\beta_L,\mathbf{u}_1,\ldots,\mathbf{u}_L,\mathbf{v}_1,\ldots,\mathbf{v}_L,b_1,\ldots,b_L,\alpha)$ are all of the weights in 2D-SLFN, $\mathbf{y}^{(p)}=\sum_{k=1}^{L}\beta_k g(\mathbf{u}_k^{\top}X^{(p)}\mathbf{v}_k+b_k)+\alpha$ is the actual output of the pth pattern, and $\mathbf{t}^{(p)}$ is the expected output. Using gradient descent, we have the iterative formula:

$$\Delta\omega=-\eta\nabla\omega. \tag{4}$$

So we need to calculate the partial derivatives of each variable of ω. Using the chain rule, we have

$$\frac{\partial E^{(p)}}{\partial \beta_{jk}} = (y_j - t_j) g(\mathbf{u}_k^\top X^{(p)} \mathbf{v}_k + b_k), \tag{5}$$

$$\frac{\partial E^{(p)}}{\partial u_{kr}} = g'(\mathbf{u}_k^\top X^{(p)} \mathbf{v}_k + b_k) \sum_{j=1}^{o} \beta_{jk} (y_j - t_j) \sum_{c=1}^{n} x_{rc} v_{ck}, \tag{6}$$

$$\frac{\partial E^{(p)}}{\partial v_{ck}} = g'(\mathbf{u}_k^\top X^{(p)} \mathbf{v}_k + b_k) \sum_{j=1}^{o} \beta_{jk} (y_j - t_j) \sum_{r=1}^{m} u_{kr} x_{rc}, \tag{7}$$

$$\frac{\partial E^{(p)}}{\partial b_k} = g'(\mathbf{u}_k^\top X^{(p)} \mathbf{v}_k + b_k) \sum_{j=1}^{o} \beta_{jk} (y_j - t_j), \tag{8}$$

$$\frac{\partial E^{(p)}}{\partial \alpha_j} = (y_j - t_j). \tag{9}$$

We used the sigmoidal activation function. In order to facilitate the implementation in Matlab, we wrote our algorithm in matrix form. Additionally, we omitted the superscript $^{(p)}$ and simply wrote g_k instead of $g(\mathbf{u}_k^\top X^{(p)} \mathbf{v}_k + b_k)$. Then we have

$$\delta_2 = \mathbf{y} - \mathbf{t}, \tag{10}$$

$$\delta_1 = G \circ (E - G) \circ (\beta^\top \delta_2), \tag{11}$$

where "$\circ$" denotes element-wise multiplication, $G = [g_1, g_2, \ldots, g_L]^\top$, $E = [1, 1, \ldots, 1]^\top$ is all 1 vector with length L, and $\beta = [\beta_1, \beta_2, \ldots, \beta_L]$.

So we can rewrite (5) and (6) as

$$\nabla \beta = \delta_2 G^\top \tag{12}$$

and

$$\nabla \mathbf{u} = S_1 \circ (\mathbf{v}^\top X^\top), \tag{13}$$

respectively. Here $S_1 = [\delta_1, \delta_1, \ldots, \delta_1]$ with the size of $L \times m$, $\mathbf{u} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_L]^\top$ and $\mathbf{v} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_L]$. Next, Eqs. (7)–(9) can be rewritten as

$$\nabla \mathbf{v}^\top = S_2 \circ (\mathbf{u} X), \tag{14}$$

$$\nabla \mathbf{b} = \delta_1, \tag{15}$$

and

$$\nabla \alpha = \delta_2, \tag{16}$$

respectively, where $S_2 = [\delta_1, \delta_1, \ldots, \delta_1]$ with the size of $L \times n$ and $\mathbf{b} = [b_1, b_2, \ldots, b_L]^\top$.

When the *momentum* filter is added to the parameter change in each iteration, we obtained the following formula for

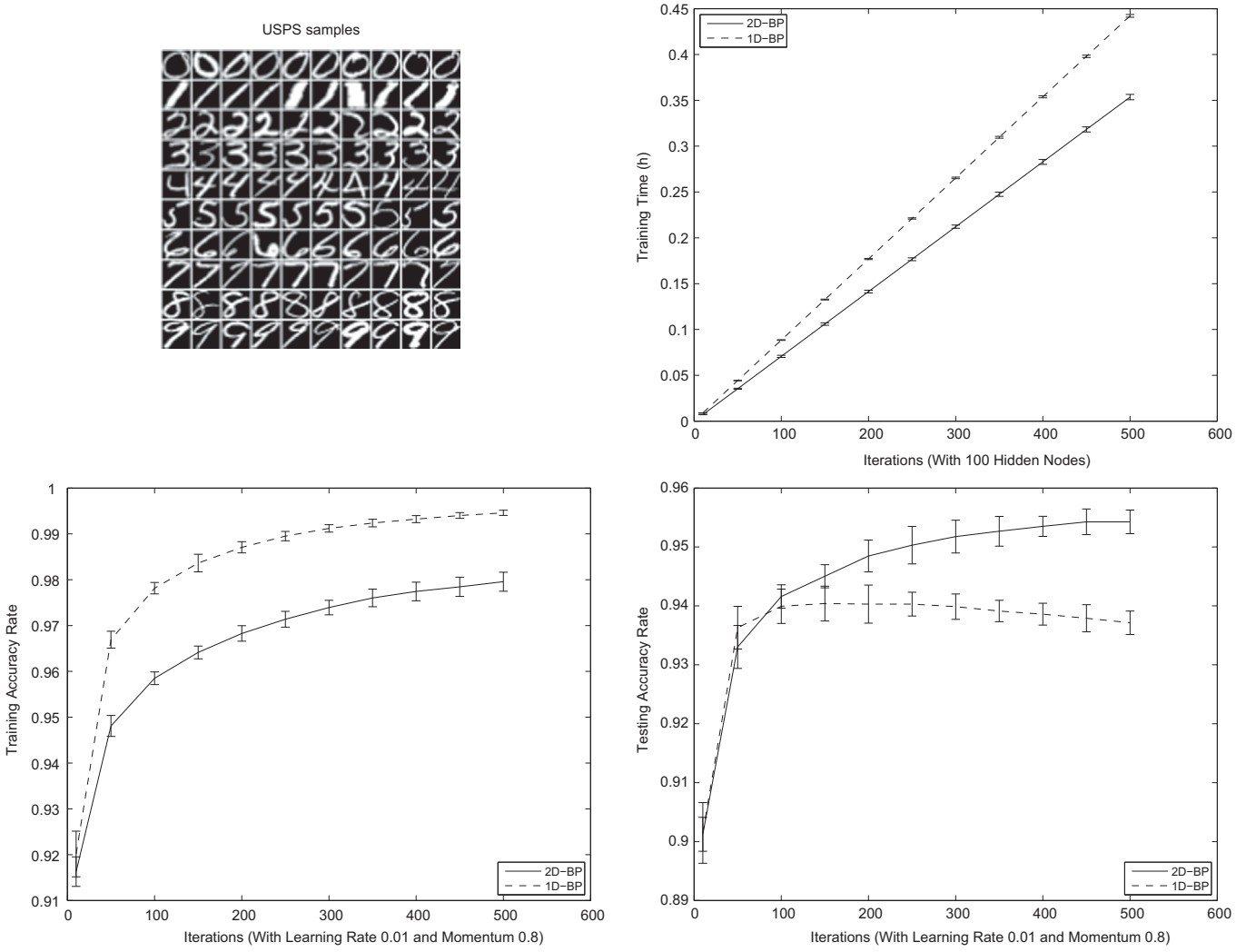

Fig. 2. Comparison of the 2D-BP and 1D-BP algorithms for different iterations on the USPS handwritten digits dataset (with 100 hidden nodes).

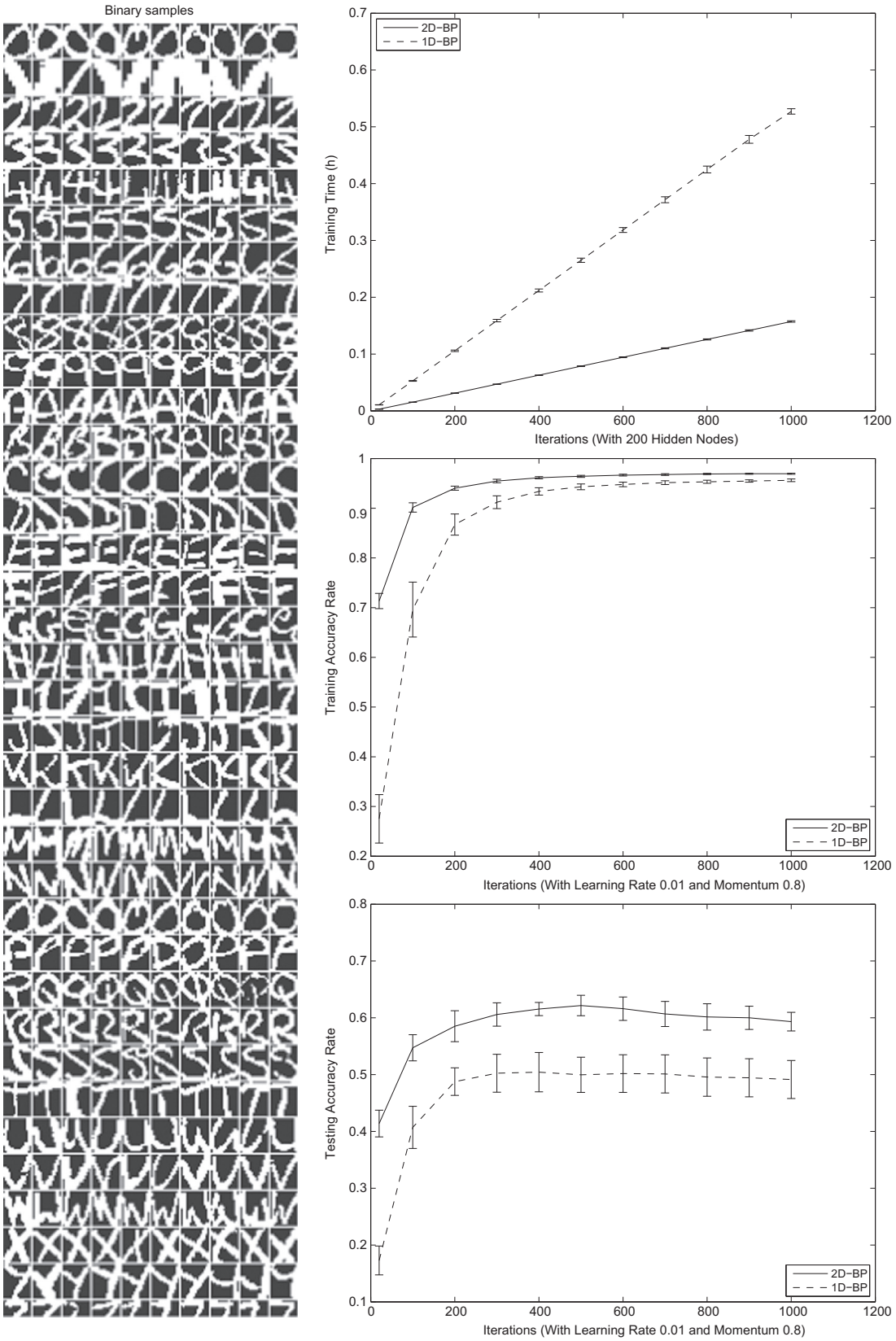

Fig. 3. Comparison of the 2D-BP and 1D-BP algorithms for different iterations on the Binary Alpha digits dataset (with 200 hidden nodes).

back-propagation with momentum modification:

$$\Delta\omega(i) = \gamma\Delta\omega(i-1) - (1-\gamma)\eta\nabla\omega, \tag{17}$$

where (i) represents the ith iteration.

Algorithm 3 describes our proposed momentum modification 2D-BP training procedure.

Algorithm 3. 2D-BP.

Require: Training dataset (N instances with matrix inputs), learning rate η, number of hidden nodes L, number of iterations s, and the momentum γ.

Ensure: Trained 2D-SLFN.

Initialize the weights of the 2D-SLFN:

$$f(X) = \sum_{k=1}^{L} \beta_k g(\mathbf{u}_k^\top X \mathbf{v}_k + b_k) + \alpha.$$

for $i \leftarrow 1$ to s **do**
  for $p \leftarrow 1$ to N **do**
    $\Delta\beta(i) \leftarrow \gamma\Delta\beta(i-1) - \eta(1-\gamma)\nabla\beta$ using Eq. (12)
    $\Delta\mathbf{u}(i) \leftarrow \gamma\Delta\mathbf{u}(i-1) - \eta(1-\gamma)\nabla\mathbf{u}$ using Eq. (13)
    $\Delta\mathbf{v}(i) \leftarrow \gamma\Delta\mathbf{v}(i-1) - \eta(1-\gamma)\nabla\mathbf{v}$ using Eq. (14)
    $\Delta\mathbf{b}(i) \leftarrow \gamma\Delta\mathbf{b}(i-1) - \eta(1-\gamma)\nabla\mathbf{b}$ using Eq. (15)
    $\Delta\alpha(i) \leftarrow \gamma\Delta\alpha(i-1) - \eta(1-\gamma)\nabla\alpha$ using Eq. (16)
    $\beta \leftarrow \beta + \Delta\beta(i)$
    $\mathbf{u} \leftarrow \mathbf{u} + \Delta\mathbf{u}(i)$
    $\mathbf{v} \leftarrow \mathbf{v} + \Delta\mathbf{v}(i)$
    $\mathbf{b} \leftarrow \mathbf{b} + \Delta\mathbf{b}(i)$
    $\alpha \leftarrow \alpha + \Delta\alpha(i)$
  end for
end for
return 2D-SLFN.

4. Performance evaluation

In this section we investigated the performance of the 2D-BP, and then compared it with two other methods: the 1D-BP and the 2D-NNRW.

4.1. Datasets

- *The US postal (USPS) handwritten digits*: They are composed of 8-bit 16×16 grayscale images of "0" through "9", 1100 examples of each class. They are available at http://www.cs.nyu.edu/~roweis/data.html.
- *The binary alpha digits*: They are composed of Binary 20×16 digits of "0" through "9" and capital "A" through "Z", 39

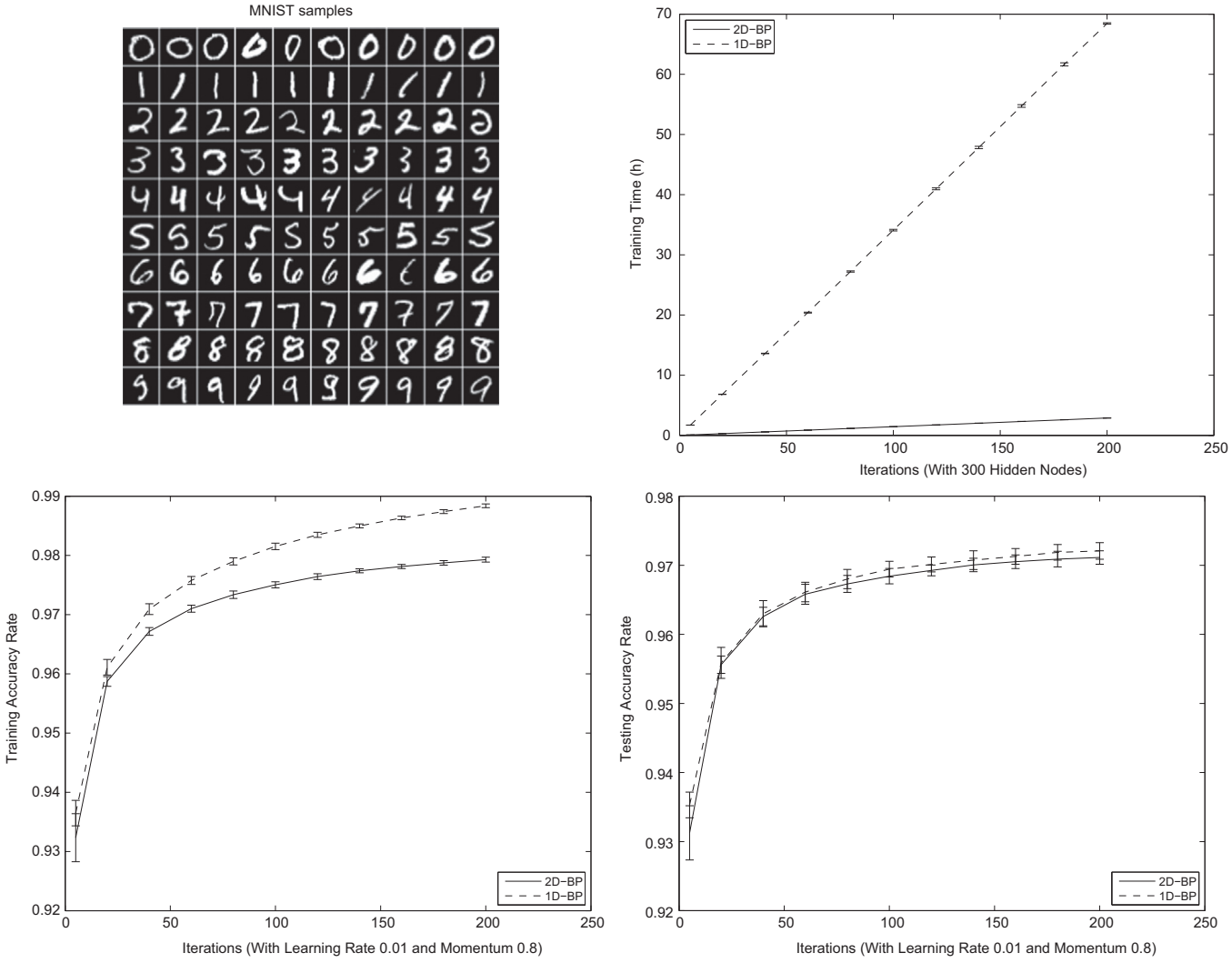

Fig. 4. Comparison of the 2D-BP and 1D-BP algorithms for different iterations on the MNIST handwritten digits dataset (with 300 hidden nodes).

examples of each class. They are available at http://www.cs.nyu.edu/~roweis/data.html.

- *The MNIST handwritten digits*: They are composed of 8-bit 28×28 grayscale images of "0" through "9". It is a training set of 60,000 examples, and a test set of 10,000 examples. They are available at http://yann.lecun.com/exdb/mnist/.
- *The Olivetti faces dataset*: It is composed of 8-bit 64×64 grayscale faces of 40 different people. Each person has 10 images, so there are 400 total images. It is available at http://www.cs.nyu.edu/~roweis/data.html.

When using the USPS dataset, we randomly selected 8000 examples for training, and the remaining 3000 examples were used for testing. We randomly selected 1084 examples from the Binary Alpha dataset for training, and the remaining 324 examples were used for testing. When using the MNIST dataset, we keep its original separation: 60,000 examples for training and 10,000 examples for testing. Finally, we selected 60% images from the Olivetti face dataset for each person for training, and the remaining 40% were used for testing. All input and output attributes were normalized to [0, 1].

4.2. Experimental setup

As mentioned above, we used the sigmoidal activation function $g(x) = 1/(1+e^{-x})$ in all methods. In the 2D-NNRW method, 2D-SLFNs were trained using the standard least-squares method described in Algorithm 2, while in the 1D-BP method and our 2D-BP method, we use the momentum incremental gradient based on the full network training for 1D-SLFNs and 2D-SLFNs, as described in Algorithms 1 and 3, respectively.

Different datasets and different methods require different parameter values. There are four parameters to be set in our 2D-BP and the 1D-BP algorithms (number of hidden nodes L, learning

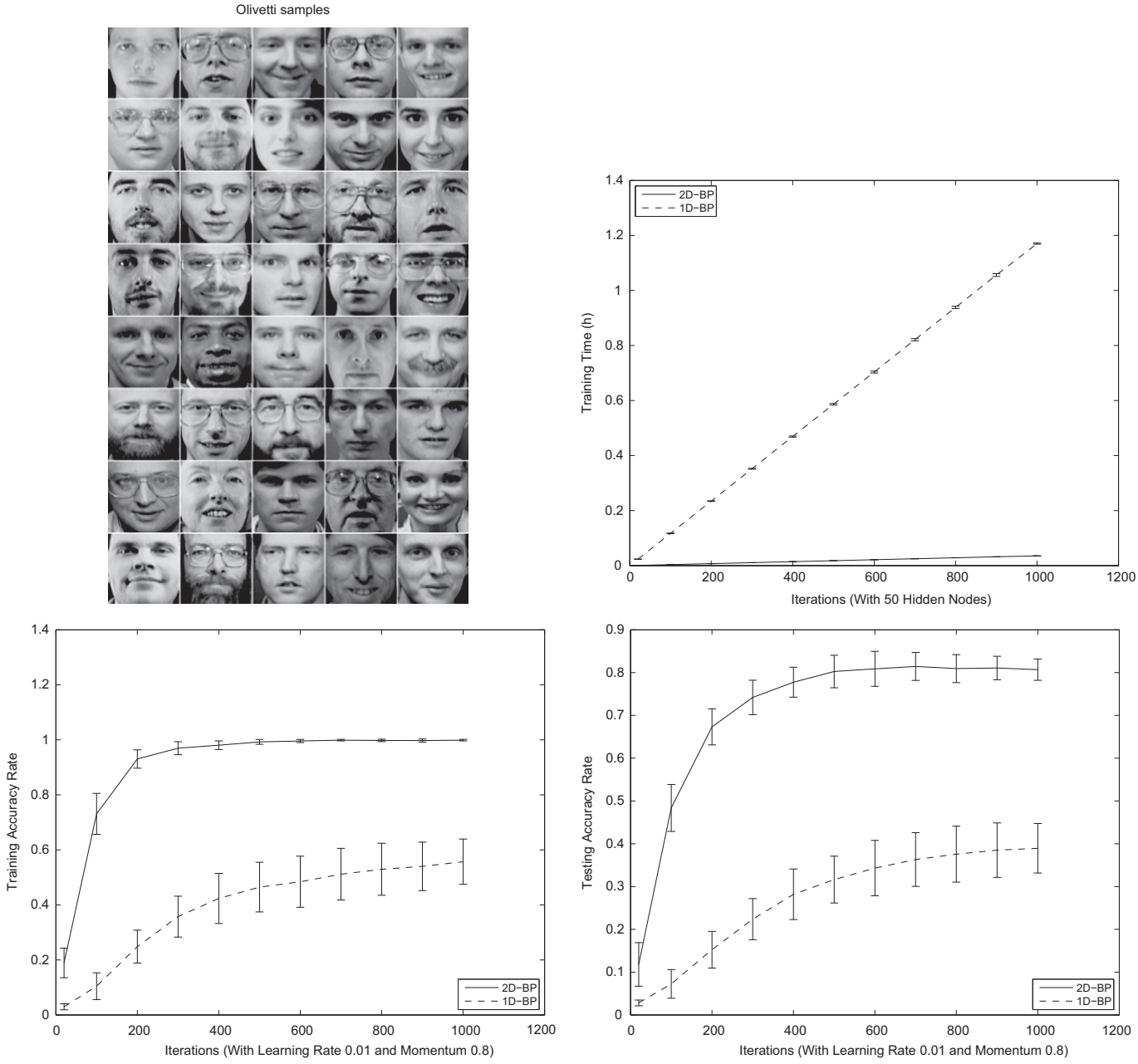

Fig. 5. Comparison of the 2D-BP and 1D-BP algorithms for different iterations on the Olivetti faces dataset (with 50 hidden nodes).

rate η, number of iterations s, and the momentum γ), and one parameter in the 2D-NNRW method (number of hidden nodes L).

Each experiment was simulated ten times to obtain more reliable results. The results from all simulations were then averaged. The simulations were executed on a computer that had 2 processers, 2.80 GHz and 2.79 GHz, and 32.0 GB of RAM. A parallel computing strategy was used to run the simulations efficiently in MATLAB R2013b.

4.3. Results and discussion

4.3.1. 2D-BP versus 1D-BP

To compare our 2D-BP algorithm with the traditional 1D-BP algorithm, the mean results of 10 independent runs on all datasets are shown in Fig. 2 (USPS data), Fig. 3 (Binary Alpha data), Fig. 4 (MNIST data), Fig. 5 (Olivetti face data).

We selected typical numbers of hidden nodes (100, 200, 300, 50) for each dataset. As we can see, for both the 2D-BP and 1D-BP algorithms, with an increasing number of iterations, the training and testing accuracy rates tend to be better and more stable. Additionally, the training accuracy rates of the 2D-BP (solid lines) on USPS and MNIST datasets are relatively low, because in 1D-SLFN, there are $L(mn+1)+o(L+1)$ free parameters, while in 2D-SLFN with the same number of hidden nodes, there are only $L(m+n+1)+o(L+1)$ free parameters (see Table 3). Thus increasing the complexity sometimes helps the network to better fit the training examples.

Interestingly, with the same number of iterations, the testing accuracy rate of our 2D-BP algorithm is always higher than the 1D-BP algorithm. Fitting the Binary Alpha dataset is a difficult task because there are 36 classes to classify, especially for "2" and "Z". Thus the testing accuracy rate is relatively low, but it still does not affect the conclusion that our 2D-BP algorithm is better than the 1D-BP algorithm.

In the 2D-SLFN approach, the mn parameters w_k were reduced to $m+n$ because of the projection vectors $\mathbf{u}_k$ and $\mathbf{v}_k$. When dealing with high-dimensional images, this reduction is astonishing. The results also suggests that the real freedom of the $m \times n$ matrix representation of an image is far less than mn, which is revealed by representing an image as an mn-dimensional vector in the general 1D-SLFN algorithm, which may also lead to over-fitting problems (see Figs. 2 and 3).

Additionally, because of the reduction of free parameters, the training time of the 2D-BP algorithm is less than the 1D-BP algorithm (see Table 3), and as the dimension of images increases, this advantages become more and more obvious. As we can see, the training time on MNIST is reduced to 2.9 h by using our 2D-BP method with 200 iterations, while the 1D-BP method needs almost 68 h, and their testing accuracy rates are almost the same as ours (see Fig. 4). Also see Fig. 5, where it is shown that the 2D-BP algorithm only takes 1/33 of training time needed in the 1D-BP algorithm, and achieves incredibly higher accuracy rates.

Table 1
Comparison of the training accuracy rates of the 2D-BP, 1D-BP and 2D-NNRW algorithms.

Dataset	2D-BP	1D-BP	2D-NNRW
USPS	0.9796 ± 0.0021	0.9946 ± 0.0006	0.9940 ± 0.0006
Binary	0.9694 ± 0.0010	0.9562 ± 0.0026	0.9722 ± 0.0000
MNIST	0.9793 ± 0.0004	0.9884 ± 0.0003	0.9613 ± 0.0008
Olivetti	0.9988 ± 0.0028	0.5571 ± 0.0825	1.0000 ± 0.0000

Table 2
Comparison of the testing accuracy rates of the 2D-BP, 1D-BP and 2D-NNRW algorithms.

Dataset	2D-BP	1D-BP	2D-NNRW
USPS	0.9543 ± 0.0020	0.9371 ± 0.0020	0.9412 ± 0.0039
Binary	0.5932 ± 0.0164	0.4914 ± 0.0334	0.5836 ± 0.0171
MNIST	0.9711 ± 0.0010	0.9721 ± 0.0012	0.9561 ± 0.0013
Olivetti	0.8069 ± 0.0249	0.3894 ± 0.0578	0.7112 ± 0.0361

Table 3
Comparison of training time and the number of free weights of the 2D-BP, 1D-BP and 2D-NNRW algorithms.

Dataset	Training time (h)			Number of free weights		
	2D-BP	1D-BP	2D-NNRW	2D-BP	1D-BP	2D-NNRW
USPS	0.3535	0.4422	0.1089	4310	26,710	20,010
Binary	0.1573	0.5271	0.0052	14,636	71,436	72,036
MNIST	2.8880	68.4555	4.1845	20,110	238,510	20,010
Olivetti	0.0354	1.1704	0.0001	8490	206,890	20,040

Table 4
Parameter values of each dataset from the results in Tables 1–3.

Dataset	2D-BP				1D-BP				2D-NNRW
	L_1[a]	η_1[b]	s_1[c]	γ_1[d]	L_2[a]	η_2[b]	s_2[c]	γ_2[d]	L_3[a]
USPS	100	0.01	500	0.8	100	0.01	500	0.8	2000
Binary	200	0.01	1000	0.8	200	0.01	1000	0.8	2000
MNIST	300	0.01	200	0.8	300	0.01	200	0.8	2000
Olivetti	50	0.01	1000	0.8	50	0.01	1000	0.8	500

[a] The number of hidden nodes.
[b] The learning rate.
[c] The number of iterations.
[d] The momentum.

4.3.2. 2D-BP versus 2D-NNRW

On the other hand, in 2D-NNRW algorithm, the number of free weights is $o(L+1)$ because of its randomness. The mean results on each dataset of 2D-NNRW are shown in Fig. 6. The training and testing accuracy rates are the average results of 10 independent runs for different number of hidden nodes.

Increasing the number of hidden nodes in the 2D-NNRW method results in better training accuracies. But for small datasets, such as the Binary Alpha dataset and the Olivetti dataset, over-fitting problems are evident (see right two plots of Fig. 6). Moreover, after hitting the bottom, their testing accuracy rates begin to increase again. However, even when the number of hidden nodes increases to a very large number, their testing accuracy rates are still unable to reach the rates achieved by our 2D-BP algorithms that used considerably fewer hidden nodes (see Tables 2 and 4).

In 2D-NNRW, the simplified training process indeed reduces the training time, which is mainly spent on calculating the hidden matrix H and its inverse. However, for high dimensional image datasets with large amounts of data (MNIST dataset), and with more hidden nodes than used in the 2D-NNRW method, this advantage becomes less obvious (see Table 3). We can also see in Table 2 that our approach is more stable.

5. Conclusions

In the structure of classical FNNs, the inputs are usually vectors, and thus FNNs cannot be used to classify matrix data directly. In this paper, a new FNNs algorithm, called the two-dimensional

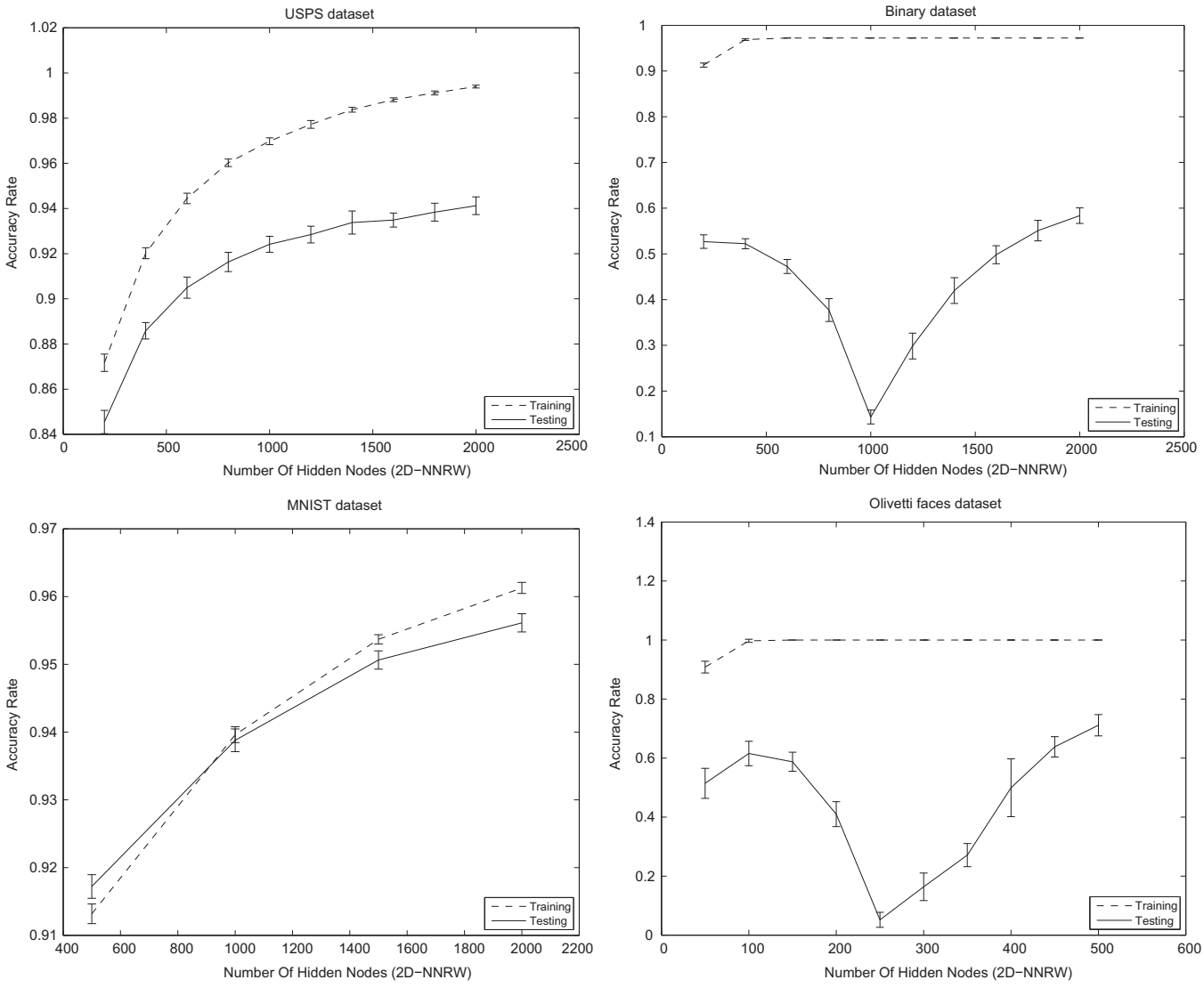

Fig. 6. Results of the 2D-NNRW algorithm for different numbers of hidden nodes on each dataset.

back-propagation (2D-BP), was proposed for classifying matrix data directly. The method of incremental gradient based learning is used to fully train the weights in extended 2D-FNNs. Since the proposed algorithm can retain important information of 2D input feature, it can be used as a classifier for image recognition. A series of experiments showed that when we compare the results of our new method with other widely used methods, our proposed approach has several advantages, including strong approximation and generalization capabilities, low computational complexity, and using less memory.

However, it should be pointed out that as the theoretical basis of the proposed new FNNs algorithm, the approximating property of this approach has not been proved yet in this paper. Since this kind of neural network is specifically designed for image recognition, its effectiveness certainly will be weaken, or indeed vanish, when the inputs are not in matrix form.

Appendix A. Supplementary data

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.engappai.2015.03.010.

References

Barron, A.R., 1993. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans. Inf. Theory 39 (3), 930–945.
Battiti, R., Masulli, F., 1990. BFGS optimization for faster and automated supervised learning. In: International Neural Network Conference, Springer, Netherlands, pp. 757–760.
Bishop, C.M., 1995. Neural Networks for Pattern Recognition.
Cybenko, G., 1989. Approximation by superposition of sigmoidal function. Math. Control, Signals Syst. 2, 303–314.
Fletcher, R., Reeves, C.M., 1964. Function minimization by conjugate gradients. Comput. J. 7 (2), 149–154.
Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (5), 359–366.
Hou, C., Nie, F., Yi, D., et al., 2013. Efficient image classification via multiple rank regression. IEEE Trans. Image Process. 22 (1), 340–352.
Hyvärinen, A., Oja, E., 2000. Independent component analysis: algorithms and applications. Neural Networks 13 (4), 411–430.
Igelnik, B., Pao, Y.H., 1995. Stochastic choice of basis functions in adaptive function approximation and the functional-link net. IEEE Trans. Neural Netw. 6 (6), 1320–1329.
Jolliffe, I., 2002. Principal Component Analysis, 2nd ed. Springer-Verlag, New York.
LeCun, Y., Bottou, L., Bengio, Y., et al., 1998. Gradient-based learning applied to document recognition. Proc. IEEE 86 (11), 2278–2324.
Li, M., Yuan, B., 2005. 2D-LDA: a statistical linear discriminant analysis for image matrix. Pattern Recognit. Lett. 26 (5), 527–532.
Lin, S.H., Kung, S.Y., Lin, L.J., 1997. Face recognition/detection by probabilistic decision-based neural network. IEEE Trans. Neural Netw. 8 (1), 114–132.
Lu, J., Zhao, J., Cao, F., 2014. Extended feed forward neural networks with random weights for face recognition. Neurocomputing 136, 96–102.

McLoone, S., Irwin, G.W., 1997. Fast parallel off-line training of multilayer perceptrons. IEEE Trans. Neural Netw. 8 (3), 646–653.
McLoone, S., Irwin, G., 2001. Improving neural network training solutions using regularisation. Neurocomputing 37 (1), 71–90.
Pao, Y.H., Park, G.H., Sobajic, D.J., 1994. Learning and generalization characteristics of the random vector functional-link net. Neurocomputing 6 (2), 163–180.
Sanguansat, P., Asdornwised, W., Jitapunkul, S., et al., 2006. Two-dimensional linear discriminant analysis of principle component vectors for face recognition. IEICE Trans. Inf. Sys. 89 (7), 2164–2170.
Schmidt, W.F., Kraaijveld, M.A., Duin, R.P.W., 1992. Feedforward neural networks with random weights. In: Proceedings of the 11th IEEE International Conference on IAPR, Pattern Recognition, Conference B: Pattern Recognition Methodology and Systems, vol. II, pp. 1–4.
Shakhnarovich, G., Darrell, T., Indyk, P., 2008. Nearest-neighbor methods in learning and vision. IEEE Trans. Neural Netw. 19 (2), 377–389.
Shang, L., Huang, D.S., Du, J.X., et al., 2006. Palmprint recognition using Fast ICA algorithm and radial basis probabilistic neural network. Neurocomputing 69 (13), 1782–1786.
Tyukin, I.Y., Prokhorov, D.V., 2009. Feasibility of random basis function approximators for modeling and control, in: Proceeding of the Multi-Conference on Systems and Control, Saint Petersburg, Russia, pp. 1391–1396.
Vapnik, V., 2000. The Nature of Statistical Learning Theory. Springer, London.
Yang, J., Zhang, D., Frangi, A.F., et al., 2004. Two-dimensional PCA: a new approach to appearance-based face representation and recognition. IEEE Trans. Pattern Anal. Mach. Intell. 26 (1), 131–137.
Yang, W., Yan, X., Zhang, L., et al., 2010. Feature extraction based on fuzzy 2DLDA. Neurocomputing 73 (10), 1556–1561.
Zhang, D., Zhou, Z.H., 2005. (2D) 2PCA: two-directional two-dimensional PCA for efficient face representation and recognition. Neurocomputing 69 (1), 224–231.