ORIGINAL RESEARCH

A three-stage learning algorithm for deep multilayer perceptron with effective weight initialisation based on sparse auto-encoder

Maysa Ibrahem Almulla Khalaf*[1,2], John Q Gan[1]

[1] School of Computer Science and Electronic Engineering, University of Essex, Essex, United Kingdom
[2] Department of Computer Science, Baghdad University, Baghdad, Iraq

Received: January 21, 2019 **Accepted:** March 19, 2019 **Online Published:** April 2, 2019
DOI: 10.5430/air.v8n1p41 **URL:** https://doi.org/10.5430/air.v8n1p41

ABSTRACT

A three-stage learning algorithm for deep multilayer perceptron (DMLP) with effective weight initialisation based on sparse auto-encoder is proposed in this paper, which aims to overcome difficulties in training deep neural networks with limited training data in high-dimensional feature space. At the first stage, unsupervised learning is adopted using sparse auto-encoder to obtain the initial weights of the feature extraction layers of the DMLP. At the second stage, error back-propagation is used to train the DMLP by fixing the weights obtained at the first stage for its feature extraction layers. At the third stage, all the weights of the DMLP obtained at the second stage are refined by error back-propagation. Network structures and values of learning parameters are determined through cross-validation, and test datasets unseen in the cross-validation are used to evaluate the performance of the DMLP trained using the three-stage learning algorithm. Experimental results show that the proposed method is effective in combating overfitting in training deep neural networks.

Key Words: Sparse auto-encoder, Deep learning, Feature learning, Effective weight initialization

1. INTRODUCTION

In recent years, big data analysis via deep learning has attracted much attention in various areas such as computer vision, speech recognition, social media analysis, fraud detection, and medical informatics.[1–7] One of the main advantages of deep learning due to the use of deep neural network structures is that it can learn feature representation, without separate feature extraction process that is a very significant processing step in pattern recognition.[8,9]

Unsupervised learning is usually required for feature learning, such as feature learning using sparse auto-encoder (SAE) and restricted Boltzmann machine (RBM).[10–12] For classification tasks, supervised learning is more desirable using support vector machine or feedforward neural networks as classifiers. How to effectively combine supervised learning with unsupervised learning is a critical issue to the success of deep learning for traditional pattern classification.[13]

Other major issues in deep learning include the overfitting problem and vanishing/exploding gradients during error back-propagation due to adopting deep neural network structures such as deep multilayer perceptron (DMLP).[13,14]

Many techniques have been proposed to solve the problems in training deep neural networks. Hinton et al.[15] introduced the idea of greedy layer-wise pre-training. Bengio et al.[16] proposed to train the layers of a deep neural net-

*Correspondence: Maysa Ibrahem Almulla Khalaf; Email: miabdu@essex.ac.uk; Address: School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, CO4 3SQ, Colchester, Essex, United Kingdom.

work in a sequence using an auxiliary objective and then "fine-tune" the entire network with standard optimization methods such as stochastic gradient descent. Martens[17] showed that truncated-Newton method has the ability to train deep neural networks from certain random initialisation without pre-training; however, it is still inadequate to resolve the challenges in training deep neural networks and most deep learning models can not be effectively trained with random initialisation.[18–21]

Effective weight initialisation or pre-training has been widely explored for avoiding vanishing/exploding gradients.[22–26] Using a huge amount of training data can overcome overfitting to some extent.[14] However, in many applications, there is no large amount of training data available or there is insufficient computer power to handle huge amount of training data, and regularisation techniques such as sparse structure and dropout technique are widely used for combatting overfitting.[27–29]

This paper proposes a three-stage learning algorithm for training DMLP with effective weight initialisation based on SAE, aiming to overcome difficulties in training deep neural networks with limited training data in high-dimensional feature space. Experiments were conducted on six datasets, with the performance of the proposed method evaluated by comparison with existing methods. This paper is organized as follows: Section 2 describes the basic principles of sparse auto-encoder, deep multilayer perceptron and the proposed approach. Section 3 presents the experimental results and discussion. Conclusion is drawn in Section 4.

2. Sparse auto-encoder, deep multilayer perceptron, and the proposed approach

2.1 Sparse auto-encoder algorithm

An auto-encoder is an unsupervised neural network trained by using stochastic gradient descent algorithms, which learns a non-linear approximation of an identity function.[12,27,28]

Figure 1 illustrates a non-linear multilayer auto-encoder network. It may have three or more hidden layers, but for simplicity a sparse auto-encoder with just a single hidden layer is described in detail as follows.

The connection weights and bias parameters can be denoted as $w = [vectorised\, W_1; vectorised\, W_2; b_1; b_2]$, where $W_1 \in R^{K\times N}$ is the encoding weight matrix and $W_2 \in R^{N\times K}$ is the decoding weight matrix, $b_1 \in R^K$ is the encoding bias vector, and $b_2 \in R^N$ is the decoding bias vector.

For a training dataset, let the output matrix of the auto-encoder be $O = [o^1, o^2, \cdots, o^m]$, which is supposed to be the reconstruction of the input matrix $X = [x^1, x^2, \cdots, x^m]$ where $o_i \in R^N$ and $x_i \in R^N$ are the output vector and input vector of the auto-encoder respectively, and m is the number of samples. Correspondingly, let the hidden output matrix be $H = [h^1, h^2, \cdots, h^m]$, where $h_i \in R^k$ is the hidden output vector of the auto-encoder to be used as feature vector in feature learning tasks.

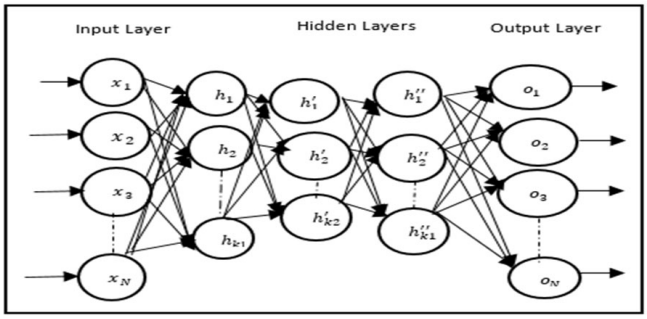

Figure 1. Multilayer auto-encoder

For the i^{th} sample, the hidden output vector is defined as

$$h^i = g\left(W_1 x^i + b_1\right) \tag{1}$$

and the output is defined by

$$o^i = g\left(W_2 h^i + b_2\right) \tag{2}$$

where $g(x)$ is the sigmoid logistic function $[1+\exp(-x)]^{-1}$.

For the sparse auto-encoder, the learning objective function is defined as follows:[12]

$$J_{sparse}(w) = \frac{1}{2m}\sum_{i=1}^{m}\left\|x^i - o^i\right\|^2 + \frac{\lambda}{2}\|w\|^2 + \beta\sum_{j=1}^{K} KL\left(p \,\|\, \hat{p}_j\right) \tag{3}$$

where p is the sparsity parameter, $\hat{p}_j$ is the average output of the j^{th} hidden node, averaged over all the samples, i.e.,

$$\hat{p}_j = \frac{1}{m}\sum_{i=1}^{m} h_j^i \tag{4}$$

λ is the coefficient for L_2 regularisation (weight decay), and β is the coefficient for sparsity control, which is defined by the Kullback-Leibler divergence:

$$KL\left(p \,\|\, \hat{p}_j\right) = p \log \frac{p}{\hat{p}_j} + (1-p)\log \frac{1-p}{1-\hat{p}_j} \tag{5}$$

The weight decay and sparsity control in the learning objective function are for combatting overfitting. The learning rule for updating the weight vector w (containing W_1, W_2, b_1,

ISSN 1927-6974 E-ISSN 1927-6982

and b_2) is error back-propagation based on gradient descent, i.e., $\Delta w = -\eta . w_{grad}$. The error gradients with respect to W_1, W_2, b_1, and b_2 are derived as follows respectively:[12,30]

$$W_1 grad = (W_2{}^T (O - X) + \beta\left(- p/\hat{p}_j + {}^{1-p}/_{1-\hat{p}_j}\right) I^T) .* g'(H) X^T / m + \lambda W_1 \quad (6)$$

$$b_1 grad = (W_2{}^T (O - X) + \beta\left(- p/\hat{p}_j + {}^{1-p}/_{1-\hat{p}_j}\right) I^T) .* g'(H) I / m \quad (7)$$

$$W_2 grad = ((O - X) H^T) / m + \lambda W_2 \quad (8)$$

$$b_2 grad = (O - X) I / m \quad (9)$$

where $g'[H] = g[H] \cdot *(1 - g[H])$ is the derivative of the sigmoid logistic function, $I = [l, l, \cdots, l]^T$ is a one vector of size m and .* represents element-wise multiplication.[30]

2.2 DMLP

A deep multilayer perceptron is a supervised feedforward neural network with multiple hidden layers. For simplicity, Figure 2 illustrates a DMLP with 2 hidden layers only (There are usually more than 2 hidden layers).

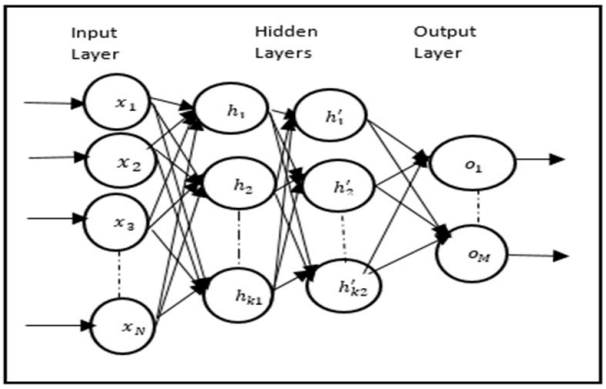

Figure 2. Deep multilayer perceptron (DMLP)

2.3 Proposed approach

Training deep neural networks usually needs a huge amount of training data, especially in high-dimensional input space. Otherwise, overfitting would be a serious problem due to the high complexity of the neural network model. However, in many applications the required huge amount of training data may be unavailable or the computer power available is insufficient to handle a huge amount of training data. With deep neural network training, there may also be local minimum and vanishing/exploding gradient problems without appropriate weight initialisation. A three-stage learning algorithm for DMLP with effective weight initialisation based on sparse auto-encoder is proposed in this paper to combat these problems, which consists of the following three stages:

(1) At the first stage, unsupervised learning is adopted to train a sparse auto-encoder with random initial weights to obtain the initial weights of the feature extraction layers of the DMLP. The auto-encoder consists of N input units, an encoder with two layers of K1 and K2 neurons in each hidden layer respectively, a symmetric decoder, and N output units. Figure 3 illustrates how it works.

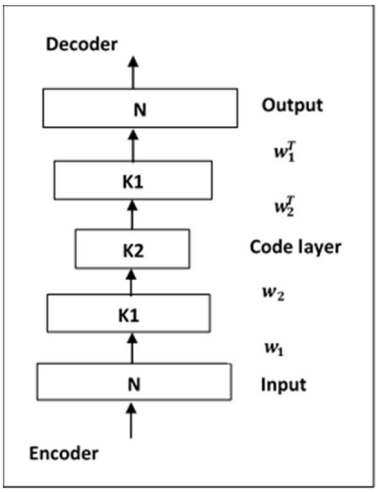

Figure 3. Training sparse auto-encoder (SAE)

(2) At the second stage, error back-propagation is employed to pre-train the DMLP by fixing the weights obtained at the first stage for its feature extraction layers (W1 and W2). The weights of higher hidden layers and output layer for feature classification (W3, W4 and W5) are trained with random initial weights. Figure 4 illustrates how it works.

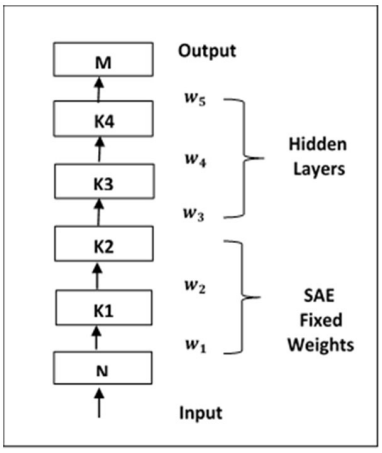

Figure 4. Pre-training DMLP with fixed W1 and W2 from SAE

(3) At the third stage, all the weights of the DMLP obtained at the first and second stages are refined by error back-propagation, without random weight initialisation. Figure 5 illustrates how it works.

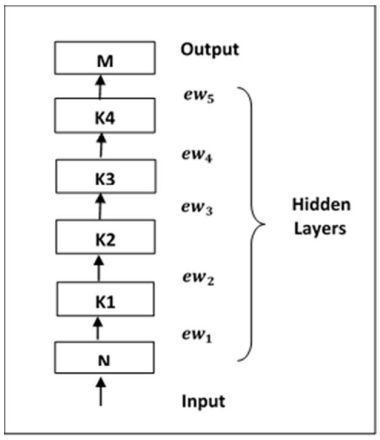

Figure 5. Refined-training of the DMLP with initial weights from the pre-trained DMLP

In our experiment, the pre-trained DMLP at the second stage, denoted as M1, and the refined DMLP obtained at the third stage, denoted as M2, are compared on several datasets. They are also compared with the DMLP trained using random initial weights for all layers, which is denoted as M3.

3. EXPERIMENTAL RESULTS AND DISCUSSION

3.1 Data sets

Six datasets were employed in the experiments to evaluate the performance of the proposed method: 1) an email_V1 dataset (`http://snap.stanford.edu/data/`),[12,30] 2) an email_V2 dataset (`https://www.kaggle.com/wcukierski/enron-email-datase`),[12,30] 3) the Reuters-21578 document dataset (`http://www.daviddlewis.com/resources/testcollection/reuters21578`),[12,30] 4) the 20Newsgroups corpus dataset which is a collection of approximately 20,000 newsgroup documents divided into 20 discussion groups (`https://archive.ics.uci.edu/ml/datasets/Twenty+Newsgroups`),[12] 5) the Musk dataset (http://archive.ics.uci.edu/ml/datasets.html),[30] and 6) a dataset of phishing technical website features (`http://khonji.org/phishing_studies`).[30] Because some news groups in the 20Newsgroups dataset are very closely related to each other, only four relatively distinguishable classes were used in the experiments.

The total number of features for email_V1, email_V2, Reuters-21578, 20Newsgroups, Musk, and the phishing technical features dataset are 750, 465, 421, 2,000, 166, and 47 respectively.

3.2 Experiment procedure

For each dataset, the experiment was repeated five times in order to assess the consistency of the results, with different data partition obtained by shuffling the data using different random seeds for each run. In each run the dataset was partitioned into a training set and a testing set, and the training set was further partitioned into estimation set and validation set for cross-validation (5-fold) to determine the optimal or appropriate network structure and hyper-parameter values (λ, β, p).

The proposed method was cross-validated with different number of hidden layers and different number of hidden neurons, and each testing set was only used once to evaluate the performance of the proposed method with the network structure trained using the hyper-parameter values chosen by the cross-validation.

The methods for comparison were named as Mi-jHL, where i = 1, 2, or 3 representing one of the three methods described in Section 2 and j = 1, 2, 3, or 4 representing the number of hidden layers. As a typical DMLP with 4 hidden layers, the numbers of hidden neurons in the first stage are K1 and K2 respectively, and the numbers of hidden neurons in the second or third stage are K1, K2, K3 and K4 respectively. For training the SAE, 8 sets of hyper-parameters (λ, β, p) were validated, as shown in Table 1, which are around the suggested default values.

Table 1. Hyper-parameters for Training SAE

Hyper-parameters	HP1	HP2	HP3	HP4	HP5	HP6	HP7	HP8
L2W (λ)	0.001	0.001	0.001	0.001	0.001	0.01	0.1	0.5
Sp. Re. (β)	2	4	1.6	0.5	0	0	0	0
Sp. Pr. (p)	0.0005	0.005	0.05	0.05	1	1	1	1

3.3 Results

Classification Accuracy: Tables 2-7 show the cross-validation (5-fold) classification accuracies of the three methods with different hyper-parameter values: Chosing the approporate hyper parameter, different number of hidden layers (M1-4HL, M1-3HL, M1-2HL, M2-4HL, M2-3HL, M2-2HL,

M3-4HL and M3-1HL) and different number of hidden neurons (K1, K2, K3, K4), on the six datasets respectively. The last two columns in each table show the average and maximum accuracy, based on which approporate network structure and hyper parameter vlaues are chosen for each method.

Table 2. Cross-Validation Accuracy on email_V1 Dataset

Methods\ Network Structure / Hyper-parameters		HP 1	HP 2	HP 3	HP 4	HP 5	HP 6	HP 7	HP 8	Ave Acc.	Max Acc.
	M1-4HL	88.9	68.9	86.4	88.9	88.7	88.9	87.1	88.4	85.7	88.9
	M1-3HL	87.2	68.4	85.1	86.1	87.5	87.2	87.0	87.1	84.4	87.5
	M1-2HL	87.1	68.1	86.8	86.0	87.1	87.1	87.0	87.1	84.5	87.1
Cross-Valid. Acc.	M2-4HL	90.2	87.9	89.3	90.7	88.5	90.2	88.9	88.5	89.2	90.7
(750 -25-20-10-5-2)	M2-3HL	89.2	86.2	88.3	88.4	88.4	89.2	87.2	88.1	88.1	89.2
	M2-2HL	89.1	86.3	88.9	87.3	88.1	89.1	88.1	88.1	88.1	89.1
	M3-4HL									87.9	88.0
	M3-1HL									87.1	87.8
	M1-4HL	89.7	78.1	87.4	89.2	89.4	91.6	84.1	89.6	87.3	91.6
	M1-3HL	89.4	80.4	86.2	83.6	91.2	89.8	83.5	89.0	86.6	91.2
	M1-2HL	88.7	79.3	86.8	91.5	88.6	89.5	84.0	88.4	87.1	91.5
Cross -Valid. Acc.	M2-4HL	91.0	92.8	87.5	92.6	91.7	90.5	90.9	92.4	91.1	92.8
(750-50-25-15-10-2)	M2-3H	91.2	90.3	87.3	90.1	91.6	89.1	88.2	90.2	89.7	91.6
	M2-2HL	89.8	88.3	90.1	90.9	88.8	88.9	89.4	90.0	89.5	90.6
	M3-4HL									88.1	88.8
	M3-1HL									87.5	87.9
	M1-4HL	88.5	86.2	85.6	87.9	88.1	88.4	88.9	89.2	87.8	89.2
	M1-3HL	87.8	86.8	84.3	87.1	88.0	87.7	87.6	88.8	87.2	88.8
	M1-2HL	87.2	85.3	85.5	86.2	87.2	87.3	87.2	88.3	86.7	88.3
Cross -Valid. Acc.	M2-4HL	88.9	89.9	87.3	88.9	89.2	89.3	90.2	90.2	89.2	90.2
(750-75-35-30-20-2)	M2-3HL	87.4	89.2	86.6	87.3	89.1	89.2	89.1	89.3	88.4	89.3
	M2-2HL	88.3	87.1	87.1	87.1	88.2	89.2	89.0	89.2	88.1	89.2
	M3-4HL									86.7	88.3
	M3-1HL									86.1	87.1

Table 3. Cross-Validation Accuracy on email_V2 Dataset

Methods\ Network Structure / Hyper-parameters		HP 1	HP 2	HP 3	HP 4	HP 5	HP 6	HP 7	HP 8	Ave Acc.	Max Acc.
	M1-4HL	85.7	87.7	86.9	87.1	89.1	87.6	88.2	89.3	87.7	89.3
	M1-3HL	87.9	85.4	86.1	86.3	88.3	88.2	87.7	88.5	87.3	88.5
	M1-2HL	83.8	86.3	85.8	85.4	87.8	88.1	87.3	87.6	86.5	88.1
Cross-Valid. Acc.	M2-4HL	91.5	88.9	87.8	90.2	89.3	91.2	89.2	89.2	89.6	91.5
(465-20-15-10-5-2)	M2-3HL	87.9	87.6	85.6	87.3	88.8	89.8	88.9	89.4	88.1	89.9
	M2-2HL	86.0	87.9	84.9	84.3	88.0	88.2	88.3	89.1	87.0	89.1
	M3-4HL									86.5	87.2.
	M3-1HL									86.1	87.0
	M1-4HL	90.1	88.8	90.7	93.5	91.4	89.9	85.6	89.6	89.9	93.5
	M1-3HL	88.3	87.3	89.4	88.8	92.5	89.8	83.5	89.9	88.6	92.5
	M1-2HL	86.9	88.5	90.8	87.4	89.7	88.9	85.1	88.7	88.2	90.8
Cross -Valid. Acc.	M2-4HL	90.9	89.8	88.7	92.3	93.5	91.9	92.7	94.4	91.7	94.4
(465-45-20-10-5-2)	M2-3HL	89.3	93.9	90.6	91.0	92.5	89.1	88.1	90.2	90.5	93.9
	M2-2HL	86.9	88.6	91.7	87.1	87.2	87.4	88.9	90.1	88.4	91.7
	M3-4HL									88.9	89.1
	M3-1HL									87.8	87.8
	M1-4HL	86.8	84.8	85.6	88.2	85.2	86.4	83.9	82.2	85.3	88.2
	M1-3HL	84.9	83.2	84.5	85.4	84.0	83.7	87.6	86.8	85.0	87.6
	M1-2HL	85.6	82.1	83.2	83.3	89.2	83.3	87.2	88.3	85.2	89.2
Cross -Valid. Acc.	M2-4HL	87.8	87.7	86.8	89.0	88.2	90.3	91.2	89.2	88.7	91.2
(465-65-30-25-15-2)	M2-3HL	86.4	91.7	86.2	84.3	88.1	89.5	89.1	88.6	87.9	91.7
	M2-2HL	87.8	88.9	84.6	86.1	87.2	89.5	89.0	86..2	87.5	89.5
	M3-4HL									85.9	86.1
	M3-1HL									85.3	85.8

Table 4. Cross-Validation Accuracy on Reuters Dataset

Methods\ Network Structure \ Hyper-parameters		HP 1	HP 2	HP 3	HP 4	HP 5	HP 6	HP 7	HP 8	Ave Acc.	Max Acc.
Cross-Valid. Acc. (421-35-25-20-15-10)	M1-4HL	78.2	75.5	69.4	76.9	67.2	78.5	69.3	62.5	72.1	78.5
	M1-3HL	76.5	71.3	67.3	76.3	66.1	67.5	77.2	61.9	70.5	77.2
	M1-2HL	76.2	70.1	64.6	74.5	60.3	69.9	68.5	76.9	70.1	76.9
	M2-4HL	77.1	78.5	75.7	77.9	71.5	77.8	77.2	76.5	76.5	78.5
	M2-3HL	74.7	76.2	73.7	77.2	70.9	79.3	76.4	77.4	75.7	79.3
	M2-2HL	73.8	73.6	71.8	75.8	70.1	77.4	76.3	76.2	74.3	77.4
	M3-4HL									68.7	77.1
	M3-1HL									68.9	75.8
Cross -Valid. Acc. (421-40-35-25-20-10)	M1-4HL	75.2	84.7	85.9	89.9	83.4	77.9	73.0	70.6	80.0	89.9
	M1-3HL	73.9	82.3	82.0	83.7	79.2	74.1	82.8	76.9	79.3	83.7
	M1-2HL	72.7	83.2	81.4	82.5	74.3	74.0	73.0	72.1	76.6	83.2
	M2-4HL	85.3	83.8	86.9	86.1	86.9	85.7	85.9	90.3	86.3	90.3
	M2-3HL	82.9	82.3	81.6	84.8	83.6	86.9	83.5	83.9	83.6	86.9
	M2-2HL	82.1	82.1	80.0	83.9	83.1	85.2	86.3	83.2	83.6	86.3
	M3-4HL									81.2	82.4
	M3-1HL									80.3	82.2
Cross -Valid. Acc. (421-65-55-45-30-10)	M1-4HL	79.5	67.2	77.4	76.8	60.4	70.7	69.1	69.2	71.2	79.5
	M1-3HL	76.8	65.6	76.7	74.1	60.2	73.9	76.7	67.8	71.4	76.8
	M1-2HL	74.6	64.9	67.5	75.7	60.1	70.4	69.9	66.3	68.6	76.7
	M2-4HL	78.3	77.3	78.2	78.2	77.1	75.4	79.5	73.9	77.2	79.5
	M2-3HL	76.8	75.2	77.8	77.1	76.3	76.6	78.9	73.5	76.5	78.9
	M2-2HL	75.8	73.9	76.8	74.6	73.3	76.7	76.8	72.2	75.0	76.8
	M3-4HL									68.6	74.3
	M3-1HL									66.8	73.9

Table 5. Cross-Validation Accuracy on Musk Dataset

Methods\ Network Structure \ Hyper-parameters		HP 1	HP 2	HP 3	HP 4	HP 5	HP 6	HP 7	HP 8	Ave Acc.	Max Acc.
Cross-Valid. Acc. (166-8-6-4-3-2)	M1-4HL	82.2	77.5	79.4	88.3	77.0	86.1	96.1	77.1	83.0	96.1
	M1-3HL	77.2	74.2	76.5	77.1	75.2	78.7	78.1	77.4	76.9	78.7
	M1-2HL	79.3	74.1	75.5	75.6	75.1	78.5	78.0	77.0	76.8	79.3
	M2-4HL	96.6	83.2	77.0	93.4	65.9	66.1	82.9	81.1	80.6	96.6
	M2-3HL	95.1	76.9	76.3	82.2	77.0	89.9	79.5	78.3	81.6	95.1
	M2-2HL	88.5	69.9	78.3	80.2	67.8	78.9	77.8	78.1	77.3	88.5
	M3-4HL									72.8	77.4
	M3-1HL									72.3	77.2
Cross -Valid. Acc. (166-10-7-5-3-2)	M1-4HL	88.8	85.2	87.8	88.1	75.1	77.5	93.8	79.9	84.5	93.8
	M1-3HL	81.5	83.2	91.1	84.2	77.7	67.5	75.3	93.9	81.8	93.9
	M1-2HL	80.4	82.4	77.0	76.3	76.7	93.2	75.2	75.2	79.5	93.2
	M2-4HL	99.2	99.6	99.8	99.2	83.0	76.7	98.9	79.7	92.0	99.8
	M2-3HL	97.5	88.2	87.9	92.1	67.7	94.6	76.4	76.3	85.0	97.5
	M2-2HL	83.1	87.5	94.7	89.8	67.2	75.2	76.2	80.1	81.7	94.7
	M3-4HL									76.8	81.9
	M3-1HL									76.1	80.8
Cross -Valid. Acc. (166-15-10-8-6-2)	M1-4HL	92.6	80.2	79.5	92.4	77.7	77.8	92.1	80.8	84.1	92.6
	M1-3HL	78.5	79.2	91.8	79.6	78.3	79.5	78.2	77.5	80.3	91.8
	M1-2HL	77.8	77.3	78.3	76.8	77.1	87.9	78.1	77.4	78.8	87.9
	M2-4HL	96.2	82.8	88.2	90.1	77.1	76.4	82.9	81.5	84.4	96.2
	M2-3HL	86.6	80.3	74.8	83.2	77.4	84.3	82.6	85.9	81.8	86.6
	M2-2HL	84.7	79.4	72.6	83.1	85.4	75.4	82.4	81.2	80.5	85.4
	M3-4HL									77.3	80.1
	M3-1HL									76.9	78.3

Table 6. Cross-Validation Accuracy on 20Newsgroups Dataset

Hyper-parameters / Methods\ Network Structure		HP 1	HP 2	HP 3	HP 4	HP 5	HP 6	HP 7	HP 8	Ave Acc.	Max Acc.
	M1-4HL	67.8	67.4	65.2	67.9	66.2	62.4	72.5	73.5	67.6	73.5
	M1-3HL	66.2	66.8	66.7	64.1	66.4	60.3	61.5	62.7	64.4	66.8
	M1-2HL	64.7	65.3	65.7	63.3	65.5	66.8	69.5	67.3	66.0	69.5
Cross-Valid. Acc.	M2-4HL	77.8	79.2	77.4	79.3	79.7	77.5	72.4	73.8	77.1	79.7
(2000-50-35-25-20-4)	M2-3HL	76.4	78.6	75.	75.1	78.2	70.4	70.6	72.1	74.5	78.6
	M2-2HL	73.8	78.4	73.6	74.3	77.2	70.1	70.3	72.5	73.7	78.4
	M3-4HL									68.8	73.1
	M3-1HL									66.2	70.3
	M1-4HL	85.3	85.4	82.1	83.1	84.5	87.9	81.4	82.5	84.0	87.9
	M1-3HL	83.8	84.8	86.6	79.0	84.1	80.3	81.5	82.3	82.8	86.6
	M1-2HL	83.2	84.2	84.7	83.7	85.2	76.9	79.5	77.7	81.8	85.2
Cross -Valid. Acc.	M2-4HL	87.6	88.9	89.0	88.1	90.4	78.9	82.4	83.5	86.1	90.4
(2000-100-75-50-25-4)	M2-3HL	84.4	85.6	89.8	86.2	88.9	80.4	80.4	82.2	84.7	89.8
	M2-2HL	83.7	89.1	84.1	84.3	86.1	80.1	80.3	82.1	84.7	89.1
	M3-4HL									78.3	80.9
	M3-1HL									74.1	80.3
	M1-4HL	73.4	76.8	79.8	79.1	79.3	74.2	75.6	72.9	76.3	79.8
	M1-3HL	74.6	75.6	75.7	76.4	74.6	70.7	72.7	71.9	74.0	76.4
	M1-2HL	70.1	74.7	74.6	73.6	72.6	76.7	79.8	72.2	74.2	79.8
Cross -Valid. Acc.	M2-4HL	77.5	78.8	79.9	78.5	79.9	73.2	73.4	74.6	76.9	79..9
(2000-150-125-75-50-5)	M2-3HL	76.2	77.9	78.2	76.3	77.1	70.2	72.3	71.7	74.9	78.2
	M2-2HL	74.2	76.8	77.2	76.7	77.2	70.1	70.3	72.5	74.3	77.2
	M3-4HL									70.9	75.9
	M3-1HL									70.2	74.6

Table 7. Cross-Validation Accuracy on Phishing Technical Feature Dataset

Hyper-parameters / Methods\ Network Structure		HP 1	HP 2	HP 3	HP 4	HP 5	HP 6	HP 7	HP 8	Ave Acc.	Max Acc.
	M1-4HL	67.2	69.7	67.6	68.5	64.0	50.3	62.9	62.7	64.1	69.7
	M1-3HL	68.6	66.7	66.7	65.8	63.8	55.3	60.3	62.5	63.7	68.6
	M1-2HL	62.7	64.8	65.8	63.3	63.5	66.9	62.5	62.2	63.9	66.9
Cross-Valid. Acc.	M2-4HL	96.9	95.2	87.8	92.1	97.0	96.7	96.8	96.8	94.9	97.0
(47-8-6-4-3)	M2-3HL	95.7	94.7	86.1	88.9	96.2	92.1	93.3	94.2	92.6	96.2
	M2-2HL	88.9	88.2	88.9	87.6	89.1	89.1	92.3	89.5	89.2	92.3
	M3-4HL									60.7	65.8
	M3-1HL									60.9	65.3
	M1-4HL	67.0	77.8	78.5	73.3	66.0	59.9	90.5	62.8	71.9	90.5
	M1-3HL	64.6	70.9	72.4	72.9	65.8	85.4	66.9	61.8	70.0	85.4
	M1-2HL	81.9	65.2	74.2	63.2	65.3	58.7	66.3	61.5	67.0	81.9
Cross -Valid. Acc.	M2-4HL	99.4	99.7	99.3	98.6	99.1	99.4	99.6	99.0	99.2	99.7
(47-15-8-7-5-2)	M2-3HL	99.3	98.9	98.5	97.2	97.8	98.2	98.8	98.3	98.3	99.3
	M2-2HL	98.9	98.7	98.2	97.0	97.9	86.4	97.5	99.1	96.7	99.1
	M3-4HL									77.2	82.3
	M3-1HL									76.5	80.4
	M1-4HL	69.7	68.4	67.5	69.5	66.0	59.9	90.5	62.8	69.2	90.5
	M1-3HL	67.8	67.6	66.8	68.2	85.9	59.8	67.9	60.3	68.0	85.9
	M1-2HL	65.4	66.3	62.7	65.6	65.3	75.9	70.9	61.5	66.7	75.9
Cross -Valid. Acc.	M2-4HL	71.4	98.1	96.4	98.4	99.4	84.3	99.3	99.1	93.3	99.4
(47-30-15-10-6-2	M2-3HL	94.8	96.7	97.4	96.3	98.3	80.9	76.8	56.1	87.1	98.3
	M2-2HL	86.8	97.3	95.8	95.2	78.1	80.7	67.6	86.5	86.0	97.3
	M3-4HL									65.3	72.4
	M3-1HL									65.1	70.6

Tables 8-13 show the corresponding average (Avg.) training and testing accuracies and standard deviation (Std.) of the methods with the appropriate network structure trained using the hyper parameter values chosen by the 5-fold cross-validation. Figure 6 compares the three methods (M1, M2, and M3) in terms of average training accuracy and testing accuracy. It can be seen from Figure 6 that the proposed three-stage learning approach, M2-jHL, achieved the highest accuracy, which have been proved to be statistically significantly better than other methods evaluated in the experiment. From Figure 6 it can be seen that the proposed method (M2) has much smaller difference between testing accuracy and training accuracy than methods M1 and M3, which can be regarded as evidence of less serious overfitting in the proposed method. It can be concluded that DMLP with effective weight initialisation can achieve significantly better performance than the standard MLP, and it is evident that the three-stage learning algorithm for DMLP can reduce overfitting.

Table 8. Performance Comparison on email_V1 Dataset

Methods	Network Structure/Hyper-Parameters	Training Avg. Acc.	Training Std.	Testing Avg. Acc.	Testing Std.
M1-4HL	750-50-25-10-5-2 / HP 6	89.5%	1.43	83.3%	1.47
M1-3HL	750-50-25-10-2 / HP 5	90.5%	0.81	82.7%	0.59
M1-2HL	750-50-25-2 / HP 4	91.5%	1.39	80.3%	1.74
M2-4HL	750-50-25-10-5-2 / HP 2	92.9%	0.29	89.9%	0.28
M2-3HL	750-50-25-10-2 / HP 5	92.1%	0.36	88.2%	0.45
M2-2HL	750-50-25-2 / HP 4	92.9%	0.63	86.2%	0.54
M3-4HL	750-50-25-10-5-2	91.7%	1.52	81.7%	1.47
M3-1HL	750-50-2	89.6%	1.49	80.9%	1.52

Table 9. Performance Comparison on email_V2 Dataset

Methods	Network Structure/Hyper-Parameters	Training Avg.Acc.	Training Std.	Testing Avg. Acc.	Testing Std.
M1-4HL	465-45-20-10-5-2 / HP 4	93.2%	1.46	86.2%	1.16
M1-3HL	465-45-20-10-2 / HP 5	93.1%	1.78	85.1%	0.96
M1-2HL	465-45-20-2 / HP 3	93.0%	1.95	84.3%	1.02
M2-4HL	465-45-20-10-5-2 / HP 8	96.5%	0.17	93.7%	0.09
M2-3HL	465-45-20-10-2 / HP 2	96.1%	0.44	93.6%	0.38
M2-2HL	465-45-20-2 / HP 3	94.9%	0.87	91.2%	0.84
M3-4HL	465-45-20-10-5-2	93.1%	1.25	85.1%	1.46
M3-1HL	465-45-2	91.3%	1.32	84.7%	1.34

Table 10. Performance Comparison on Reuters Dataset

Methods	Network Structure/Hyper-Parameters	Training Avg Acc.	Training Std.	Testing Avg Acc.	Testing Std.
M1-4HL	421-40-35-25-20-10 / HP 4	93.9%	4.92	79.1%	4.89
M1-3HL	421-40-35-25-10 / HP 7	84.3%	2.84	75.5%	3.67
M1-2HL	421-40-35-10 / HP 2	83.9%	2.867	75.1%	3.52
M2-4HL	421-40-35-25-20-10 / HP 8	88.3%	1.71	86.2%	0.56
M2-3HL	421-40-35-25-10 / HP 6	85.3%	1.85	85.2%	0.65
M2-2HL	421-40-35-10 / HP 7	85.2%	1.78	84.1%	0.95
M3-4HL	421-40-35-25-20-10	83.9%	2.03	74.8%	2.76
M3-1HL	421-40-10	83.4%	2.34	74.4%	2.03

Table 11. Performance Comparison on Musk Dataset

Methods	Network Structure/Hyper-Parameters	Training Avg. Acc.	Training Std.	Testing Avg. Acc.	Testing Std.
M1-4HL	166-10-7-5-3-2 / HP 5	94.8%	2.99	82.1%	2.18
M1-3HL	166-10-7-5-2 / HP 8	94.7%	3.21	81.8%	2.78
M1-2HL	166-10-7-2 / HP 6	92.2%	3.32	81.1%	2.82
M2-4HL	166-10-7-5-3-2 / HP 7	96.2%	0.24	90.2%	0.17
M2-3HL	166-10-7-5-3-2 / HP 6	96.8%	0.26	87.5%	0.21
M2-2HL	166-10-7-5-3-2 / HP 3	94.3%	0.32	86.6%	0.25
M3-4HL	166-10-7-5-3-2	97.5%	2.17	80.6%	1.25
M3-1HL	166-10-7-5-3-2	96.2%	2.28	79.4%	1.34

Table 12. Performance Comparison on 20Newsgroups Dataset

Methods	Network Structure/Hyper-Parameters	Training Avg. Acc.	Training Std.	Testing Avg. Acc.	Testing Std.
M1-4HL	2000-100-75-50-25-4 / HP6	87.1%	2.54	81.8%	2.67
M1-3HL	2000-100-75-50-4 / HP 3	86.9%	2.76	81.3%	2.35
M1-2HL	2000-100-75-4 / HP 5	85.9%	2.88	80.3%	2.63
M2-4HL	2000-100-75-50-25-4 / HP 5	94.4%	1.42	93.2%	1.36
M2-3HL	2000-100-75-50-4 / HP 3	89.8%	1.56	89.1%	1.53
M2-2HL	2000-100-75-4 / HP 2	89.1%	1.24	86.4%	1.23
M3-4HL	2000-100-75-50-25-4	86.4%	2.05	81.6%	2.85
M3-1HL	2000-25-4	86.7%	2.56	80.3%	2.65

Table 13. Performance Comparison on Phishing Technical Features Dataset

Methods	Network Structure/Hyper -Parameters	Training Avg. Acc.	Training Std.	Testing Avg. Acc.	Testing Std.
M1-4HL	47-15-8-7-5-2 / HP 7	92.8%	4.30	67.6%	5.82
M1-3HL	47-15-8-7-2 / HP 6	92.3%	4.29	65.6%	5.27
M1-2HL	47-15-8-2 / HP 1	92.1%	4.32	64.9%	5.25
M2-4HL	47-15-8-7-5-2 / HP 2	99.7%	1.23	96.8%	0.41
M2-3HL	47-15-8-7-2 / HP 1	97.3%	1.27	95.9%	0.46
M2-2HL	47-15-8-2 / HP 8	97.9%	1.45	94.7%	0.47
M3-4HL	47-15-8-7-5-2	98.6%	3.42	61.5%	4.41
M3-1HL	47-15-2	97.4%	4.36	61.1%	4.54

Statistical Significance Test: To assess whether the performance differences among the methods are statistically significant, we applied T-test and the Wilcoxon's rank-sum test to determine whether two sets of accuracy data are significantly different from each other. The statistical tests were conducted on three paired methods (M2 vs. M1, M2 vs. M3, and M1 vs. M3) in terms of testing classification accuracy. Tables 14 and 15 show the p-values from these tests, which demonstrate that, in terms of classification performance, M2 significantly outperformed M1 and M3, and M1 significantly outperformed M3.

Table 14. Statistical Test Results (t-test)

Methods for comparison	*p*-value
M2 vs. M1	2.7431e-08
M2 vs. M3	2.8853e-06
M1 vs. M3	0.0026

Table 15. Statistical Test Results (rank-sum)

Methods for comparison	*p*-value
M2 vs. M1	0.0037
M2 vs. M3	0.0046
M1 vs. M3	0.0452

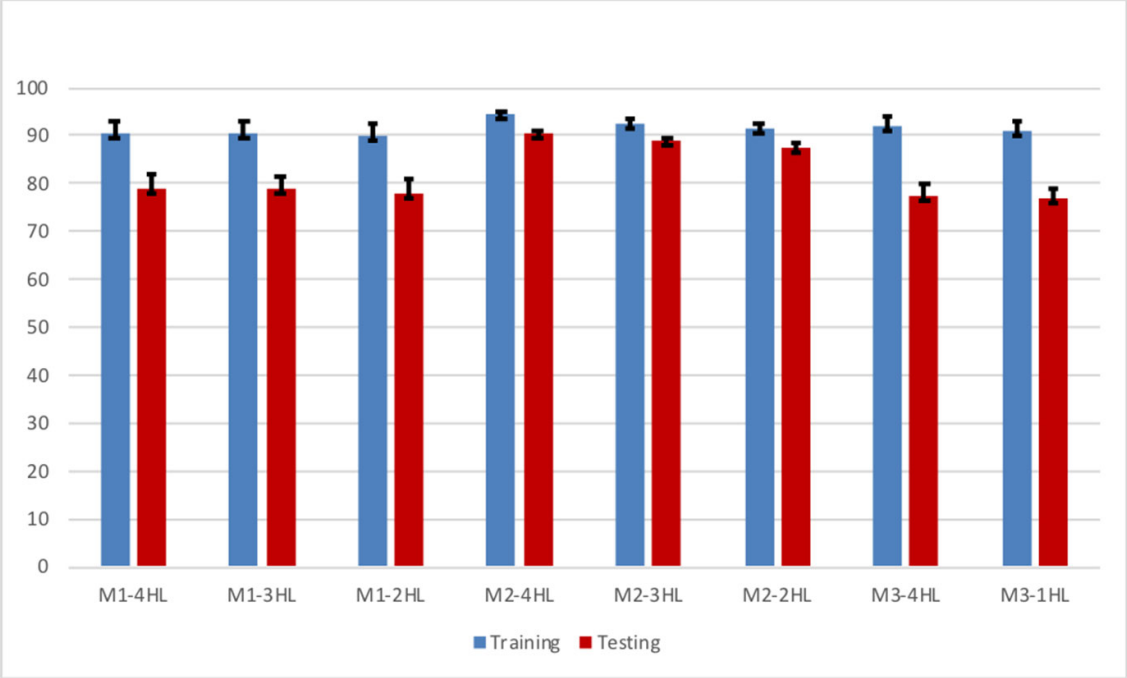

Figure 6. Comparison of three methods in terms of average training and testing accuracy and standard deviation

4. Conclusion

This paper proposes a three-stage learning approach for training deep multilayer perceptron with effective weight initialisation based on sparse auto-encoder. This approach can combat possible overfitting and vanishing/exploding gradient problems in deep learning with limited training data. It is evident from the experimental results that the deep multilayer perceptron trained using the proposed algorithm significantly outperformed the standard multilayer perceptron and its combination with sparse auto-encoder as well. Preliminary experimental results have demonstrated the advantages of the proposed method. Further tests on this algorithm would be applied to deep neural networks with more layers and for other applications as well would be conducted in future investigations.

REFERENCES

[1] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015; 521: 436-444. PMid:26017442.

[2] Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science. 2006; 313 (5786): 504-507. PMid:16873662. https://doi.org/10.1126/science.1127647

[3] Najafabadi MM, Villanustre F, Khoshgoftaar TM, et al. Deep learning applications and challenges in big data analytics. Journal of Big Data. 2015; 2(1): 1-21.

[4] Chen XW, Lin X. Big data deep learning challenges and perspectives. IEEE Access. 2014; 2: 514-525. https://doi.org/10.1109/ACCESS.2014.2325029

[5] Hinton GE, et al. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Processing Magazine. 2012; 29: 82-97.

[6] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proceedings of Advances in Neural Information Processing Systems. 2012; 25: 1090-1098.

[7] Ravì D, et al. Deep learning for health informatics. IEEE Journal of Biomedical and Health Informatics. 2017; 21: 4-21.

[8] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013; 35(8): 1798-1828. PMid:23787338. https://doi.org/10.1109/TPAMI.2013.50

[9] Bengio Y. Deep learning of representations: looking forward. Proceedings of International Conference on Statistical Language and Speech Processing, Spain. Lecture Notes in Computer Science (LNCS). 2013; 7978: 1-37.

[10] Salakhutdinov R, Hinton GE. Deep Boltzmann machines. Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, Florida, USA; 2014. pp. 448-455.

[11] Lee H, Ng AH, Koller D, et al. Unsupervised feature learning via sparse hierarchical representations. Ph.D. Thesis, Dept. of Comp. Sci., Stanford University; 2010.

[12] Abdulhussain MI, Gan JQ. Class specific pre-trained sparse autoencoders for learning effective features for document classification. Proceedings of the 8th Computer Science and Electronic Engineering Conference (CEEC), University of Essex, UK; 2016. pp. 36-41.

[13] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), Sardinia, Italy. 2010; 9: 249-256.

[14] Geman S, Bienenstock E, Doursat R. Neural networks and the bias/variance dilemma. Neural Computation. 1992; 4: 1-58. https://doi.org/10.1162/neco.1992.4.1.1

[15] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Computation. 2006; 18: 1527-1554. PMid:16764513. https://doi.org/10.1162/neco.2006.18.7.1527

[16] Bengio Y, Lamblin P, Popovici D, et al. Greedy layer-wise training of deep networks. Proceedings of Advances in Neural Information Processing Systems. MIT press; 2007. pp. 153-160.

[17] Martens J. Deep learning via Hessian-free optimization. Proceedings of the 27th International Conference on Machine Learning (ICML-10); 2010. p. 735-742.

[18] Martens J, Sutskever I. Training deep and recurrent networks with Hessian-free optimization. In Neural Networks: Tricks of the Trade, Springer, LNCS; 2012. p. 479-535.

[19] Mohamed A, Dahl GE, Hinton GE. Acoustic modeling using deep belief networks. IEEE Transactions on Audio, Speech, and Language Processing. 2010; 20: 14-22.

[20] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics; 2010. p. 249-256.

[21] Chapelle O, Erhan D. Improved preconditioner for hessian free optimization. Proceedings of the NIPS Workshop on Deep Learning and Unsupervised Feature Learning; 2011. p. 1-8.

[22] Yam JYF, Chow TW. A weight initialization method for improving training speed in feedforward neural network. Neurocomputing. 2000; 30(4): 219-232.

[23] Sutskever I, Martens J, Dahl G, et al. On the importance of initialisation and momentum in deep learning. Proceedings of the 30th International Conference on Machine Learning, Atlanta, USA; 2013.

[24] Fernandez-Redondo M, Hernandez-Espinosa C. Weight initialization methods for multilayer feedforward. Proceedings of European Symposium on Artificial Neural Networks (ESANN), Bruges Belgium; 2001. p. 119-124.

[25] Sousa CA. An overview on weight initialization methods for feedforward neural networks. Proceedings of International Joint Conference on Neural Networks (IJCNN), Vancouver, Canada; 2016. p. 52-59.

[26] Sodhi SS, Chandra P, Tanwar S. A new weight initialization method for sigmoidal feedforward artificial neural networks. Proceedings of International Joint Conference on Neural Networks (IJCNN), Beijing, China; 2014. p. 291-298.

[27] Zhang F, Du B, Zhang L. Saliency-guided unsupervised feature learning for scene classification. IEEE Transactions on Geoscience and Remote Sensing. 2015; 53(2): 2175-2184.

[28] Shu M, Fyshe A. Sparse autoencoders for word decoding from magnetoencephalography. Proceedings of the 3rd NIPS Workshop on Machine Learning and Interpretation in NeuroImaging (MLINI), Lake Tahoe, USA; 2013. p. 20-27.

[29] Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research. 2014; 16: 1929-1958.

[30] Almulla khalaf MI, Gan JQ. Deep classifier sructures with autoencoder for higher-level feature extraction. Proceedings of IJCCI 2018, University of Seville, Spain; 2018.