Neural Comput & Applic
DOI 10.1007/s00521-016-2401-x

RECENT ADVANCES IN PATTERN RECOGNITION AND ARTIFICIAL INTELLIGENCE

Text classification based on deep belief network and softmax regression

Mingyang Jiang[1,2] · Yanchun Liang[1,3] · Xiaoyue Feng[1] · Xiaojing Fan[4] · Zhili Pei[2] · Yu Xue[5] · Renchu Guan[1,3]

Received: 3 January 2016 / Accepted: 30 May 2016

Abstract In this paper, we propose a novel hybrid text classification model based on deep belief network and softmax regression. To solve the sparse high-dimensional matrix computation problem of texts data, a deep belief network is introduced. After the feature extraction with DBN, softmax regression is employed to classify the text in the learned feature space. In pre-training procedures, the deep belief network and softmax regression are first trained, respectively. Then, in the fine-tuning stage, they are transformed into a coherent whole and the system parameters are optimized with Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm. The experimental results on Reuters-21,578 and 20-Newsgroup corpus show that the proposed model can converge at fine-tuning stage and perform significantly better than the classical algorithms, such as SVM and KNN.

Keywords Deep belief networks · Softmax model · Restricted Boltzmann machines · L-BFGS · Feature learning

1 Introduction

In the 1990s, the neural network researchers mainly focused on the shallow structures which are typically characterized by a single nonlinear feature extraction layer and suitable for solving problems with small-scale data. When dealing with a large number of complex voice, text, and image problems in today's era of big data, the limitations are particularly prominent compared with shallow neural networks. Recently the deep structure models are proposed to extract features and build representations on large-scale datasets due to their strong adaptive and nonlinear feature extraction capabilities.

In 2006, Hinton et al. proposed a greedy unsupervised training method to tackle the weights optimization problem in deep belief networks (DBN). The effectiveness of the method was then proved in a number of applications and laid the theoretical foundation for the application and development of the deep structure of neural networks [1]. After that, deep learning became one of the most famous feature learning methods in machine learning area. Compared with traditional machine learning algorithms, the research results are particularly remarkable in speech recognition and image processing. For example, Deng et al. [2] and Sivaram et al. [3] successfully applied deep learning technology to address phone recognition problems; Yu et al. [4] and Dahl et al. [5] used deep neural networks in large vocabulary speech recognition, and the results are satisfactory; In Large Scale Visual Recognition Challenge 2012 competition, Krizhevsky established a

✉ Renchu Guan
guanrenchu@jlu.edu.cn

[1] Key Laboratory for Symbol Computation and Knowledge Engineering of National Education Ministry, College of Computer Science and Technology, Jilin University, Changchun 130012, China

[2] College of Computer Science and Technology, Inner Mongolia University for the Nationalities, Tongliao 028000, China

[3] Zhuhai Laboratory of Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Zhuhai College of Jilin University, Zhuhai 519041, China

[4] College of Mechanical Engineering, Inner Mongolia University for the Nationalities, Tongliao 028000, China

[5] School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China

Published online: 14 June 2016

large-scale convolution neural network (CNN) with 60,000,000 weights and 650,000 neurons, which trained 1.2 million high-resolution image data. Those images contain 1000 different species, and the results on 50,000 testing images show that the proposed deep learning method greatly reduces recognition errors for high-resolution images [6]. The essence of deep learning is to extract the hierarchical features or representations from raw data.

DBN is a type of deep neural network with a plurality of hidden layers, which is different from traditional shallow structures. During the learning processes, unlabeled samples are used for training and few labeled data are used for fine-tuning the entire network. Lawrence McAfee proposed that DBN network is not an ideal algorithm for text classification [7], although DBN had not achieved any significant result on image and speech recognitions. Liu Tao presented a hybrid text classification approach based on DBN and support vector machine (SVM), in which SVM is trained on the learned deep features. The hybrid method outperforms the classical SVM [8]. Hinton designed a deep generative model in which the lowest layer represents the word count vector of a document and the top layer represents a learned binary code for that document. By using short binary codes as addresses, Hinton's model can perform text retrieval on very large document sets at one time [9]. Chenchen Huang et al. used DBNs to automatically extract emotional features in speech signals. The learned features are the input of nonlinear SVM classifier, and a hybrid speech emotion recognition classifier system is achieved [10]. Shusen Zhou et al. proposed a semi-supervised learning algorithm called active hybrid DBN that addressed the semi-supervised sentiment classification problem. They constructed a two-part network, in which the previous hidden layers extract features with restricted Boltzmann machines (RBM), and the following hidden layers learn the comments information with convolutional restricted Boltzmann machines (CRBM). This method is effective for sentiment classification [11].

Text classification task is to determine the category of each document in the collection according to predefined categories [12]. With increasing scientific papers, Internet information, and other text-format data, automatic text categorization plays an important role in information retrieval, data mining and machine learning [13]. Commonly used classification methods are back propagation neural network, decision trees, K-nearest neighbor (KNN), naive Bayes and SVM, and especially SVM achieves good performance on the effectiveness and stability of classification [14–25, 26–28]. However, most of them are supervised learning algorithms and training data or labeled samples often demand great human efforts in practical applications. The unlabeled samples are easier to obtain than labeled ones. But facing such large number of unlabeled samples (such as texts in our case) with tens of thousands of features (such as words/phases), completely unsupervised learning is lack of relevant information to guide the results [29, 30]. Moreover, the text data are often semi-structured or unstructured, high dimensional, sparse and unlabeled, which can result in huge time and space complexity for text representation, feature reduction and text classification tasks.

In this paper, to solve the problem of the high-dimensional sparse matrix computing of text classification and the dilemma between the choice of supervised and unsupervised learning model, we proposed a novel hybrid algorithm combining a DBN and softmax regression. With a specially selected parameters optimization method–Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), the two algorithms cooperate well. The experimental results show that the proposed method can converge at fine-tuning stage and perform significantly better than the classical classification algorithms on different data scales of Reuters-21,578 and 20-Newsgroup.

The rest of the paper is organized as follows: In the next section, DBN and softmax regression are briefly introduced. In Sect. 3, we present more details of the proposed hybrid algorithm. In Sect. 4, the experimental setup and results on the benchmark datasets are discussed. Finally, we conclude this paper in the last section.

2 Background

2.1 Deep belief network

Deep belief network (DBN) is a deep learning structure, which is superposed by RBM. Each two adjacent layers in DBN is paired as a RBM. It is a two-layer neural network. Nodes in the same layer are not connected with each other, while they are fully connected with the nodes of other layers. The input layer is used to train the connection weights between the two layers, and the output layer is to construct the input of the next RBM. Table 1 lists the main symbols and their definitions in our paper.

2.1.1 Restricted Boltzmann machines

Boltzmann machine (BM) is proposed in 1986 based on random neural networks (RNN) proposed by Hinton and Sejnowski [31]. The output of the random neurons in RNN has two probability-determined states, active or inactive, which are often represented by 0 and 1. BM has great unsupervised learning ability to learn complex data structures. However, it demands long training time to achieve the ability. To overcome this drawback, Smolensky proposed RBM [32]. As shown in Fig. 1, RBM can be

Table 1 Main symbols' definition

Symbol	Definition	Symbol	Definition
n	Number of visible units in one RBM	h_n	DBN output layer
m	Number of hidden units in one RBM	x	Input vector
v_i	Text feature state of visible unit i	$p(y = j\|x)$	Probability of each x belonging to each category j
h_j	Text feature state of hidden unit j	k	Number of Categories
a_i	Visible units bias	s_θ	Function parameter
b_j	Hidden units bias	sw_{ij}	Weight of softmax regression
w_{ij}	Weight between visible unit and hidden unit	sb_j	Bias of output layer
$\delta(x)$	Activation probability	$x^{(i)}$	Output of DBN
ε	Learning rate	m	Number of training samples
Δw_{ij}	Updating value of the weight	$y_j^{(i)}$	Label belonging to j class for the i-th sample
$\Delta \mathrm{a}_i$	Updating value of visible units	$h_{s\theta j}$	Output of softmax regression
Δb_j	Updating value of hidden units	λ	Penalty factor
$<.>_{\text{data}}$	Values of visible units i multiplied by hidden units j before reconstruction	θ_p	Weight of the RBMs in DBN
$<.>_{\text{recon}}$	Values of visible units i multiplied by hidden units j after reconstruction	tf	Term frequency
D	Training set	idf	Inversed document frequency
$z^{(i)}$	Training data	N	Number of text
$y^{(j)}$	Label of training data	n_f	Number of the feature appearing in texts
$p(y = k\|z)$	probability of each z belonging to each category k	$W(f, \vec{d})$	Word f weight in document $\vec{d}$
v	Input layer	$tf(f, \vec{d})$	Word f term frequency in document $\vec{d}$

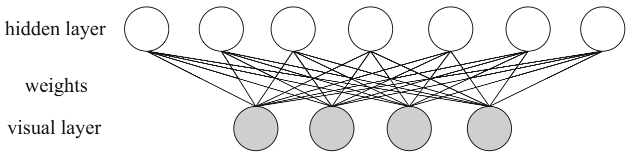

Fig. 1 RBM model

regarded as an undirected graph model, which consists of a visible layer, a hidden layer, and connection weights between the two layers.

2.1.2 Training RBM

Let n be the number of visible units and m the number of hidden units in a RBM. Visible units are the input of RBM, which is represented by the vector v. The output is the hidden units represented by the vector h. For a given state (v, h), the energy function is defined as follows [33]:

$$E(v,h) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} v_i W_{ij} h_j \tag{1}$$

where $\theta = \{w_{ij}, a_i, b_j\}$ are the parameters, v_i and h_j are text feature states of visible unit i and hidden unit j, a_i, b_j are their biases and w_{ij} is the weight between them. To fit the given training data, RBM training is demanded to optimize E. DBN is composed of multi-RBM units, and the outputs of a RBM's hidden units are the inputs of the visible units of the next RBM. When we extract features from data, the output of the hidden layer units is its activation probability and the reconstruction value of the visible layer is its activation probability of reconstruction. They are calculated by using the sigmoid function.

$$\delta(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

In many application domains, to obtain the connection weights, the contrastive divergence (CD) learning with K (or CD-K) has been shown to work quite well [34, 35]. The updating methods for the parameters are as follows:

$$\Delta W_{ij} = \varepsilon\left(\left\langle v_i h_j \right\rangle_{\text{data}} - \left\langle v_i h_j \right\rangle_{\text{recon}}\right) \tag{3}$$

$$\Delta a_i = \varepsilon\left(\left\langle v_i \right\rangle_{\text{data}} - \left\langle v_i \right\rangle_{\text{recon}}\right) \tag{4}$$

$$\Delta b_j = \varepsilon\left(\left\langle h_j \right\rangle_{\text{data}} - \left\langle h_j \right\rangle_{\text{recon}}\right) \tag{5}$$

where ε is a learning rate, ΔW_{ij} is the updating value of the weight, Δa_i and Δb_j are updating values of their biases, $<.>_{\text{data}}$ represents the values of visible units i multiplied by hidden units j before reconstruction, and $<.>_{\text{recon}}$ represents the values after, which reflects the distribution of the reconstructed model. To obtain high efficiency, we select CD-1 learning strategy in our model, where the

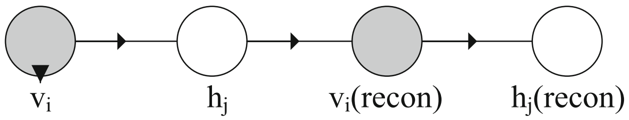

Fig. 2 CD-1 processes

reconstruction is only computed once. These calculation processes are shown in Fig. 2.

Because of its excellent feature learning performance, DBN has been combined with SVM or KNN to promote their classification accuracy [36, 37]. The advantage of such dissevered model is that it can be quickly established. However, different models work independently, which makes it difficult to obtain global optimized parameters and fine-tune the overall system. Moreover, the classification accuracy of the "hard-hybrid" models highly depends on DBN results especially for high-dimensional data; however, the classification error cannot be fed back to DBN model. In other words, DBN can rarely take advantage in the training data, which is often of great value. If we can use a portion of the labeled data for fine-tuning in the training process of DBN-based classification model, both the feature learning and classification results can be further improved.

2.2 Softmax regression

Softmax regression is generated from logistic regression for multi-classification problems. Since it is easy to perform, it has been widely used in practical applications, such as MNIST digit classification.

Let $D = \{(z^{(1)}, y^{(1)}), \ldots, (z^{(n)}, y^{(n)})\}$ be the training set, $z^{(i)}(i \in \{1, 2, \ldots, n\})$ represents the training data, $y^{(j)}(j \in \{1, 2, \ldots, n\})$ represents the label of training data. Given a sample z, to estimate the probability $p(y = k|z)$, the softmax regression classification model is shown in Fig. 3. Softmax regression is composed of input, classifier and output. From Fig. 3, it can be seen that softmax regression and DBN can be seamlessly connected as a soft-hybrid system, which can make full use of labeled samples and achieve much higher accuracy with global optimization.

3 The hybrid model of deep belief network and soft regression

As the aforementioned introduction, for classification problem, if DBN could use a portion of the labeled data for fine-tuning in the training process, we could obtain a more accurate model with global optimization. Therefore, we proposed a novel hybrid model of DBN and soft regression, which is shown in Fig. 4.

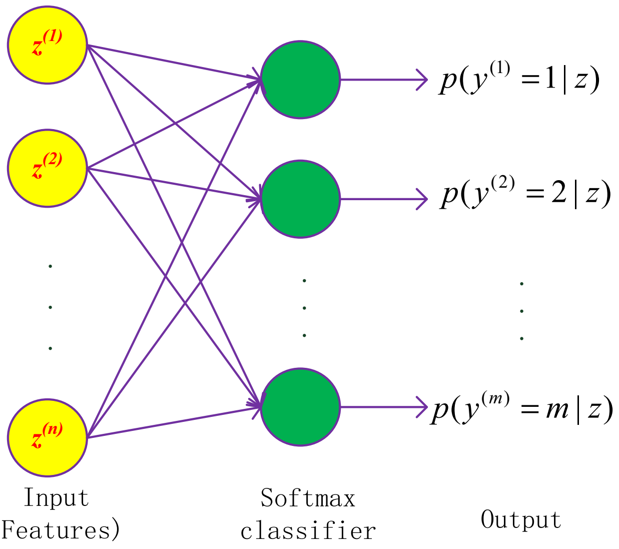

Fig. 3 Softmax regression model

Both unlabeled and a few labeled data are used to train the model, in which unlabeled data are used to train DBN model and labeled data are used to train softmax regression model and fine-tune the coherent whole system.

The input layer v is the text samples, and the output of h_n is the feature learning results produced by DBN. And the output of layer h_n is the input of softmax regression. Therefore, for the new classification model, DBN is trained first, and then, partial labeled data are used to train softmax regression. In other words, the two components of the hybrid model can be trained, respectively. This strategy feeds softmax regression initial weights instead of randomly generated. Finally, the whole system can be seen as a deeper neural network for fine-tuning to optimize.

For softmax regression training, the input is x, the hypothesis function is to compute the probability $p(y = j|x)$ for each x belong to each category j. For k classification problem, it is assumed that the output is k dimensional vector. In order to distinguish the parameters of DBN, the function parameter is defined as s_θ, and the output of softmax regression is as follows [38]:

$$h_{s\theta} = \begin{bmatrix} p(y^{(i)} = 1)|x^{(i)}; & s\theta \\ p(y^{(i)} = 2)|x^{(i)}; & s\theta \\ \vdots & \\ p(y^{(i)} = k)|x^{(i)}; & s\theta \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{s\theta_j^T x^{(i)}}} \begin{bmatrix} e^{s\theta_1^T x^{(i)}} \\ e^{s\theta_2^T x^{(i)}} \\ \vdots \\ e^{s\theta_k^T x^{(i)}} \end{bmatrix} \tag{6}$$

where s_θ includes the weights of softmax regression sw_{ij} and the bias of output layer sb_j. From the perspective of the whole model, $x^{(i)}$ is the output of DBN. Then, the cost function definition is defined as follows [38].

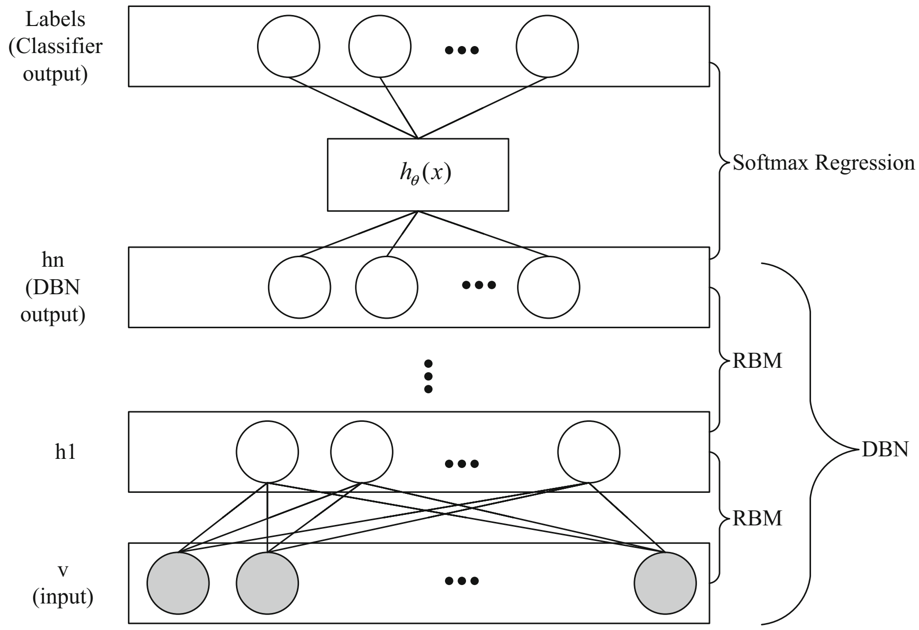

Fig. 4 Hybrid model of deep belief network and soft regression

$$J(s\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} y_j^i \log(h_{s\theta j}) + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}(s\theta_{ij})^2 \qquad (7)$$

where k is the number of categories, m is the number of training samples, $y_j^{(i)}$ is the label which belongs to j class for the i-th sample, $h_{s\theta j}$ is the output of softmax regression, $s_{\theta ij}$ is parameters of the model, the followed term is a penalty to punish large parameter, λ is penalty factor. In Eq. (7), only if $i = j$, $y_j^{(i)}$ is 1, otherwise, $y_j^{(i)}$ is 0. Then the first term in Eq. (7) equals 0.

When DBN and softmax regression training is completed, respectively, we use labeled data to fine-tune all the parameters. During fine-tuning stage, the hybrid model can be seen as a whole, which is similar to a back propagation neural network with more hidden layers [39, 40]. The process of fine-tuning is to achieve a global minimum for the cost function of the whole model. In our experiments, the gradient descent algorithm and L-BFGS algorithm are both used to optimize the global cost function.

During the fine-tuning process, the system cost function is as follows:

$$J(s\theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} y_j^i \log(h_{s\theta j}) + \frac{\lambda}{2}\sum_{i=1}^{k}\sum_{j=0}^{n}(s\theta_{ij})^2 + \frac{\lambda}{2}\sum_{p=1}^{q-1}\mathrm{\theta}_p^2 \qquad (8)$$

where the first two terms in the equation are the same as softmax regression, and the last one is a penalty in the DBN model; θ_p indicates the weight of the RBMs in DBN.

4 Experimental results

To validate the performance of the new proposed model, we implement experiments on the two benchmark text data Reuters-21,578 and 20-Newsgroup. Moreover, the classical algorithms KNN and SVM with polynomial kernel (SVM) are used to compare with the DBN + Softmax(1) and DBN + Softmax(2), which are optimized by gradient descent algorithm and Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS), respectively.

4.1 Description of datasets

Reuters-21,578 is collected from the Reuters newswire in 1987, and the original collection has 21,578 samples which belong to 135 classes. We used “ModApte” version [41]. It removed multi-labeled documents and contains 8293 samples belonging to 65 classes. There are 5946 testing documents, 2347 training documents, and 18,933 different words in the corpus.

20-Newsgroups corpus contains 18,845 postings taken from the Usenet newsgroup collection. This corpus is partitioned fairly into 20 different newsgroups, and each group is corresponding to a different topic. The data were split by date into 11,314 training and 7531 test articles [42].

For these two corpuses, after the preprocessing of removing stop words, stemming, selected the top 2000 most frequently used words in the training and testing dataset. Each document corresponds to a 2000-dimensional vector, and each vector contains the number of word frequency in the document. Then, we use TFIDF method to calculate the weight for each word in each article, which is the input of DBN model. The TFIDF method considers three factors as follows:

1. Term frequency (*tf*): The times of the term occurs in a document, usually called *tf* factor.
2. Inversed document frequency (*idf*): It is a measure of how much information the word provides, that is, whether the term is common or rare across all documents in a finite dataset. Commonly used equation is log ($N/n_f + 0.01$), where N is the total number of text, n_f is the number of the feature appears in texts.
3. Normalization factor: Each component of the text vector is normalized.

According to the three factors mentioned above, the TFIDF weights equation can be drawn

$$W(f,\vec{d}) = \frac{tf(f,\vec{d}) \times \log(N/n_f + 0.01)}{\sum_{f\in\vec{d}}\left[tf(f,\vec{d}) \times \log(N/n_f + 0.01)\right]^2} \tag{9}$$

where $W(f,\vec{d})$ and $tf(f,\vec{d})$ are the word f weight and term frequency, respectively in document $\vec{d}$, N is the total number of documents, n_f is the number that f appears in the document, and the denominator is a normalization factor.

In our experiments, in order to test the performance on different scales of datasets, we take out different numbers of documents from datasets, and check the training effectiveness. At the same time, we use classification accuracy and the changes in mean square error to evaluate the performance of the model. The result of the proposed method is compared with those from the classical SVM and KNN. The experimental framework is shown in Fig. 5.

4.2 The experiment on tiny Reuters

In this experiment, the number of document category is 10, the quantity of documents is 600, in which training set is 440 and testing set is 160. We select 40 labeled documents (9.09 %) from the training set to train softmax regression. The typological structure of DBN is 2000-1000-10. In the training process of DBN, the iteration of RBM is set to 100. In the fine-tuning step, 300 labeled documents are used. To check and compare their effectiveness, both the gradient descent and L-BFGS algorithm are used to minimize the cost function. The results are as follows:

In Table 2, DBN + Softmax(1) represents the model optimized by gradient descent algorithm for fine-tuning and DBN + Softmax(2) indicates the model adopting L-BFGS. After the fine-turning, DBN + Softmax(2) gets an accuracy of 82.50 %, which increased 36.2 % and get 78.38 % higher than KNN. Moreover, it achieves 2.36 % higher than SVM. Through the comparisons of these two classical algorithms, it can be seen that the new proposed model is superior to them on classification accuracy.

On the other hand, for the two fine-tuning strategy i.e., gradient descent algorithm and L-BFGS, DBN + Softmax(2) increased 29.38 % compared with DBN + Softmax(1), which is 55.31 % higher than the latter one. From their comparison, we can find that at first with the same physical structure, L-BFGS is much better than gradient descent algorithm; secondly, L-BFGS is one of quasi-Newton methods, which uses an estimation to the inverse Hessian matrix and considers second-order Taylor expansion for the cost function. On the contrary, gradient descent is a first-order method which takes an indirect route.

In Fig. 6, the blue line depicts the error rate when we use gradient descent algorithm for fine-tuning, while red curve is L-BFGS. In the fine-tuning process, when we use gradient descent algorithm, the error rate decreases slowly before 150 iterations. From 150 to 300, it declines faster; however, it fluctuates sharply. More importantly, even it has reached 300 iterations, the DBN + Softmax (1) is still oscillating, and that is why it gets worse results. In other words, DBN + Softmax (1) gets stuck at a local minima.

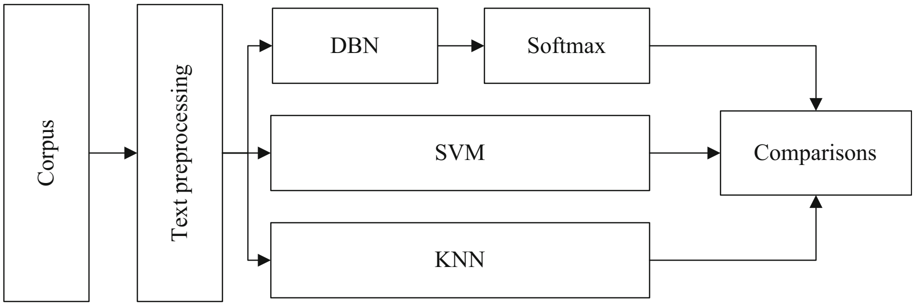

Fig. 5 Framework of experiments

Table 2 Tiny Reuters classification accuracy (%)

Model	KNN (%)	SVM (%)	DBN + Softmax(1) (%)	DBN + Softmax(2) (%)
Accuracy	46.25	80.60	53.12	82.50

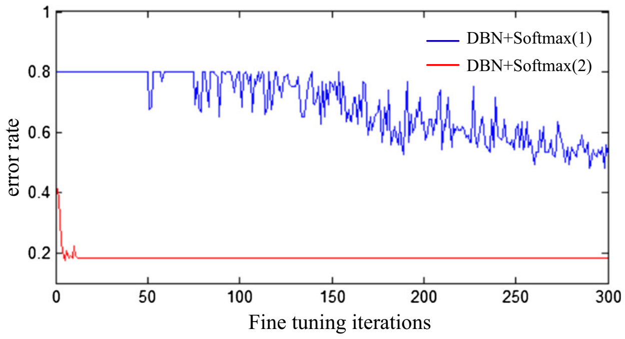

Fig. 6 Error rate comparison

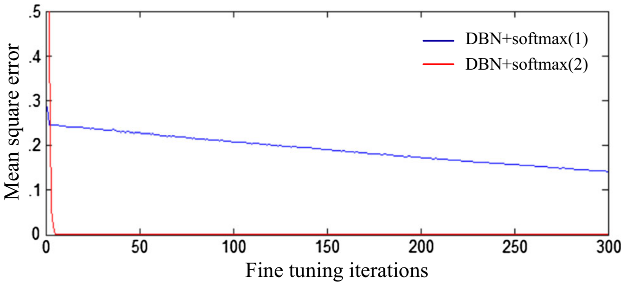

Fig. 7 Mean square error comparison

On the contrary, the error rate of DBN + Softmax(2) with L-BFGS algorithm drops down quickly from the beginning. After a few fluctuations before 12 iterations, the fine-tuning process reaches the minimum error rate and keeps steady with the iterations increasing.

In the fine-tuning process, if we use the mean square error, the learning tendency is more obvious. In Fig. 7, the blue curve indicates the mean square error line of DBN + Softmax(1) fine-tuning using gradient descent algorithm. The red curve is the DBN + Softmax(2) using L-BFGS. From the figure, it can be seen that the algorithm with L-BFGS declined much faster than the one with gradient descent algorithm. The former has converged in a very short time (12 iterations), but for gradient descent algorithm, the mean square error decreases slowly and has not converged after 300 iterations with high error rate.

Through the above comparison, it can be seen that the new proposed hybrid algorithm can be used for text classification and achieve satisfactory results. For the fine-tuning optimization, L-BFGS algorithm is more suitable for DBN + Softmax model. It can effectively improve the classification accuracy and converge faster than gradient descent algorithm.

4.3 The experiment on large scale Reuters

In the large-scale datasets, the number of documents is 6000, training set is set to 4000, and testing set is 1600. We use 400 labeled documents (6.67 %) from the training set to train softmax regression. Then, we use 800 (13.33 %) labeled documents to fine-tune the whole model. The system structure is 2000-1000-100-10, and the cost function minimizing methods are the same as small scales.

From Table 3, it can be inferred that compared with small-scale datasets experiment, accuracies of all the four algorithms are improved. However, DBN + Softmax(2) still achieves the highest accuracy 86.88 %, which is 40.40 % higher than KNN, 4.44 % higher than SVM, and 41.84 % than DBN + Softmax(1), respectively.

In Fig. 8, the blue curve is the error rate using gradient descent algorithm for fine-tuning and the red curve is the error rate of L-BFGS. From Fig. 8, it can be seen that when we use gradient descent algorithm during fine-tuning, the error rate has not changed a lot from beginning to 300 iterations. It shows that the accuracy rate is not significantly improved. For the L-BFGS optimized model, the error rate goes down significantly at start-up. Its fine-tuning converges at 13.12 % after the 27th iteration and achieved the minimum error rate.

The mean square error of the comparison is shown in Fig. 9. The blue curve uses gradient descent algorithm for fine-tuning, and red curve is the L-BFGS algorithm's mean square error. From Fig. 9, it can be seen that the mean square error got a small declination at the very beginning and then went steadily at a high mean square error. In other words, DBN + Softmax using gradient descent optimized at a local minimum. By contrast, the mean square error of the hybrid algorithm with L-BFGS decreases fast and converges at much small value (near to zero).

From the experiments on Reuters, we can find that the hybrid algorithm DBN + Soft Regression (2) can achieve higher classification accuracy than other models such as KNN and SVM. In addition, because the gradient descent takes an indirect route to optimize in searching space and may fall into local minimum, it can be seen that L-BFGS algorithm is more suitable for fine-tuning, which could effectively improve the classification accuracy of the system and converge fast.

Table 3 Large-scale Reuters Classification Accuracy (%)

Model	KNN (%)	SVM (%)	DBN + Softmax(1) (%)	DBN + Softmax(2) (%)
Accuracy	61.88	83.19	61.25	86.88

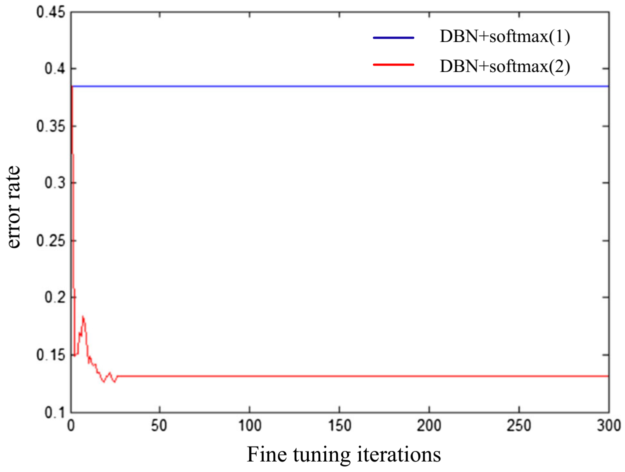

Fig. 8 Error rate comparison

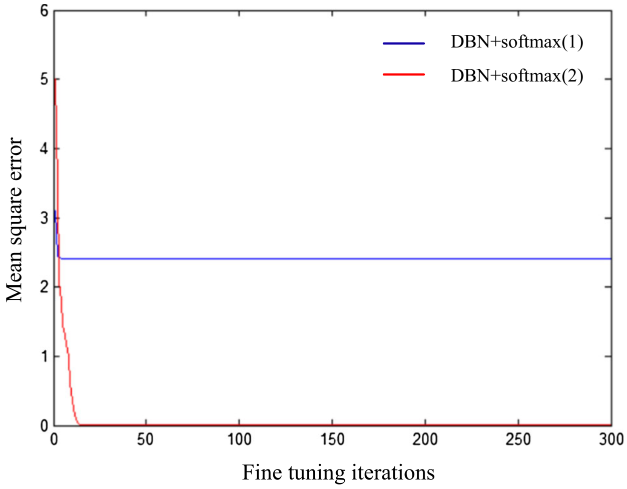

Fig. 9 Mean square error comparison

4.4 20-Newsgroups experiment

For 20-Newsgroups datasets, we use eight different scale datasets to verify the performance of the proposed text classification method. The classification accuracies of the four algorithms are shown in Table 4, which also includes the architecture and labeled samples for fine-tuning.

Table 4 shows that with the increasing size of the dataset, the descending order of the classification accuracy is DBN + Softmax(2), SVM, DBN + Softmax(1), and KNN. For example, on the 5000 training samples data, DBN + Softmax(2) is 37.37 % higher than KNN, 8.51 % higher than SVM, and 24.54 % higher than DBN + Softmax(1). The results are the same as what we got on Reuter-21,578 datasets. Moreover, the L-BFGS algorithm is more effective in the fine-tuning process of the novel hybrid model. In addition, we can find that the new proposed model only uses no more than 50 % labeled documents (27.27 % for the last experiment 11,000 training data), but achieves higher accuracy than KNN and SVM which employ all the training samples.

In addition, to compare our hybrid model with other deep networks, we borrow the encoder-based deep network results from the article in the 25th ICML [43]. It also employs the 20-Newsgroups and Reuters-21,578 corpus. Take 20-Newsgroup as example, the encoder + SVM in [43] get the accuracy 76.3 % which is 8.30 % lower than DBN + Softmax(2).

5 Conclusion

This paper presents a novel hybrid model based on DBN and softmax regression. Although DBN completes the feature learning to solve the high dimension and sparse matrix problem and softmax regression is employed to classify the texts, they will work as a coherent whole model with fine-tuning and achieve the final results. With a few labeled samples ($\leq$50 % of the train data) the new proposed algorithm could get higher accuracy (e.g. 8.51 % higher than SVM) than the classical KNN and SVM on the whole training sets. Moreover, the proposed method uses L-BFGS algorithm to optimize the weights, which is one of

Table 4 20-Newsgroup test results

Testing data	KNN (%)	SVM (%)	DBN + Softmax(1) (%)	DBN + Softmax(2) (%)	Unlabeled training data	Labeled data in fine-tuning	Architecture
750	57.33	74.67	65.33	81.33	1000	500	2000-1000-20
1500	58.67	78.00	62.73	80.13	2000	900	2000-1000-20
2000	56.50	79.50	66.25	82.00	3000	1000	2000-1000-20
3000	56.67	80.00	61.83	83.07	4000	1500	2000-1000-20
3500	62.29	78.86	68.71	85.57	5000	1500	2000-1000-20
4500	60.78	82.33	69.56	83.38	6000	1800	2000-1000-500-20
5000	63.20	80.60	71.50	82.40	7000	2200	2000-1000-500-20
7500	61.87	81.60	65.33	82.63	11,000	3000	2000-1000-500-20

quasi-Newton methods and uses estimation to the inverse Hessian matrix. It can avoid optimizing in indirect route and early trap into local minimum.

Acknowledgments This work is supported by the National Natural Science Foundation of China (61163034, 61373067, 61572228, 61272207, 61472158), the 321 Talents Project of the two level of Inner Mongolia Autonomous Region (2010), the Inner Mongolia Talent Development Fund (2011), the Natural Science Foundation of Inner Mongolia Autonomous Region of China (2016MS0624), the Research Program of Science and Technology at Universities of Inner Mongolia Autonomous Region (NJZY16177), and Science and Technology Development Program of Jilin Province (20140101195JC, 20140520070JH, 20160101247JC).

References

1. Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18(7):1527–1554. doi:10.1162/neco.2006.18.7.1527
2. Deng L, Li X (2013) Machine learning paradigms for speech recognition: an overview. IEEE Trans Audio Speech Lang Process 21(5):1060–1089. doi:10.1109/TASL.2013.2244083
3. Sivaram G, Hermansky H (2012) Sparse multilayer perceptron for phoneme recognition. IEEE Trans Audio Speech Lang Process 20(1):23–29. doi:10.1109/TASL.2011.2129510
4. Yu D, Wang S, Karam Z, Deng L (2010) Language recognition using deep-structured conditional random fields. Acoust Speech Signal Process 41(3):5030–5033. doi:10.1109/ICASSP.2010.5495072
5. Dahl G, Yu D, Deng L, Acero A (2011) Large vocabulary continuous speech recognition with context-dependent DBN-HMMS. In: Proceedings of international conference on acoustics, speech and signal processing, pp 4688–4691. doi:10.1109/ICASSP.2011.5947401
6. Krizhevsky A, Sutskever I, Hinton G (2012) ImageNet classification with deep convolutional neural networks. Neural Inf Process Syst 25(2):1106–1114
7. Lawrence McAfee (2008) Document classification using deep belief nets. http://nlp.stanford.edu/courses/cs224n/2008/reports/10. Accessed 4 June 2008
8. Liu T (2010) A novel text classification approach based on deep belief network. In: Proceedings of the 17th international conference on neural information processing, pp 314–321. doi:10.1007/978-3-642-17537-4_39
9. Hinton GE, Salakhutdinov R (2011) Discovering binary codes for documents by learning deep generative models. Top Cogn Sci 3(1):74–91. doi:10.1111/j.1756-8765.2010.01109.x1
10. Huang CC, Gong W, Fu WL, Feng DY (2014) A research of speech emotion recognition based on deep belief network and SVM. Math Probl Eng 2014(2014):1–7. doi:10.1155/2014/749604
11. Zhou S, Chen Q, Wang X (2014) Active semi-supervised learning method with hybrid deep belief networks. PLoS One 9(9):e107122. doi:10.1371/journal.pone.0107122
12. Yang YM (1999) An evaluation of statistical approaches to text categorization. Inf Retr 1(1):69–90. doi:10.1023/A:1009982220290
13. Sebastiani F (2002) Machine learning in automated text categorization. ACM Comput Surv 34(1):1–47. doi:10.1145/505282.505283
14. Chakrabarti S, Roy S, Soundalgekar M (2003) Fast and accurate text classification via multiple linear discriminant projections. VLDB J 12(2):170–185. doi:10.1007/s00778-003-0098-9
15. Wu H, Phang TH, Liu B, Li X (2002) A refinement approach to handling model misfit in text categorization. In: Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining, pp 207–216. doi:10.1145/775047.775078
16. Gu B, Sheng VS, Tay KY, Romano W, Li S (2015) Incremental support vector learning for ordinal regression. IEEE Trans Neural Netw Learn Syst 26(7):1403–1416. doi:10.1109/TNNLS.2014.2342533
17. Tan S, Cheng X, Wang B, Xu H, Ghanem MM, Guo Y (2005) Using dragpushing to refine centroid text classifiers. In: Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval, pp 653–654. doi:10.1145/1076034.1076174
18. Debole F, Sebastiani F (2004) An analysis of the relative hardness of reuters-21578 subsets. J Am Soc Inf Sci Technol 56(6):584–596. doi:10.1002/asi.20147
19. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features. In: 10th european conference on machine learning, Chemnitz, Germany, pp 137–142. doi:10.1007/BFb0026683
20. Gu B, Sheng VS (2016) A robust regularization path algorithm for ν-support vector classification. IEEE Trans Neural Netw Learn Syst. doi:10.1109/TNNLS.2016.2527796
21. Lewis DD, Li F, Rose T, Yang Y (2004) RCV1: a new benchmark collection for text categorization research. J Mach Learn Res 5(2):361–397. doi:10.1145/122860.122861
22. Forman G, Cohen I (2004) Learning from little: Comparison of classifiers given little training. In: 8th European conference on principles and practice of knowledge discovery 3203, pp 161–172. doi:10.1007/978-3-540-30116-5_17
23. Gu B, Sun XM, Sheng VS (2016) Structural minimax probability machine. IEEE Trans Neural Netw Learn Syst. doi:10.1109/TNNLS.2016.2544779
24. Zheng W, Qian Y, Lu H (2013) Text categorization based on regularization extreme learning machine. Neural Comput Appl 22(3–4):447–456. doi:10.1007/s00521-011-0808-y
25. Wang W, Yu B (2009) Text categorization based on combination of modified back propagation neural network and latent semantic analysis. Neural Comput Appl 18(8):875–881. doi:10.1007/s00521-008-0193-3
26. Wu S, Er MJ (2000) Dynamic fuzzy neural networks: a novel approach to function approximation. IEEE Trans Syst Man Cybern 30(2):358–364. doi:10.1109/3477.836384
27. Er MJ, Wu S, Lu J, Toh HL (2002) Face recognition using radial basis function (RBD) neural networks. IEEE Trans Neural Netw 13(3):697–710. doi:10.1109/CDC.1999.831240
28. Chen W, ER MJ, Wu S (2006) Illumination compensation and normalisation for robust face recognition using discrete cosine transform on logarithm domain. IEEE Trans Syst Man Cybern Part B Cybern A Publ IEEE Systems Man Cybern Soc 36(2):458–66. doi:10.1109/TSMCB.2005.857353
29. Larochelle H, Bengio Y, Louradour J et al (2009) Exploring strategies for training deep neural networks. J Mach Learn Res 10(10):1–40. doi:10.1145/1577069.1577070
30. Guan R, Shi X, Marchese M, Yang C, Liang Y (2011) Text clustering with seeds affinity propagation. IEEE Trans Knowl Data Eng 23(4):627–637. doi:10.1109/TKDE.2010.144
31. Hinton G E, Sejnowski T (1986) Learning and relearning in Boltzmann machines. In: Parallel distributed processing: explorations in the microstructure of cognition. vol 1. Foundations, MIT Press, Cambridge, MA, pp 282–317

32. Smolensky P (1986) Information processing in dynamical systems: foundations of harmony theory. In: Parallel distributed processing: explorations in the microstructure of cognition, vol 1. Foundations, MIT Press, Cambridge, MA, pp 194–281
33. Hinton GE (2010) A practical guide to training restricted boltzmann machines. Neural Netw: Tricks Trade 9(1):599–619. doi:10.1007/978-3-642-35289-8_32
34. Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Comput 14(8):1771–1800. doi:10.1162/089976602760128018
35. Sarikaya R, Hinton GE, Deoras A (2014) Application of deep belief networks for natural language understanding. IEEE/ACM Trans Audio, Speech Lang Process 22(4):778–784. doi:10.1109/TASLP.2014.2303296. DOI: 10.1109/TNN.2005.844909
36. Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297. doi:10.1007/BF00994018
37. Altman N (1992) An introduction to kernel and nearest-neighbor nonparametric regression. Am Stat 46(3):175–185. doi:10.1080/00031305.1992.10475879
38. Ng A, Ngiam J, et al (2013) UFLDL tutorial. IOP Stanford. http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial. Accessed 7 Apr 2013
39. Wu S, Er MJ, Gao Y (2001) A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. IEEE Trans Fuzzy Syst 9(4):578–594. doi:10.1109/CDC.1999.831240
40. Er MJ, Chen W, Wu S (2005) High-speed face recognition based on discrete cosine transform and RBF neural networks. IEEE Trans Neural Netw 16(3):679–691. doi:10.1109/TNN.2005.844909
41. Joachims T (1999) Making large-scale support vector machine learning practical. In: Schölkopf B, Burges CJC, Smola AJ (eds) Advances in Kernel methods-support vector learning, chapter 11. MIT Press, Cambridge, pp 169–184
42. Salakhutdinov R (2009) Learning deep generative models. Annu Rev Stat Appl 2(1):74–91. doi:10.1146/annurev-statistics-010814-020120
43. Ranzato MA, Szummer M (2008) Semi-supervised learning of compact document representations with deep networks. In: Proceedings of the twenty-fifth international conference, pp 792–799. doi:10.1145/1390156.1390256