Knowledge-Based Systems 21 (2008) 900–904

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Latent semantic analysis for text categorization using neural network

Bo Yu [a,*], Zong-ben Xu [b], Cheng-hua Li [c]

[a] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[b] Institute for Information and System Science, School of Science, Xi'an Jiaotong University, Xi'an 710049, China
[c] Department of Information & Communication Engineering, Chonbuk National University, Chonju 561756, South Korea

ARTICLE INFO

Article history:
Received 18 December 2007
Accepted 30 March 2008
Available online 4 April 2008

Keywords:
Latent semantic analysis
Neural network
Text categorization

ABSTRACT

New text categorization models using back-propagation neural network (BPNN) and modified back-propagation neural network (MBPNN) are proposed. An efficient feature selection method is used to reduce the dimensionality as well as improve the performance. The basic BPNN learning algorithm has the drawback of slow training speed, so we modify the basic BPNN learning algorithm to accelerate the training speed. The categorization accuracy also has been improved consequently. Traditional word-matching based text categorization system uses vector space model (VSM) to represent the document. However, it needs a high dimensional space to represent the document, and does not take into account the semantic relationship between terms, which can also lead to poor classification accuracy. Latent semantic analysis (LSA) can overcome the problems caused by using statistically derived conceptual indices instead of individual words. It constructs a conceptual vector space in which each term or document is represented as a vector in the space. It not only greatly reduces the dimensionality but also discovers the important associative relationship between terms. We test our categorization models on 20-newsgroup data set, experimental results show that the models using MBPNN outperform than the basic BPNN. And the application of LSA for our system can lead to dramatic dimensionality reduction while achieving good classification results.

1. Introduction

With the rapid development of the web, large numbers of electronic documents are available on the Internet. Text categorization becomes a key technology to deal with and organize large numbers of documents.

In recent years, there have been extensive study and rapid progress in automatic text categorization, including the traditional machine learning approaches such as Bayesian [8,9], decision tree [11], and *k*-nearest neighbor classifier [10]. Even these methods are not so significant compared to those new approaches, however, they have the advantages of simple algorithms and relatively high efficiency, the modified and improved methods based on these traditional approaches are continuing to cause concern.

Some new approaches and models have been proposed recently, for example, maximum entropy models [4,17] and fuzzy theory based approaches [5,20] also have good results.

The application of SVM is one of the important progresses in text categorization. SVM is very popular and has been proved to be one of the best algorithms for text categorization [13,16]. Compared to the other approaches, SVM [12,15] has the superiority of stability and the ability of generalization which can overcome some negative affects of skewed distribution of the samples as well as overfitting. Neural network is also a popular classification method, it can handle linear and nonlinear problems for text categorization, and both of linear [2] and nonlinear [7] classifier can achieve good results.

Text representation is an important process to perform text categorization. A major problem of text representation is the high dimensionality of the feature space. The feature space with a large number of terms is not only unsuitable for neural networks but also easily to cause the overfitting problem. The ambiguity meaning of terms can also prohibit the classifier to choose the categories deterministically, and will directly decrease the categorization accuracy. Latent semantic analysis [3] uses singular value decomposition (SVD) technique to decompose a large term-document matrix into a set of k orthogonal factors, it is an automatic method that can transform the original textual data to a smaller semantic space by taking advantage of some of the implicit higher-order structure in associations of words with text objects [18]. These derived indexing dimensions, rather than individual words, can greatly reduce the dimensionality and have the semantic relationship between terms. So, even two documents don't have any common words, we also can find the associative relationship between them, because the similar con-

* Corresponding author. Tel./fax: +86 10 68949443.
E-mail addresses: xjtuyubo@gmail.com (B. Yu), zbxu@mail.xjtu.edu.cn (Z.-b. Xu), lchjk@msn.com (C.-h. Li).

0950-7051/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2008.03.045

texts in the documents will have similar vectors in the semantic space.

Latent semantic analysis has been applied to text categorization in many previous works, Yang [14] used SVD for noise reduction so as to improve the computational efficiency in text categorization, Zelikovitz and Hirsh [6] performed LSA expanded term-by-document matrix in conjunction with background knowledge in text categorization, more recently, the supervised LSA [1] has been proposed to improve the performance in text categorization.

The remainder of this paper is organized as follows: The detail algorithm of neural networks is described in Section 2. Section 3 explains how to generate the semantic vector space for text categorization. Experiment results are given in Section 4. Conclusions are given in Section 5.

2. Basic BPNN and modified BPNN algorithms

2.1. Basic BPNN algorithm

The back-propagation neural network plays an important role in the neural networks as a tool to solve wide kinds of problems. In general, these feed-forward-nets consist of at least three layers (one input, one output, and at least one hidden layer) and use back-propagation as learning mechanism. The structure of the three layered back-propagation neural network is shown in Fig. 1.

The training of a network by back-propagation involves several stages: calculation of input and output value, calculation of activation function, target function and back-propagation of the associated error, the adjustment of the weight and the biases. These stages are explained in detail as follows:

2.1.1. Input and output calculation

For each neuron j, the input I_j and output O_j are defined as:

$$I_j = \sum_i w_{ij} O_i \tag{1}$$

$$O_j = f(I_j + \theta_j) \tag{2}$$

where w_{ij} is the weight of the connection from the i^{th} neuron in the previous layer to the neuron j,$f(I_j + \theta_j)$ is activation function of the neurons, and the O_i,O_j are the output of previous neuron i and the neuron j,θ_j is the biases input to the neuron.

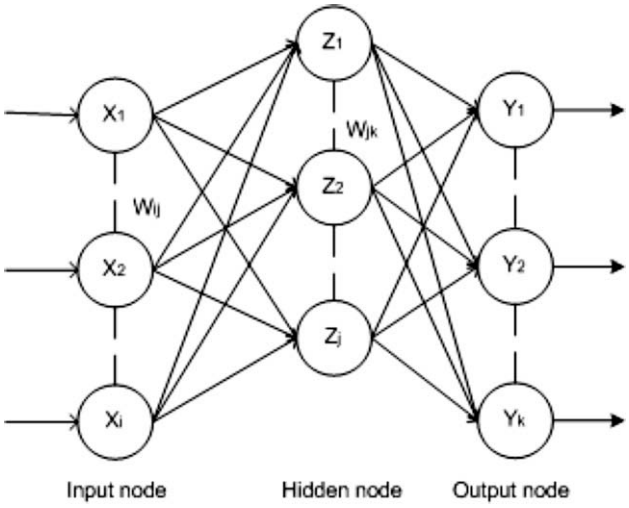

Fig. 1. Typical three layered back-propagation neural network.

2.1.2. Activation and target function

We use bipolar sigmoid activation function in our networks. And the activation function is defined as formula (3). The total absolute error E and mean absolute error in output layer can be calculated as formula (4) and (5):

$$f(x) = \frac{2}{(1+\exp(-x))} - 1 \tag{3}$$

$$E = \frac{1}{2}\sum_k \sqrt{(T_k - O_k)^2} \tag{4}$$

$$E_m = \frac{1}{2n}\sum_k \sqrt{(T_k - O_k)^2} \tag{5}$$

where n is the number of training patterns. The absolute error is used to evaluate the learning effects, the training will keep up until the absolute error falls below some threshold or tolerance level. Calculate the back-propagation error both in output layer δ_k and hidden layer δ_j as following formulas:

$$\delta_k = \lambda(T_k - O_k)f'(O_k) \tag{6}$$

$$\delta_j = \lambda\sum_i \delta_k w_{ji} f'(O_j) \tag{7}$$

where T_k is the target of the k^{th} output neuron, O_k is the actual output in the output layer, O_j is the actual output value in the hidden layer, and λ is the parameter of activation function. The back-propagation error is used to update the weights and biases both in output layer and hidden layer.

2.1.3. Weights and biases adjustment

The weights w_{ji} and biases θ_i can be adjusted as the following formulas:

$$w_{ji}(k+1) = w_{ji}(k) + \eta\delta_j y_i \tag{8}$$

$$\theta_i(k+1) = \theta_i(k) + \eta\delta_i \tag{9}$$

where k is the number of epochs, and η is the learning rate.

2.2. Modified BPNN algorithm

The learning process proceeds very quickly in the beginning, and can make rapid progress, however it slows down in the later stages [19]. We use two methods to improve the speed of training for BPNN.

2.2.1. Introduce momentum into the network

Convergence is sometimes faster if a momentum term is added to the weight update formulas. The weight update formula for a BPNN with momentum is

$$W_{ij}(k+1) = W_{ij}(k) + \eta\delta_i x_j + u(W_{ij}(k) - W_{ij}(k-1)) \tag{10}$$

where momentum parameter u is constrained to be in the range from 0 to 1. The new weights for the training step $k+1$ are based on the weights at training steps k and $k-1$.

2.2.2. Using adaptive learning rate to adjust the learning rate

The role of the adaptive learning rate is to allow each weight to have its own learning rate, and to let the learning rate vary with time as training progress. The formulas for a BPNN with an adaptive learning rate is

$$\eta^{(k+1)} = \eta^{(k)} \times \frac{E^{k-1}}{E^k} \tag{11}$$

where k is the epoch during the training process, and E is the absolute error in each epoch. When E decreases, the learning effect will increase (the weight may change to a greater extent). Otherwise, the learning effect will decrease.

These two kinds of methods accelerate the convergence of the BPNN to some extent.

3. Semantic vector space generation

Latent semantic analysis is a technique that projects the original high dimensional document vectors into a space with "latent" semantic dimensions. Once a term-by-document matrix is constructed, LSA requires the singular value decomposition of this matrix to construct a semantic vector space which can be used to represent conceptual term-document associations.

3.1. Singular value decomposition

From the training documents, we can get the term by document matrix $A(m \times n)$, it means there are m distinct terms in a n documents collection $m \geqslant n$. The singular value decomposition of A is defined as

$$A = U\Sigma V^{\mathrm{T}} \tag{12}$$

where U and V are the matrices of the term vectors and document vectors. $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ is the diagonal matrix of singular values.

3.2. Reduced vector space

For reducing the dimensions, we can simply choose the k largest singular values and the corresponding left and right singular vectors, the best approximation of A with rank-k matrix is given by

$$A_k = U_k \Sigma_k V_k^{\mathrm{T}} \tag{13}$$

where U_k is comprised of the first k columns of the matrix U and V_k^{T} is comprised the first k rows of matrix V^T, $\Sigma_k = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ is the first k factors, the matrix A_k captures most of the important underlying structure in the association of terms and documents while ignoring noise due to word choice.

3.3. LSA for information retrieval

When LSA is used for information retrieval [3,18], a query is represented as a vector in k-dimensional space in the same way with the document collection. And then the query is compared to the documents. The query is represented by

$$\hat{q} = q^T U_k \Sigma_k^{-1} \tag{14}$$

And each document is represented by

$$\hat{d} = d^T U_k \Sigma_k^{-1} \tag{15}$$

Once the query and the documents are represented, the relevance value between the query and documents can be computed using cosine similarity coefficient. Those documents exceeding some cosine threshold are returned as relevance to the query.

3.4. Latent semantic analysis for our system

LSA is originally proposed as an information retrieval method, nowadays, it is also widely used in text categorization.

When LSA is used for text categorization, we can refer to the document collection as the training examples and the query as a test example. However, our experimental results show that the traditional LSA method representation of training and test example done by multiplying the transpose of the document vectors of the training and test documents with matrices U_k and Σ_k^{-1} is not as good as by multiplying the single U_k matrix. In our experiment, we remove the Σ_k^{-1} matrix, so each training and test example is represented by

$$\hat{d} = d^{\mathrm{T}} U_k \tag{16}$$

Once all of the training and test examples are represented by this way, the new document vectors are performed by the neural network classifier, and the neural network algorithm will decide which category the test samples should be assigned into.

4. Experiments

4.1. Experiments design

In order to measure the performance of our system, in all of our experiments, we use a subset of the documents from the 20-newsgroup (20-news-18828 version) test collection for training and testing our text categorization model. We choose 1000 documents from 10 categories in the 20 news-group data set. About 600 documents are used for training and 400 documents are used for testing.

After word stemming, we merge the sets of stems from each of the 600 training documents and remove the duplicates. This results in a set of 12146 indexing terms in the vocabulary.

In order to create the set of initial feature vectors to represent the 600 training documents, we measure the term weight for each of the 12146 indexing terms. The feature vectors are then formed term weights, and each of the feature vectors is of the form

$$d_j = \langle w_{j,1}, w_{j,2}, \ldots\ldots\ldots w_{j,12146} \rangle \tag{17}$$

where $w_{j,i}$ is the term weight of the i^{th} indexing terms in document j.

Dimensionality of 12146 is too high for neural networks, so we reduce this size by choosing the highest term weights. Finally, 1200 terms are selected as the neural network's input based on our experience. So the reduced feature vector can be represented as:

$$d_j = \langle w_{j,1}, w_{j,2}, \ldots\ldots\ldots w_{j,1200} \rangle \tag{18}$$

In our experiments, we employ Porter's stemming algorithms for word stemming, and logarithmic function as the term weight

$$\mathrm{weight} = \log(1 + tf_{ij}) \tag{19}$$

where tf_{ij} is the i^{th} indexing term in document j.

When we apply the LSA, each document in the original feature vectors $(1 \times m)$ can transform to our desired k-dimensional vectors $(1 \times k)$ by

$$\underset{1\times k}{\hat{d}} = \underset{1\times m}{d^{\mathrm{T}}} \cdot \underset{m\times k}{U_k} \tag{20}$$

where k is the reduced dimensional, and m is original feature vectors.

4.2. Results and analysis

In our experiments, we compare the performance by varying the number of dimension k from 10, 15, 20, 30, 40, 50, 60, 80, 100, 120, 150, 200, 250, 300, 350 to 400, the neural network's input nodes number is equal to the dimension of the document vectors, for LSA, the dimensions is range from 10 to 400, and for VSM is 1200. So the input nodes for our neural networks are ranged from 10 to 1200. The number of hidden nodes, output nodes and some parameters used in our networks are shown as Table 1. Our neural networks will stop training after 500 epochs. The reason why we use 500 epochs as stop condition is that, from our experiments we find that 500 epochs can get the tradeoff between categorization accuracy and efficiency.

The performance of our categorization system is evaluated by precision, recall and micro-averaging F-measure. The value of mi-

Table 1
The size of the networks and parameters using in our networks.

Neural networks	#Input nodes	#Hidden nodes	#Output nodes
BPNN/MB PNN	SVD10–V SM 1200	15	10
Neural Networks	# Iterations	Learning Rate	Momentum Parameter
BPNN/MB PNN	500	0.01	0.8

cro-averaging *F*-measure is based on the value of precision and recall. In our experiment, we compare the performance of basic BPNN using LSA with dimension of 10–400, basic BPNN using VSM with dimension of 1200, the modified BPNN (MBPNN) using LSA with dimension of 10–400, modified BPNN using VSM with dimension of 1200. The categorization performance is given in Fig. 2 and the Computational time is given in Table 2.

Form the Fig. 2, we can see that the *F*-measure of BPNN using VSM with 1200 dimensions is 0.830. The *F*-measure of BPNN using LSA is enhanced when the number of dimensions is increased. When the dimensions are more than 80, the *F*-measure of BPNN using LSA is better than the *F*-measure of BPNN using VSM, and at dimension 200, it get the best performance of 0.862. The performance of MBPNN has been enhanced when compared to the performance of BPNN, even it is not so significant. The *F*-measure of MBPNN using VSM with 1200 dimensions is 0.839. When the dimensions are more than 70, the *F*-measure of MBPNN using LSA is better than the *F*-measure of MBPNN using VSM, and at dimensions of 250, it get the best performance of 0.870.

The computational time includes the training time and SVD decomposition time. In our experience, we perform the SVD on the OpenCV [20] (Open Source Computer Vision) which is a library of programming functions mainly aiming at real time computer vision. From Table 2, we can see that the computational time on neural network with LSA method is increased when the number of the dimension is increased. For all number of dimensions, neural network with LSA method is faster than the original neural network using vector space model (152.7 s). The test time of our system is very fast, for all dimensions, the test time is within one second.

5. Conclusions

In this paper, we develop text categorization models using neural network and latent semantic analysis method. The modified BPNN accelerates the training speed as well as improves the performance of basic BPNN algorithm. The introducing of latent semantic analysis not only reduces the dimension drastically but also overcomes the problems existing in commonly used vector space model method for text representation, the categorization performance has been improved consequently, the reduced size of the vectors greatly decreased the computational time in the back-propagation neural network.

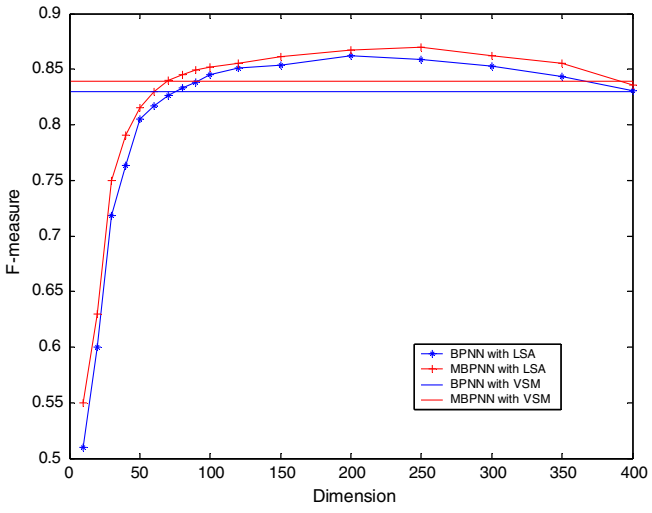

Fig. 2. Categorization performance according to the number of dimension.

Table 2
Computational time according to the number of dimension (seconds)

Number of dimension	Computational time	Number of dimension	Computational time
SVD 10	7.40	SVD 100	18.6
SVD 20	8.90	SVD 120	21.0
SVD 30	10.13	SVD 150	24.56
SVD 40	11.26	SVD 200	30.70
SVD 50	12.42	SVD 250	37.01
SVD 60	13.72	SVD 300	42.90
SVD 70	14.95	SVD 350	49.15
SVD 80	16.13	SVD 400	55.10
SVD 90	17.30	VSM 1200	152.7

Acknowledgment

The research work in this paper is supported by National Natural Science Foundation of China, under the Grant No. 70531030.

References

[1] Jian-Tao Sun, Zheng Chen, Hua-Jun Zeng, Yuchang Lu, Chun-Yi Shi, Wei-Ying Ma, Supervised latent semantic indexing for document categorization, in: ICDM, IEEE Press, 2004, pp. 535–538.
[2] H.T. Ng, W.B. Goh, K.L. Low, Feature selection, perception learning, and a usability case study for text categorization, in: Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 1997, pp. 67–73.
[3] S.C. Deerwester, S.T. Dumais, T.K. Landauer, G.W. Furnas, R.A. Harshman, Indexing by latent semantic analysis, Journal of the American Society of Information Science 41 (6) (1990) 391–407.
[4] R. Li, J. Wang, X. Chen, X. Tao, Y. Hu, Using maximum entropy model for Chinese text categorization, Journal of Computer Research and Development 42 (1) (2005) 94–101 (in Chinese with English abstract).
[5] W.Y. Liu, N. Song, A fuzzy approach to classification of text documents, Journal of Computer Science and Technology 18 (5) (2003) 640–647.
[6] S. Zelikovitz, H. Hirsh, Using LSI for text classification in the presence of background text, in: Proceedings of the tenth international conference on Information and knowledge management, ACM Press, 2001, pp. 113–118.
[7] Savio L.Y. Lam, Dik Lun Lee, Feature reduction for neural network based text categorization, in: Sixth International Conference on Database Systems for Advanced Applications (DASFAA'99), 1999, p. 195.
[8] A. Rafael Calvo, Jae-Moon Lee, Xiabo Li, Managin content with automatic document classification, Journal of Digital Information, 5(2) (2004) Article No. 282.
[9] M. Lu, L. Diao, Y. Lu, L. Zhou, The design and implementation of an excellent text categorization system, in: Proceeding of the fourth world congress intelligent control and automation, vol. 1, 2002, pp. 10–14.
[10] P. Soucy, G.W. Mineau, A simple k-NN algorithm for text categorization, in: Proceeding of the first IEEE International Conference on Data Mining (ICDM_01), vol. 28, 2001, pp. 647–648.
[11] L. Ma, J. Shepherd, Y. Zhang, Enhancing text classification using synopses extraction, in: Proceeding of the fourth international conference on web information systems engineering, 2003, pp. 115–124.
[12] T. Joachims, Text categorization with support vector machines: learning with many relevant features, in: C. Nedellec, C. Rouveirol, (Eds.), Proc. of the 10th European Conf. on Machine Learning (ECML-98), Springer-Verlag, Chemnitz, 1998, pp. 137–142.
[13] Evgeniy Gabrilovich, Shaul Markovitch, Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5, in: ICML, ACM Press, 2004, pp. 321–328.
[14] Y. Yany, Noise reduction in a statistical approach to text categorization, in: Proc. of the 18th ACM International Conference on Rexorch ond Development in Information Retrieval, New York, 1995, pp. 256–263.

[15] S. Chakrabarti, S. Roy, M. Soundalgekar, Fast and accurate text classification via multiple linear discriminant projections, International Journal on Very Large Data Bases 12 (2) (2003) 170–185.
[16] Y. Yang, X. Liu, A Re-examination of Text Categorization Methods, Proceedings of SIGIR'99 (1999) 42–49.
[17] J. Kazama, J. Tsujii, Maximum entropy models with inequality constraints: a case study on text categorization, Machine Learning 60 (1–3) (2005) 159–194.
[18] M.W. Berry, S.T. Dumais, G.W. O'Brien, Using linear algebra for intelligent information retrieval, SIAM Review 37 (4) (1995) 573–595.
[19] Wei Wu, Guorui Feng, Zhengxue Li, Yuesheng Xu, Deterministic convergence of an online gradient method for BP neural networks, IEEE Transactions on Neural Networks 16 (No. 3) (2005).
[20] The OpenCV is available on sourceforge http://sourceforge.net/projects/opencvlibrary/.