패턴인식 문제에 대한 다층퍼셉트론의 설계 방법
Design of Multilayer Perceptrons for Pattern Classifications

오상훈
목원대학교 정보통신공학과

Sang-Hoon Oh(shoh@mokwon.ac.kr)

요약

다층퍼셉트론 혹은 전방향 신경회로망이 임의의 함수를 근사시킬 수 있다는 이론적 연구결과에 기초하여 많은 분야에 응용되고 있다. 이 다층퍼셉트론을 실제 문제에 응용하는 경우에 여러 가지 파라미터 혹은 학습 방법 등을 결정하여야 한다. 이 논문에서는 패턴인식 문제에 다층퍼셉트론을 적용하는 경우에 실제 결정하여야 할 파라미터의 결정방법과 학습 방법에 대하여 논의한다. 이 논의는 각층의 노드 수 결정 방법, 다층퍼셉트론의 가중치 초기화, 그리고, 성능향상을 위하여 학습에 사용되는 여러 가지 오차 함수, 데이터 불균형 문제의 학습, 깊은 구조 등을 다루었다.

■ 중심어 : | 다층퍼셉트론 | 패턴인식 | 설계방법 | 노드 수 | 오차함수 |

Abstract

Multilayer perceptrons(MLPs) or feed-forward neural networks are widely applied to many areas based on their function approximation capabilities. When implementing MLPs for application problems, we should determine various parameters and training methods. In this paper, we discuss the design of MLPs especially for pattern classification problems. This discussion includes how to decide the number of nodes in each layer, how to initialize the weights of MLPs, how to train MLPs among various error functions, the imbalanced data problems, and deep architecture.

■ keyword : | Multilayer Perceptron | Pattern Classification | Design Rule | Node Number | Error Function |

I. Introduction

It is well known that multilayer perceptrons(MLPs) or feed-forward neural networks have the capability of function approximation through series of weighted sums and non-linear transformations[1]. Based on the characteristics, MLPs have been widely applied to pattern recognition[2], time series prediction[3], speech recognition[4], data mining[5], bio-informatics[6], etc.

When implementing MLPs for such application problems, there are many things to be decided for designing the structure of MLPs and choices to select training methods. In this paper, we discuss the design of MLPs for pattern classification problems. This includes how to decide the number of nodes in each layer, how to initialize weights, which error function

* 본 논문은 한국콘텐츠학회 ICCC2009 국제학술대회 우수논문입니다.

접수번호 : #100324-003
접수일자 : 2010년 03월 24일
심사완료일 : 2010년 05월 18일
교신저자 : 오상훈, e-mail : shoh@mokwon.ac.kr

we can use for performance improvement, how to handle imbalanced data problems, and deep architecture neural networks.

In section II, the architecture of MLPs and EBP(error back-propagation) training algorithms are introduced. In section III, we will discuss how to decide the number of nodes in each layer. Section IV discusses the initialization of MLPs. The various error functions for performance improvement are in section V. Classification of imbalanced data is introduced in section VI. Deep architecture is briefly introduced in section VII. Finally, section VIII concludes this paper.

II. Architecture of MLP and EBP Algorithm

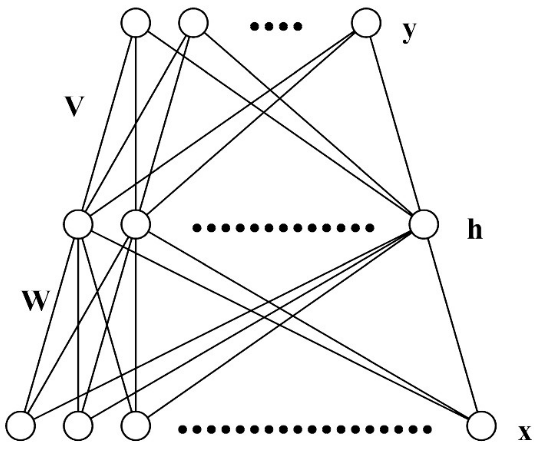

Fig. 1. The architecture of MLP.

Consider an MLP consisted of N inputs, H hidden, and M output nodes, as shown in [Fig. 1]. Here, each node has a value between -1 and +1. When an input vector $\boldsymbol{x}=[x_1,x_2,...,x_N]$ is presented to the MLP, the jth hidden node value is

$$h_j = f(\hat{h}_j) = \tanh(\hat{h}_j/2), j=1,2,...,H, \quad (1)$$

where $f(.)$ is the sigmoid nonlinear function and

$$\hat{h}_j = \sum_{i=0}^{N} w_{ji} x_i \quad (2)$$

is the weighted sum to h_j. Here, w_{ji} denotes the weight connecting x_i to h_j, $x_0=1$, and w_{j0} denotes the bias to x_i. Also, the weighted sum to the kth output node is given by

$$\hat{y}_k = \sum_{j=0}^{H} v_{kj} h_j, k=1,2,...,M, \quad (3)$$

where v_{kj} denotes the weight connecting h_j to y_k, $h_0=1$, and v_{k0} denotes the bias to y_k. Finally, the kth output node value is given by

$$y_k = f(\hat{y}_k) = \tanh(\hat{y}_k/2), k=1,2,...,M. \quad (4)$$

When the pth training pattern $\boldsymbol{x}^{(p)}(p=1,2,...,P)$ is presented to the MLP, we train the MLP whose output vector $\boldsymbol{y}^{(p)}(p=1,2,...,P)$ approaches the desired output vector $\boldsymbol{t}^{(p)} = [t_1^{(p)}, t_2^{(p)}, \ldots, t_M^{(p)}]$. Let the pattern set which belongs to the class k be C_k . Then, the class where $\boldsymbol{x}^{(p)}$ originates from is coded as follows:

$$t_k^{(p)} = \begin{cases} +1, \text{if } \boldsymbol{x}^{(p)} \in C_k \\ -1, otherwise. \end{cases} \quad (5)$$

Conventionally, we use the MSE(mean-squared error) function defined by

$$E_{MSE}^{out} = \frac{1}{2} \sum_{p=1}^{P} \sum_{k=1}^{M} (t_k^{(p)} - y_k^{(p)})^2 \quad (6)$$

as a distance measure between $\boldsymbol{y}^{(p)}$ and $\boldsymbol{t}^{(p)}$. To minimize E_{MSE}^{out} , output weights are updated as

$$\Delta v_{kj}(\boldsymbol{x}^{(p)}) = -\eta \frac{\partial E_{MSE}^{out}}{\partial v_{kj}} = \eta \delta_k^{(p)} h_j^{(p)} \qquad (7)$$

where η is the learning rate and

$$\delta_k^{(p)} = -\frac{\partial E_{MSE}^{out}}{\partial \widehat{y_k^{(p)}}} = (t_k^{(p)} - y_k^{(p)})^n f'(\widehat{y_k^{(p)}}) \qquad (8)$$

is the error signal for the kth output node. Hidden weights are updated as

$$\Delta w_{ji}(\boldsymbol{x}^{(p)}) = \eta f'(\widehat{h_j^{(p)}}) x_i^{(p)} \sum_{k=1}^{M} v_{kj} \delta_k^{(p)} \qquad (9)$$

We can minimize E_{MSE}^{out} through iterative updating of weights for all training patterns. This is the EBP(error back-propagation) algorithm[7].

III. How to Decide the Number of Nodes in Each Layer

In order to train the MLP for application problems, firstly we should determine the number of nodes in each layer. After initializing the weights of MLP with given number of nodes, the EBP algorithm is generally used for training. In this section, we discuss how to decide the number of nodes in input, hidden, and output layers.

1. The Number of Input Nodes

Usually, the number of input nodes, N, corresponds to the dimension of input data. The input data can be raw data or features extracted from raw data. There are two general approaches for reducing input dimensionality, i.e. feature extraction and feature selection[8]. Feature extraction is transforming the existing features into a lower dimensional space, while feature selection is selecting a subset of the existing features without a transformation.

PCA(Principal Component Analysis)[9], ICA (Independent Component Analysis)[10][11], and NMF(Non-negative Matrix Factorization)[12] can be categorized into feature extraction. In order to find the PCA transform, we firstly make a correlation matrix of input data and find eigenvectors with associated eigenvalues. Since eigenvectors are mutually orthogonal, we can transform input data to an orthogonal space using the eigenvectors. Here, we can reduce the data dimension with minimum transformation error if we use a portion of eigenvectors with larger associated eigenvalues[9]. For reducing the data dimension through ICA, we can use the projection pursuit algorithm[11] or $N \times N$ ICA algorithm[10] after dimensional reduction of input data using PCA[13]. NMF[12] or sNMF(sparse NMF)[8] also are excellent dimensional reduction algorithms which have a constraint that all variables are positive.

When we compare the characteristics of image features extracted by PCA, ICA, NMF, and sNMF[13], PCA shows global shape with spacial frequency components. ICA shows line-shape features similar to the simple cell characteristics of V1 layer in the human visual pathway. Features extracted by cascaded processing of PCA and ICA are more complex than the ICA features. If we want to extract very local features, we can use NMF algorithm which transform the input data to a lower dimensional positive elements vectors. In addition to, sNMF has the ability to control the degree of sparseness in the NMF algorithm[8].

MI(Mutual Information) is a good tool for feature selection[14]. Feature selection is a problem to find from the observation data space a subset of features, that optimally characterizes the target classification variable c. If a feature has a high mutual information with c, the feature is important for the classification and it must be selected. If not, the feature may be useless for the classification.

2. The Number of Output Nodes

In pattern recognition applications, the number of output nodes, M, is generally equal to the number of classes. In the limit $P \rightarrow \infty$, the minimizer of E_{MSE}^{out} converges (under certain regularity conditions, Theorem 1 in [15]) toward the minimizer of the function

$$\Delta_m = E\left\{ \frac{1}{2} \sum_{k=1}^{M} \left(T_k - y_k(\boldsymbol{X}) \right)^2 \right\} \tag{10}$$

where $E\{.\}$ is the expectation operator, $\boldsymbol{X}$ is the random vector denoting an input pattern, and T_k is the random variable denoting the target. Thus when network parameters are chosen to minimize Δ_m, $y_k(\boldsymbol{x})$ estimates $E\{T_k|\boldsymbol{x}\}$[16]. Since the target are coded as in Eq. (5),

$$E\{T_k|\boldsymbol{x}\} = 2Q_k(\boldsymbol{x}) - 1 \tag{11}$$

where $Q_k(\boldsymbol{x})$ is the posterior probability

$$Q_k(\boldsymbol{x}) = \Pr\left[\boldsymbol{X} \in C_k | \boldsymbol{X} = \boldsymbol{x} \right] \tag{12}$$

Hence the Bayes classifier can be defined by

$$\text{decide k, if k = max } y_k(\boldsymbol{x}) \tag{13}$$

If the number of output nodes per class is increased from 1 to many, there might be an improvement of performance. This effect will be intensified when the outputs are mutually independent[17].

3. The Number of Hidden Nodes

The objective for determining the number of hidden nodes is to find the smallest architecture that accurately fits the true function described by the training data[18]. A too large architecture may accurately fit the training data, but may have bad generalization due to over-fitting of the training data. On the other hand, a too small architecture will save on computational costs, but may not have enough processing elements to accurately approximate the true function.

The simplest way to determine the number of hidden nodes is the method based on PCA[19]. When applying PCA to the training data, the number of large eigenvalues may correspond to the number of hidden nodes and the associated eigenvectors are used as initial weights of hidden layer. Besides this direct method[19], there have been many approaches to select the number of hidden nodes, which can be categorized into brute-force approach, regularization, network construction, and pruning[18].

The brute-force approach is finding the global optimum architecture through an exhaustive search over all possible architecture. However, the search space explodes with the number of weights[20]. Neural network regularization involves the addition of a penalty term to the objective function to be minimized[21].

This regularization requires a delicate balance between the normal error term and the penalty term. Network construction algorithms start training with a small network and incrementally add hidden nodes during training when the network is trapped in a local minimum[20]. Crucial to the success of construction algorithms is effective criteria to trigger when to add a new unit, when to stop the growing process, where and how to connect the new unit to the existing architecture and how to avoid restarting training[18].

Neural network pruning algorithms start with an oversized network and remove unnecessary network parameters, either during training or after convergence to a local minima[22]. Sensitivity measure of hidden nodes or weights can be a good criteria for pruning.

IV. Initialization of MLPs

In pattern recognition applications, the desired output value of MLP is one of the two extreme values of the sigmoid function. If the weighted sum to any output node is near the wrong extreme value, we say the node is "incorrectly saturated". When an output node is incorrectly saturated, the amount of weight change is small due to the small gradient of the sigmoid activation function and the error remains nearly unchanged[25]. This causes the critical drawbacks of the EBP algorithm, that is, slow learning speed and convergence to local minima.

Especially, when output nodes are incorrectly saturated in the initial stage of learning, we can find the "premature saturation" which is a phenomenon that the error of MLP stays significantly high constant for some period of time during training. The probability of premature saturation can be derived in terms of the maximum value of initial weights, the number of nodes in each layer, and the maximum slope of the sigmoid activation function[25]. Based on this result, the premature saturation can be avoided with proper initial weight settings such that output nodes are in the linear region of sigmoid activation function in the initial epoch[25].

Also, there were proposals[19][26][27] to initialize the weights of hidden layer based on the interpretation that the hidden layer is a kind of feature extraction stage[24]. In this sense, hidden weights can be initialized with PCA[19][23], ICA[26] or NMF feature vectors[27].

V. Error Functions for Performance Improvement

Among many error functions proposed to improve the performance of EBP algorithm, the cross-entropy(CE) error function was interesting because it could remove the incorrect saturation of output nodes during training[28]. However, the CE suffers from overspecialization for training patterns since the error signal for a correctly saturated output node is too strong.

In order to resolve this problem, an nth order extension of the cross-error function(nCE) was proposed[16]. This nCE error function accelerates the learning speed of EBP algorithm by reducing the probability that output nodes are near wrong extreme values. It also prevents the overspecialization of learning for training patterns by generating a weak error signal for output node near the desired value. Here, the error signal of output layer can be represented by an nth order function of the difference between desired and actual output values.

Generally, decreasing of error function does not correspond to the increasing of classification ratio. In order to resolve this problem, "classification figure of merit(CFM)" cost function was proposed[29]. The CFM method also generates a very weak error signal for correctly saturated output node to prevent overspecialization for training patterns.

VI. Classification of Imbalanced Data

In many classification problems, unusual or interesting class is rare among a general population. This data imbalance has been reported in a wide range of applications such as remote sensing, bio-medical diagnoses, etc. Many methods for classification of the imbalanced data can be categorized into the data level approach, algorithmic level, and ensemble scheme.

Among the above approaches, developing a better classifier at the algorithmic level is critical because it is the essential part in the data level approach or ensemble of classifiers. Recently, a new error function for classification of imbalanced data was proposed which intensifies weight-updating for the minority class and weakens weight-updating for the majority class[30]. This method shows relatively stable and superior performance than the other training methods for MLPs.

VII. Deep Architecture

There have been reports that well-trained MLPs show poor generalization performance. Or in some cases, MLPs can not fit the true-function described by the training data. To resolve these problems, one may need deep architectures. Deep architectures are sorts of feed-forward neural networks with many hidden layers[31]. Restricted Boltzmann Machine is proposed as an initial training algorithm of the deep neural networks[32]. Also, EBP algorithm is necessary to finely tune the deep neural networks.

VIII. Conclusion

In this paper, we discussed the design of MLPs for pattern classification problems. Regarding the architecture of MLPs, inputs could be reduced by feature extraction or feature selection. The number of hidden nodes could be determined by the brute-force, regularization, network construction, or pruning techniques. We could avoid the premature saturation by proper initial weight settings. Also, hidden weights can be initialized for certain feature extraction purposes. For better performance, various error functions were introduced. Also, there was an error function suggested for classification of imbalanced data problems. Finally, the recent deep neural network is introduced for more complex application problems.

참 고 문 헌

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feed-forward networks are universal approximators," Neural Networks, Vol.2, pp.359-366, 1989.

[2] R. P. Lippmann, "Pattern classification using neural networks," IEEE Communication Magazine, pp.47-64, 1989.

[3] J. B. Hamshire II and A. H. Waibel, "A novel objective function for improved phoneme recognition using time-delay neural networks," IEEE Trans. Neural Networks, Vol.1, pp.216-228, 1990.

[4] A. S. Weigend and N. A. Gershenfeld, *Time Series Prediction: Forecasting the future and understanding the past,* Addison-Wesley Publishing Co., 1994.

[5] Y.-M. Huang, C.-M. Hung, and H. C. Jiau, "Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem," Nonlinear

Analysis, Vol.7, pp.720-747, 2006.

[6] Z. R. Yang and R. Thomson, "Bio-basis function neural netwrok for prediction of protease cleavage sites in proteins," IEEE Trans. Neural Networks, Vol.16, pp.263-274, 2005.

[7] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*, MIT Press, Cambridge, MA, 1986.

[8] H. Kim and H. Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," Bioinformatics, Vol.23, pp.1495-1502, 2007.

[9] Keinosuke Fukunaga, Introduction to Statistical Pattern Recognition, Elsevier, 1990.

[10] T.-W. Lee, et al., "A unifying information-theoretic framework for independent component analysis," Computers & Mathematics with Applications, Vol.31. pp.1-21, 2000.

[11] M. Girolami, A. Cichocki, and S.-I. Amari, "A common neural-network model for unsupervised exploratory data analysis and independent component analysis," IEEE Trans. Neural Networks, Vol.9, No.6, pp.1495-1501, 1998.

[12] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," Natute, Vol.401, pp.788-791, 1999.

[13] S.-H. Oh, "Comparisons of linear feature extraction methods," Journal of the Korea Contents Association, Vol.9, No.4, pp.121-130, 2009.

[14] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," IEEE Trans. PAMI, Vol.27, No.8, pp.1226-1238, 2005.

[15] H. White, "Learning in artificial neural networks: a statistical perspective," Neural Computation, Vol.1, pp.425-464. 1989.

[16] S.-H. Oh, "Improving the error backpropagation algorithm with a modified error function," IEEE Trans. Neural Networks, Vol.8, pp.799-803, 1997.

[17] S.-H. Oh, "Performance improvement of multilayer perceptrons with increased output nodes," Journal of the Korea Contents Association, Vol.9, No.1, pp.123-130, 2009.

[18] A. P. Engelbrecht, "A new pruning heuristic based on variance analysis of sensitivity information," IEEE Trans. Neural Networks, Vol.12, pp.1386-1399, 2001.

[19] Y. R. Park, T. J. Murray, and C. Chen, "Predicting Sun Spots Using A Layered Perceptron Neural Networks," IEEE Trans. Neural Networks, Vol.7, pp.501-505, 1996(3).

[20] J. Moody and P. J. Antsaklis, "The dependence identification neural network construction algorithm," IEEE Trans. Neural Networks, Vol.7, pp.3-15, 1996.

[21] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural network architecture," Neural Computation, Vol.7, pp.219-269, 1995.

[22] X. Zeng and D. S. Yeung, "Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure," Neurocomputing, Vol.69, pp.825-837, 2006.

[23] J. V. Shah and C.-S. Poon, "Linear independence of internal representations in multilayer percpetrons," IEEE Trans. Neural Networks, Vol.10, No.1, pp.10-18, 1999.

[24] S.-H. Oh and Y. Lee, "Effect of nonlinear transformations on correlation between weighted sums in multilayer perceptrons,"

IEEE Trans. Neural Networks, Vol.5, pp.508-510, 1994.

[25] Y Lee, S.-H. Oh, and M. W. Kim, "An analysis of premature saturation in back propagation learning," Neural Networks, Vol.6, pp.719-728, 1993.

[26] Y.-F. Yam, "An independent component analysis based weight initialization method for multilayer perceptrons," Neurocomputing, Vol.48, pp.807-818, 2002.

[27] P. C. Barman and S.-Y. Lee, "Nonnegative matrix factorization (NMF) based supervised feature selection and adaptation," LNCS, Vol.5326, pp.120-127, 2008.

[28] A. van Ooyen and B. Nienhuis, "Improving the convergence of the backpropagation algorithm," Neural Networks, Vol.5, pp.465-471, 1992.

[29] J. B. Hampshire II and A. H. Waibel, "A novel objective function for improved phoneme recognition using time-delay neural networks," IEEE Trans. Neural Networks, Vol.1, pp.216-228, 1990.

[30] S.-H. Oh, "Classification of imbalanced data using multilayer perceptrons," Journal of the Korea Contents Association, Vol.9, pp.141-148, 2009.

[31] Y. Bengio, "Learning deep architecture for AI," to appear in Foundations and Trends in Machine Learning

[32] G. E. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, Vol.313, pp.504-507, 2006.

저 자 소 개

오 상 훈(Sang-Hoon Oh) 종신회원

- 1986년 2월 : 부산대학교 전자공학과(공학사)
- 1988년 2월 : 부산대학교 대학원 전자공학과(공학석사)
- 1999년 8월 : 한국과학기술원 전기 및 전자공학과(공학박사)
- 1988년 1월 ~ 1989년 12월 : LG 반도체(주) 사원
- 1990년 1월 ~ 1998년 6월 : 한국전자통신연구원 기초기술연구부 및 이동통신기술연구소 선임연구원
- 1999년 8월 ~ 2000년 3월 : 한국과학기술원 뇌과학연구센터 연구원
- 2000년 4월 ~ 2000년 10월 : 일본 RIKEN, Brain Science Institute, Research Scientist
- 2000년 10월 ~ 20001년 10월 : (주)엑스텔테크놀리지 연구소장
- 2001년 11월 ~ 2002년 2월 : 한국과학기술원 초빙교수
- 2002년 3월 ~ 현재 : 목원대학교 정보통신공학과 부교수
- 2008년 8월 ~ 2009년 8월 : 조지아공대 College of Computing, Div. Computational Science and Eng. 방문교수

<관심분야> : 신경회로망, 독립성분분석, NMF, 패턴인식, 음성신호 처리, 바이오 인포매틱스