Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 31 (2004) 1411–1426
computers & operations research
www.elsevier.com/locate/dsw

A neural network model with bounded-weights for pattern classification

Yi Liao*, Shu-Cherng Fang, Henry L.W. Nuttle

Operations Research and Industrial Engineering, North Carolina State University, Raleigh, NC 27695-7906, USA

Abstract

A new neural network model is proposed based on the concepts of multi-layer perceptrons, radial basis functions, and support vector machines (SVM). This neural network model is trained using the least squared error as the optimization criterion, with the magnitudes of the weights on the links being limited to a certain range. Like the SVM model, the weight specification problem is formulated as a convex quadratic programming problem. However, unlike the SVM model, it does not require that kernel functions satisfy Mercer's condition, and it can be readily extended to multi-class classification. Some experimental results are reported.

Scope and purpose

For the past decade, there has been increasing interest in solving nonlinear pattern classification problems. Among the various approaches, Multi-layer perceptrons, radial basis function networks and support vector machines have received most attention due to their tremendous success in real-world applications. Compared with the other two, The support vector machines approach is relatively new and often performs better in many applications. However, it also has some limitations, for example, kernel functions are required to satisfy Mercer's condition and it is not easily applicable for multi-class classification. In this paper, we propose a new neural network model which overcomes these limitations.

Keywords: Pattern classification; Neural networks; Multi-layer perceptrons; Radial basis function networks; Support vector machines

1. Introduction

Multi-layer perceptrons (MLP), radial basis function (RBF) Networks, and support vector machines (SVM) are three major approaches for nonlinear pattern classification. The MLP model [1,2] is

* Corresponding author.
E-mail address: yliao2@eos.ncsu.edu (Y. Liao).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0305-0548(03)00097-2

perhaps the most widely used neural network model, being easy to understand and easy to implement. Its main drawback is that the training procedure often gets stuck at a local optimum of the cost function. The RBF model [3–6] is another popular neural network model, it avoids the difficulty of local optima by conducting the training procedure in two steps. The locations of the center vectors are found in the first step; then the values of the weights are optimized in the second step. Nevertheless, since there are usually some correlations between the center vectors and the weights, this training procedure often leads to a suboptimal solution. The SVM [7–11] is a relatively new pattern classification technique. It formulates the weight specification problem as a convex quadratic programming problem, thus the difficulties of local optima and suboptimal solutions are avoided. However, the SVM is basically for two-class classification only, and the kernel functions in the SVM model are required to satisfy Mercer's condition.

Based on the concepts of MLP, RBF, and SVM models, a new neural network model is proposed in this paper. This neural network model adopts the least squared error as the optimization criterion in training, while the magnitude of the weights on the links is limited to a certain range. Similar to the SVM model, the weight specification for the proposed model is also formulated as a convex quadratic programming problem. Thus the difficulties of local optima and suboptimal solutions are avoided. However, unlike the SVM model, it does not require that the kernel functions satisfy Mercer's condition, and it can be readily extended to multi-class classification.

This paper is organized as follows. The basic models for MLP, RBF and SVM are briefly introduced in Section 2. The proposed bounded neural network (BNN) model is discussed in Section 3. The experimental results are reported in Section 4, and the conclusions are given in Section 5.

2. Basic approaches

In this paper, we consider the following setting of a pattern classification problem. Each sample is assumed to come from one of c possible classes Ω_j, $j = 1, 2, \ldots, c$. We are given n training samples $(\mathbf{x}_i, y_i), i = 1, \ldots, n$, where $\mathbf{x}_i \in R^d$ represents the data and $y_i \in \{1, 2, \ldots, c\}$ the corresponding class label. The objective is to design a decision function $g(\cdot)$ on R^d from these training samples such that $g(\mathbf{x})$ can accurately predict the class label for any given test sample $\mathbf{x} \in R^d$.

2.1. Multi-layer perceptrons (MLP)

When used for pattern classification, the basic MLP model provides a nonlinear transformation of a pattern $\mathbf{x} \in R^d$ to $g(\mathbf{x}) \in R^c$, such that

$$g_j(\mathbf{x}) = \sum_{k=1}^{m} w_{jk} \phi \left(\sum_{i=1}^{d} w_{ki} x_i + w_{k0} \right) + w_{j0}, \quad j = 1, \ldots, c, \tag{1}$$

where m is a constant representing the number of hidden nodes, w_{jk} and w_{ki} are weights on links, w_{j0} and w_{k0} are biases on nodes. The nonlinear function $\phi(\cdot)$ is usually of the form of the logistic function:

$$\phi(z) = \frac{1}{1 + \exp(-az)} \tag{2}$$

where $z \in R$ and $a > 0$ is a constant called the gain parameter. Often in practice, the number of outputs, c, is taken as the number of classes, with each output corresponding to one class. A test sample is assigned to the class with the largest output value.

In the MLP model, the optimal weights and biases are found by optimizing a criterion function. Least squared error is often used. It is defined as

$$\min J = \frac{1}{2}\sum_{i=1}^{n} \|\mathbf{t}_i - g(\mathbf{x}_i)\|^2, \tag{3}$$

where $g(\mathbf{x}_i)$ is the output vector for the input $\mathbf{x}_i$ and $\mathbf{t}_i = (t_{i1}, \ldots, t_{ic})^{\mathrm{T}}$ is the corresponding target vector. Usually, the class labels are coded in such a way that $t_{ij} = 1$, if $\mathbf{x}_i$ is in class Ω_j, and $t_{ij} = 0$ otherwise.

In the MLP model, the procedure to find the optimal weights and biases is called "backpropagation". Since the objective function (3) is not convex with respect to its parameters, the backpropagation algorithm often gets stuck at a local optimum. Often in practice, to find a good solution, the backpropagation algorithm needs to be run several times, each time with different initial weights, this creates a heavy computational load and is often considered to be the major drawback of the MLP model.

2.2. Radial basis function (RBF) networks

Like the MLP model, the basic RBF model provides a nonlinear transformation of a pattern $\mathbf{x} \in R^d$ to $g(\mathbf{x}) \in R^c$. In this case,

$$g_j(\mathbf{x}) = \sum_{i=1}^{m} w_{ji}\phi\left(\frac{\|\mathbf{x} - \mu_i\|}{h}\right) + b_j, \quad j = 1, \ldots, c, \tag{4}$$

where m is a constant representing "the number of basis functions", w_{ji} is a weight, b_j is a bias, $\phi(\cdot)$ is a radially symmetric basis function, $\mu_i \in R^d$ is called a center vector, and $h \in R$ is a smoothing parameter. (4) has almost the same mathematical form as (1), the key difference being that the logistic function is replaced by a radial basis function, which is often taken to be the Gaussian function $\phi(z) = \exp(-z^2)$, where $z = \|\mathbf{x} - \mu\|/h$.

With the RBF model, the least squared error, (3), is usually adopted as the optimization criterion. Unlike the MLP model where all the parameters are optimized at the same time, in the RBF model, the center vectors and the weights are found in two separate steps. In the first step, the center vectors are found using some existing pattern recognition techniques (such as clustering or the Gaussian mixture models). In the second step, a set of linear equations is solved to find the optimal weights and biases. Since a set of linear equations is solved, the training of the RBF model does not run into the problem of local optima as with the MLP model. Therefore, the training time for RBF models is reported to be shorter. However, since there often exist some correlations between the locations of center vectors and the weights, the RBF approach may lead to a suboptimal solution.

2.3. Support vector machines (SVM)

The SVM is a relatively new approach for pattern classification. As pointed out in [7,8], it is basically a two-class classifier based on the ideas of "large margin" and "mapping data into a higher

Table 1
Commonly used Kernel functions

Name	Mathematical form
Polynomial kernel	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + \delta)^q$
Gaussian kernel	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\lvert \mathbf{x}_i - \mathbf{x}_j \rvert^2}{2h}\right)$
Sigmoid kernel	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + \delta)$

dimensional space", although some extensions have been proposed for the multi-class case [12,13]. In the SVM model, the training procedure is formulated as the following convex quadratic programming problem:

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \kappa(\mathbf{x}_i, \mathbf{x}_j) - \sum_{i=1}^{n} \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} y_i \alpha_i = 0, \\ & 0 \leqslant \alpha_i \leqslant C, \quad i = 1, \ldots, n, \end{aligned} \tag{5}$$

where α_i are the variables, $y_i \in \{+1, -1\}$ is the class label, $C > 0$ is a constant picked by the user, and $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ is a kernel function which is required to satisfy Mercer's condition [13–15]. Some commonly used kernel functions which satisfy Mercer's condition are listed in Table 1.

One interesting property of Mercer kernels is that they give rise to positive semi-definite matrices $[\kappa_{ij}]$ with $\kappa_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ for $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ [16]. Therefore, problem (5) becomes a convex quadratic programming problem and global optimality is guaranteed.

With the SVM model, the decision rule assigns a test sample $\mathbf{x}$ to class 1 if $g(\mathbf{x}) > 0$, and to class 2 if $g(\mathbf{x}) < 0$, where the discriminant function, $g(\mathbf{x})$, is defined as

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \kappa(\mathbf{x}_i, \mathbf{x}) + b,$$

where b is a threshold value, which is computed elsewhere in the SVM model. Since only the training samples $(\mathbf{x}_i, y_i)$ with positive α_i are used in the discrimination function, we call these training samples as "support vectors". Interestingly, when $\kappa(\cdot, \cdot)$ is a sigmoid kernel function, the SVM model produces a MLP discrimination function (1), but unlike the MLP model, a "global optimum" is guaranteed since the SVM model is formulated as a convex programming problem. On the other hand, when $\kappa(\cdot, \cdot)$ is a Gaussian kernel function, the SVM model leads to an RBF discrimination function (4), but unlike the RBF model, the centers and the weights are found at the same time. Although the SVM model has the above advantages over the MLP and RBF models, it is basically only for two-class classification, and the kernel functions are required to satisfy Mercer's condition.

3. Neural networks with bounded weights

3.1. Motivation

From a neural network perspective, support vector machines can be regarded as a special type of feedforward neural network. Like other feedforward neural networks, the SVM model also has a layered structure, in which a data vector is fed into the input nodes and nonlinearly transformed at the hidden nodes. Then at the output nodes, the outputs of the hidden nodes are linearly combined to produce the final output. The special characteristic of the SVM model is that it uses each training sample to define a hidden node, with a subset of them being selected by solving a convex quadratic programming problem.

Examining the problem formulation of the SVM model (5), we notice that the bound constraint, $0 \leqslant \alpha_i \leqslant C$, $i = 1, \ldots, n$, plays an important role in the selection of hidden nodes. Since each α_i is bounded below by 0, some α_i become 0 at the optimal solution. Since α_i is the weight of the link connecting the output node and the hidden node i, when α_i becomes 0, the hidden node i actually makes no contribution to the network output, i.e., these hidden nodes with $\alpha_i = 0$ are pruned during training.

Through the bound constraint $0 \leqslant \alpha_i \leqslant C$, the parameter C limits the magnitudes of the weights and thereby controls the learning capability of the SVM model. A large value for C allows the SVM model to have more freedom to learn the nonlinearities embedded in the training samples, but with greater chance of overfitting. On the other hand, a small value for C causes the SVM model to have less learning capability, but with a small chance of overfitting. The idea of using small weights to avoid overfitting is similar to that used in the weight decay method [17] for the MLP model.

Based on the above observations, we speculate that the concept of bounded weights may be successfully applied to neural network models which adopt objective functions different to that used in the SVM model. In this paper, we propose a neural network model in which the least squared error is adopted as the optimization criterion for weight specification, and the magnitudes of the weights on the links between the hidden nodes and output nodes are limited to a range $[0, C]$.

3.2. Proposed model

3.2.1. Two-class case

Let us first consider the two-class classification problem, in which we are given n training samples consisting of input data vectors $\mathbf{x}_i \in R^d$ together with corresponding labels $y_i \in \{-1, 1\}$, $i = 1, 2, \ldots, n$. The goal is to find some decision function $g(\mathbf{x})$ which accurately predicts the label of any given data $\mathbf{x}$.

For this problem, we propose a three-layer neural network model as shown in Fig. 1. The network has d input nodes, n potential hidden nodes (each training sample corresponds to one hidden node), and 1 output node. Each component x_i^j of data vector $\mathbf{x}_i \in R^d$ is entered at input node j and passed to each of the hidden node. At the hidden node k the data is fed into the transformation function $z_k = \kappa(\mathbf{x}_k, \mathbf{x})$, where $z_k \in R$ is the output of the hidden node k, $\mathbf{x}_k$ is the kth training sample, and $\kappa(\cdot, \cdot)$ is a nonlinear function (we call it a kernel function). The output node delivers the output as $\sum_{k=1}^{n} \alpha_k y_k z_k + b$, where $\alpha_k \in R$ is the weight of the link connecting the hidden node k and the output node, y_k is the class label of training sample $\mathbf{x}_k$, and $b \in R$ is the bias of the output node. As

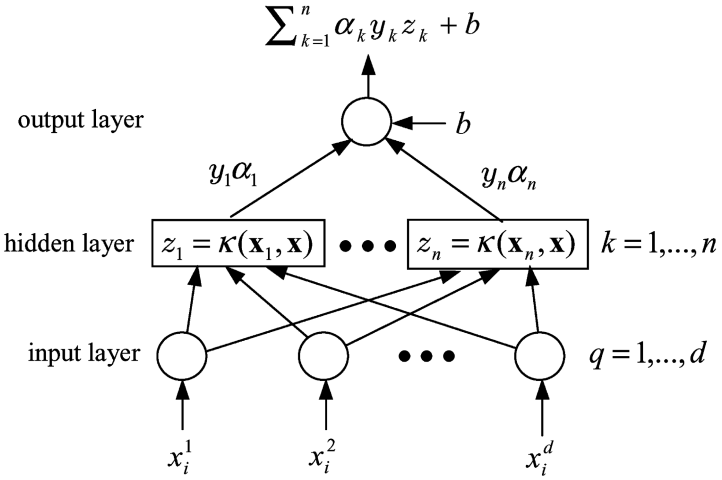

Fig. 1. Bounded neural network: two-class case.

we can see from Fig. 1, the actual weight of the link connecting the hidden node k and the output node is $\alpha_k y_k$, rather than α_k. But since y_k is either -1 or 1 and is known, we simply refer to α_k as the "weight".

Similar to the basic MLP and RBF models, we adopt the least squared error as the optimization criterion for weight specification for the proposed model. Also similar to the SVM model, we limit the magnitudes of the weights α_k, $k = 1, 2, \ldots, n$, to the range $[0, C]$, where $C > 0$ is a constant chosen by the user. Since the weights are bounded, we call the proposed model a "Bounded Neural Network" (BNN).

Specifically, in the training stage, we try to find the optimal parameters (weights α_k, $k = 1, \ldots, n$, and bias b) by solving the following problem:

$$\min_{(\alpha,b)} \quad \frac{1}{2}\sum_{i=1}^{n} \left|\left| y_i - \left(\sum_{k=1}^{n} \alpha_k y_k \kappa(\mathbf{x}_k, \mathbf{x}_i) + b \right) \right|\right|^2$$

$$\text{s.t.} \quad 0 \leqslant \alpha_k \leqslant C, \quad k = 1, 2, \ldots, n. \tag{6}$$

For prediction, the decision rule is

$$\text{assign } \mathbf{x} \text{ to } \begin{cases} \text{class 1} & \text{if } g(\mathbf{x}) \geqslant 0, \\ \text{class 2} & \text{if } g(\mathbf{x}) < 0, \end{cases}$$

where the discrimination function, $g(\mathbf{x})$, is defined by

$$g(\mathbf{x}) = \sum_{k=1}^{n} \alpha_k y_k \kappa(\mathbf{x}_k, \mathbf{x}) + b. \tag{7}$$

As with the SVM model, the weight $\alpha_k \in [0, C]$, some weights may become 0 after training. Since a hidden node with 0 weight plays no role in the discrimination function (7), such a hidden node is actually pruned after the training. Therefore, the actual number of hidden nodes in the trained

network may be less than n, and we call the training samples $(\mathbf{x}_i, y_i)$ with positive α_i as "support vectors".

While in the SVM model, the kernel functions are required to satisfy Mercer's condition, the BNN model does not have this requirement. To see this, let us first write the BNN model in the matrix form

$$\begin{aligned} \min_{(\alpha,b)} \quad & \tfrac{1}{2}\|\mathbf{y} - (\kappa(\mathbf{X},\mathbf{X})\mathbf{Y}\alpha + b\mathbf{e})\|^2 \\ \text{s.t.} \quad & 0 \leqslant \alpha \leqslant C\mathbf{e}, \end{aligned} \tag{8}$$

where $\mathbf{y} = (y_1, \ldots, y_n)^{\mathrm{T}}$ and $\alpha = (\alpha_1, \ldots, \alpha_n)^{\mathrm{T}}$ are n-dimensional vectors, $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^{\mathrm{T}}$ is an $n \times d$ matrix with the training sample $\mathbf{x}_i^{\mathrm{T}}$ being the ith row, $\kappa(\mathbf{X},\mathbf{X})$ is an $n \times n$ matrix with $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ being its (i,j)th element, $\mathbf{Y} = diag(y_1, \ldots, y_n)$ is a diagonal matrix with y_i as its (i,i) element, and $\mathbf{e}$ is an n-dimensional column vector of 1's. For ease of representation, let us denote $\kappa(\mathbf{X},\mathbf{X})$ as simply $\mathbf{K}$. The objective function of the problem (8) becomes

$$\begin{aligned} J &= \tfrac{1}{2}\|\mathbf{y} - (\mathbf{K}\mathbf{Y}\alpha + b\mathbf{e})\|^2 \\ &= \tfrac{1}{2}\alpha^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}}\mathbf{K}^{\mathrm{T}}\mathbf{K}\mathbf{Y}\alpha + b\mathbf{e}^{\mathrm{T}}\mathbf{K}\mathbf{Y}\alpha + \tfrac{1}{2}b\mathbf{e}^{\mathrm{T}}\mathbf{e}b - \mathbf{y}^{\mathrm{T}}(\mathbf{K}\mathbf{Y}\alpha + b\mathbf{e}) + \tfrac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{y} \\ &= \tfrac{1}{2}\hat{\alpha}^{\mathrm{T}}\hat{\mathbf{K}}^{\mathrm{T}}\hat{\mathbf{K}}\hat{\alpha} - \mathbf{q}^{\mathrm{T}}\hat{\alpha} + \tfrac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{y}, \end{aligned}$$

where $\hat{\alpha}^{\mathrm{T}} = (\alpha^{\mathrm{T}}, b)$ is an $(n+1)$-dimensional vector, $\hat{\mathbf{K}} = (\mathbf{K}\mathbf{Y} \mid \mathbf{e})$ is an $n \times (n+1)$ matrix, $\mathbf{q}^{\mathrm{T}} = (\mathbf{y}^{\mathrm{T}}\mathbf{K}\mathbf{Y}, \mathbf{y}^{\mathrm{T}}\mathbf{e})$ is an $(n+1)$-dimensional vector. Notice that $\frac{1}{2}\mathbf{y}^{\mathrm{T}}\mathbf{y}$ is a constant. Thus problem (8) is equivalent to

$$\begin{aligned} \min_{(\alpha,b)} \quad & \tfrac{1}{2}\hat{\alpha}^{\mathrm{T}}\hat{\mathbf{K}}^{\mathrm{T}}\hat{\mathbf{K}}\hat{\alpha} - \mathbf{q}^{\mathrm{T}}\hat{\alpha} \\ \text{s.t.} \quad & 0 \leqslant \alpha \leqslant C\mathbf{e}. \end{aligned} \tag{9}$$

Since the $\hat{\mathbf{K}}^{\mathrm{T}}\hat{\mathbf{K}}$ is positive semi-definite, problem (9) is a convex quadratic programming problem, no matter what kernel function $\kappa(\cdot,\cdot)$ we choose. Consequently, Mercer's condition is not required for the kernel functions. Thus, compared with the SVM model, the BNN model provides greater flexibility in choosing the form of the kernel function, which in turn provides better potential for learning.

Notice that while the weight specification problems in both the BNN model and the SVM model are formulated as convex quadratic programming problems, the BNN problem (9) only has the bound constraint, $0 \leqslant \alpha \leqslant C\mathbf{e}$. For the convex quadratic programming problem with bound constraints, a variety of algorithms [18,19] are readily available. The SVM problem (5) has the equality constraint $\mathbf{y}^{\mathrm{T}}\alpha = 0$ in addition to the bound constraint $0 \leqslant \alpha \leqslant C\mathbf{e}$, thus special care needs to be taken in designing an algorithm to solve the SVM problem [20].

3.2.2. Multi-class case

The model for the two-class classification problems can be easily extended to multi-class classification. In the multi-class classification, each sample is assumed to come from one of the c ($c \geqslant 2$) possible classes Ω_j, $j = 1, 2, \ldots, c$. We are given n training sample inputs $\mathbf{x}_i \in R^d$ together with corresponding labels $\mathbf{y}_i = (y_{i1}, \ldots, y_{ic})^{\mathrm{T}}$, where $y_{ij} = 1$ if $\mathbf{x}_i$ is in class Ω_j, and $y_{ij} = -1$ if $\mathbf{x}_i$ is not in

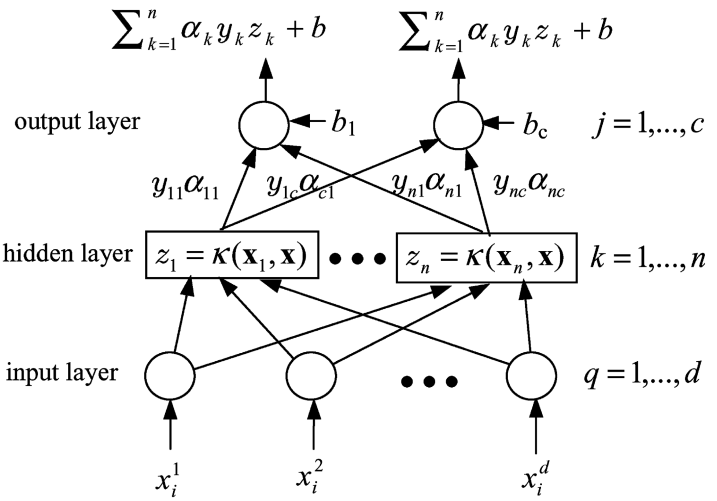

Fig. 2. Bounded neural network: multi-class case.

class Ω_j. The goal is to find some decision function $g(\mathbf{x})$ which accurately predicts the class label of any given data $\mathbf{x}$.

For the multi-class classification problem, the BNN model for the two-class case needs to be modified to have c output nodes, as shown in Fig. 2. Similar to the two-class case, the input data $\mathbf{x}$ is nonlinearly transformed at the hidden node k as $z_k = \kappa(\mathbf{x}_k, \mathbf{x})$, where $z_k \in R$ is the output of the hidden node k, and $\kappa(\cdot,\cdot)$ is a nonlinear kernel function. At each output node j, the output is delivered as $\sum_{k=1}^{n} \alpha_{kj} y_{kj} z_k + b_j$, where $\alpha_{kj} \in R$ is the weight of the link connecting the hidden node k to the output node j, $b_j \in R$ is the bias of the output node j. Similar to the two-class case, the actual weight of the link connecting the hidden node k to the output node j is $\alpha_{kj} y_{kj}$, rather than α_{kj}. But for ease of exposition, we call α_{kj} the "weight".

In the training stage, we try to find the optimal parameters (weights α_{kj}, $k = 1,\ldots,n, j = 1,\ldots,c$, and biases b_j, $j = 1,\ldots,c$) for the following problem:

$$\min_{(\alpha,b)} \quad \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{c}\left\| y_{ij} - \left(\sum_{k=1}^{n}\alpha_{kj} y_{kj}\kappa(\mathbf{x}_i, \mathbf{x}_k) + b_j\right)\right\|^2$$

$$\text{s.t.} \quad 0 \leqslant \alpha_{kj} \leqslant C, \quad k = 1,2,\ldots,n, \ \ j = 1,2,\ldots,c. \tag{10}$$

For prediction, the decision rule is

$$\text{assign } \mathbf{x} \text{ to class } k \text{ if } g_k(\mathbf{x}) \geqslant g_j(\mathbf{x}), \quad \forall j = 1,\ldots,c,$$

where the discrimination function, $g_j(\mathbf{x})$, is defined by

$$g_j(\mathbf{x}) = \sum_{k=1}^{n}\alpha_{kj} y_{kj}\kappa(\mathbf{x}, \mathbf{x}_k) + b_j, \quad j = 1,\ldots,c.$$

Notice that in problem (10), there are $n \times c$ weight variables α_{kj}, $k = 1,\ldots,n, j = 1,\ldots,c$. This problem can be recast in the following form

$$\min_{(\tilde{\alpha},\tilde{\mathbf{b}})} \quad \tfrac{1}{2}\|\tilde{\mathbf{y}} - (\tilde{\mathbf{K}}\tilde{\mathbf{Y}}\tilde{\alpha} + \tilde{\mathbf{E}}\tilde{b})\|^2$$

$$\text{s.t.} \quad 0 \leqslant \tilde{\alpha} \leqslant C\tilde{\mathbf{e}}, \tag{11}$$

where $\tilde{\alpha}=(\alpha_{11}\cdots\alpha_{n1},\ldots,\alpha_{1c}\cdots\alpha_{nc})^{\mathrm{T}}$ and $\tilde{\mathbf{y}}=(y_{11}\cdots y_{n1},\ldots,y_{1c}\cdots y_{nc})^{\mathrm{T}}$ are nc-dimensional vectors, $\tilde{\mathbf{b}} = (b_1,\ldots,b_c)^{\mathrm{T}}$ is a c-dimensional vector, $\tilde{\mathbf{e}}$ is an nc-dimensional vector of 1's. $\tilde{\mathbf{Y}} = diag(\tilde{\mathbf{y}})$ is an $nc \times nc$ diagonal matrix with $\tilde{y}_i$ being its ith diagonal element.

$$\tilde{\mathbf{K}} = \begin{pmatrix} \kappa(\mathbf{X},\mathbf{X}) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \kappa(\mathbf{X},\mathbf{X}) \end{pmatrix}$$

is an $nc \times nc$ block diagonal matrix which has c blocks of the matrix $\kappa(\mathbf{X},\mathbf{X})$ as its diagonal elements. $\kappa(\mathbf{X},\mathbf{X})$ is an $n \times n$ matrix with $\kappa(\mathbf{x}_i,\mathbf{x}_j)$ being its (i,j) element, and $\tilde{\mathbf{E}}$ is an $nc \times c$ matrix whose (i,j)th element is 1 if $(j-1)\times n + 1 \leqslant i \leqslant j \times n$, and 0 otherwise.

The problem as stated in (11) can be further simplified as

$$\min_{(\tilde{\alpha},\tilde{b})} \quad \tfrac{1}{2}\hat{\alpha}^{\mathrm{T}}\hat{\mathbf{K}}^{\mathrm{T}}\hat{\mathbf{K}}\hat{\alpha} - \mathbf{q}^{\mathrm{T}}\hat{\alpha}$$

$$\text{s.t.} \quad 0 \leqslant \tilde{\alpha} \leqslant C\tilde{\mathbf{e}}, \tag{12}$$

where $\hat{\alpha}^{\mathrm{T}} = (\tilde{\alpha}^{\mathrm{T}}, \tilde{\mathbf{b}})$ is an $(nc + c)$-dimensional vector, $\hat{\mathbf{K}} = (\tilde{\mathbf{K}}\tilde{\mathbf{Y}} \mid \tilde{\mathbf{E}})$ is an $nc \times (nc + c)$ matrix, and $\mathbf{q}^{\mathrm{T}} = (\tilde{\mathbf{y}}^{\mathrm{T}}\tilde{\mathbf{K}}\tilde{\mathbf{Y}}, \tilde{\mathbf{y}}^{\mathrm{T}}\tilde{\mathbf{E}})$ is an $(nc + c)$-dimensional vector. As with the 2-class case, (12) is a convex quadratic programming problem, no matter what kernel function we take.

4. Experimental results

To demonstrate the competitiveness of the proposed model, we conducted several numerical experiments. In our experiments, the quadratic programming solver of Matlab 5.3 was used to solve the weight specification problem for the BNN model, while the LIBSVM2.2 [21] was used for the SVM model. Since our objective is to compare the classification accuracy of the BNN and SVM models, it does not matter that much which solver to choose as long as it gives us the correct results.

4.1. Small example

For both the BNN and SVM models, we define the support vectors as the training samples with $\alpha_i > 0$. The first experiment was designed to show the different location of the support vectors for the two models. In this experiment, we randomly generated two classes of data, both of which are normally distributed but with different mean vectors, one at $(0,0)^{\mathrm{T}}$ and the other at $(2.5,2.5)^{\mathrm{T}}$, and the same covariance matrix,

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

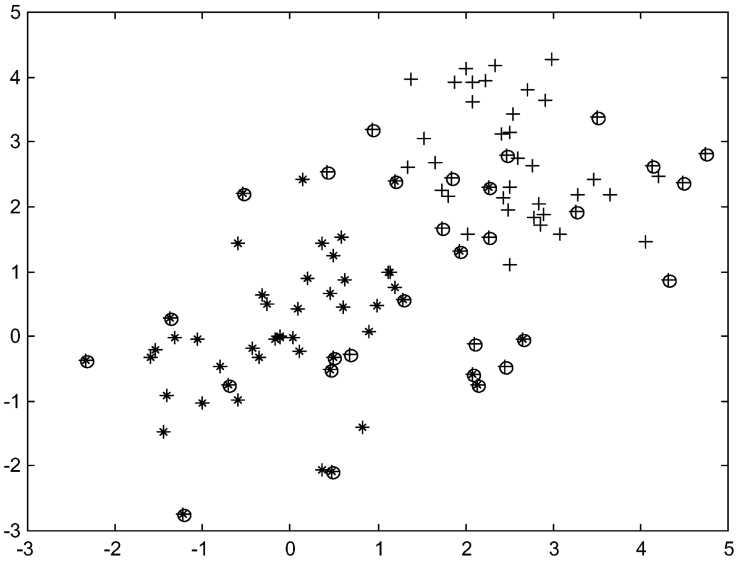

Fig. 3. The support vectors for the BNN model.

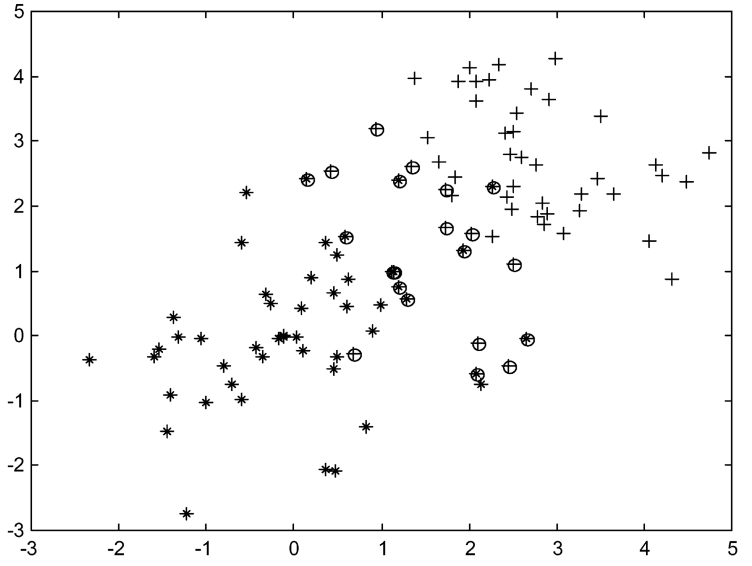

Fig. 4. The support vectors for the SVM model.

For each class, we randomly generated 50 data points. They are plotted as "+" and "★" in Figs. 3 and 4. Then we used the BNN model (6) and the SVM model (5) to train the data for the support vectors. For comparison purposes, we set the parameter C to be 1 in both models. We also adopted the same kernel function, namely the simple inner product function, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$.

The support vectors for the BNN and SVM models are shown in Figs. 3 and 4, respectively, plotted as circles. As we can see from the figures, the support vectors for the SVM model lie close to the boundary between the two classes, while the support vectors for the BNN model are scattered throughout the input space. Since in classification problems, the key issue is to obtain the classification boundary, intuitively, it would seem better for the support vectors to lie close to boundary rather than scattered around. However, this is not always the case, as we show in the examples in the next section.

4.2. Real-world examples

To demonstrate the competitiveness of the proposed model, we tested the BNN model on five benchmark real-world data sets which are publicly available from a University of California at Irvine (UCI) data repository [22]. The results were compared with those resulted from the SVM model. All of the five data sets are for two-class classification. They are listed as follows:

- Wisconsin Breast Cancer. This breast cancer data set was from the University of Wisconsin Hospital, Madison. The objective is to detect whether a breast tumor is "benign" or "malignant". The data set contains 699 points, each point consisting of 9 features. All features are continuous variables. The "benign" class contains 458 points; the "malignant" class contains 241 points.
- Cleveland Heart Disease. This data set was collected by the Cleveland Clinic Foundation. It was used to diagnose heart disease. The data set contains 303 points, each point consisting of 13 features. All features are continuous variables. The "positive" class (heart disease) contains 164 points; the negative class (no heart disease) contains 139 points.
- Liver Disorders. This data set was collected by the BUPA Medical Research Ltd.. It was used to detect the liver disorders caused by excessive alcohol consumption. It contains 345 points, each point consisting of 6 features. The first 5 features are the results of blood test which are thought to be sensitive to liver disorders. The sixth feature is the number of half-pint equivalents of alcoholic beverages drunk per day. All features are continuous variables. The "positive" class (liver disorders) contains 145 points; the "negative" class (no liver disorders) contains 200 points.
- Ionosphere. This data set was collected by a radar system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. The "good" radar signals are those showing evidence of a particular type of structure in the ionosphere. The "bad" signals are those that do not. The data set contains 351 points, each point consisting of 34 features. All features are continuous variables. The "good" signals class contains 225 points; the "bad" signals class contains 126 points.
- Votes. This data set was from the 1984 United States Congressional Voting Records Database. It contains the voting records for each of the members of U.S. House of Representatives on the 16 key votes identified by the Congressional Quarterly Almanac. The objective is to identify whether the voting person is a "democrat" or a "republican". The data set contains 435 points, each consisting of 16 features. All features are binary variables. The "democrats" class contains 267 points; the "republicans" class contains 168 points.

For the above data sets, the feature values are frequently not in the same range. In our experiments, we normalized all feature values to the range [0, 1].

Our experiments were conducted using the ten-fold cross-validation technique, which is a commonly used technique to evaluate the performance of a pattern classification algorithm [23]. We conducted the ten-fold cross-validation in the following way: we randomly partitioned each data set into ten groups, each group having approximately the same number of points. Then we ran the BNN (or SVM) model ten times. Each time, one different group of data was held out for use as the test set; the other nine groups were used as the training set. The reported results are average values for these ten runs.

In our experiments, we tried a set of the values for the parameter, C. The reported results are the best results obtained.

For each data set, we tested the BNN and SVM models with three different types of kernel functions: the Gaussian kernel function, the polynomial kernel function and the sigmoid kernel function. For the Gaussian kernel function,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{2h}\right),$$

we set the kernel width $h = d/2$, where d is the number of features. For the polynomial kernel function,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\gamma \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + \delta)^q,$$

we set $q = 3$, $\gamma = 1$, and $\delta = 1$. For the sigmoid kernel function,

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\gamma \mathbf{x}_i^{\mathrm{T}} \mathbf{x}_j + \delta),$$

we set $\gamma = 1$ and $\delta = 1$. Tables 2–4 list the experimental results for the Gaussian kernel function, polynomial kernel function, and sigmoid kernel function, respectively. In these tables, the training correctness (or test correctness) is defined as the ratio of the number of correctly classified training (or test) samples divided by the total number of training (or test) samples. "SV" denotes the number of support vectors, while "BSV" denotes the number of bounded support vectors, which is defined as the support vector with $\alpha_i = C$.

As we can see from the tables, the BNN model and the SVM model have comparable training and test correctness. For these two models, the number of support vectors were also approximately in the same range. For the "Liver Disorders" test, the BNN model outperformed the SVM model in both training and testing correctness, and the number of support vectors was much smaller than that of the SVM model. For other tests, the performance of the BNN model is close to that of the SVM model.

From these results, we may conclude that the performance of these two models are comparable. Interestingly, it seems there were not much difference in performance with the three different kinds of kernel functions. An interesting topic for future research is to find a mathematical explanation.

4.3. General kernel functions

The next experiment was conducted to demonstrate that the kernel functions in the BNN model are not required to satisfy Mercer's condition. In this experiment, we tested the BNN model with an "invalid" Mercer kernel function (i.e., a kernel function which does not satisfy Mercer's condition). The result was compared with the one obtained using a valid Mercer kernel function.

Table 2
The experimental results for the Gaussian kernel function

Data set $n \times d$	Algorithm	Training correctness (%)	Test correctness (%)	Number of SVs	Number of BSVs
Breast Cancer	BNN	98.15	96.43	36	7
699×9	SVM	97.97	96.37	33	19
Cleveland Heart	BNN	89.15	84.87	98	64
297×13	SVM	86.87	85.11	114	91
Liver Disorders	BNN	76.12	72.67	136	128
345×6	SVM	73.48	71.35	236	226
Ionosphere	BNN	95.37	92.33	123	85
351×34	SVM	98.37	93.12	89	50
Votes	BNN	96.56	95.15	76	72
435×16	SVM	96.79	96.11	60	58

Table 3
The experimental results for the polynomial kernel function

Data set $n \times d$	Algorithm	Training correctness (%)	Test correctness (%)	Number of SVs	Number of BSVs
Breast Cancer	BNN	98.21	96.33	36	9
699×9	SVM	97.85	96.37	37	20
Cleveland Heart	BNN	88.18	84.67	98	65
297×13	SVM	88.04	84.67	113	83
Liver Disorders	BNN	75.36	73.42	138	127
345×6	SVM	74.88	72.85	235	226
Ionosphere	BNN	95.37	91.73	123	82
351×34	SVM	97.23	92.15	93	58
Votes	BNN	96.02	94.58	74	71
435×16	SVM	96.23	95.46	62	60

The valid Mercer kernel function we tested is

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + 1)^3, \tag{13}$$

while the invalid Mercer kernel function we tested is

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = -(\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j + 1)^3. \tag{14}$$

The kernel matrix $\kappa(\mathbf{X}, \mathbf{X})$ is defined as the $n \times n$ matrix with $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ being its (i, j)th element. Function (13) is a valid Mercer kernel function since the kernel matrix $\kappa(\mathbf{X}, \mathbf{X})$ is positive semi-definite.

Table 4
The experimental results for the sigmoid kernel function

Data set $n \times d$	Algorithm	Training correctness (%)	Test correctness (%)	Number of SVs	Number of BSVs
Breast Cancer	BNN	97.85	96.52	36	11
699×9	SVM	97.93	96.23	34	19
Cleveland Heart	BNN	87.32	85.21	94	59
297×13	SVM	86.67	85.17	114	89
Liver Disorders	BNN	76.23	73.35	139	129
345×6	SVM	75.84	73.17	238	228
Ionosphere	BNN	96.13	93.56	121	89
351×34	SVM	97.75	94.37	91	55
Votes	BNN	96.13	94.56	73	70
435×16	SVM	96.34	95.63	60	59

Table 5
The experimental results for general kernel functions

Data set $n \times d$	Kernel	Training correctness (%)	Test correctness (%)	Number of SVs	Number of BSVs
Breast Cancer	Valid	98.21	96.33	36	9
699×9	Invalid	98.06	96.23	33	9
Cleveland Heart	Valid	88.18	84.67	98	65
297×13	Invalid	86.98	84.67	51	26
Liver Disorders	Valid	75.36	73.42	138	127
345×6	Invalid	75.23	73.56	108	95
Ionosphere	Valid	95.37	91.73	123	82
351×34	Invalid	94.87	92.12	134	95
Votes	Valid	96.02	94.58	74	71
435×16	Invalid	96.38	94.26	81	76

Function (14) is not a valid Mercer kernel function because the kernel matrix $\kappa(\mathbf{X}, \mathbf{X})$ is negative semi-definite.

We tested the BNN model on the five data sets using each of these two kernel functions. As we expected, the BNN model was able to find the solution to problem (9) with both kernel functions. The test results are listed in Table 5. As we can see from the table, the two kernel functions lead to similar training and testing correctness. Interestingly, for the Cleveland Heart data set, the number of support vectors for the invalid Mercer kernel is much smaller.

Table 6
The experimental results for the Iris Flower classification

Algorithm	Training correctness				Test correctness			
	Class 1	Class 2	Class 3	Total	Class 1	Class 2	Class 3	Total
BNN	100%	98.33%	87.56%	95.30%	100%	97.23%	82.89%	93.37%
1-NN	NA	NA	NA	NA	100%	94.00%	92.00%	95.33%

From this test, we may conclude that the kernel functions, indeed, are not required to satisfy Mercer's condition in the BNN model. Thus the BNN model provides more flexibility in choosing the kernel functions than the SVM model.

4.4. Multi-class classification

The final experiment was designed to demonstrate the capability of the BNN model for multi-class classification. In this experiment, we tested the BNN model on the IRIS flower classification problem. In this problem, we are given three classes of data, which represent three different types of iris flowers. Each class contains 50 data points, with each point consisting of four features. The objective is to predict the class label based on the feature information.

We applied the BNN model to this problem. Again, we conducted the experiment by using the ten-fold cross-validation technique. The kernel function adopted was the Gaussian kernel function, $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|/2h)$, with the kernel width $h = d/2$, where d is the number of features, i.e., in this case $h = 2$. For the purpose of comparison, we applied the 1-nearest neighbor algorithm [24] to the Iris flower problem. The results are given in Table 6. The "Total" column contains the average value of the (training or test) correctness for class 1, class 2 and class 3.

As we can see from the table, the performance of the BNN model is not as good as that of the 1-nearest neighbor algorithm. While further investigation is needed, these results show that the BNN model can, indeed, perform multi-class classification.

5. Conclusion

We have proposed a new neural network model which has bounded weights for the links between hidden nodes and output nodes. It is trained by using the least squared error as the optimization criterion. Compared with the SVM model, the proposed BNN model does not require the kernel functions to satisfy Mercer's condition, and it can handle multi-class classification problems. The experimental results support the effectiveness of the proposed model.

References

[1] Rumelhart DE, Hinton GE, Williams RJ. Learning internal representation by error propagation. In: Rumelhart DE, McClelland JL, PDP Research Group, editors. Parallel distributed processing: explorations in the microstructure of cognition. Cambridge, MA: MIT Press, 1986;1:318–62.

[2] Bishop CM. Neural networks for pattern recognition. Oxford: Oxford University Press, 1995.
[3] Powell MJD. Radial basis functions for multivariable interpolation: a review. In: Mason JC, Cox MG, editors. Algorithms for approximation. Oxford: Clarendon Press, 1987. p. 143–67.
[4] Lowe D. Radial basis function networks. In: Arbib MA, editor. The handbook of brain theory and neural networks. Cambridge, MA: MIT Press, 1995. p. 779–82.
[5] Broomhead DS, Lowe D. Multi-variable functional interpolation and adaptive networks. Complex Systems 1998;2(3):269–303.
[6] Moody J, Darken CJ. Fast learning in networks of locally-tuned processing units. Neural Computation 1989;1: 281–94.
[7] Vapnik V. The nature of statistical learning theory. New York: Springer, 1995.
[8] Vapnik V. Statistical learning theory. New York: Wiley, 1998.
[9] Cortes C, Vapnik V. Support vector networks. Machine Learning 1995;20:273–97.
[10] Burges CJC, Scholkopf B. Improving the accuracy and speed of support vector learning machines. In: Mozer M, Jordan M, Petsche T, editors. Advances in neural information processing systems, vol. 9. Cambridge, MA: MIT Press, 1997. p. 375–81.
[11] Blanz V, Scholkopf B, Bulthoff H, Burges C, Vapnik V, Vetter T. Comparison of view-based object recognition algorithms using realistic 3D models. In: von der Malsburg C, von Seelen W, Vorbruggen JC, Sendhoff B, editors. Artificial neural networks—ICANN'96. Lecture Notes in Computer Science, vol. 1112. Berlin: Springer, 1996. p. 251–6.
[12] Suykens JAK, Vandewalle J. Multiclass least squares support vector machines. IJCNN'99 International Joint Conference on Neural Networks. Washington, DC, 1999.
[13] Boser BE, Guyon I, Vapnik V. A training algorithm for optimal margin classifiers. In: Haussler D, editor. Proceedings of the Fourth Workshop on Computational Learning Theory. San Mateo, CA: ACM Press, 1992. p. 144–52.
[14] Mercer J. Functions of positive and negative type and their connection with the theory of integral equations. Philosophical Transactions of the Royal Society of London, Series A 1999;209:415–46.
[15] Aizerman M, Braverman E, Rozonoer L. Theoretical foundations of the potential function method in pattern recognition learning. Automation and Remote Control 1964;25:821–37.
[16] Saitoh S. Theory of reproducing kernels and its applications. Harlow, England: Longman Scientific & Technical, 1988.
[17] Hinton GE. Learning distributed representations of concepts. In: Proceedings of the Eighth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Lawrence Erlbaum Associates, 1986. p. 1–12.
[18] More JJ, Toraldo G. On the solution of large quadratic programming problems with bound constraints. SIAM Journal of Optimization 1991;1(1):93–113.
[19] Ni Q, Yuan Y. A subspace limited memory quasi-Newton algorithm for large-scale nonlinear bound constrained optimization. Mathematics of Computation 1997;66(220):1509–20.
[20] Fang S-C, Puthenpura S. Linear optimization and extensions: theory and algorithms. New Jersey: Prentice-Hall, 1993.
[21] Chang CC, Lin C-J. LIBSVM: a library for support vector machines, 1991. [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm.
[22] Murphy PM, Aha DW. UCI repository of machine learning databases, 1992. [Online]. Available: http://www.ics.uci./edu/∼mlearn/MLRepository.html.
[23] Duda R, Hart P, Stork D. Pattern classification, 2nd ed. New York: Wiley, 2000.
[24] Fukunaga K. Introduction to statistical pattern recognition, 2nd ed. London: Academic Press, Inc., 1990.