Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 4 – 18
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Multi-class pattern classification using neural networks

Guobin Ou, Yi Lu Murphey*

Department of Electrical and Computer Engineering, The University of Michigan-Dearborn, Dearborn, MI 48128-1491, USA

Received 10 October 2005; received in revised form 10 March 2006; accepted 28 April 2006

Abstract

Multi-class pattern classification has many applications including text document classification, speech recognition, object recognition, etc. Multi-class pattern classification using neural networks is not a trivial extension from two-class neural networks. This paper presents a comprehensive and competitive study in multi-class neural learning with focuses on issues including neural network architecture, encoding schemes, training methodology and training time complexity. Our study includes multi-class pattern classification using either a system of multiple neural networks or a single neural network, and modeling pattern classes using one-against-all, one-against-one, one-against-higher-order, and P-against-Q. We also discuss implementations of these approaches and analyze training time complexity associated with each approach. We evaluate six different neural network system architectures for multi-class pattern classification along the dimensions of imbalanced data, large number of pattern classes, large vs. small training data through experiments conducted on well-known benchmark data.

Keywords: Machine learning; Pattern recognition; Multi-class classification; Neural networks

1. Introduction

Multi-class pattern recognition is a problem of building a system that accurately maps an input feature space to an output space of more than two pattern classes. Multi-class pattern recognition has a wide range of applications including handwritten digit recognition, object classification [1–3], speech tagging and recognition [4,5], bioinformatics [6–8], text categorization and information retrieval [9]. While two-class classification problem is well understood, multi-class classification is relatively less investigated. Many pattern classification systems were developed for two-class classification problems and theoretical studies of learning have focused almost entirely on learning binary functions [10] including the well-known support vector machines (SVM) [11,12], artificial neural network algorithms such as the perceptron and the error backpropagation (BP) algorithm [13–15]. For most of these algorithms, the extension from two-class to the multi-class pattern classification problem is non-trivial, and often leads to unexpected complexity or weaker performances [16–19]. The most popular approach used in multi-class pattern classification is to decompose the problem into multiple two-class classification problems. There are a number of different approaches to decompose a K-class pattern classification problem into two-class problems [12,16,19–22]. However, this is not necessarily the best approach to certain application problems.

This paper presents a comprehensive and competitive study in multi-class neural network classification using supervised learning. Our research is focused on the following important issues: approaches for modeling a multi-class pattern classification problem, neural network architectures, methods for encoding multi-class patterns, decision modules, learning complexity and system generalization. We present two major system architectures, a single neural network system and a system of multiple neural networks, and three types of approaches for modeling pattern classes, one-against-one (OAO), one-against-all (OAA) and P-against-Q

* Corresponding author. Tel.: +1 313 593 5028; fax: +1 313 593 9967.
E-mail address: yilu@umich.edu (Y.L. Murphey).

(PAQ). In addition we evaluate the learning capabilities of different neural network systems with respect to imbalanced training data, number of pattern classes, and small vs. large training data. We build the theoretical analysis based on the assumption that BP is the learning algorithm used in all neural networks. We will show that different architectures, modeling approaches and implementations with the same neural learning algorithm can give quite different performances in terms of system complexity, training time, system accuracy, and generalization. The theoretical analysis and comparison are accompanied by a large number of experiments. We have implemented at least one neural network system from each architecture and modeling category and conducted experiments using benchmark data that include six data collections from the UCI machine learning databases, a protein data set obtained from a well-known protein bank and the handwritten digit set provided by NIST. The theoretical analysis and experiments results show that the best neural network system seems to be problem-dependent: balance of data distribution, training data size, and the number of pattern classes.

This paper is organized as follows. Section 2 gives an overview of multi-class pattern classification problem using neural network systems and the three categories of multi-class modeling approaches. Sections 3 and 4 present multi-class pattern classifications using single and multiple neural networks, respectively. Section 5 analyzes the training time complexity and presents the performances of the neural network systems with different modeling approaches and architectures on the well-known benchmark data, and Section 6 concludes the study.

2. An overview of neural network systems for multi-class pattern classification

A multi-class, denoted as K-class, neural network classification problem can be described formally as follows. For a given d-dimensional feature space, Ω, and a training data set $\Omega_{tr} \subset \Omega$, where each element $\bar{x}$ in Ω_{tr} is associated with a class label cl, $cl \in Class\_Labels = \{cl^1, cl^2, \ldots, cl^K\}$, where $cl^j \neq cl^h$ for all $h \neq j$ and $K > 2$, a neural network system F can be trained on Ω_{tr} such that for any given feature vector $\bar{x} \in \Omega$, $F(\bar{x}) \in Class\_Labels$. F can be a system of neural networks or a single neural network whose weights are determined by a neural learning algorithm. In this paper, we use a multi-layered feed forward neural network with BP as our basis for studying system complexities and performance analysis. To facilitate our discussion, the following notations will be used throughout the paper. We denote the input and output at a hidden node j as

$$a_j^h = \sum_i w_{ji}^h x_i, \quad z_j = g^h(a_j^h), \quad j = 1, \ldots, H,$$

where x_i is the ith input of feature vector $\bar{x}$, w_{ji}^h is the weight associated with the input x_i to the jth hidden node, H is the number of hidden nodes, $g^h(\bullet)$ is the activation function used in the hidden layer. In the output layer, each node O_k has the input and output as follows:

$$a_k^o = \sum_j w_{kj}^o z_j, \quad y_k = g^o(a_k^o), \quad k = 1, \ldots, M,$$

where z_j is the output value from the jth hidden unit, w_{kj}^o is the weight associated with the jth hidden node and the kth output node, M is the number of the output nodes, and $g^o()$ is the activation function used in the output layer. Throughout this paper, we assume the activation functions are the same: the well-known logistic sigmoid function.

A K-class pattern classification problem can be implemented in either one of the two neural network architectures, a single neural network system with M outputs, where $M > 1$ (see Fig. 1(a)) or a system of multiple neural networks (see Fig. 1(b) and (c)). In Fig. 1(a) the number of the output nodes, M, is determined by the encoding scheme for pattern classes, and is not necessary equal to K. Fig. 1(b) illustrates a system of M binary neural networks with a decision module that integrates the results from the M binary neural networks, and Fig. 1(c) illustrates a system of neural networks each with multiple output nodes. Note the feature vectors in different neural networks in Fig. 1(b) and (c) can be different from each other. As we pointed out above that a K-class pattern classification problem can be modeled using one of the three types of schemes: OAA, OAO and PAQ. The OAA modeling, in which each of the K pattern classes is trained against all other classes, can be implemented in either a single neural network system (see Fig. 1(a)), or in a system of K binary neural networks (see Fig. 1(b)). When it is implemented in a single neural network system, the resulting neural network system should have $M = K$ output nodes, when it is implemented in a system of binary neural networks, the resulting system should have K binary neural networks.

In the OAO modeling, each of the K pattern classes is trained against every one of the other pattern classes. The OAO modeling can be implemented only in a system of $K(K-1)/2$ binary neural networks, which has an architecture similar to the one illustrated in Fig. 1 (b). In the PAQ modeling, a neural network is trained by using P of the K pattern classes against the other Q of the K pattern classes. The training process can be repeated several times, each time a mix of P different pattern classes against Q different pattern classes is used to train a neural network. The PAQ modeling can be implemented in either a single system or a system of multiple neural networks, which can be either binary or multiple classes.

Fig. 2 illustrates the possible classification boundaries drawn by four major neural network systems for multi-class pattern classification. In Fig. 2(a), a classification boundary is drawn by a trained neural network using class 2 against all other classes, which is similar to a figure in Ref. [23]. In this illustration, classification boundary drawn by the neural network was optimal: it provided maximum separation between

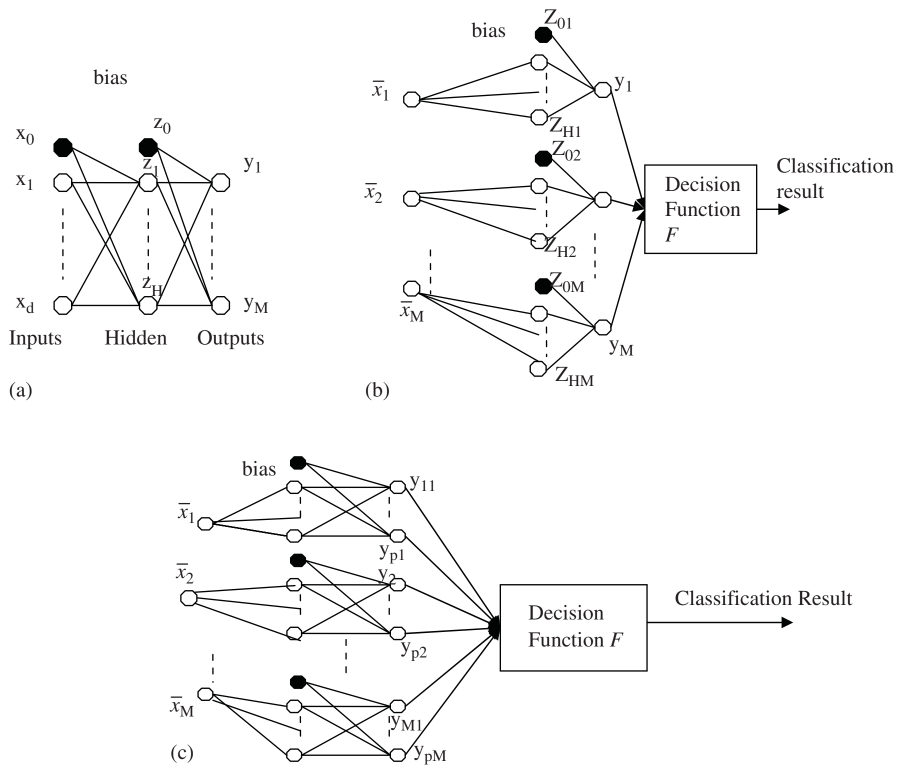

Fig. 1. Different neural network architectures for implementing *K*-class pattern classification: (a) a single neural network for *K*-class pattern classification; (b) *M* binary neural networks used to classify *K* object classes; and (c) a system of multiple neural networks for multi-class pattern classification.

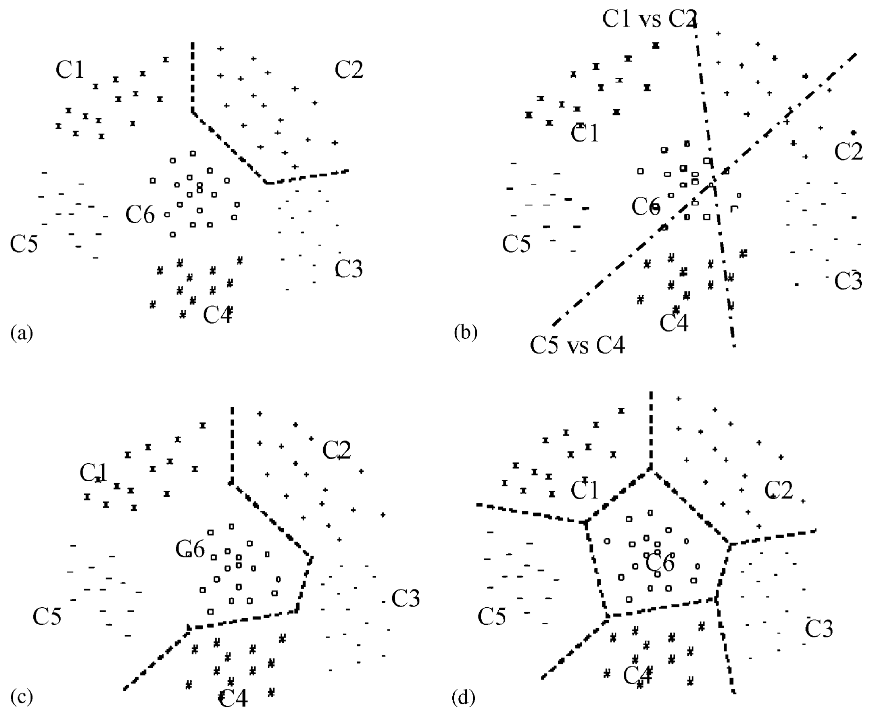

Fig. 2. Illustration of various classification boundaries generated using different training methodologies for *K*-class pattern classification: (a) a classification boundary generated by a neural network trained with a OAA methodology; (b) two classification boundaries generated by two neural networks trained using OAO; (c) a classification boundary generated by a neural network trained with a PAQ methodology; and (d) an optimal classification boundary that separates all six classes in the feature space.

class 2 and all other five classes. However, when we have a system of neural networks trained on OAA, the composite of such classification boundaries is very likely not optimal, which will be discussed in a later section. Fig. 2(b) shows two classification boundaries drawn by two neural networks modeled with OAO, one was trained using class 1 against class 2, and the other class 4 against class 5. To each neural network, its classification boundary was optimal: it provides the largest separated of the two classes it was trained on without making any error. However, if we combine the classification boundaries with data examples of other classes, as shown in this figure, these classification boundaries cut through the regions of other classes, which can cause potential classification errors. Fig. 2(c) shows an optimal classification boundary drawn by a neural network trained with classes 1, 5, and 6 against classes 2, 3, and 4. However, when a neural network system is trained using PAQ, the classification boundaries of these neural networks will likely overlap. Fig. 2(d), which is similar to a figure in Ref. [23], shows an optimal classification boundary for a six-class pattern classification problem, which is possible to be drawn by a single neural network system since it is trained with the presence of the knowledge of all pattern classes. The following two sections discuss various implementations of these modeling schemes in two different neural network architectures: a system of multiple neural networks combined with a decision module, or a single K-class neural network with multiple output nodes, and the advantages and disadvantages of each approach.

3. K-class pattern classification using a system of multiple neural networks

A K-class pattern recognition problem can be implemented in a system of $M > 1$ neural networks. The M neural networks are trained independently using relevant subsets of a given training data set. A decision module is usually needed to integrate the results of M neural networks to produce the final system output. The exact value of M and the training methodology are determined by the modeling scheme. Multiple neural network systems are powerful in the sense that they can implement all three modeling schemes, OAA, OAO, and PAQ.

3.1. *K-class pattern recognition in a system of K neural networks modeled using OAA*

The OAA modeling scheme uses a system of $M = K$ binary neural networks, NN_i, $i = 1, \dots, K$, and each neural network, NN_i has one output node O_i with output function f_i being modulated based on y_i to output $f_i(\bar{x}) = 1$ or 0 to represent whether the input pattern $\bar{x}$ belongs to class i or does NOT belong to class i. Every neural network is trained with the same data set but with different class labels. To train the ith neural network NN_i, the training data Ω_{tr} is decomposed to two sets, $\Omega_{tr} = \Omega^i_{tr} \cup \bar{\Omega}^i_{tr}$, where Ω^i_{tr} contains all the class i examples, which are labeled as "1", and $\bar{\Omega}^i_{tr}$ contains all the examples belonging to all other classes, which are labeled as "0."

There are three possible patterns of output from the K neural networks, $f_1, \dots, f_M$. The first output pattern is the most ideal, $f_i = 1$, and $f_j = 0$ for all j such that $i \neq j$. The decision function F for the system output can be easily made, $F(\bar{x}, f_1, f_2, \dots, f_M) = \arg\max_{i=1,\dots,M}(f_i)$. The second output pattern is that all $f_i =$ " 0" for $i = 1, \dots, M$. In this case the system output should be "don't know." Or the decision function could look at the output of the activation function at each neural network, and output the class label that corresponding neural network that has the largest output value by the activation function at the output node. Mathematically,

$$F(\bar{x}, y_1, y_2, \dots, y_M) = \arg \max_{i=1,\dots,M} (y_i),$$

where y_i is the output of the activation function used in the output layer of the ith neural network. The third output pattern is that more than one of M neural networks output "1". In this case several possible decisions can be made. The simplest system output is to indicate a "tie" among the classes output "1". If a "tie" is not acceptable, the decision function can use the same formula above to output a classification result.

3.1.1. *System analysis*

A system of K binary neural networks trained with OAA has a number of advantages. Since all K neural networks are trained independently, this system architecture provides a lot of flexibility:

- Each neural network can have its own feature space as illustrated in Fig. 1(b). A special feature extraction function can be designed to best fit each neural network.
- Each neural network can have its own architecture such as the number of hidden layers and the number of hidden nodes, activation functions, etc.
- The training of these K binary neural networks can be conducted simultaneously on different computers to speed up the total system training time.

A K binary neural network system has two major drawbacks, it may have problems to learn knowledge of minority classes if the training data are imbalanced, and the system classification boundaries generated by the K binary neural networks may uncover or overlap regions in a feature space, which are being discussed in detail below.

3.1.2. *Imbalanced training data*

One problem associated with a system of M neural networks modeled using OAA is that the training data for individual neural networks can be highly imbalanced. Even when the number of the training samples in each class is

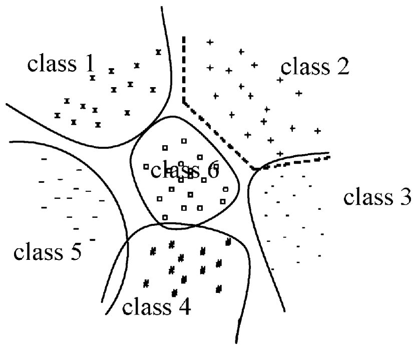

Fig. 3. An illustration of classification boundaries drawn by six binary neural networks trained with OAA.

approximately equal, the ratio of examples in Ω_{tr}^{i} and in $\bar{\Omega}_{tr}^{i}$ is 1:$(K-1)$. When K is large, the training data for neural network i is highly imbalanced, for $i=1,\ldots,M$. This problem is more serious when the training data is imbalanced. The neural network representing a minority class i will have extremely imbalanced data since $|\Omega_{tr}^{i}| \ll |\Omega_{tr}-\Omega_{tr}^{i}|$. Neural learning from imbalanced trained data can result in totally ignoring the minority classes. Our research showed that a neural network trained from imbalanced data using the BP algorithm can be biased toward the majority class, which is the "other classes" in NN_i when training data is noisy [25].

Another problem associated with imbalanced training data is the rate of convergence. Since the majority class contains far more samples than the minority class, the rate of convergence of the neural network output error is very low. This is because the gradient vector computed by the BP algorithm for an imbalanced training set responds slowly to the error generated by the training examples of the minority class [16]. A number of techniques have been proposed to deal with this problem, one type of approaches is to generate extra training examples for the minority class. In Ref. [25], we discussed three different techniques: prior duplications, snowball and Gaussian CPS methods that can be used to boost up the minority training data examples.

3.1.3. *Uncovered and overlapped classification regions in feature space*

One biggest drawback for a system consisting of K neural networks trained with OAA is that the classification boundary of each neural network is drawn independently from others due to the separate training processes. This may result in a situation that a portion of the feature space, which we assume all neural networks use the same feature space, is not covered by any neural networks, which is referred to as an uncovered region in the feature space, or a portion of the feature space is covered by more than one classes, which is referred to as an overlapped region.

Fig. 3 illustrates these two scenarios in a feature space by using an example of classification boundaries generated by six neural networks trained independently using OAA methodology. There are a number of overlapped regions: a region between class 1 and 2, class 2 and 3, class 3 and 4, etc. There are also a number of regions uncovered by neither neural networks: region between class 1 and 5, class 5 and 6, etc. For the feature vectors of the patterns in test data that fall into an overlapped region in the feature space, each of them can be claimed by more than one neural networks as their trained classes, which causes ambiguity. For the feature vectors that fall into the uncovered regions, they are not claimed by any neural networks, and therefore, can be rejected by all neural networks as "other classes." A neural network system that leaves uncovered and overlapped regions in its feature space is not going to generalize well on test data.

3.2. *K-class pattern classification using a system of M neural networks modeled using OAO*

A popular approach to model a K-class pattern classification problem is to decompose it into $K(K-1)/2$ two-class classification problems using the OAO modeling method. This OAO modeling approach, also known as pair-wise method [19,24] or round robin method [23], is very popular among researchers in SVM, Adaboost, decision trees, etc. [12,18,23]. Let us denote these $K(K-1)/2$ two-class neural networks as $NN_m(i,j)$, $1 \leqslant m \leqslant M = K(K-1)/2$, and $NN_m(i,j)$, representing a neural network trained to discriminate class i from class j, for $1 \leqslant i < j \leqslant K$, is trained with data examples of class i and j, and its output, $f_m(i,j)$, is binary indicating whether the input vector $\bar{x}$ is either class i or j. The collective output from these neural networks for $\bar{x}$ represent a combination of $K(K-1)/2$ votes for the K classes and a decision module needs to be designed to decide what class $\bar{x}$ belongs to.

3.2.1. *Decision functions in OAO systems*

Since OAO modeling approach presents abundant redundancies in classification, the posterior decision function can make a significant impact on the final system performance. Research has been active in designing effective decision modules for a multi-class pattern classification system modeled by OAO. These decision functions were originally proposed for machine learning systems in general not specifically for a system of neural networks.

The simplest decision function is a majority vote or max-win scheme. The decision function counts the votes for each class based on the output from the $K(K-1)/2$ neural networks. The class with the most votes is the system output. Friedman showed that in some circumstances this algorithm is Bayes optimal [26]. An extension of the majority vote scheme is to consider the confidence $y_{i,j}$ value for an input vector being class i generated by the activation function [27], $y_{i,j} = g_{i,j}^{o}(a_{i,j}^{o})$, for neural network $NN(i,j)$, where $NN(i,j)$ is the neural network trained with class i against class j for $i=1,\ldots,K-1$, and $j=i+1,\ldots,K$. With these notations the decision function can be written as follows:

$$F(\bar{x}) = \arg\max_p \times \left\{ \left(\sum_{j=p+1}^{K} y_{p,j} + \sum_{j=1}^{p-1} (1-y_{j,p}) \right| p=1,\ldots,K \right\},$$

where, as indicated above, $y_{i,j}$ is the confidence value of $\bar{x}$ being class i and $(1-y_{i,j})$ is the confidence value of $\bar{x}$ being class j for any $i=1,\ldots,K-1$ and $j=i+1,\ldots,K$. We have found this decision function is quite effective, therefore it is used in all our experiments associated with the systems modeled with OAO presented in Section 5.

Many other decision modules have been developed for a system of classifiers modeled by OAO, mostly in the research areas of SVM and decision tress [18,19,27–30]. These modeling approaches can be also implemented in a system of neural networks.

3.2.2. System analysis

The major advantage of the OAO approach is that the independently trained $K(K-1)/2$ binary neural networks provide redundancy to the prediction of pattern classes, which can be used to improve system generalization. In a system of OAO, a pattern class is trained by $K-1$ different neural networks. If one makes a classification mistake, a good decision module may still be able to derive a correct classification based on the outputs of the other neural networks. Similar to a system of K binary neural networks trained with OAA, a neural network system modeled with OAO provides the same flexibility in terms of independent feature spaces, independent neural network architectures, and simultaneous training of multiple neural networks on different computers. However, a system of neural networks trained with OAO does not suffer as much from the imbalanced data. Since each neural network is trained with class i data against class j data, unless these two classes have imbalanced data, the trained system does not suffer by imbalanced data as much as the OAA modeling. Because of the redundancy in the training of pattern classes, the feature space is less likely to have uncovered areas as indicated in the OAA modeling method.

Another important merit of an OAO neural network system is that it has the capability of incremental class learning. Let the $K(K-1)/2$ neural networks be $NN(i,j)$, where $i=1,\ldots,K-1$, and $j=i+1,\ldots,K$. When the existing system is requested to learn a new class, denoted as class $K+1$, we need to train K new neural networks, $NN(i,K+1)$, for $i=1,\ldots,K$ without affecting the existing $K(K-1)/2$ neural networks.

One concern about the OAO approach is the fast growing of the binary neural networks as the pattern classes increase: the number of the binary neural networks grows in the order of K^2. When K is large, the training time of a system modeled with OAO is longer than other systems modeling using OAA or PAQ. Our experiments show that when the pattern classes increase to more than 20, the training time of a system modeled with OAO is indeed much longer than the systems modeled by OAA and PAQ. The long training time is due to the I/O time and network initialization required by the large number of binary neural networks. This finding is a further step from what is shown by Furnkranz in Ref. [23].

3.3. PAQ neural networks

The K-class pattern classification problem can be implemented in a system of M neural networks, and each neural network has binary output trained for P classes against Q classes, where $P \geqslant 1$ and $Q \geqslant 1$. A PAQ modeling scheme can be described by a truth table of K codewords with length M. The content of a codeword can be either 0, 1 or *don't need*. Each bit in a codeword is the output of a two-class neural network trained based on the combination of all the codewords at that bit. Let $f_0, f_1, \ldots, f_{M-1}$ be the output functions of M two-class neural networks, and $cw_1, cw_2, \ldots, cw_K$ the codewords for K pattern classes. To train the jth neural network, its output function $f_j(x)$ is learnt by re-labeling the training examples as $(x_1, f_j(x_1)), (x_2, f_j(x_2)), \ldots, (x_n, f_j(x_n))$, where

$$f_j(x_i) = \begin{cases} 1 & x_i \text{ has codeword } cw_p \text{ and } j\text{th bit of } cw_p \text{ is } 1, \\ 0 & x_i \text{ has codeword } cw_p \text{ and } j\text{th bit of } cw_p \text{ is } 0. \end{cases}$$

If the jth bit of a class codeword is *don't need*, then the training data examples that belong to that class is not included in the training of the jth neural network. As a result of the neural learning, we obtain M neural networks with output function $\hat{f}_j(\bar{x})$, for $j=0,1,\ldots,M-1$. For a test data example $\bar{x}$, the M neural networks collectively produce a binary string of M length, and a decision module is required to produce a system output. All PAQ modeling approaches allow M neural networks in a PAQ system to have their own feature spaces and neural network architectures, and independent training processes.

There are many different PAQ modeling schemes being explored including the hierarchical classification systems developed for multi-class pattern classification [31–33]. A simple type of PAQ neural networks is to encode K classes into $M=\log_2 K$ bits. For an eight-class problem each class is encoded in $M=3$ bits. The problem with this PAQ scheme is that it provides no redundancy in the codewords. Since the minimum Hamming distance between two codewords is 1, any error made by any single neural network will result in a system error. Two more sophisticated PAQ modeling methods are introduced below.

3.3.1. One-against-higher-order modeling

For a K-class pattern classification problem, one-against-higher-order (OAHO) modeling approach trains a system of $(K-1)$ binary neural networks by using the following algorithm. Let K classes be in a list *class_list*$=\{C_1, C_2, \ldots, C_K\}$.

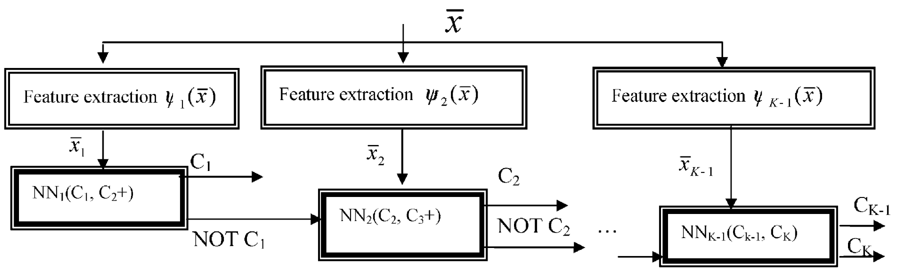
$\bar{x}$
Feature extraction $\psi_1(\bar{x})$
Feature extraction $\psi_2(\bar{x})$
Feature extraction $\psi_{K-1}(\bar{x})$
$\bar{x}_1$
$\bar{x}_2$
$\bar{x}_{K-1}$
NN1(C1, C2+)
NN2(C2, C3+)
NNK-1(CK-1, CK)
C1
NOT C1
C2
NOT C2
CK-1
CK
…

Fig. 4. Classification process in a multi-class pattern classification system modeled by OAHO.

The first neural network $NN_1(C_1, C_2+)$ is trained with examples of class C_1 marked as class "1" and all other classes as class "0", the second neural network $NN_2(C_2, C_3+)$ is trained with examples of class C_2 marked as class "1", and examples of the classes in the higher orders, $C_3, C_4, \ldots, C_K$, are marked as class "0", and in general, neural network $NN_i(C_i, C_i+)$ is trained using the examples from class C_i as class "1" and examples from higher order classes, $C_{i+1}, \ldots, C_K$, as class "0". Fig. 4 illustrates the classification process in an OAHO system. A test data $\bar{x}$ is first sent to the feature extraction functions associated with the individual neural networks to be transformed into the respective feature vectors $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{K-1}$. This allows each neural network to define its own feature space. First neural network $NN_1(C_1, C_2+)$ is activated. If it predicts $\bar{x}_1$ as class C_1 the system outputs the result and stops the process. Otherwise, $NN_2(C_2, C_3+)$ is activated. If this neural network makes a prediction, then the system stops, otherwise, the process goes on as illustrated in Fig. 4. When the process continues to the last neural network, $NN_{K-1}(C_{K-1}, C_K)$, a classification result is produced to indicate whether $\bar{x}$ belongs to class C_{K-1} or C_K.

The classification process in an OAHO system is hierarchical, which implies that if any neural network system in the hierarchy makes a prediction mistake, it is a system error that cannot be corrected in any further processes. Therefore the neural network systems at higher levels in the OAHO hierarchy should be designed and trained to be as reliable as possible. The OAHO modeling can be very effective if the constraint used to order the classes in *class_list* is well defined to meet the need of a particular application problem. In general the classes can be ordered either randomly, based on training data properties, importance of each class or prediction of accuracy for each class. We introduce an OAHO modeling designed to reduce the impact of imbalanced training data. In order to minimize the impact of imbalanced training data, we order the K classes based on the size of the available training examples in each class such that *class_list* $= \{C_1, C_2, \ldots, C_K\}$ if and only if $|\Omega^i_{tr}| \leqslant |\Omega^{i-1}_{tr}|$, $i = 1, 2, \ldots, K - 1$, where Ω^i_{tr} is the training data of class C_i, for $i = 1, 2, \ldots, K$. In this modeling, the classes with smaller training sizes are used together as negative training examples against the examples of a single larger class in the training of the neural networks at the higher levels of the hierarchy. Statistically it reduces the impact of imbalanced training data as we discussed in Section 3.1.

Another important feature of OAHO modeling scheme is that it has the capability of incrementally learn a new pattern class based on an already trained neural network system. Let a new class be C_0. The $K + 1$ pattern classification system is exactly the same as in Fig. 4 with a newly trained neural network, $NN(C_0, C_1+)$ being added at the highest level.

3.3.2. Error-correcting output code (ECOC)

A PAQ modeling can provide redundancy to give error tolerance by using more neural networks than the number of pattern classes, i.e. $M > K$. Based on the information theory, the redundancy embedded in codewords gives the variance in the input feature vectors [32], which makes the classifier less sensitive to noise and generalize better. One type of such approaches is referred to as the error-correcting output code (ECOC) decomposition [2,32,34,35]. In an ECOC approach each class is assigned a binary string of M bits referred to as codeword, $M > K$. An ECOC scheme can be represented in a $K \times M$ table, where all the classes and their associated codewords are listed in K rows, and the bit positions of all codeword are in M columns. Dietterich and Bakiri proposed a 15-bit ECOC scheme [35] for the 10-class digit recognition problem, in which each class is assigned a unique 15 bit codeword (a row of the table), and a total of 15 neural networks were trained to implement the ECOC.

A good ECOC should have both good row and column separation. A good row separation implies that each codeword is well separated in Hamming distance from every other codeword. A good column separation implies that each neural network output, f_i should be uncorrelated with any of other neural network outputs, f_j such that $i \neq j$. If two columns i and j were similar or identical, the neural networks with output equal to f_i and f_j would make similar or correlated mistakes. In summary the column separation condition attempts to ensure that columns are neither identical nor complementary. Algorithms for constructing good ECOCs can be found in Ref. [35]. Dietterich pointed out that ECOC can make a machine learning system to have better generalization capability and be less sensitive to noise by increasing the minimum Hamming distance between the codewords. Dietterich and Bakiri showed that the neural network system modeled by their

ECOC approach out-performed the standard OAO BP neural networks [32,35].

4. K-class pattern classification using a single neural network

A K-class pattern classification problem can be implemented in a single neural network with an architecture of d input nodes and M output nodes (see Fig. 1(a)), where d is the dimension of an input feature vector and M is the number of output nodes in the neural network system. The number of output nodes $M > 1$ is determined by the number of bits in codewords representing the K classes. Here we want to point out that it is possible to model the K-class with a neural network of one output which is modulated to produce K different values, each presents a class. However, this is in generally not considered as good in generalization and therefore is not very popular. In this paper we only analyze the neural network architectures that have $M > 1$ outputs. The modeling approaches OAA and PAQ discussed in Section 3 are applicable to the single neural network implementation, but not OAO. The following subsections describe the implementation of these modeling approaches in a single neural network system.

4.1. A single neural network trained with OAA scheme

When a K-class single neural network, *NN*, is modeled by the OAA scheme, NN has $M = K$ output nodes, each is denoted as $O_1, O_2, \ldots, O_K$. Every pattern class is encoded in a codeword of K bits, and the codeword for the ith class is, $O_1 = \ldots O_{i-1} = 0$, $O_i = 1$, $O_{i+1} = \ldots, O_K = 0$, for $i = 1, \ldots, K$. For a training data example $\bar{x}$, the expected output of the neural network at the output node f_i is set to 1 if and only if $\bar{x}$ belongs to class i, otherwise it is set to 0, for $i = 1, \ldots, K$. Since we use only one neural network to model multiple classes, the architecture of the neural network should have higher degree of complexity than those used in the binary neural networks. During the pattern classification stage, a test example $\bar{x}$ is assigned a class label whose binary codeword has the smallest Hamming distance to the neural network output $F(\bar{x})$.

This single neural network system modeled by OAA shares many features with its counter part: a system of multiple binary neural networks modeled by OAA. For example it also ignores the minority classes when the training data are imbalanced since the output node corresponding to a minority class was set to “1” in much less time than a node corresponding to a majority class. However, it is different from the multiple neural network systems in the following aspects.

The presence of training data from all classes and the updates of all weights related to all classes during the neural learning process for a single neural network provide an opportunity to train a neural network with an optimal decision boundary such as the one shown in Fig. 2(d). The single neural network architecture, as pointed out by Caruana [36], allows features developed in the hidden layer being shared by the multiple classes, and some hidden units being specialized for just one or a few classes, which can be ignored by other classes by keeping the weights connected to them small. If trained properly, the uncovered and overlapped regions in a feature space can be minimized. However, in order to achieve optimal classification boundaries as those illustrated in Fig. 2(d), we need not only effective neural network modeling but also effective neural network architecture and learning process. We will show in Section 5 that a single neural network modeled with OAA is a good architecture when K is small and the training data size is not too large. In general the training process for a single neural network modeled with OAA is easy to handle, since there is only one training data set and one neural network to train. However, the training time will be very long when the training data are many and the number of pattern classes is large, which makes the fine tuning of the neural network architecture and learning parameters difficult.

4.2. A single neural network trained with PAQ scheme

The PAQ scheme can also be implemented in a single neural network with M output nodes, where $M > K$. The output functions at the output nodes, $f_1, \ldots, f_M$, collectively represent the codewords assigned to different pattern classes. For example, for the 15-bit ECOC scheme for the 10-class handwritten digit recognition [35], a single neural network with 15 output nodes can be constructed to implement this ECOC modeling. For a training data example $\bar{x}$, the neural network's output functions output nodes are set to the codeword of the class to which $\bar{x}$ belongs. During the pattern classification stage, a test example $\bar{x}$ is assigned a class label whose binary codeword has the smallest Hamming distance to the neural network output $F(\bar{x})$.

This neural network architecture has similar characteristics as the system architecture modeled using OAA in a single neural network. However, it will require even more training time since it has more output nodes as evidenced in our experiments (see Section 5). In general a single neural network system modeled using PAQ is not recommended when the number of pattern classes is large and/or training data are many.

4.3. Summary of single neural network systems for K-class pattern classification

A single neural network with multiple output nodes can implement both OAA and PAQ modeling approaches. As discussed earlier, a single neural network system has the capability of being trained to produce optimal classification boundaries for K-class pattern classification for $K > 2$. Its training process is easy to handle, however it requires that all pattern classes use the same feature space. In some application problems such as text document categorization, the feature vector dimension for all K classes is much larger than the dimension of the feature spaces for individual classes. A higher dimension in feature space generally requires higher complexity in a neural network architecture, which is already high in a single neural network system that has the number of output nodes K. As we will show in the experimental section the training time is very high for a single neural network system in an application that has a large number of pattern classes, high dimension feature space and large training data. Since there is no general rule to select the number of hidden nodes/layers in a neural network for a given application problem, in most practices a neural network architecture is determined by a try-and-error strategy. If the training time takes weeks, it is not likely an appropriate architecture can be found easily for a neural network system.

5. System performance analysis

In this section, we first analyze time complexity for the training of various neural network systems used for multi-class pattern classification, and then present the experimental results of various multi-class neural network classification systems conducted on well-known benchmark data. All our discussion builds on a base architecture of one hidden layer neural networks with feed-forward BP learning. We have implemented neural network systems modeled using OAA and PAQ in single neural network systems, OAO, OAA and PAQ in multiple neural network systems. The experiments were designed to evaluate the following learning capabilities of various neural network systems:

- imbalanced training data,
- large number of pattern classes,
- small vs. large training data sets.

5.1. Time complexity of neural learning for multi-class pattern classification

This section analyzes the computational complexity in training a multi-class neural network system modeled using either OAA, OAO or PAQ. The computational cost for training a multi-class neural network system is determined by the five factors: the number of neural networks within the system, the input feature dimension, d, the number of hidden nodes, H, and output nodes, O, in each network, the number of training data, N, and the number of epochs needed for the training process to converge. Note the number of pattern classes K is related to the number of neural networks, and sometimes the number of the output nodes. Since our intention is to compare the training time complexity among different neural network architectures, our analysis is focused on the number of multiplications required in training each system per epoch. Let W^h be the total number of weights from the input to the hidden layer, and W^o the number of weights from the hidden to the output layer. We have, assuming the hidden layer has a bias input,

$$\begin{aligned}W^h &= d * H + H = H(d+1),\\ W^o &= H * O + O = O(H+1) \quad \text{and}\\ W &= W^h + W^o.\end{aligned}$$

For each data example in Ω_{tr}, the BP algorithm requires a forward calculation and backward calculation. During the forward calculation, each training data example requires $W = W^h + W^o$ multiplications. It can also be shown that the backward weight update for each input training example requires $2 * W + W^o$ multiplications. In summary the number of multiplications for the training of a neural network in terms of weights for a training date size of N is

$$\Gamma^{weights}(W^o, W^h, N) = (3 * W + W^o) * N. \qquad (1)$$

The computational complexity for the training of a neural network can be expressed in terms of the dimension of feature space d, number of hidden nodes H, output nodes O, and the number of training data examples, N, as follows:

$$\begin{aligned}&\Gamma^{nodes}(d, H, O, N)\\ &\quad = \{3 * [H * (d+1) + O * (H+1)]\\ &\qquad + O * (H+1)\} * N\\ &\quad = [3 * d * H + 4 * O * (H+1) + 3 * H] * N. \qquad (2)\end{aligned}$$

From Eq. (2), we derived formulas that describe the training time complexity of neural network systems modeled by OAA, OAO and PAQ, which are summarized in Table 1. The single neural network modeled by PAQ with a binary coding scheme requires the least training time for K classes, and the multiple neural networks modeled by the binary coding is even better if $H^{1\text{-}nets}_{binary} > H^{multi\text{-}nets}_{binary} * \log_2 K$. In general practice it is likely $H^{1\text{-}net}_{OAA} \gg H^{k\text{-}nets}_{OAA} > H_{OAHO} > H_{OAO}$. In this case the single neural network modeled with OAA requires the most training time, and the neural network system modeled with OAO requires the least. However, we want to point out that in implementation, the training of each neural network requires an additional amount of time for overhead operations such as I/O and creating the network. When the number of pattern classes becomes large, this overhead can be significant, which will be shown in the experiment results conducted on data sets with large number of pattern classes.

5.2. Performance evaluation of multi-class neural network systems on imbalanced data

A training data is imbalanced if the number of the training data examples in at least one class is much less than the examples in another class or classes. We derived the following imbalance measures, β^{OAO}_{Ω} and β^{OAA}_{Ω}, on a given data set

Table 1
Computational complexity analysis of multi-class pattern classification systems

Neural network system	Number of multiplications required in training per epoch
A single neural network modeled using OAA	$C_{OAA}^{1\text{-}net} = [3 * d * H_{OAA}^{1\text{-}net} + 4 * K * (H_{OAA}^{1\text{-}net} + 1) + 3 * H_{OAA}^{1\text{-}net}] * N$
A system of K neural networks modeled using OAA	$C_{OAA}^{k\text{-}nets} = [3 * d * H_{OAA}^{k\text{-}nets} + 7 * H_{OAA}^{k\text{-}nets} + 4] * K * N$
A system of multiple neural networks modeled using OAO	$C_{OAO} = [3 * d * H_{OAO} + 7 * H_{OAO} + 4] * (K - 1) * N$
PAQ: binary coding Single neural network	$C_{binary}^{1\text{-}net} = [3 * d * H_{binary}^{1\text{-}net} + 4 * \log_2 K * (H_{binary}^{1\text{-}net} + 1) + 3 * H_{binary}^{1\text{-}net}] * N$
PAQ: binary coding Multiple neural networks	$C_{binary}^{multi\text{-}nets} = 3 * d * H_{binary}^{multi\text{-}nets} + 7 * H_{binary}^{multi\text{-}nets} + 4] * \log_2 K * N$
PAQ: OAHO	$C_{OAHO} < [3 * d * H_{OAHO} + 7 * H_{OAHO} + 4] * \frac{K+1}{2} N$

Ω used in the training of a neural network system modeled with OAO and OAA, respectively:

$$\beta_{\Omega}^{OAO} = \frac{\min\{n_i \mid i = 1, \ldots, K\}}{\max\{n_i \mid i = 1, \ldots, K\}} \quad \text{and}$$

$$\beta_{\Omega}^{OAA} = \min\left\{ \frac{n_i}{\sum_{j=1,\ldots,K, j \neq i} n_j} \middle| i = 1, \ldots, K \right\},$$

where n_i and n_j denote the number of data examples in class i and j in Ω. When $\beta_{\Omega}^{OAO} \to 1$, Ω is well balanced, and when $\beta_{\Omega}^{OAO} \to 0$, Ω is most imbalanced. The maximum value for B_{Ω}^{OAA} is $1/(K - 1)$, which indicates that Ω is equally distributed over K classes. When $B_{\Omega}^{OAA} \to 0$, the data set is most imbalanced.

We use two data sets, "Glass" and "Shuttle", obtained from the UCI database, to evaluate the effectiveness of the neural network systems trained on imbalanced data. The distributions of these two data sets are shown in Table 2. The imbalance measures on the *Shuttle* data are $\beta_{shuttle}^{OAO}=0.00018$ and $\beta_{shuttle}^{OAA} = 0.00014$, which indicate that both the systems modeled with OAO and OAA have highly imbalanced training data. On the "Glass" data, we have $\beta_{Glass}^{OAO} = 0.12$ and $\beta_{Glass}^{OAA} = 0.044$, which also indicate an imbalance, but are not as bad as the *Shuttle* data.

We implemented four neural network systems: an OAO system; an OAA system of binary neural networks; a single OAA neural network; and an OAHO neural network system. The performances of these four neural network systems on "Glass" and "Shuttle" are shown in Tables 3 and 4, respectively. For the "*Glass*" collection, we experimented with all four neural network systems with various numbers of hidden nodes ranging from 3 to 30. The results shown in Table 3 are the best performances generated by the OAO system that contains 15 binary neural networks with five hidden nodes in each: the single OAA system that has 20 hidden nodes and six output nodes; the OAHO system that contains five binary neural networks with five hidden nodes in each; and the OAA system of six binary neural networks with five hidden nodes in each. Since the UCI site does not provide a separate test set, the performances listed in Table 3 were obtained through a 10-fold cross validation process. In this collection, the class gives the most trouble to all neural network systems is a minority class, class 3. Although it has more training examples than class 4 and 5, class 3 in the validation sets was completely ignored by all neural network systems except the OAHO system, which gave a recognition rate of 17.65%. Based on our earlier research [25], it is likely that training data in class 3 contain more noise than the other minority classes. For the overall performance the OAA system of six binary neural networks gave the best performance, and the OAO system the worst.

For the "*Shuttle*" collection, we experimented with all four neural network systems with various numbers of hidden nodes ranging from 3 to 30. Table 4 lists the neural network systems that gave the best performances: the OAO system that contains 21 binary neural networks with 10 hidden nodes in each; the OAA system of seven binary neural networks with 20 hidden nodes in each; the single OAA system that contains 30 hidden nodes; the OAHO system that contains six neural networks such that NN_0 with 30 hidden nodes was trained by class 1 against {class 2, 3, 4, 5, 6, 7}, NN_1 with 20 hidden nodes was trained by class 4 against {class 2, 3, 5, 6, 7}, NN_2 with 15 hidden nodes was trained by class 5 against {class 2, 3, 6, 7}, NN_3 with 10 hidden nodes was trained by class 3 against {class 2, 6, 7}, NN_4 with 10 hidden nodes was trained with class 2 against {class 6, 7}, and NN_5 with 10 hidden nodes was trained with class 7 against class 6. Although the OAO system gave the best overall performance, the OAHO system gave the best performance on the minority classes 2, 3, 6 and 7. The other three neural network systems completely failed to recognize the minority classes 2, 6 and 7. The system modeled using OAO did better than the two systems modeled with OAA on the minority class 3.

To explore further on learning from imbalanced data, we implemented two additional approaches to solve the imbalanced data problem, one is a prior duplication of training data in minority classes and the other is the Snowball approach. The experiments are conducted on the "Shuttle" data. The prior duplication approach simply repeated the training examples in minority classes 2, 3, 6 and 7 a number of times that resulted in 3700 examples in class 2, 3960 in class 3, 2688 in class 6 and 2970 in class 7. The description of the snowball approach can be found in Ref. [25].

Table 2
Training and test data distribution in "Glass" and "Shuttle"

	Class 1	Class 2	Class 3	Class 4	Class 5	Class 6	Class 7
Glass/training	70	76	17	13	9	29	N/A
Shuttle/training data	34,108	37	132	6748	2458	6	11
Shuttle/test data	11,478	13	39	2155	809	4	2

Table 3
Summary of performances on *Glass* collection

	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	Total (%)
OAA, 6 nets	84.29	69.74	0	69.23	66.67	86.21	71.03
OAO, 15 nets	68.57	68.42	0	61.54	66.67	82.76	64.95
OAHO, 5 nets	85.71	57.89	17.65	69.23	55.56	82.76	67.76
OAA, 1 net	84.29	69.74	0	69.23	66.67	86.21	70.56

Table 4
Summary of performances on test data of the *Shuttle* collection

	1 (%)	2 (%)	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)	Total (%)
OAA, 7 nets	100	0	15.38	100	99.75	0	0	99.63
OAO, 21 nets	100	0	33.33	100	100	0	0	99.69
OAHO	99.99	7.69	38.46	100	99.75	0	0	99.69
OAA, 1 net	100	0	12.82	100	99.88	0	0	99.63
OAA, 1 net with prior duplication	99.94	30.77	64.1	100	99.93	100	100	99.77
OAA, 1 net, snowball	100	92.31	94.87	100	99.88	100	100	99.97

The experiments were conducted using a single neural network system and the results shown in Table 4 indicate that these two approaches can indeed boost up the recognition rate on the minority classes. The snowball method gave the best performance: it boosted the performances on the minority classes without any loss on the other classes. Its overall performance has surpassed the performances generated by the SVM presented in Ref. [12], and is very close to the best performance, 99.99%, posted at the UCI website.

5.3. Evaluation of multi-class neural network systems on large number of pattern classes

To evaluate the learning capabilities of different neural network systems on large number of pattern classes, we conducted experiments on two problems: 10-class handwritten digit recognition and 26-class English letter recognition, and the results are described as follows.

5.3.1. Handwritten digit recognition

The data collection provided by NIST (http://yann.lecun.com/exdb/mnist/index.html) contains 60,000 gray scale images of handwritten digits in the training set and 10,000 in the test set. In addition to the four neural networks modeled by OAO, OAA multi-nets, OAA 1-net and OAHO, we implemented a single neural network and a system of binary neural networks both modeled using the ECOC method described in Ref. [35]. All these neural networks use the same feature vectors of 49 dimensions calculated from 7×7 grids superimposed on each image. Each element in the feature vector is the average grayscale value within one grid. For all the binary neural networks we tried different numbers of hidden nodes ranging from 10 to 60, and for the single neural network systems we tried different number of hidden nodes ranging from 100 to 600. Table 5 lists the best performances by the neural network systems in the six categories: the OAO system that contains 45 binary neural networks with 20 hidden nodes in each; the OAHO system that contains nine binary neural networks with 20 hidden nodes in each; the OAA system of 10 binary neural networks 15 hidden nodes in each; and the single OAA system that contains 400 hidden nodes and 10 output nodes; the ECOC system of 15 binary neural networks with 15 hidden nodes in each; and the single ECOC system that contains 300 hidden nodes and 15 output nodes. It appears that the OAO system gave the best performance. Among the two single neural network systems, the OAA system gave better performance than the ECOC system. In comparison to the benchmark performances, the OAO neural network system and the single OAA neural network have outperformed all of the 2 or 3-layered neural network systems posted at http://yann.lecun.com/exdb/mnist/index.html that did not use special pre-processing. The training time was obtained on a PC Pentium II 450 MHz with 512 MB RAM.

Table 5
System performances of neural networks on 10-class digit recognition problem

	ECOC 15 nets	OAA 10 nets	OAO 45 nets	OAHO 9 nets	ECOC 1 net	OAA 1 net
Accuracy (%)	96.4	96.50	97.54	96.24	96.85	97.39
Training time (h)	6.5	4.2	7	18.1	30	23

Table 6
Performances of four neural networks on the "Letter" collection

	OAA 26 nets	OAO 325 nets	OAHO 25 nets	OAA 1 net
Classification accuracy (%)	86.34	93.96	89.58	87.38
Training time (h)	2	2.3	1.16	0.45

Table 7
Classification accuracy of four neural network systems on three small data collections

	OAA 3 nets (%)	OAO 3 nets (%)	OAHO 2 nets (%)	OAA 1 net (%)
Iris	96	98	96	98
Wine	98.31	97.75	98.31	98.31
Vehicle	84.75	83.69	84.16	83.92

The training time for each of the single neural network systems, which have far more hidden nodes, is significantly more than any of those systems containing multiple neural networks. The 1-net ECOC system took the most time, and the 10-net OAA system took the least. The system modeled with OAO gave the best recognition rate and its training time is reasonable.

5.3.2. UCI letter database

The "Letter" database obtained from the UCI website has a training set of 15,000 and a test set of 5000 data examples distributed rather evenly among the 26 classes. We implemented four neural network systems: an OAO system of 325 binary neural networks; a OAA system of 26 binary neural networks; a OAHO system of 25 binary neural networks; and a single OAA neural network. We experimented with various numbers of hidden nodes ranging from 10 to 60 for all four neural network systems. Table 6 presents the performances of the best neural network systems in the four categories: each of the binary neural networks in the OAO system; the OAA system or the OAHO system has 15 hidden nodes; and the single OAA neural network has 40 hidden nodes. The OAO system has a strong lead in classification accuracy. All these experiments were conducted on a PC which is a Pentium Centrino with 1.6 GHz and 512 MB RAM. The OAA with a single neural network took the least training time, the OAO system took the most. This is because the overhead time, which includes the time for I/O, creating the networks and initializing the weights, needed for all 325 neural networks is significant.

Based on the discussions above, it appears that OAO is better suited for the problems with large number of pattern classes. When the pattern classes increase to a very large number, the training time for an OAO system be a quadruple of the training time for a single neural network system due to the overhead required for the large number of neural networks in an OAO system. When the training data size is large such as in the 10-class digit recognition problem, the overhead time for the training of an OAO system is less significant in comparison to the computational time required for weight updates.

5.4. Performances of neural network systems trained on small and large data sets

One concern in machine learning community is that a system trained on small samples may not perform well on test data. On the other hand, if training data sets are too large, our concern is how well and efficiently a system can learn. The objective of this study is to find what neural network systems are better suited for applications that have small or large training data.

For studying neural learning from small training data we chose three data collections, *iris*, *wine*, and *vehicle* from the UCI databases. All three collections have rather balanced distribution among all classes, and the number of pattern classes is not too large. The *iris* collection contains 50 data examples for each of its three pattern classes. The *wine* collection has 59 data examples for class 1, 71 for class 2 and 48 for class 3. The *vehicle* collection has 240 data examples for each of class 1, 2 and 3, and 226 data examples for class 4. Since there were no separate test data, performances shown in Table 7 were obtained through a 10-fold cross validation process. For the *iris* collection, each neural network system has been tried with hidden nodes ranging from 5 to 30. The best results presented in Table 7 are generated by an

Table 8
Performances of four neural network systems on a large training data set

	OAA 3 nets	OAO 3 nets	OAHO 2 nets	OAA 1 net
Classification accuracy (%)	65.64	80.35	48.88	65.14
Training time (h)	21	4.32	30.6	90

OAA system of three binary neural networks with 10 hidden nodes in each, an OAO system of three binary neural networks with five hidden nodes in each, an OAHO system of two binary neural networks with 10 hidden nodes in each, and the single OAA neural network with 20 hidden nodes. The single OAA neural network and the OAO system gave the best performances. For the *wine* collections, the best results were generated from the following system architectures: each of the binary neural networks used in the systems of OAA, OAO and OAHO contains five hidden nodes, and the single OAA neural network contains 10 hidden nodes. The OAO system is slightly behind the other three systems. For the *vehicle* collection, each neural network system has been tried with different number of hidden nodes ranging from 10 to 40. The best performances listed in Table 7 were generated by the following system architectures: each of the binary neural networks contained in the systems modeled with OAA, OAO and OAHO contains 15 hidden nodes, and the single OAA neural network has 40 hidden nodes. The best performance was given by the 1-net OAA and the 3-net OAA system. Based on these three experiments we conclude that the single OAA neural network systems generally perform well for small training data sets, its training procedure is simple, and the training time is nominal.

To evaluate the learning capabilities of neural network systems from large training data, we chose to use data extracted from a well-known protein data bank (PDB) [37]. Let a protein sequence be $p = p_1, p_2, \ldots, p_m$, where p_i is an amino acid whose secondary structure can be categorized into three consolidated classes: helix, strand and coil [6]. The training data set contains 110,530 data examples extracted from 300 protein sequences and the test data set contains 110,530 data examples extracted from 100 protein sequences. A data example consists of 19 amino acids and is represented by a feature vector of 95 dimensions [8]. Table 8 shows the best performances of the neural networks in each category: each of the three binary neural networks in the OAO system contains 40 hidden nodes; each of the three binary neural networks in the OAA system contains 20 hidden nodes; the neural network in OAHO modeled with class 1 against class 2 and 3 contains 60 hidden nodes and the one modeled with class 2 against class 3 contains 30 hidden nodes; and the single OAA neural network contains 60 hidden nodes. The OAO neural network system gave over 15% more accuracy than the next best system, the 3-net OAA system. The OAO system's performance even surpassed the published prediction accuracy range of 60–75% [6,7]. Since the training set is huge, training time is very much related to the number of epochs used in the training. For a fair comparison, we used 1000 epochs in training all these systems running on the same PC, which is a Pentium Centrino 1.6 GHz with 512 MB RAM. The OAO system required the least training time, which is expected since it has only two binary neural networks and each uses only two classes of training data. Our conclusion is that when training data are huge, an OAO neural network system is the best.

6. Conclusions

We have presented an in-depth study on *K*-class pattern classification using neural networks. Specifically, we discussed two different architectures, systems of multiple neural networks and single neural network systems, and three types of modeling approaches, OAA, OAO, and PAQ. We showed through theoretical analysis and experiments that these different system implementations have their own strength over different application problems along the directions of system generalization within feature space, incremental class learning, learning from imbalanced data, large number of pattern classes, and small and large training data. We also presented training time complexity analysis for each system, and comparative performance analysis based on well-known benchmark data sets. Our findings are summarized as follows:

- A single neural network system can be modeled using either OAA or PAQ but not OAO. But a system of multiple neural networks can be modeled using either OAA, PAQ or OAO.
- A *K*-class pattern classification problem implemented in a system of multiple neural network classifiers modeled by either OAO, OAA, OAHO or ECOC share the following advantages:
    - ○ Individual neural networks are likely to be simpler than a single neural network system for *K* classes, they can be trained and operated simultaneously on distributed computers.
    - ○ The individual neural networks can be modeled independently. Different neural networks can have different feature spaces, feature dimensions and system architectures.
    - ○ The architectures of individual neural networks can be changed and the individual neural networks be retrained without affecting the other neural networks in the same system.
- A neural network system modeled with either OAO or OAHO has the capability of incremental class learning. The new neural networks trained with new class data can be added to the existing neural network system without affecting the other neural networks in the system.

- A major drawback associated with the architecture of multiple neural networks is that individual neural networks are trained only on local knowledge, namely knowledge of relevant pattern classes. This can result in ambiguity and/or uncovered feature space regions.
- A single neural network for *K*-class pattern classification provides a neural learning process with all *K*-class information, which can result in, in theory, an optimal classification if the feature space is well defined. Our experiments showed this system architecture performed well when the training data set is not too large and the pattern classes are not too many. When the training data set is large, the number of hidden nodes in a single neural network needs to be increased significantly, which results in high training time.
- OAHO can be designed to be effective in learning from imbalanced training data. For other types of modeling techniques such as OAA performances on minority classes can be boosted up by using additional pre-processing methods such as snowball.
- An OAO system coupled with a powerful posterior decision unit is very effective in learning from large training data and large number of pattern classes.
- Since the number of binary neural networks in an OAO system increases in a quadratic rate with *K*, when *K* is large, the training time for the OAO system can be very high due to the increased overhead time needed to train the large number of the neural networks.

Acknowledgment

This work is supported in part by a URP grant from the Ford Motor Company.

References

[1] Y. Le Cun, B. Boser, J. Denker, D. Hendersen, R. Howard, W. Hubbard, L. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[2] D. Aha, R. Bankert, Cloud classification using error-correcting output codes, Artif. Intell. Appl. Nat. Resour. Agric. Environ. Sci. 11 (1) (1997) 13–28.
[3] Y.L. Murphey, Y. Luo, Feature extraction for a multiple pattern classification neural network system, IEEE International Conference on Pattern Recognition, August 2002.
[4] E. Brill, Some advances in transformation-based part of speech tagging, In: AAAI Conference, vol. 1, 1994, pp. 722–727.
[5] Y. Even-Zohar, D. Roth, A sequential model for multiclass classification, in: SIGDAT Conference on Empirical Methods in Natural Language Processing, 2001, pp. 10–19.
[6] P. Baldi, B. Pollastri, A machine learning strategy for protein analysis, IEEE Intell. Syst. 17 (2) (2002) 28–35.
[7] A. Clare, R.D. King, Knowledge discovery in multi-label phenotype data, in: Fifth European Conference on Principles of Data Mining and Knowledge Discovery, Springer, Berlin, 2001.
[8] B. Zhang, Z. Chen, Y.L. Murphey, Protein secondary structure prediction using machine learning, IEEE International Joint Conference on Neural Networks, July 2005.
[9] C. Apte, F. Damerau, S.M. Weiss, Automated learning of decision rules for text categorization, Inf. Syst. 12 (3) (1994) 233–251.
[10] B.K. Natarajan, Machine Learning: A Theoretical Approach, Morgan Kaufmann, Los Alamitos, CA, 1991.
[11] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[12] C. Hsu, C. Lin, A comparison of methods for multiclass support vector machines, IEEE Trans. Neural Networks 13 (2) (2002) 415–425.
[13] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, Oxford, 1995.
[14] D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in: D. E. Rumelhart, J. L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructures of Cognition, vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.
[15] M. Anthony, P. Bartlett, Neural Network Learning: Theoretical Foundations, Cambridge University Press, Cambridge, UK, 1999.
[16] R. Anand, K. Mehrotra, C.K. Mohan, S. Ranka, Efficient classification for multiclass problems using modular neural networks, IEEE Trans. Neural Networks 6 (1995) 117–124.
[17] E. Gelenbe, K.F. Hussain, Learning in the multiple class random neural network, IEEE Trans. Neural Networks 13 (6) (2002) 1257–1267.
[18] E.L. Allwein, R.E. Schapire, Reducing multiclass to binary: a unifying approach for margin classifiers, J. Mach. Learn. Res. 1 (2000) 113–141.
[19] D. Price, S. Knerr, Pairwise neural network classifiers with probabilistic outputs, Neural Inf. Process. Syst. 7 (1994).
[20] B. Lu, M. Ito, Task decomposition and module combination based on class relations: a modular neural network for pattern classification, IEEE Trans. Neural Networks 10 (5) (1999) 1244–1256.
[21] E. Allwein, R.E. Schapire, Y. Singer, Reducing multiclass to binary: a unifying approach for margin classifiers, in: Proceedings of the 17th International Conference on Machine Learning, 2000.
[22] K. Crammer, Y. Singer, On the learnability and design of output codes for multiclass problems, in: Computational Learning Theory, 2000, pp. 35–46.
[23] J. Furnkranz, Round robin classification, J. Mach. Learn. Res. (2) (2002) 721–747.
[24] T. Hastie, R. Tibshirani, Classification by pairwise coupling, in: Conference on Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 1998, pp. 507–513.
[25] Y.L. Murphey, H. Guo, L.A. Feldkamp, Neural learning from imbalanced data, Appl. Intell. Neural Networks Appl. 21 (2) (2004) 117–128.
[26] W. Maass, On the computational power of winner-take-all, Neural Comput. 12 (11) (2000) 2519–2536.
[27] S. Knerr, L. Personnaz, G. Dreyfus, Single-layer learning revisited: a stepwise procedure for building and training a neural network, in: Neurocomputing: Algorithms, Architectures and Applications, NATO ASI Series, Springer, Berlin, 1990.
[28] J.C. Platt, N. Cristianini, Large margin DAGs for multiclass classification, Advances in Neural Information Processing Systems, 12th ed., 2000, pp. 547–553.
[29] S. Har-Peled, D. Roth, Constraint classification: a new approach for multiclass classification and ranking, in: 13th International Conference on Algorithmic Learning Theory, 2002, pp.365–379.
[30] T.-F. Wu, C.-J. Lin, R.C. Weng, Probability estimates for multi-class classification by pairwise coupling, J. Mach. Learn. Res. 5 (2004) 975–1005.
[31] H. Blockeel, M. Bruynooghe, S. Dzeroski, J. Ramon, J. Struyf, Hierarchical multi-classification, in: First International Workshop on Multi-Relational Data Mining (KDD-2003), 2003.
[32] E.B. Kong, T.G. Dietterich, Error-correcting output coding corrects bias and variance, in: Proceedings of the 12th International Conference on Machine Learning, 1995, pp. 313–321.

[33] K. Wang, S. Zhou, S.C. Liew, Building hierarchical classifiers using class proximity, in: 25th International Conference on Very Large Data Bases, Morgan Kaufmann, Los Alamitos, CA, 1999, pp. 363–374.
[34] G. James, Majority vote classifiers: theory and applications, Ph.D. Thesis, Stanford University, 1998.
[35] T.G. Dietterich, G. Bakiri, Solving multiclass learning problems via error-correcting output codes, J. Artif. Intell. Res. (2) (1995) 263–286.
[36] R. Caruana, Multitask learning, Mach. Learn. 28 (1997) 41–75.
[37] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, The protein data bank, Nucleic Acids Res. 28 (2000) 235–242.