Decision Support Systems 52 (2012) 657–663

Contents lists available at SciVerse ScienceDirect

Decision Support Systems

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

Multiple costs based decision making with back-propagation neural networks

Guang-Zhi Ma [a], Enmin Song [a], Chih-Cheng Hung [b,*], Li Su [a], Dong-Shan Huang [a]

[a] Huazhong University of Science and Technology, 1037 Luoyu Road, Wuhan, HB 430074, China
[b] Southern Polytechnic State University, 1100 South Marietta Parkway, Marietta, GA 30060-2896, USA

ARTICLE INFO

Article history:
Received 27 September 2010
Received in revised form 22 September 2011
Accepted 23 October 2011
Available online 26 October 2011

Keywords:
Cost-sensitive
Neural networks
Multiple costs
Misclassification

ABSTRACT

The current research investigates a single cost for cost-sensitive neural networks (CNN) for decision making. This may not be feasible for real cost-sensitive decisions which involve multiple costs. We propose to modify the existing model, the traditional back-propagation neural networks (TNN), by extending the back-propagation error equation for multiple cost decisions. In this multiple-cost extension, all costs are normalized to be in the same interval (i.e. between 0 and 1) as the error estimation generated in the TNN. A comparative analysis of accuracy dependent on three outcomes for constant costs was performed: (1) TNN and CNN with one constant cost (CNN-1C), (2) TNN and CNN with two constant costs (CNN-2C), and (3) CNN-1C and CNN-2C. A similar analysis for accuracy was also made for non-constant costs; (1) TNN and CNN with one non-constant cost (CNN-1NC), (2) TNN and CNN with two non-constant costs (CNN-2NC), and (3) CNN-1NC and CNN-2NC. Furthermore, we compared the misclassification cost for CNNs for both constant and non-constant costs (CNN-1C vs. CNN-2C and CNN-1NC vs. CNN-2NC). Our findings demonstrate that there is a competitive behavior between the accuracy and misclassification cost in the proposed CNN model. To obtain a higher accuracy and lower misclassification cost, our results suggest merging all constant cost matrices into one constant cost matrix for decision making. For multiple non-constant cost matrices, our results suggest maintaining separate matrices to enhance the accuracy and reduce the misclassification cost.

1. Introduction

In the traditional artificial neural network (TNN) learning, TNN classifiers try to minimize the error with training data sets [22]. This concept will only make sense when the cost of information acquisition is not taken into account [12,20], and/or different costs for misclassification are treated equally [12,14]. This is not true in many real-world applications of decision making. For instance, in medical applications, predicting a woman to be cancer free, but who actually has breast cancer, is far more costly than misdiagnosing a woman with breast cancer when she is in fact healthy. The misclassification in the former case may result in the loss of the patient's life. Therefore, the cost-sensitive neural network (CNN) has attracted much attention in machine learning and data mining communities.

In the past few years, several cost-sensitive learning methods have been developed, and many of them are mainly developed from decision tree cost-sensitive models [2,7,16,24]. To make a TNN cost-sensitive, four different methods have been proposed [13,17]. The first leaves the learning procedure intact but modifies the network's probability estimates in classifying new data sets. This method can be used to make any cost-insensitive decision models cost-sensitive [5,6,8,23,27]. The second is to adapt the output of a network to give higher impact on connection weights for certain classes with higher expected misclassification costs [13,15,17]. The third is to adjust the learning rate so that the prevalence of samples with a higher cost will be increased significantly in training data sets [13,26]. The last is to take the misclassification costs into account by modifying the learning algorithm [3,9,13,17].

As far as we know, the CNN has focused on only one cost [1-21,23-28]. In many real world applications, there are usually multiple costs associated with the decision-making. However, these costs might not be combined into one equivalent cost. For example, the economical cost and the societal cost cannot be combined into one cost as these costs can hardly be measured in a single metrological system. Furthermore, it is difficult to judge which cost is more important than another. To implicate such different costs into account simultaneously, this paper introduces multiple costs into the back-propagation error equation of the TNN. As the importance of each cost cannot be evaluated precisely, we assume that all the costs are of equal importance and normalize their values into the same interval.

To investigate the performance of CNNs, we compare the average accuracy between TNNs and CNNs. The CNN is constructed with one or two cost matrices which are either constant or non-constant. These comparisons are carried out based on experiments of eight discrete UCI data sets (http://archive.ics.uci.edu/ml).

The remainder of this paper is organized as follows. Section 2 briefly describes some related works. Section 3 introduces some necessary definitions to describe the neural network's error equations,

* Corresponding author. Tel.: +1 6789153574.
E-mail address: chung@spsu.edu (C.-C. Hung).

0167-9236/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2011.10.023

and then accommodates the misclassification costs into the error equations. Section 4 describes the cost sensitive learning and weight adjusting with respect to misclassification. Experimental results are reported and analyzed in Section 5. Finally, conclusions are given in Section 6.

2. Related work

In this section, we describe the data acquisition and the misclassification cost, and the cost processing methods applied in constructing cost sensitive models.

2.1. Type of costs

Cost-sensitive decision making may take different kinds of costs into account. Turney [25] provides a comprehensive survey of different types of costs, including data acquisition, misclassification, active learning, computation, and human-computer interaction. Generally, the data acquisition and misclassification costs are extensively applied in building decision models [17].

Data acquisition cost is synonymous with test cost, i.e., the cost occurred in testing of samples to get attributes, measurements, or features [12,20,24]. Some medical tests such as PET/CT and direct-to-consumer genetic testing are very expensive, so it is worthwhile to find low-costive features to build diagnosing models [10]. Also, it is desirable to build models for determining when to stop testing for other features after balancing the test cost and the risk of misdiagnosis [12].

Although the costs for some medical tests are very expensive, they are still much lower than the misdiagnosis cost, which is usually identified as misclassification cost. Many decision models have been built based on the misclassification cost, such as regressions [1], decision trees [2,7,16,24], Bayesian networks [4], neural networks [13,14,19,21,26,28], and support vector machines [18], etc. These models mainly concentrate on how to reduce the average misclassification costs when trained with a single misclassification cost.

The misclassification cost can be either a fixed value or a function, and it can be associated with a sample or a class [11]. If a cost is associated with a sample, such a cost is suitable to make any cost-insensitive decision models cost-sensitive, but faces a higher risk of over-fitting to training data sets when the cross validation technique is used [17,23].

The fixed cost is usually associated with a class, and thus it cannot describe special samples and conditions. However, it has a lower risk of the so-called over-fitting problem. It has been widely applied to all kinds of decision models and can be expressed with a misclassification matrix [25].

2.2. Cost processing methods

There are four methods available to make a cost-insensitive TNN cost-sensitive [13]. One method is to leave learning algorithms intact and make the trained decision models cost-sensitive, by changing the proportion of negative samples [8], relabeling the classes of samples [6,27], weighting the class of samples [5], and adjusting the threshold for minimizing misclassification cost [23]. As this method leaves learning algorithms intact, it can be applied to almost all existing decision models.

The second method is to adjust the outputs of decision models by taking into account the impact of the misclassification cost. The output is adjusted by imposing a higher impact on learning process for classes with higher expected misclassification costs [13], or corrected by utilizing thresholds to minimize misclassification costs [15], or replaced by a different class that has minimum estimated conditional risk concerned with misclassification costs [17].

The third method is to increase the learning rate for those samples with higher costs [13,26]. This is equivalent to a sample with a higher cost which will be learned more times than that which has a lower cost [26]. To ensure the convergence of the modified back-propagation procedure, Kukar and Kononenko use a method to normalize the replaced learning rate [13].

The fourth method is to modify the learning algorithm to take misclassification costs into account. This may be done by using cost sensitive splitting criteria [7] and making cost-sensitive pruning [2] in building a cost-sensitive decision tree, or by applying the additional cost adjustment function into the weight updating rule which is used to transform the cost-insensitive AdaBoost into a cost-sensitive adaptive boosting AdaCost [9]. Similar methods are proposed for two-class and multiple-class versions of cost-sensitive bagging [3]. By introducing misclassification cost into the TNN's error equation, Kukar and Kononenko modify the TNN training algorithm into a cost sensitive algorithm [13].

All methods mentioned above only take one kind of cost into account, and few of them have investigated the impact of costs against the model's accuracy. In this paper, we study the CNN which is constructed with one or more fixed costs, and attempt to find the impact of the costs relating to the accuracy as well as against one another. We compare the impact of the fixed constant and non-constant costs and analyze the possibility of merging them when encountered with multiple costs.

To deal with misclassification costs, we introduce cost matrices for constant and non-constant misclassification costs in Section 3, and propose a method for weighting one or more costs compared with the classification error in Section 4. In addition, the CNN training method is also presented in Section 4.

3. Defining cost matrices

The cost matrix stores costs that are functions from the actual classes (targeted classes) to the output classes (predicted classes). For a cost matrix C, the element $C[i,j]$stands for the cost of misclassifying a sample of the actual class i as the output class j. When the CNN correctly classifies a sample, i.e., the actual class i of the sample is the same as the output class j, the misclassification cost is zero, e.g., $C[i,j]=0$ for all $i=j$.

If the misclassification costs are equal to c, and the number of the classes is N, we define a constant $N\times N$ cost matrix as in the following [13]:

$$C[i,j]=\begin{cases} c, i\neq j \\ 0, i=j, \end{cases} i,j=1,2,\ldots,N. \tag{1}$$

The cost $E[i]$ is used to stand for the expected cost of misclassifying the sample that belongs to the ith class. When making a decision in TNN, the expected cost of making a correct decision is the same as making a wrong decision:

$$E[i]=\frac{1}{1-P(i)}\sum_{j\neq i}^{N}P(j)\cdot C[i,j],\ \ i=1,2,\ldots,N, \tag{2}$$

where $P(i)$ and $P(j)$ are the estimated prior probabilities that a sample belongs to the ith and the jth class respectively. In the case of a constant cost matrix, the uniform cost vector is as follows:

$$E[i]=c,\ \ i=1,2,\ldots,N. \tag{3}$$

The TNN performance criterion is the error rate or the classification accuracy. But, when the cost is not constant for a data set, we are more concerned with the average cost of CNN trained by this data set.

The average cost is a very important performance measure for cost sensitive learning. It should be evaluated in our experiments and is defined as below:

$$A(S) = \frac{1}{|S|}\sum_{s\in S} C[TC(s), PC(s)], \tag{4}$$

where $|S|$ is the number of the test samples in data set S, $TC(s)$ is the targeted class of sample s, and $PC(s)$ is the predicted class of sample s. When the constant cost matrix is used in Eq. (4), the error rate may be viewed as a special case of the average cost. Usually, the TNN minimizes the squared error of its output vector for sample $s\in S$.

$$e(s) = \frac{1}{2}\sum_{k=1}^{N}(t_k - o_k)^2, \tag{5}$$

where o_k is the kth component of the predicted class of the TNN, and t_k is the binary classification of the kth class on sample s, i.e., $t_k = 1$ if $k = TC(s)$ or 0 otherwise.

4. Cost sensitive learning

Since the cost is increased along with the error, we may take the cost into account at the same time by multiplying the error with the cost. Assuming that we use the sigmoid activation function for both the hidden layer and the output layer; the outputs of the neural network will be automatically normalized within interval [0, 1]. To treat the cost and error with equal importance, we normalize the cost within the interval of [0, 1] and extend Eq. (5) for both errors and costs as follows [13]:

$$e(s) = \frac{1}{2}\sum_{k=1}^{N}(t_k - o_k)^2 \cdot [\tilde{C}(TC(s), PC(s))]^2, \tag{6}$$

where $\tilde{C}(TC(s), PC(s))$ is the normalized cost of $C[TC(s), PC(s)]$. In TNN, the cost of correct classification is the same as that of incorrect classification, thus the normalized cost $\tilde{C}(i,j)$ is defined as follows:

$$\tilde{C}(i,j) = \begin{cases} \dfrac{E[i]}{\max_{q,r}(C[q,r])}, i = j \\ \dfrac{C[i,j]}{\max_{q,r}(C[q,r])}, i \neq j, \end{cases} \quad q,r,i,j = 1,2,\ldots,N, \tag{7}$$

where $\max_{q,r}(C[q,r])$ is the maximum misclassification cost in matrix C.

From Eq. (3), it is obvious that $\tilde{C}(i,j)$ will be 1 for any constant cost matrix C. As a result of using constant cost matrices, the cost-sensitive learning should theoretically become cost-insensitive learning, i.e., Eq. (5) is a special case of Eq. (6). However, as the average cost in Eq. (4) is closely related to the corresponding classification accuracy, and the accuracy may be different for each training data set and testing data set when the cross validation is used, we would like to explore whether the constant cost matrix will have an impact on the accuracy of cost-sensitive learning and vice versa.

Assume that there are two kinds of costs stored in cost matrices C_1 and C_2, respectively. There are two ways to take these two kinds of costs into account. The first is to merge two kinds of costs into one equivalent cost denoted by C_3:

$$C_3[i,j] = C_1[i,j] + C_2[i,j], \quad i,j = 1,2,\ldots,N \tag{8}$$

When two kinds of costs cannot be merged, we treat these two costs with equal importance by normalizing them within the interval of [0, 1], and take them into consideration independently. We extend the Eq. (6) with the equal importance for two kinds of costs as follows:

$$e(s) = \frac{1}{2}\sum_{k=1}^{N}(t_k - o_k)^2 \cdot \left[\tilde{C}_1(TC(s), PC(s)) \cdot \tilde{C}_2(TC(s), PC(s))\right]^2 \tag{9}$$

Similar to Eq. (6), if C_1, and C_2 are all constant cost matrices, the behavior of Eq. (9) should be theoretically identical to that of original Eq. (5). Also, we would like to answer whether Eq. (9) still behaves the same as what is expected for constant cost matrices. Furthermore, we would like to learn the behavior of Eq. (9) if the cost matrix is not constant.

At the nth TNN learning iteration, the momentum learning algorithm modifies the l-layer synaptic weight $w_{hk}^{(n)}(l)$ of the hth neuron for the kth output of the l-1 layer according to the delta rule [22], by computing the local gradient $\delta_h(l)$. Please note that for the hidden layer $l = 1$, and the output layer $l = 2$ in a 3-layer TNN.

$$\begin{aligned} \Delta w_{hk}^{(n+1)}(l) &= \alpha \cdot \Delta w_{hk}^{(n)}(l) + (1-\alpha)\cdot\eta\cdot\delta_h(l)\cdot o_k(l-1), \\ w_{hk}^{(n+1)}(l) &= w_{hk}^{(n)}(l) + \Delta w_{hk}^{(n+1)}(l), \end{aligned} \tag{10}$$

where η is the learning rate that controls the magnitude of the weight changes, α is the momentum coefficient, and $o_k(l)$ is the kth component of the output of layer l. In a 3-layer TNN, the computation of $\delta_h(\bullet)$ of the output layer differs from that of $\delta_h(\bullet)$ of the hidden layer.

Assume that the sigmoid activation function is used for both the output layer and the hidden layer. The computation of $\delta_h(l)$for Eq. (5) should be as follows:

$$\begin{aligned} \delta_h(2) &= (t_h - o_h)\cdot o_h \cdot (1-o_h), && \text{for output layer,} \\ \delta_h(1) &= o_h(1)\cdot(1-o_h(1))\cdot\sum_{k=1}^{N}\delta_k(2)\cdot w_{hk}(2), && \text{for hidden layer.} \end{aligned} \tag{11}$$

The computation of $\delta_h(l)$for Eq. (6) can be formulated as follows:

$$\begin{aligned} \delta_h(2) &= \left[\tilde{C}(TC(s),PC(s))\right]^2 \cdot (t_h - o_h)\cdot o_h\cdot(1-o_h), && \text{for output layer,} \\ \delta_h(1) &= o_h(1)\cdot(1-o_h(1))\cdot\sum_{k=1}^{N}\delta_k(2)\cdot w_{hk}(2), && \text{for hidden layer.} \end{aligned} \tag{12}$$

Similarly, the computation of $\delta_h(l)$for Eq. (9) can be written as in the following:

$$\begin{aligned} \delta_h(2) &= \left[\tilde{C}_1(TC(s),PC(s))\cdot\tilde{C}_2(TC(s),PC(s))\right]^2 \cdot (t_h - o_h)\cdot o_h\cdot(1-o_h), \\ &\qquad\qquad \text{for output layer,} \\ \delta_h(1) &= o_h(1)\cdot(1-o_h(1))\cdot\sum_{k=1}^{N}\delta_k(2)\cdot w_{hk}(2), \quad \text{for hidden layer.} \end{aligned} \tag{13}$$

If the normalized cost matrices $\tilde{C}$, $\tilde{C}_1$, and $\tilde{C}_2$ used in Eqs. (12) and (13) are originally constant, Eqs. (12) and (13) will be identical to Eq. (11).

Instead of treating the error and cost equally, we may give different weights for the error and costs. Assume the weight for error, costs $\tilde{C}_1$ and $\tilde{C}_2$ are α, β and γ, respectively, Eq. (9) will be reformulated as follows:

$$e(s) = \frac{1}{2}\sum_{k=1}^{N}(t_k - o_k)^{2\alpha}\cdot\left[\tilde{C}_1(TC(s),PC(s))\right]^{2\beta}\cdot\left[\tilde{C}_2(TC(s),PC(s))\right]^{2\gamma} \tag{14}$$

And the computation of $\delta_h(l)$ for Eq. (14) can be written as in the following:

$$\delta_h(2) = \alpha\cdot\left[\tilde{C}_1(TC(s),PC(s))\right]^{2\beta}\cdot\left[\tilde{C}_2(TC(s),PC(s))\right]^{2\gamma}\cdot(t_h-o_h)\cdot o_h\cdot(1-o_h), \quad \text{for output layer,}$$
$$\delta_h(1) = o_h(1)\cdot(1-o_h(1))\cdot\sum_{k=1}^{N}\delta_k(2)\cdot w_{hk}(2), \quad \text{for hidden layer.} \tag{15}$$

In case we cannot differentiate important degrees about the error, the costs $\tilde{C}_1$ and $\tilde{C}_2$, we may assume $\alpha=\beta=\gamma=1$. Then, Eq. (15) will be reduced to Eq. (13).

5. Experiments

We compare the classification accuracy between TNN and CNN using eight UCI data sets. To compare the performance of the classification accuracy and the average misclassification cost defined in Eq. (4), we construct three-layer neural networks for the testing of errors and misclassification costs, assuming that the misclassification costs are as equally important as the classification error.

The number of hidden neurons in a TNN or CNN is $4\cdot\sqrt{m\cdot n}$ (m is the number of inputs, n is the number of classes). We use 10-fold cross-validation to evaluate the accuracies and costs. The following parameters are used for the neural networks in all the experiments. The learning rate is 0.25, the momentum is 0.07, the important degrees are $\alpha=\beta=\gamma=1$, and the number of iterations is 5000. We are aware that by fixing those parameters it may result in the networks under-fitting or over-fitting for some data sets.

5.1. Data sets and cost matrices

Our experiments were carried out on eight data sets from UCI repository, which are Original Breast Cancer (BC), Congressional Voting (CV), Lymphography (LG), SPECT Heart (SH), Poker Hand (PH), Haberman's Survival (HS), Balance Scale (BS), and Car Evaluation (CE). The class label or the category attribute is not counted in the number of attributes. When training a neural network, the reduced attributes outputted by an attribute reduction algorithm are used as its inputs in our experiments, in order to achieve higher predication accuracy in case of a small number of samples in most of the data sets, as shown in Table 1.

We provide constant and non-constant matrices to compare our CNNs. The constant matrix C_0 is used to test whether the accuracy of such a CNN-1C is the same as that of a TNN. C_0 is a 0–1 constant cost matrix, which is defined as follows:

$$C_0[i,j] = \begin{cases} 1, & i\neq j, \\ 0, & i=j. \end{cases} \tag{16}$$

Constant matrix C_1 is constructed as $C_1=2\cdot C_0$, which is used to compare the one-cost CNN-1C from constant C_1 with the two-cost CNN-2C from two constant C_0s. The non-constant matrix C_3 is used to train CNN-1NC, which is defined as $C_3=C_0+C_2$, where the non-constant matrix C_2 is constructed as follows:

$$C_2[i,j] = \begin{cases} 1 & ,i<j, \\ 0 & ,i=j, \\ i-j+1 & ,i>j. \end{cases} \tag{17}$$

Similarly, the non-constant matrix C_5 is defined as $C_5=C_0+C_4$, and the non-constant matrix C_4 is constructed as follows:

$$C_4[i,j] = \begin{cases} j-i+1 & ,i<j, \\ 0 & ,i=j, \\ 1 & ,i>j. \end{cases} \tag{18}$$

In summary, the cost matrix C_1 is equivalent toC_0+C_0, the cost matrix C_3 is equivalent to C_0+C_2, and the cost matrix C_5 is equivalent to C_0+C_4. If a CNN is trained by two cost matrices, and both of them are constant, this CNN is called a CNN-2C. Otherwise, it is called a CNN-2NC.

5.2. Results and analysis

The first experiment is to compare the accuracy between TNN and CNN, in which cost matrices are applied to Eqs. (6) and (9). As shown in Table 2, the accuracies in the second column TNN are higher than the corresponding accuracies in column CNN-1C [C_0] for data sets BC, LG, PH, HS, and BS; the accuracy in column TNN for the data set SH is identical to the accuracy in column CNN-1C [C_0]; and the accuracies in column TNN for the data sets CV and CE is less than the corresponding accuracies in column CNN-1C [C_0]. Thus, the accuracies of the TNNs are usually higher than those of the CNN-1Cs.

A similar observation can be made between accuracies in column TNN and column CNN-1C [C_1]. Moreover, we find that the accuracies of CNN-1Cs are usually higher than those of CNN-2Cs. Comparing between the lower cost matrix C_0 and the higher cost matrix C_1, the accuracies of CNN-1C [C_0] are almost equal to those of the CNN-1C [C_1] (4 higher and 4 lower of 8 comparisons). This is because they are both one-cost and the normalized costs (for Eq. (6)) are the same for C_0 and C_1. From this experiment, we learn that in most cases even if the cost-sensitive learning with constant cost matrices can be theoretically viewed as a cost-insensitive learning, these cost matrices still show their negative effects to the CNN accuracies in the learning process.

Similar to most TNNs, the CNNs trained with two-class data sets converge much faster than the CNNs trained with multi-class data sets when the same learning rate, the same momentum, and the same number of iterations are used. In other words, no matter whether the neural network is a TNN or a CNN, experiments show that the convergence speed is mainly determined by the number of classes rather than the number of costs. For example, data set LG has only 148 learning samples, but has as many as 4 classes,its accuracy is

Table 1
Basic characteristics of the datasets showing the number of samples, number of attributes, number of attributes used, number of classes, and percentage of samples in the major class.

Data set	No. of samples	No. of attributes	No. of attributes used	No. of classes	% major class
BC	699	9	4	2	0.66
CV	435	16	9	2	0.61
LG	148	18	8	4	0.45
SH	80	22	22	2	0.50
PH	25010	10	5	10	0.50
HS	306	3	3	2	0.74
BS	625	4	4	3	0.46
CE	1728	6	6	4	0.70

Table 2
Average accuracies of TNNs and CNNs from constant cost matrices: TNN column obtained by Eq. (5), columns CNN-1C [C_0] and CNN-1C [C_1] obtained by Eq. (6), CNN-2C [C_0+C_0] column obtained by Eq. (9).

Cost matrices	TNN	CNN-1C [C_0]	CNN-2C [C_0+C_0]	CNN-1C [C_1]
BC	0.964 ± 0.019	0.947 ± 0.030	0.943 ± 0.024	0.956 ± 0.026
CV	0.945 ± 0.034	0.952 ± 0.036	0.939 ± 0.034	0.959 ± 0.023
LG	0.360 ± 0.118	0.353 ± 0.130	0.360 ± 0.167	0.313 ± 0.100
SH	0.663 ± 0.177	0.663 ± 0.196	0.688 ± 0.169	0.600 ± 0.227
PH	0.456 ± 0.039	0.445 ± 0.038	0.432 ± 0.033	0.443 ± 0.039
HS	0.732 ± 0.051	0.723 ± 0.061	0.710 ± 0.065	0.732 ± 0.083
BS	0.886 ± 0.032	0.876 ± 0.047	0.868 ± 0.034	0.860 ± 0.034
CE	0.660 ± 0.056	0.666 ± 0.046	0.689 ± 0.036	0.675 ± 0.047

the lowest among all data sets. This is because the classification performance decreases with the increase of the number of classes [19].

The second experiment is to compare the accuracies between two CNNs which are constructed with non-constant cost matrices. Remember that matrix C_3 is equal to C_0+C_2 and C_5 is equal to C_0+C_4. As shown in Table 3, we can observe the average accuracies of CNN-2NCs are higher than those of CNN-1NCs, because accuracies with two-cost are higher than the corresponding accuracies with one-cost in 9 comparisons, and lower in 6 comparisons out of all 16 pairs. If we further compare the accuracies between CNN-1NC and its equivalent CNN-2NC (as comparison pair in black-edged square), we may find that there are 6 pairs in which the accuracy increase with respect to CNN-1NC is more than 0.016, but there are only 3 pairs in which the accuracy decrease with respect to CNN-1NC is more than 0.016. In addition, if we take each data set as a comparison unit, the accuracy of CNN-2NC is higher than the corresponding accuracy of CNN-1NC in 4 data sets, and lower in 2 data sets among all 8 data sets. There is no clear distinction for the remaining 2 data sets. The abrupt accuracy decrease of data set HS (as comparison pair in black-edged square) with respect to CNN-1NC is owing to its large imbalance of class distribution and few number of learning samples [19].

The third experiment is to compare the costs between CNNs with different constant cost matrices. As shown in Table 4, columns CNN-1C [C_0] and CNN-1C [C_1] are obtained by using Eq. (6), and the CNN-2C [C_0+C_0] column is obtained by using Eq. (9). Because cost matrix C_1 is the double of C_0, the average costs in column CNN-1C [C_1] should be also the double of the costs in column CNN-1C [C_0] if their CNNs have the same performance on misclassification cost. Also, we observe that the costs are somehow related with the accuracies shown in Table 2; the higher the prediction accuracy, the lower the misclassification cost. We can derive the following results based on our analysis; (1) there are more CNN-1Cs in columns CNN-1C [C_0] and CNN-1C [C_1] whose costs are relatively lower than those of CNN-2Cs in column CNN-2C [C_0+C_0] (as comparison pair in black-edged square); and (2) between constant cost matrices C_0 and C_1, the costs of CNN-1Cs trained by the lower cost C_0 are lower than those of CNN-1Cs trained by the higher cost C_1.

The fourth experiment is to compare the costs between CNN-1NCs and CNN-2NCs, as is shown in Table 5. Remember that matrix C_3 is equal to C_0+C_2 and C_5 is equal to C_0+C_4, where C_2 and $C_4$are defined in Eqs. (17) and (18) respectively. If we compare the costs between CNN-1NC [C_3] and CNN-2NC [C_0+C_2], and the costs between CNN-1NC [C_5] and the CNN-2NC [C_0+C_4] in Table 5, we may find that there are many more CNN-1NCs whose costs are larger than the sum of two costs of their equivalent CNN-2NCs (11 larger out of 16 comparisons).

6. Conclusions

In this paper, the effects of multiple constant and non-constant costs in training CNNs have been studied empirically on eight discrete UCI data sets. The TNN, one-cost and two-cost CNNs are constructed and tested. Our results show that the greater the number of cost matrices, the slower the convergence of the related CNNs. Furthermore, our experiments confirmed that the convergence speed is mainly determined by the number of classes rather than the number of the costs [19].

In general, when the neural networks are trained with the same data sets and the same learning parameters, the TNNs are usually more accurate than CNN-1Cs and CNN-2Cs. And based on our experiments, the CNN-1Cs are usually more accurate than the CNN-2Cs. This is resulted from the competition of the two equivalent constant costs. For constant one-cost matrices, the accuracy of the CNN-1Cs trained with lower costs is almost equal to those with higher costs. The reason is that the number of cost matrices is same and the normalized cost matrices are also same according to Eq. (6).

As it is well known in data mining literature, the misclassification costs and the classification accuracy are inversely related. We obtained similar conclusions from our experiments: (1) the misclassification costs of CNN-1Cs are usually lower than those of the equivalent CNN-2Cs, and the accuracies of CNN-1Cs are usually higher than those of the equivalent CNN-2Cs; and (2) for constant one-cost matrices, the misclassification costs of the CNN-1Cs trained with lower costs are usually lower than those of the CNN-1Cs trained with higher costs, this is because their accuracies are almost identical, but the cost of the latter is higher than that of the former.

Between CNN-1NCs and their equivalent CNN-2NCs, the accuracies of the CNN-1NCs are lower than those of the CNN-2NCs, and the misclassification costs of the CNN-1NCs are significantly higher than those of the CNN-2NCs. For the CNN-2NC, because one of its cost is smaller than the other, and it is likely that the competition will be over quickly, the CNN-2NC will have more time to improve its accuracy. Therefore, when there are two or more non-constant costs, it is better to keep the costs separated to obtain higher accuracy and lower cost CNNs.

Table 4
Average costs of CNNs from constant matrices: columns CNN-1C [C_0] and CNN-1C [C_1] obtained by Eq. (6), CNN-2C [C_0+C_0] column obtained by Eq. (9).

Cost matrices	CNN-1C [C_0]	CNN-2C [C_0+C_0]		CNN-1C [C_1]
		C_0	C_0	
BC	0.053 ± 0.030	0.057 ± 0.024	0.057 ± 0.024	0.089 ± 0.051
CV	0.048 ± 0.036	0.061 ± 0.034	0.061 ± 0.034	0.082 ± 0.047
LG	0.647 ± 0.130	0.640 ± 0.167	0.640 ± 0.167	1.373 ± 0.199
SH	0.338 ± 0.196	0.313 ± 0.169	0.313 ± 0.169	0.800 ± 0.453
PH	0.555 ± 0.038	0.568 ± 0.033	0.568 ± 0.033	1.115 ± 0.078
HS	0.277 ± 0.061	0.290 ± 0.065	0.290 ± 0.065	0.535 ± 0.167
BS	0.124 ± 0.047	0.132 ± 0.033	0.132 ± 0.033	0.279 ± 0.068
CE	0.334 ± 0.046	0.311 ± 0.036	0.311 ± 0.036	0.650 ± 0.095

Acknowledgments

This study is done in the Chinese Education Ministry Key Lab of Image Processing and Intelligent Control and is funded and supported by grant 2006AA02Z347 for the Chinese National 863 Target-oriented

Table 3
Average accuracies of CNNs from non-constant cost matrices: columns CNN-1NC [C_3] and CNN-1NC [C_5] obtained by Eq. (6), columns CNN-2NC [C_0+C_2] and CNN-2NC [C_0+C_4] obtained by Eq. (9).

Cost matrices	CNN-1NC [C_3]	CNN-2NC [C_0+C_2]	CNN-1NC [C_5]	CNN-2NC [C_0+C_4]
BC	0.956 ± 0.014	0.951 ± 0.030	0.954 ± 0.020	0.933 ± 0.030
CV	0.932 ± 0.044	0.950 ± 0.034	0.941 ± 0.024	0.964 ± 0.027
LG	0.320 ± 0.125	0.307 ± 0.095	0.320 ± 0.088	0.420 ± 0.122
SH	0.597 ± 0.195	0.613 ± 0.109	0.638 ± 0.199	0.638 ± 0.150
PH	0.420 ± 0.007	0.421 ± 0.009	0.444 ± 0.032	0.438 ± 0.046
HS	0.706 ± 0.069	0.277 ± 0.065	0.732 ± 0.071	0.684 ± 0.073
BS	0.878 ± 0.039	0.887 ± 0.027	0.870 ± 0.021	0.895 ± 0.026
CE	0.657 ± 0.042	0.682 ± 0.049	0.675 ± 0.036	0.758 ± 0.038

Table 5
Average costs of CNNs from non-constant cost matrices: columns CNN-1NC [C_3] and CNN-1NC [C_5] obtained by Eq. (6), columns CNN-2NC [C_0+C_2] and CNN-2NC [C_0+C_4] obtained by Eq. (9).

Cost matrices	CNN-1NC [C_3]	CNN-2NC [C_0+C_2]		CNN-1NC [C_5]	CNN-2NC [C_0+C_4]	
		C_0	C_2		C_0	C_4
BC	0.100 ± 0.036	0.049 ± 0.030	0.059 ± 0.039	0.123 ± 0.055	0.067 ± 0.030	0.084 ± 0.041
CV	0.168 ± 0.122	0.050 ± 0.034	0.050 ± 0.034	0.159 ± 0.071	0.036 ± 0.027	0.036 ± 0.027
LG	1.833 ± 0.350	0.693 ± 0.095	0.900 ± 0.158	1.907 ± 0.318	0.580 ± 0.122	0.920 ± 0.231
SH	1.050 ± 0.540	0.388 ± 0.109	0.500 ± 0.118	0.888 ± 0.469	0.363 ± 0.150	0.513 ± 0.260
PH	1.275 ± 0.018	0.579 ± 0.009	0.702 ± 0.014	2.023 ± 0.221	0.562 ± 0.046	1.012 ± 0.170
HS	0.881 ± 0.206	0.723 ± 0.065	0.723 ± 0.065	0.535 ± 0.143	0.316 ± 0.073	0.316 ± 0.073
BS	0.284 ± 0.091	0.113 ± 0.027	0.149 ± 0.040	0.349 ± 0.071	0.105 ± 0.026	0.140 ± 0.039
CE	0.861 ± 0.123	0.319 ± 0.049	0.398 ± 0.070	0.835 ± 0.092	0.242 ± 0.038	0.283 ± 0.058

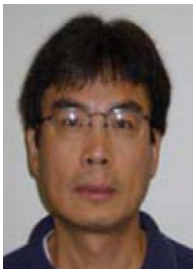
Project “Grid Based Digitalized Medical Treatment Decision Making Support System”,and grant 61075010 for the project “Research on Non-rigid Registration Method for Multi-model Cardiac Images Based on Different Direction Features” of Natural Science Foundation of China.

References

[1] G. Bansal, A. Sinha, H. Zhao, Tuning data mining methods for cost-sensitive regression: a study in loan charge-off forecasting, Journal of Management Information Systems 25 (3) (2008) 315–336, doi:10.2753/MIS0742-1222250309.
[2] J.P. Bradford, C. Kuntz, R. Kohavi, C. Brunk, C.E. Brodley, Pruning decision trees with misclassification costs, Proceedings of the 10th European Conference on Machine Learning, 1998, pp. 131–136, Chemnitz, Germany.
[3] L. Breiman, Bias, Variance, and Arcing Classifiers, Technical Report, Department of Statistics, University of California, Berkeley, 1996.
[4] N. Cesa-Bianchi, G. Valentini, Hierarchical cost-sensitive algorithms for genome-wide gene function prediction, Journal of Machine Learning Research 8 (2010) 14–29.
[5] N. Chen, B. Ribeiro, A. Vieira, J. Duarte, J. Neves, Weighted learning vector quantization to cost-sensitive learning, Lecture Notes in Computer Science 6354 (2010) 277–281, doi:10.1007/978-3-642-15825-4_33.
[6] P. Domingos, MetaCost: a general method for making classifiers cost-sensitive, Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 155–164, San Diego, CA, USA.
[7] C. Drummond, R.C. Holte, Exploiting the cost in sensitivity of decision tree splitting criteria, Proceedings of the 17th International Conference on Machine Learning, 2000, pp. 239–246, San Francisco, CA, USA.
[8] C. Elkan, The foundations of cost-sensitive learning, Proceedings of the 7th International Joint Conference on Artificial Intelligence, 2001, pp. 973–978, San Francisco, CA, USA.
[9] W. Fan, S.J. Stolfo, J. Zhang, P.K. Chan, AdaCost: misclassification cost-sensitive boosting, Proceedings of the 16th International Conference on Machine Learning, 1999, pp. 97–105, San Francisco, USA.
[10] R. Goetschalckx, K. Driessens, S. Sanner, Cost-sensitive parsimonious linear regression, Proceedings of the 8th IEEE International Conference on Data Mining, 2008, pp. 809–814, doi:10.1109/ICDM.2008.76, Pisa, Italy.
[11] J. Hollmén, M. Skubacz, M. Taniguchi, Input dependent misclassification costs for cost-sensitive classifiers, in: N. Ebecken, C. Brebbia (Eds.), DATA MINING II - Proceedings of the 2nd International Conference on Data Mining, 495–503, 2000, Cambridge, England.
[12] S. Ji, L. Carin, Cost-sensitive feature acquisition and classification, Journal of Pattern Recognition 40 (5) (2007) 1474–1485.
[13] M. Kukar, I. Kononenko, Cost-sensitive learning with neural networks, Proceedings of the 13th European Conference on Artificial Intelligence, 1998, pp. 445–449, Brighton, UK.
[14] J. Lan, M.Y. Hu, E. Patuwo, G.P. Zhang, An investigation of neural network classifiers with unequal misclassification costs and group sizes, Decision Support Systems 48 (4) (2010) 582–591.
[15] J. Langford, A. Beygelzimer, Sensitive error correcting output codes, Lecture Notes in Computer Science 3559 (2005) 21–49.
[16] S. Lomax, S. Vadera, An empirical comparison of cost-sensitive decision tree induction algorithms, Expert Systems 28 (3) (2011) 227–268, doi:10.1111/j.1468-0394.2010.00573.x.
[17] D.D. Margineantu, Methods for cost-Sensitive learning, Ph.D. Dissertation, Corvallis, OR: Oregon State University, 2001.
[18] H. Masnadi-Shirazi, N. Vasconcelos, Risk minimization, probability elicitation, and cost-sensitive SVMs, Proceedings of the 27th International Conference on Machine Learning, 2010, pp. 204–213, Haifa, Israel.
[19] G. Ou, Y.L. Murphey, Multi-class pattern classification using neural networks, Journal of Pattern Recognition 40 (1) (2007) 4–18.
[20] P.C. Pendharkar, Hybrid approaches for classification under information acquisition cost constraint, Decision Support Systems 41 (1) (2005) 228–241.
[21] P.C. Pendharkar, A threshold varying bisection method for cost sensitive learning in neural networks, Journal of Expert Systems with Applications 34 (2) (2008) 1456–1464.
[22] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning internal representations by error propagation, in: D.E. Rumelhart, J.L. McClelland, the PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Cambridge, MIT Press, MA, 1986, pp. 318–362.
[23] V.S. Sheng, C.X. Ling, Thresholding for making classifiers cost-sensitive, in: Anthony Cohn (Ed.), Proceedings of the 21st International Conference on Artificial Intelligence, 476–481, 2006, Boston, Massachusetts, USA.
[24] P.D. Turney, Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm, Journal of Artificial Intelligence Research 2 (1995) 369–409.
[25] P.D. Turney, Types of cost in inductive concept learning, Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning (WCSL at ICML-2000), Stanford University, California, USA, 2000, (NRC #43671).
[26] C. Wan, L. Wang, K.M. Ting, Introducing cost sensitive neural networks, Proceedings of the 2nd International Conference on Information and Communication Security, Sydney, Australia, 1999, [Online] Available at, http://www.gscit.monash.edu.au/~kmting/Papers/Icics99.ps.
[27] I.H. Witten, E. Frank, Data mining—practical machine learning tools and techniques with Java implementations, Morgan Kaufmann Publishers, San Francisco, 2005.
[28] Z.H. Zhou, X.Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Transactions on Knowledge and Data Engineering 18 (1) (2006) 63–77.

Guang-Zhi Ma is an associate professor in Computer Science and Technology at Huazhong University of Science and Technology, China. He received his M.S. and Ph.D. in Computer Science from Huazhong University of Science and Technology. He had been trained in North Illinois University for IBM S390 System. His research interests include decision support using data warehousing and data mining, computer aided diagnose using neural network, multiple object optimization using genetic algorithm, and database and network applications in health information systems.

En-Min Song received his PhD in 1999 and engaged in postdoctoral research in University of California at San Francisco for 2 years. He became a senior member of IEEE in 2002. Since 2004, He has been a professor at the college of computer science and technology, Huazhong University of Science and Technology (HUST), P. R. China. He is the director of the Center for Biomedical Imaging and Bioinformatics in HUST. His research interests include computability theory, medical image Processing and artificial Intelligence.



Chih-Cheng Hung is professor of Computer Science at Southern Polytechnic State University. He received his B.S. in business mathematics from Soochow University, and his M.S. and Ph.D. in Computer Science from the University of Alabama in Huntsville.

Li Su is professor in Biological Sciences at Huazhong University of Science and Technology, China. She received her Ph.D. from Kyoto University in Japan, and trained as a postdoctoral Research Scientist at Kyoto University and Stanford Research Institute.

Dong-Shan Huang is a Ph.D. student of Computer School of Science and Technology in Huazhong University of Science and Technology, China. He received his M.S. in Computer Software and Theory from Huazhong University of Science and Technology. His research interests include machine learning and data mining and health related decision support applications.