A cost-sensitive semi-supervised learning model based on uncertainty

Hongyu Zhu, Xizhao Wang*

Big Data Institute, College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

A B S T R A C T

Aiming at reducing the total cost in cost-sensitive learning, this paper introduces a semi-supervised learning model based on uncertainty of sample outputs. Its central idea is (1) to categorize the samples which are not in training set into several groups based on the uncertainty-magnitude of their outputs, (2) to add the group of samples which have the least uncertainty together with their predicted labels in the original training set, and (3) to retain a new classifier for total cost reduction. The ratio of costs between classes and its impact on learning system improvement is discussed. Theoretical analysis and experimental demonstration show that the model can effectively improve the performance of a cost-sensitive learning algorithm for a certain type of classifiers.

© 2017 Elsevier B.V. All rights reserved.

1. Introduction

Technology of classification which belongs to a topic of machine learning has been widely applied in a lot of domains such as pattern recognition, knowledge discovery, intelligent control, network security, gene engineering, bioinformatics, and so on. From literature we can find many approaches to designing classifiers such as decision tree induction [1,2], hypothesis version spaces [3], Bayesian networks [4,5], evolutionary computing [6], logistic regressions [7,8], support vector machines [9,10], neural networks [11], and deep learning technique [12,13]. The most important index for evaluating a designed classifier is the generalization ability, i.e., the rate of correctly classifying samples which are not in training set.

The concept of cost-sensitive classification can be introduced into the process of classifier design [14–18]. The “cost-sensitive” refers to that a class (feature or object) has the different cost in comparison with another class (feature or object respectively) in classification process. For example, the cost of wrongly classifying a patient as non-cancer class is much bigger than the cost of wrongly classifying the patient as cancer class. From the viewpoint of loss function, cost-sensitive classifier learning is to minimize a cost-loss function but cost-insensitive learning is to maximize the correct rate of classification. Under some certain conditions, these two objective functions of optimization can be equivalent.

The following is an incomplete survey on cost-sensitive learning of classification problems. Reference [14] proposed a principled method for making a general classifier cost-sensitive by wrapping a cost-minimizing procedure called Meta-Cost around it. It is confirmed that class-imbalance often affects the performance of cost-sensitive classifiers in [15]. While most studies of cost-sensitive have only paid attention how to deal with misclassification costs, article [16] put forward to handle the equally important issue: the test costs associated with querying the missing values in a test case. A theoretical analysis on F-measures was presented in [17] for binary, multiclass and multi-label classification while these performance measures are non-linear. Paper [18] proposed a cost-sensitive rotation forest algorithm for gene expression data classification. Three classification costs, namely misclassification cost, test cost and rejection cost, are embedded into the rotation forest algorithm.

Under the framework of semi-supervised learning, this paper introduces a cost-sensitive learning model based on uncertainty-magnitude of sample outputs. Suppose we have selected a classifier training algorithm and trained a basic classifier. The fundamental operations of this model include three steps. The first step is categorizing the testing samples into several groups based on the uncertainty of their outputs given by the basic classifier; the second step is determining a group of samples with smallest uncertainty and adding these samples and their labels predicted by the basic classifier in the original training set, and the third step is retraining a classifier on the enlarged training set through the selected training algorithm. We expect that the new classifier has a reduced total cost in comparison with the basic classifier when
a class-cost matrix for wrong classification among the classes is given.

The algorithm of training a basic classifier is selected in this paper as Extreme Learning Machine (ELM) which is a recently popular scheme for training a single hidden layer feed forward neural network. The reason is that the ELM has the non-iterative training mechanism, which has been studied intensively and extensively in recent decade. ELM was firstly proposed in [19], which randomly chooses weights of hidden nodes and analytically determines the output weights of a single-hidden layer feed-forward network (SLFN). ELMs have both better generalization performance and much faster learning speed than traditional algorithms. ELMs were originally developed for the SLFNs [20] and then extended to the generalized SLFNs which need not be neuron alike [21]. To tackle with the data with imbalanced class distribution, paper [22] proposed a weighted ELM which can be generalized to cost sensitive learning by distributing different weights for individual examples. The dissimilar ELM (D-ELM) was developed by introducing misclassification costs into the classifier and the cost-sensitive D-ELM (CS-D-ELM) was proposed to increase the classification stability [23].

One can find from references many specific forms of uncertainty representation for a vector within which each component is a number between 0 and 1. The uncertainty in this work is chosen as the fuzziness which basically is a variant of the entropy, the typical representation of uncertainty for a probability distribution. It is worth pointing out that our developed learning model is basically insensitive to the selection of both the classifier training algorithms and the specific representation forms of uncertainty.

The rest of this paper is organized as follows. Section 2 gives a brief review on uncertainty of a vector. Section 3 introduces the ELM training algorithm and then incorporates the cost-sensitive class into the ELM training. Section 4 demonstrates some observations between uncertainty amount and classification cost, and then proposes our new training model for class cost-sensitive learning. Section 5 lists the experimental verification and provides some analysis on our model. Section 6 finally concludes this paper.

### 2. Uncertainty

Uncertainty is usually referred to as that a concept cannot be described clearly and exactly. We do not find a general definition of uncertainty mathematically, but under different settings, specific definitions of uncertainty can be given. The following is a brief summary of several uncertainties well modeled mathematically (Table 1).

We now focus on a typical uncertainty called fuzziness for a fuzzy set [32]. Fuzziness is used to describe the unclear degree between two terms such as hot and cold, which was first mentioned in 1968 by Zadeh who developed the fuzzy set theory [24]. The essential idea in Zadeh proposed fuzzy set theory is the membership degree which extends a 0–1 valued function to a function taking values within interval [0, 1]. It is noted that the membership function is determined subjectively to a great extent. Following the fuzzy set theory, Luca and Termini in 1972 deemed that fuzziness was a type of the uncertainty described by the fuzzy set, and furthermore [25], defined the quantitative measure of fuzziness by non-probabilistic entropy resembled to the information entropy of Shannon. They also put forward that fuzziness should satisfy three properties which indicate that the degree of fuzziness achieves maximum when all the elements are equal to each other and achieves minimum when all the elements are either 0 or 1. In addition, Luca and Termini expanded the definition of the entropy on fuzzy sets [26]. It can be not only a numerical quantity but also a column matrix or a vector.

Fuzziness is considered as a kind of cognitive uncertainty which emerged from the transition of uncertainty from one linguistic term to another, where the linguistic term is a fuzzy set defined on a certain universe of discourse. For instance, the weather can be expressed as windy, rainy, sunny and so on. Here the windy, rainy, sunny are fuzzy sets defined on a universal space such as a number of days. Suppose that X is a discrete finite universal space, μ and σ denote two fuzzy sets defined on X, and F(i) is a fuzziness function defined on all fuzzy sets of X. As stated in [27], the function F(i) must satisfy the following axioms:

1. \( F(\mu) = 0 \) if and only if \( \mu \) is a crisp set;
2. \( F(\mu) \) gets its maximum if and only if \( \mu(x) = 0.5 \) for each \( x \) in \( X \);
3. \( F(\mu) \geq F(\sigma) \) if \( \mu \leq \sigma \);
4. \( F(\mu) = F(\mu') \) where \( \mu'(x) = 1 - \mu(x) \) for each \( x \) in \( X \);
5. \( F(\mu + \sigma) + F(\mu \cap \sigma) = F(\mu) + F(\sigma) \).

Regarding the third axiom, the sharpened order "≤" is defined as [25]:

\[
\mu \leq \sigma \Leftrightarrow \min(0.5, (\mu(x))) \geq \min(0.5, (\sigma(x))) \quad \text{and} \quad \max(0.5, (\mu(x))) \leq \max(0.5, (\sigma(x)))
\] (2.1)

**Definition 1.** Let \( S = \{\mu_1, \mu_2, ..., \mu_n\} \) be a fuzzy set. According to the opinion of Luca and Termini [25], the fuzziness of S can be formulated as:

\[
F(S) = \frac{1}{n} \sum_{i=1}^{n} (\mu_i \log \mu_i + (1 - \mu_i) \log (1 - \mu_i))
\] (2.2)

Eq. (2.2) has many equivalent forms where the “equivalent” means the same extreme values and same monotonicity. For instance, the following formulas (2.3) or (2.4) can be considered as a simple form of fuzziness when \( n = 2 \) and \( S \) is normalized (i.e. \( \mu_1 + \mu_2 = 1 \)).

\[
F_1(S) = 1 - \mu_1^2 - (1 - \mu_1)^2
\] (2.3)

\[
F_2(S) = \begin{cases} 
\frac{\mu_1}{1 - \mu_1} & 0 \leq \mu_1 \leq 0.5 \\
\frac{1 - \mu_1}{\mu_1} & 0.5 \leq \mu_1 \leq 1
\end{cases}
\] (2.4)

The fuzziness of a fuzzy set defined by (2.2) achieves maximum when the element \( \mu_i = 0.5 \) for each \( i \) (1 ≤ \( i \) ≤ \( n \)), and achieves minimum when the element \( \mu_i = 1 \) or \( \mu_i = 0 \) for every \( i = 1, 2, ..., n \).

We connect the classifier output together with the fuzziness of a fuzzy set [28]. In literatures we can find that many classifiers can have the output forms of fuzzy vector which means a discrete fuzzy set. Although the classifiers directly output a real
vector in which each element can be a real member, it is easy to make a normalization for the output such that the real vector becomes a fuzzy vector, where the dimension of the vector is the number of classes for a classification problem. This paper does not focus on the normalization process and supposes that the output of a classifier is a fuzzy vector in which each component represents the degree of the case belonging to the corresponding class. Given a set of training samples \( \{X_i\}_{i=1}^{N} \) from which a classifier is trained with outputs of fuzzy vectors, we can obtain a membership matrix \( U = (\mu_{ij})_{c \times N} \) where \( c \) represents the number of classes. After a normalization process, the matrix has the following properties:

\[
\sum_{i=1}^{c} \mu_{ij} = 1, \quad 0 < \sum_{i=1}^{c} \mu_{ij} < N, \quad \mu_{ij} \in [0, 1]
\]

(2.5)

where \( \mu_{ij} \) denotes the membership of \( j \)th sample \( X_j \) belonging to the \( i \)th class.

We highlight in this section that the matrix \( U \) is obtained after training and, for each sample, the trained classifier will produce an output vector in a form of fuzzy set \( \mu_i \) which is a row in matrix \( U \). Based on (2.2), the fuzziness of trained classifier on \( X_j \) can be evaluated by

\[
F(\mu_j) = -\frac{1}{c} \sum_{j=1}^{c} (\mu_{ij} \log \mu_{ij} + (1 - \mu_{ij}) \log (1 - \mu_{ij}))
\]

(2.6)

3. Extreme learning machine with cost-sensitive class

Neural networks are widely used in many fields because of their strong ability to approximate complex nonlinear mappings. The approximation ability of randomly weighted neural networks is confirmed in [20,21,29]. Extreme learning machine (ELM) [19] is an algorithm for training single-hidden-layer feedforward neural networks (SLFNs), which is reported in many references to have faster learning speed and better generalization performance than the traditional algorithms of training feed-forward neural networks. The traditional algorithms for training SLFN such as BP learning algorithm need to tune parameters artificially and therefore are time consuming.

Algorithm 1. The initial ELM algorithm for training an SLFN is described as follows. Input: A training set \( S \) with \( N \) training samples which can be represented a feature matrix \( X_{N \times d_f} \), \( d_f \) is the dimension of feature vector, and a tag matrix \( T_{N \times d_o} \) do is the number of classes; a predefined interval \([a, b] \) where \( a \) and \( b \) are two real numbers \((a > b) \)

Output: A function \( f(x_1, x_2, \cdots, x_d) \) which is corresponding to Fig. 1

(1) Generate the hidden layer weight matrix \( A_{d_f \times d_h} \), \( d_h \) is the dimension of hidden layer, and bias matrix \( B_{N \times d_h} \) by randomly selecting real numbers from \([a, b] \).

(2) The output matrix of hidden layer can be computed through the following formula:

\[
H_{N \times d_h} = h(x) = h(XA + B)
\]

(3.1)

where,

\[
X_{N \times d_f} = \begin{bmatrix}
X_{11} & X_{12} & \cdots & X_{1d_f} \\
X_{21} & X_{22} & \cdots & X_{2d_f} \\
\vdots & \vdots & \ddots & \vdots \\
X_{N1} & X_{N2} & \cdots & X_{Nd_f}
\end{bmatrix}
\]

\[
A_{d_f \times d_h} = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1d_h} \\
\vdots & \vdots & \ddots & \vdots \\
a_{d_f1} & a_{d_f2} & \cdots & a_{d_fd_h}
\end{bmatrix}
\]

\[
B_{N \times d_h} = \begin{bmatrix}
b_{11} & b_{12} & \cdots & b_{1d_h} \\
\vdots & \vdots & \ddots & \vdots \\
b_{N1} & b_{N2} & \cdots & b_{Nd_h}
\end{bmatrix}
\]

The function \( h(x) \) is the activation function which is used to increase the nonlinear impact of ELM model. Generally, the function \( h(x) \) can be the following forms:

(1) Liner function: \( h(x) = k \times x + c \)

(2) Ramp function: \( h(x) = \begin{cases} T, x > c \\ k \times x, |x| \leq c \\ -T, x < -c \end{cases} \)

(3) Threshold function: \( h(x) = \begin{cases} 1, x \geq c \\ 0, x < c \end{cases} \)

(4) Sigmoid function: \( h(x) = \frac{1}{1 + \exp(-x)} (0 < h(x) < 1) \)

In this paper, we use sigmoid function to be the activation function.

(3) The weights of output layer nodes are represented as a matrix \( \beta_{d_o \times d_h} \) which can be obtained by solving a least square problem

\[
\min_{\beta} \|H\beta - T\| \quad \text{(3.2)}
\]

By solving its regular equations, the solution can be represented as:

\[
\beta = (H^T H)^{-1} H^T \quad \text{(3.3)}
\]

The problem can also be transformed to solve a Moore–Penrose generalized inverse of matrix \( H \). In this setting, it corresponds to a minimum norm least square problem with the solution \( \beta = H^+ T \) where \( H^+ \) represents the Moore–Penrose generalized inverse matrix of \( H \).

(4) The output function can be written as

\[
f(x) = h(x)\beta \quad \text{(3.4)}
\]

We start to incorporate the cost-sensitive concept into the ELM training. From reference we can know different types of cost-sensitive training. Usually it can be categorized three models, i.e., the learning is sensitive to specific examples, to particular features, or to different classes respectively. We focus on the third one which indicates that, for any example, the cost of classifying it wrongly from class A to class B is different from the cost of classifying it wrongly from class B to class A.
Suppose that the classification problem has \( m \) classes and the class–cost matrix is represented as
\[
C = (C_{ij})_{m \times m} = \begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{m1} & C_{m2} & \cdots & C_{mm} \end{pmatrix}_{m \times m}
\] (3.5)
where \( C_{ij} \) is the cost that the true class of an example is the \( i \)th but it is wrongly classified as the \( j \)th class where \( i \) and \( j \) can be any integers in \([1, m]\). Obviously each diagonal element in the matrix \( C \) is zero.

Let \( k_j = C_{1j} + C_{2j} + \cdots + C_{mj} \) which denotes weight of wrongly classifying an example of which the true class is the \( j \)th. We embed these weights into the ELM training listed in Algorithm 1. Examples in the training set are first sorted according to their class labels. All of training examples are categorized as \( m \) groups. Suppose that the \( j \)th group (corresponding to class \( # j \)) has \( n_j \) examples \((j = 1, 2, \ldots, m)\). We rewrite the matrix \( H \) in Algorithm 1 as \( \widetilde{H} = (H^1, H^2, \ldots, H^m)^T \) where \( H^j \) is a block of matrix \( H \) with \( n_j \) rows and \( d_l \) columns \((j = 1, 2, \ldots, m)\). Then, we consider the optimization problem formulated below.

\[
\min_{\beta} \left\| \begin{pmatrix} k_1H_1 \beta \\ k_2H_2 \beta \\ \vdots \\ k_mH_m \beta \end{pmatrix} - \begin{pmatrix} k_1T_1 \\ k_2T_2 \\ \vdots \\ k_mT_m \end{pmatrix} \right\| \\
= \min_{\beta} (k_1\|H_1 \beta - T_1\| + k_2\|H_2 \beta - T_2\| + \cdots + k_m\|H_m \beta - T_m\|)
\] (3.6)

This is a weighted least square problem. Let \( H_\lambda = (k_1H_1^T, k_2H_2^T, \ldots, k_mH_m^T)^T \) and \( T_\lambda = (k_1T_1^T, k_2T_2^T, \ldots, k_mT_m^T)^T \). As same as in Algorithm 1, the solution can be represented as:
\[
\beta = (H_\lambda^T H_\lambda)^{-1} H_\lambda^T T_\lambda
\] (3.7)

Also the solution can be represented via Moore–Penrose generalized inverse. That is, the solution can be denoted as \( \beta = H_\lambda^+ T_\lambda \) where \( H_\lambda^+ \) represents the Moore–Penrose generalized inverse matrix of \( H_\lambda \). In summary we have the algorithm for training a class-sensitive ELM.

Algorithm 2. Let \( k = (k_1, k_2, \ldots, k_m) \) be a given weight vector corresponding to the cost of misclassification with respect to \( m \) classes, \( H_\lambda = (k_1H_1^T, k_2H_2^T, \ldots, k_mH_m^T)^T \) and \( T_\lambda = (k_1T_1^T, k_2T_2^T, \ldots, k_mT_m^T)^T \). Replacing matrices \( H \) and \( T \) in Algorithm 1 with \( H_\lambda \) and \( T_\lambda \) respectively, we have the solution
\[
\beta = (H_\lambda^T H_\lambda)^{-1} H_\lambda^T T_\lambda
\] (3.8)

It is noted that Algorithm 2 will become Algorithm 1 when the weight vector is equal to \((1, 1, \ldots, 1)\), i.e., each class is equally treated for the cost of misclassification.

4. Sample categorization based on uncertainty

This section reports an experimental observation regarding the training/testing cost among sample categories with different uncertainty. We can use Algorithm 2 to train an ELM with a given class cost matrix and a training set of classification problem. The trained ELM can have a vector output for any example (training or testing), and the vector can further be normalized as a fuzzy set. Then all examples (training or testing) are grouped as several categories based on the sorting of sample uncertainty. It is experimentally observed that a significant difference of total cost (training or testing) exists among the several categories.

Algorithm 3. Sample categorization based on uncertainty.

1. Divide the data set samples into two parts randomly based on a given ratio, i.e., the training set and testing set respectively.
2. Use Algorithm 2 to train an ELM for a given class cost matrix.
3. Match each sample (training/testing) to the trained ELM and get a fuzzy vector output.
4. Compute the fuzziness for each sample’s output, and based on the amount of fuzziness, sort samples.
5. Group the samples (training and testing respectively) as 3 categories based on the fuzziness amount sorting, i.e., low, middle, and high fuzziness categories of samples.
6. Calculate respectively averaged cost of training and testing for each of 3 categories.
7. Calculate respectively the averaged fuzziness of training and testing for each of 3 categories.
8. Repeat steps 1–7 to get averaged training and testing cost for low and high fuzziness categories.

Some notes on Algorithm 3. In step 3) the output of the trained ELM may not be a fuzzy vector. Suppose the initial output vector is \( \mu = (\mu_1, \mu_2, \ldots, \mu_n) \), then the fuzzy vector \( v = (v_1, v_2, \ldots, v_n) \) is given by (4.1) while we make the appointment: \( \log(0) = 0 \).

\[
\mu_i = \begin{cases} 
\mu_i & 0 < \mu_i < 1 \\
0 & \mu_i \leq 0 \\
1 & \mu_i \geq 1 
\end{cases}
\] (4.1)

In step (4) the formula of computing fuzziness is listed in Section 2 (25). In step (5) the number of samples for each category depends on two thresholds. With different thresholds, the results may be far different with each other. Using appropriate thresholds can better demonstrate the experimental results. In our experiments, we define the thresholds equal to the values which evenly divide the sample set into three parts based on their fuzziness degrees. Honestly speaking, it is hard to find the best thresholds to reach an optimal result of experiments. We try to study this issue in the future work deeply.

We now experimentally observe the averaged cost of low and high fuzziness categories respectively both for training and testing sets. The datasets used for our experiments are listed in Table 2 and the experimental results are summarized in Table 3.

Figs. 2–5 show the change of training and testing cost with experimental times for some datasets. From Table 3 and Figs. 2–5 we can clearly see that the averaged cost of high fuzziness category is significantly bigger than that of low fuzziness category both for training and testing. These illustrate that the averaged cost is closely related to the uncertainty. Thus, the averaged cost for a certain class with high fuzziness degree is always bigger than those classes with low fuzziness degree. It is verified that the cost with different fuzziness is significantly different from each other, and the higher fuzziness degree is, the bigger total cost is (both for training and testing).

5. Our approach and experimental demonstration

Section 4 indicates experimentally an interesting phenomenon, that is, the testing cost of the low fuzziness category of samples is generally less than the testing cost of high fuzziness category for a given class cost matrix. Carefully studying this phenomenon, we have three approaches to reduce the total cost of testing.

1. Divide-and-conquer strategy: It means we can regularly use the trained classifier to predict classes of samples of low fuzziness category and use an enhanced technique such as ensemble learning to particularly predict the high fuzziness samples.
Table 2
Datasets used for our experiments.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Total sample</th>
<th>Input features</th>
<th>Class</th>
<th>Hidden nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>pima</td>
<td>768</td>
<td>7</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>credit(0–1)21</td>
<td>690</td>
<td>6</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>magic(0–1)</td>
<td>19,020</td>
<td>10</td>
<td>2</td>
<td>150</td>
</tr>
<tr>
<td>sonar21</td>
<td>208</td>
<td>60</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>musk01(0–1)21</td>
<td>476</td>
<td>166</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>Spambase(p10)21</td>
<td>458</td>
<td>57</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>AU1_21</td>
<td>1000</td>
<td>20</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>wilt</td>
<td>500</td>
<td>5</td>
<td>2</td>
<td>50</td>
</tr>
<tr>
<td>wineQW(0–1)435</td>
<td>4535</td>
<td>11</td>
<td>3</td>
<td>150</td>
</tr>
<tr>
<td>wineQW(p10)435</td>
<td>449</td>
<td>11</td>
<td>3</td>
<td>50</td>
</tr>
<tr>
<td>wineQR(0–1)345</td>
<td>1518</td>
<td>11</td>
<td>3</td>
<td>100</td>
</tr>
<tr>
<td>page(0–1)254</td>
<td>532</td>
<td>10</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>Abalone-r(0–1)91,113</td>
<td>1379</td>
<td>7</td>
<td>3</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 3
Experimental results of Algorithm 3.

<table>
<thead>
<tr>
<th>Pima</th>
<th>Training cost</th>
<th>Testing cost</th>
<th>Hidden nodes</th>
<th>Class cost matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low fuzziness category</td>
<td>0.1508</td>
<td>0.3023</td>
<td>50</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.6111</td>
<td>0.6047</td>
<td></td>
<td></td>
</tr>
<tr>
<td>credit(0–1)21</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.0957</td>
<td>0.3739</td>
<td>100</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.8174</td>
<td>0.7478</td>
<td></td>
<td></td>
</tr>
<tr>
<td>magic(0–1)</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.1587</td>
<td>0.1164</td>
<td>150</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.5410</td>
<td>0.5726</td>
<td></td>
<td></td>
</tr>
<tr>
<td>sonar21</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.0303</td>
<td>0.1944</td>
<td>50</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.2727</td>
<td>0.6944</td>
<td></td>
<td></td>
</tr>
<tr>
<td>musk01(0–1)21</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.1795</td>
<td>0.2500</td>
<td>50</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.5641</td>
<td>0.7625</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spambase(p10)21</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.1600</td>
<td>0.2338</td>
<td>10</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.7733</td>
<td>0.7922</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AU1_21</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.2108</td>
<td>0.5663</td>
<td>100</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.5181</td>
<td>0.7289</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wilt</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.0120</td>
<td>0.0482</td>
<td>50</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 3 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.3614</td>
<td>0.3494</td>
<td></td>
<td></td>
</tr>
<tr>
<td>wineQW(p10)435</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.5697</td>
<td>0.4331</td>
<td>100</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 2 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.6335</td>
<td>0.6457</td>
<td></td>
<td></td>
</tr>
<tr>
<td>page(0–1)254</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.3864</td>
<td>0.1011</td>
<td>10</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 2 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.8182</td>
<td>0.4719</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abalone-r(0–1)91,113</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.5833</td>
<td>0.5628</td>
<td>100</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 2 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.8158</td>
<td>0.8225</td>
<td></td>
<td></td>
</tr>
<tr>
<td>winequality-white657</td>
<td>Training cost</td>
<td>Testing cost</td>
<td>Hidden nodes</td>
<td>Class cost matrix</td>
</tr>
<tr>
<td>Low fuzziness category</td>
<td>0.5007</td>
<td>0.5013</td>
<td>50</td>
<td>\begin{bmatrix} 0 &amp; 1 \ 2 &amp; 0 \end{bmatrix}</td>
</tr>
<tr>
<td>High fuzziness category</td>
<td>0.8093</td>
<td>0.7024</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Three way decision model [30]: It sets up a threshold. If a sample has fuzziness bigger than the threshold, no decision is made. It is explained that the current information for the sample is not sufficient to predict its class.

Semi-supervised learning technique [31]: It thinks of that the low fuzziness samples are quality samples and are good enough to participate in training. These testing samples together with their predicted class labels are added in the training set for retraining.

Approach 1, i.e., the divide-and-conquer, which is a general strategy suitable for large scale problems. It is often criticized to have difficulties of lacking specific partition and fusion algorithms. Approach 2, i.e., the three way decision, refuses to make decision for high uncertainty samples which are usually very important to the model development. In comparison with approaches 1 and 2, approach 3, i.e., the semi-supervised learning is simpler, easier to implement, and more effective for reducing the total cost.

Algorithm 4. Adding low fuzziness samples together with their predicted class labels in the training set and retraining.

(1) Divide the data set samples into two parts randomly based on a given ratio, i.e., the training set and testing set respectively.
(2) Use Algorithm 2 to train an ELM for a given class cost matrix, and compute the fuzziness of each sample’s output both for training and testing.
(3) Calculate the averaged cost both for training and testing, then marked as first training averaged cost and first testing averaged cost.
(4) Group the samples (training and testing respectively) as 3 categories based on the fuzziness magnitude sorting, i.e., low, middle, and high fuzziness categories of samples.
(5) Add a number of testing samples with low fuzziness and their class labels predicted by the basic classifier in the original training set, and retraining an ELM on the enlarged training set.
(6) Calculate the averaged cost both for training and testing for the new classifier, and marked as second training averaged cost and second testing averaged cost.

Please cite this article as: H. Zhu, X. Wang, A cost-sensitive semi-supervised learning model based on uncertainty, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.010
(7) Compare the first averaged cost with the second averaged cost both for testing and training.

Some notes on Algorithm 4. The main idea of Algorithm 4 is putting some testing samples with lowest fuzziness and their predicted class labels into the original training samples, and retraining on an enlarged training set.

One problem we should pay attention to is that the data samples we add into the original training set may have the same category. In this situation, adding low fuzziness samples to the original training set and then retaining may not result in a classifier performance improvement. And furthermore, the algorithm is invalid and unavailable. To solve this problem, we change the algorithm which obtains the data samples according only to fuzziness of output vector of each testing sample which we add into the original training set. In addition to the fuzziness, we need to consider the distribution of class labels of added samples. In summary, the handling strategy in this situation is to keep the balance of the categories of the data samples with the lowest fuzziness.

Datasets used in Algorithm 4 and other related information including the structure of neural networks are listed in Table 2. The class cost matrix for 2 and 3 classes are given by

\[
C_{22} = \begin{bmatrix} 0 & 1 \\ 3 & 0 \end{bmatrix}, \quad C_{33} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 2 & 0 & 0 \\ 1 & 4 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\]  

Experimental results are listed in Table 4.

The change of averaged training (resp. testing) costs before and after adding low fuzziness samples with the experimental number of times for some datasets is shown in Figs. 6–9.

Furthermore we experimentally observe the change of cost difference between low and high fuzziness categories with the class cost matrix. From Figs. 10 and 11, it is not hard to see that the difference of total averaged cost between before-adding and after-adding low fuzziness samples is not sensitive to the cost matrix.

Keeping the first class misclassification cost unchanged and increasing the second class misclassification cost from 1 to 3, we ob-

---

### Table 4

Experimental results of Algorithm 4.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Before adding samples</th>
<th>After adding samples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Training cost</td>
<td>Testing cost</td>
</tr>
<tr>
<td>pima</td>
<td>0.5833</td>
<td>0.6074</td>
</tr>
<tr>
<td>Phoneme-r(0-1)21</td>
<td>0.4327</td>
<td>0.3946</td>
</tr>
<tr>
<td>magic(0-1)</td>
<td>0.4110</td>
<td>0.4151</td>
</tr>
<tr>
<td>wineQR(0-1)345</td>
<td>0.4801</td>
<td>0.5475</td>
</tr>
<tr>
<td>wineWQ(0-1)435</td>
<td>0.5309</td>
<td>0.5576</td>
</tr>
<tr>
<td>Abalone-r(0-1)9113</td>
<td>0.5613</td>
<td>0.5183</td>
</tr>
<tr>
<td>winequality-white657</td>
<td>0.5503</td>
<td>0.5405</td>
</tr>
</tbody>
</table>

---

**Fig. 6.** Training cost for magic(0-1).

**Fig. 7.** Testing cost for magic(0-1).

**Fig. 8.** Training cost for winequality-white657.
serve the corresponding experimental results. The difference between the first cost and the second cost does not vary sharply both for training and testing processes. Generally, the misclassification cost from one class to another is given by users according to specific situations. The experiments confirm that the total cost change is not sensitive to the misclassification cost perturbation which should be under a controllable range. It is not meaningful to discuss the total cost increase for large change of misclassification inputs. We speculate that the non-sensitivity is resulted from the stability of ELM model and its random selection mechanism of connection weights. Unfortunately we have not yet had a model to analyze this sensitivity.

6. Concluding remarks

This paper empirically confirms that, for classification problem with cost-sensitive classes, samples with low output-fuzziness are usually of high quality. It will reduce the total cost of prediction if these high quality samples together with their predicted labels are added in the original training set and a retraining on the enlarged training set is conducted. We highlight that it is an empirical observation. We have not yet a mathematical equation to model this observation and then to logically prove the corresponding results. Some initial study on building the model indicates that it is interesting but very difficult.

It may lead to an open problem that, under the condition of learning consistency (i.e., with probability 1, the performance of classifier learned from A is better than that from B if B is a subset of A), how to build a mathematical model to prove that the performance of classifier trained from S+ will be better than that from S? Here S is the original training set which is categorized as two parts, i.e., the low uncertainty and high uncertainty parts and S+ is the union of S and the part of low uncertainty samples (with predicted labels). It assumes that the prediction accuracy of the S-trained classifier on the low uncertainty parts is much higher than the training accuracy.

Acknowledgments

This study was supported by Basic Research Project of Knowledge Innovation Program in Shenzhen (JCYJ201503241440036825), and National Natural Science Foundations of China (71371063).

References


Please cite this article as: H. Zhu, X. Wang, A cost-sensitive semi-supervised learning model based on uncertainty, Neurocomputing (2017), http://dx.doi.org/10.1016/j.neucom.2017.04.010

Hongyu Zhu is a postgraduate of college at Computer Science and Software Engineering in Shenzhen University under the supervision of Professor Xi-zhao Wang. She received her bachelor’s degree from Jangsu Normal University in 2011. Her research interests include machine learning, cost sensitive model, fuzzy sets and three-way decision.

Xi-Zhao Wang (M’03 – SM’04 – F’12) received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1998. He is currently a professor with the College of Computer Science and Software Engineering, Shenzhen University, Guangdong, China. From September 1998 to September 2001, he was a research fellow with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. He has more than 180 publications, including four books, seven book chapters, and more than 100 journal papers in the IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Systems, Man, and Cybernetics, IEEE Transactions on Fuzzy Systems, Fuzzy Sets and Systems, Pattern Recognition, etc. His H-index is 20 (up to June 2014). His current research interests include learning from examples with fuzzy representation, fuzzy measures and integrals, neurofuzzy systems and genetic algorithms, feature extraction, multiclassifier fusion, and applications of machine learning. Dr. Wang has been the PI/Co-PI for 16 research projects supported partially by the National Natural Science Foundation of China and the Research Grant Committee of Hong Kong Government. He was the BoG Member of the IEEE Systems, Man, and Cybernetics Society (IEEE SMCS) in 2005, 2007–2009, and 2012–2014; the Chair of the IEEE SMC Technical Committee on Computational Intelligence, an associate editor of IEEE Transactions on Cybernetics; an associate editor of Pattern Recognition and Artificial Intelligence; a Member of the Editorial Board of Information Sciences; and an Executive Member of the Chinese Association of Artificial Intelligence. He was the recipient of the IEEE SMCS Outstanding Contribution Award in 2004 and the recipient of IEEE SMCS Best Associate Editor Award in 2006. He is the general Co-Chair of the 2002–2014 International Conferences on Machine Learning and Cybernetics, cosponsored by IEEE SMCS. He is a distinguished lecturer of the IEEE SMCS.