Extension of Learning Vector Quantization to Cost-sensitive Learning

Ning Chen, Bernardete Ribeiro, Armando Vieira, Jo ão Duarte, and Jo ão C. Neves

***Abstract*—Learning vector quantization (LVQ) is an effective network model to solve classification tasks in a wide variety of real world applications. The usage of LVQ has been extended to hybrid data type. In this paper, we propose a weighted version of BNCLVQ, which incorporates the cost matrix into prototype learning and labeling by means of instance weighting. Empirical results show the superiority of proposed algorithm over original NBCLVQ and some variants on both binary-class data and multi-class data.**

***Index Terms*—Classification, Cost-sensitive learning, Learning vector quantization, Hybrid data type.**

I. Introduction

The misclassification cost plays an important role in decision making due to the fact that the costs of different errors are usually unequal. For example, in medical diagnosis misclassifying a life-threatening disease as healthy is much more serious than false alarm. In bankruptcy prediction, the missing prediction of a bankruptcy company will result in a higher loss than the opposite type of error. In software detect process, the misclassification of defect-prone modules usually incurs higher cost than that of not-detect-prone ones. This makes the classification cost occupy a unique position in the research of cost-sensitive learning.

Many classification methods are able to handle certain type of data. For data with hybrid numeric and nominal features, a transformation procedure is the commonly used approach, but usually damages the nature of data [1]. As an extension of standard LVQ, BNCLVQ (Batch Numeric and Categorical Learning Vector Quantization) is able to classify hybrid numeric and categorical data [2]. In this paper we advance the usage of BNCLVQ to cost-sensitive learning. We present a weighted BNCLVQ algorithm (wBNCLVQ), integrating the instance weights which convey the cost of errors into the network learning and labeling. A number of data sets with both binary-class and multi-class are used in the experiments. The proposed algorithm is compared with original BNCLVQ and some cost-sensitive implementations in terms of classification error and expected misclassification cost. Experimental results demonstrate that the proposed algorithm outperforms the competing algorithms on lowering misclassification cost and improving the robustness in different situations. The paper is organized as follows. In section II, an introduction study of LVQ and cost-sensitive learning is presented. In section III, the methodology of weighted BNCLVQ algorithm is explained. In section IV, the empirical study is reported. The contributions and future remarks are addressed in section V.

Manuscript received July 30, 2010; revised March 17, 2011. This work was supported by project C2007-FCT/442/2006-GECAD/ISEP (Knowledge Based, Cognitive and Learning Systems).

Ning Chen is with GECAD, Instituto Superior de Engenharia do Porto, Instituto Politecnico do Porto (telphone: 351-22-8340500; fax: 351-22-8321159; email: ningchen74@gmail.com).

Bernardete Ribeiro is with CISUC, Department of Informatics Engineering, University of Coimbra, Portugal (email: bribeiro@dei.uc.pt).

Armando Vieira is with Instituto Superior de Engenharia do Porto, Instituto Politecnico do Porto (email: asv@isep.ipp.pt).

Jo ão Duarte is with GECAD, Instituto Superior de Engenharia do Porto, Instituto Politecnico do Porto (email: jmmd@isep.ipp.pt).

Jo ão C. Neves is with ISEG, School of Economics, Technical University of Lisbon, Portugal (email: jcneves@iseg.utl.pt).

II. Related Work

Classification is a supervised machine leaning method to separate the instances into predefined classes. It is usually performed in the manner that derives the inherent models from a training data set of previously labeled samples and predicts the class of new samples based on the learned model. In the literature, a large number of algorithms have been proposed to solve the classification tasks, such as decision tree, neural network, support vector machine, case-based reasoning, fuzzy logic, rough set, discriminant analysis, bayesian networks, hidden markov model, hybrid and ensemble approach. Meantime, the application of classification has covered almost all domains in the real world including image analysis, drug discovery, medical diagnosis, computer vision, speech and handwriting recognition, biometric identification, credit scoring, fraud detection etc.

Learning vector quantization (LVQ) was invented by Kohonen as an improvement over labeled vector quantization. The main objective is to approximate the data distribution with a number of prototypes. Since its origination from LVQ1 to LVQ2, LVQ2.1, and LVQ3 [3], a lot of variants have been implemented to improve the convergence, diminish the errors, simplify the network processing and advance the usage. Some representative studies are as follows. Generalized LVQ (GLVQ) uses a stochastic gradient descent to minimize an explicit error function [4]. Concerning the limitation of Euclidian metric, the importance of input dimensions is adapted in distinction sensitive LVQ (DLVQ), or determined based on Hebbian learning in common with standard LVQ (RLVQ) [5] and generalized LVQ (GRLVQ) [6]. In an initialization insensitive LVQ [7], the commonly used nearest neighbor distance is substituted with harmonic average distance. In [8], a subset of points located in risk zones contributes to the learning of prototypes and consequently decreases the misclassification errors. In [2], a batch type LVQ algorithm

(BNCLVQ) is proposed for classifying data with hybrid numeric and nominal values. Nowadays, LVQ is widely applied in many domains, such as bankruptcy prediction [9], gene expression analysis [10], creditworthiness evaluation [11] and image segmentation [12]. It is reported to achieve comparable performance with other neural networks, support vector machines and multivariate statistical methods [13].

Cost-sensitive learning becomes a hot research topic in the recent study of classification. Many evidences have demonstrated the necessity of incorporating different types of cost into classification. Among them, the cost of misclassification error is the mostly concerned one. As indicated in [14], there are different categories of misclassification cost, e.g., constant error cost, conditional error cost on individual case, time, other instances or feature. The present-day research concentrates on the former, in which the instances belonging to a class have the same cost. The proposed algorithms in this paper are related to the class-dependent cost formalization.

The so far studies on cost-sensitive learning mainly follow such methodologies as sampling, weighting and threshold-moving. The misclassification cost can be conveyed in the distribution of training data by means of the often-used sampling techniques, such as over-sampling and under-sampling. For example, over-sampling is applied to obtain sensitive and accurate support vector machine classifier on drug data [15]. Over-sampling duplicates the expensive examples artificially, but simultaneously increases the training time and leads to overfitting. On the contrary, under-sampling discards inexpensive examples, which usually results in a general loss of information and potentially general rules [16]. These limitations can be ameliorated using intelligent sampling, which attempts to remove redundant inexpensive examples or generate new expensive examples by interpolation [17].

A widely applied method is introducing some weights into the underlying learning algorithms. The main issue is how to set appropriate value of the weights and adapt the learning methodology to specified weights. Regarding decision tree, the pioneer method may be the cost-sensitive tree induction employing the greedy divide-conquer algorithm [18]. The instance-weighting C4.5 decision tree assigns the samples of different classes with different weights proportional to the corresponding costs [19]. The cost matrix is incorporated into back-propagation neural network [20]. A cost-sensitive regularized least square algorithm uses weights to penalize different fractions of classes in medical diagnosis [21]. Weighting strategies are employed to introduce cost into the weight-updating of a boosting algorithm [17]. The weighted-instance approaches of online LVQ and batch LVQ are described in [22] and achieve comparable performance. Besides, some researchers pointed out that the cost-sensitive classification can be solved by an optimization procedure with well-defined objective functions for particular classifiers [23]. By modifying the objective functions, the cost is integrated into mathematical programming and genetic algorithm based neural network [24].

Threshold-moving is a simple way to make a minimum error tree cost-sensitive. A new instance is assigned to a class with the minimal expected misclassification cost instead of the majority class in the leaf node [25]. It is noted that threshold-moving can be applied to classifiers with real-valued output, such as neural network and boosting method. The cost-relevant threshold moves the classification output of an Adaboost algorithm towards the costly class [26]. In [27], the cost matrix is integrated with basic LVQ algorithm using sampling and threshold-moving. The matter at issue is how to select a desired threshold. For this purpose, the knowledge to the specific classifier and application domain is required. A bisection method is developed to detect the optimal threshold that minimizes the overall misclassification cost of neural networks [28].

III. Weighted BNCLVQ Algorithm

Weighting is applicable to learning methods which can accept weights of samples. If there are two classes and the cost of a false positive is λ times larger than the cost of a false negative, a widely accepted assignment is to put a weight of λ on each negative training example. Afterwards, the learning algorithm is applied as usual. Although LVQ does not accept weighted input directly, it can be implemented by a simple way. In this section, we adapt the BNCLVQ algorithm to embedding the instance weights into the learning and labeling of the network.

A. *Weight Assignment*

Normally, the cost of misclassification error can be represented by a matrix, in which the value C(i, j), i, j=1,..., k indicates the cost of error that classifying an instance to class j, while it belongs to class i in fact. In this study, we simplify the cost matrix to a cost vector: $\{S_i \mid i=1,..., k\}$, where S_i means the cost of misclassifying a class i instance to others. The conversion is intuitional by summing the values of the cost matrix by rows followed by normalization so that the minimal cost is one. An example of cost matrix with 4 classes is shown in Table I, in which the last column gives the cost vector.

$$S_i = \sum\nolimits_j C(i,j), \quad S_i = \frac{S_i}{\min_j S_j} \tag{1}$$

Let N be the total number of instances in the training data, N_j the number of class j instances. The weight of an instance x belonging to class i can be calculated from the cost vector, so that the costly class instances have high weights and the sum of all instance weights is N [19]. Through this transformation, the weights are distributed to all instances proportional to the corresponding class cost.

$$\omega(x \in C_i) = S_i \frac{N}{\sum_j S_j N_j} \tag{2}$$

TABLE I An Example of Cost Matrix and Cost Vector.

Actual class	Predicted class				S_i
	C_1	C_2	C_3	C_4	
C_1	0	9	4	10	2.56
C_2	3	0	2	4	1
C_3	8	3	0	8	2.11
C_4	1	8	4	0	1.44

B. Cost-based Labeling Principle

The traditional majority voting labeling principle aims to minimize the classification errors. Differently, the cost-based labeling principle tries to minimize the expected cost of prediction with the consideration of cost matrix. Let P(i|x) (i=1,..., k) denote the probability of an instance x in class i, thereby the conditional risk R(i|x) implies the expected cost due to the prediction of x as i. As described in [29], it can be calculated as:

$$\mathrm{R}(i\mid x)=\sum\nolimits_{i\neq j}P(j\mid x)C(j,i) \tag{3}$$

Thus, the optimal label for x is the one with the minimal expected cost:

$$\min\nolimits_{\mathrm{i}} R(i\mid x) \tag{4}$$

The cost-sensitive labeling principle has been applied in various classification methods. Regarding LVQ, P(i|m$_p$) denotes the probability of neuron m$_p$ in class i, and can be estimated as the percent of the class i instances belonging to V$_p$ (the Voronoi set of the neuron m$_p$), then the expected cost of each class is calculated, and m$_p$ is labeled by the class with the minimal expected cost. The cost-based labeling is described as follows.

1) Calculate the probability for each class i:

$$\mathrm{P}(i\mid m_p)=|\,x\in V_p\wedge Class(x)=i\,|\,/\,|\,V_p\,|$$

2) Calculate the expected cost of each class:

$$\mathrm{R}(i\mid m_p)=\sum\nolimits_{i\neq j}P(j\mid m_p)C(j,i)$$

3) Label the neuron as the class with minimal expected cost:

$$\mathrm{Class(m_p)}=\min\nolimits_{\mathrm{i}} R(i\mid m_p)$$

C. Weighted-instance LVQ Training

As a prototype representation based on nearest neighbor approach, the distance metric plays an important role to LVQ. Some non-Euclidean distance metrics to classification methods are discussed in [30]. In BNCLVQ [2], the distance is based on squared Euclidean distance on numeric features and mismatch measurement on nominal features (0 for match and 1 for no match). The distance is able to enhance the accuracy of standard LVQ, especially on data sets with the mixture of data types.

We propose a weighted learning vector quantization approaches for hybrid data types (wBNCLVQ). The instance weights are incorporated into the updating of prototypes so that instances with higher weights have more influence on neurons.

In one batch round, the Voronoi set of each map neuron is computed by projecting the input data to its best matching unit (BMU), then the prototype is updated according to learning laws depending on class label and feature type. For numeric attribute j, the new value can be calculated in the following.

$$\mathrm{m_{pj}}=\frac{\sum_{x_i\in V_p}\omega(x_i)s_{ip}x_{ij}}{\sum_{x_i\in V_p}\omega(x_i)s_{ip}} \tag{5}$$

The function s$_{ip}$ denotes the class agreement between m$_p$ and x$_i$, having the value 1 if they are in the same class, and -1, otherwise.

$$\mathrm{s_{ip}}=\begin{cases}1 & if\ Class(m_p)=Class(x_i)\\ -1 & otherwise\end{cases} \tag{6}$$

For a nominal attribute j with the values { $\alpha_j^1,\ldots,\alpha_j^{n_j}$ }, a set of counters keeps the frequencies of variant values, taking the instance weights into account.

$$\mathrm{F}(\alpha_{\mathrm{j}}^{\mathrm{r}},m_{pj})=\sum\nolimits_{x_i\in V_p}v(\omega(x_i)s_{ip}\mid x_{ij}=\alpha_{\mathrm{j}}^{\mathrm{r}}) \tag{7}$$

Function *v(y| COND)* is y when *COND* holds and zero otherwise. This way, $\mathrm{F}(\alpha_{\mathrm{j}}^{\mathrm{r}},m_{pj}),r=1,\ldots,n_j$ represents a weighted vote regarding each value $\alpha_{\mathrm{j}}^{\mathrm{r}}$. For nominal features, the best category $\mathrm{c}=\max_{r=1}^{n_j}\mathrm{F}(\alpha_{\mathrm{j}}^{\mathrm{r}},m_{pj})$ is the new value for the next iteration. The learning rule on nominal variables is formulated as follows.

$$\mathrm{m_{pj}}=\begin{cases}c & if\ F(c,m_{pj})>0\\ m_{pj} & otherwise\end{cases} \tag{8}$$

The description of wBNCLVQ algorithm is shown as follows. The difference from BNCLVQ is that the instance weights are incorporated into the updating of prototypes. Moreover, the cost-based labeling is used to minimize the expected cost rather than classification error.

N: number of instances; m: number of map neurons; k: number of attributes; m$_p$: the prototype of neuron p; Class(m$_p$): class label of neuron p; λ: maximal number of iteration; ω(x): weight of instance x

```
1) For p = 1,…, m Initialize $m_p$ and Class($m_p$)
2) t = 0
3) For p = 1,…, m $V_p = \phi$
4) For i = 1,…, N
   a) Input $x_i$ to the map
   b) For j = 1,…, m Calculate d($x_i$, $m_j$)
   c) $\mathrm{m_p}=\arg\min_{\mathrm{j}} d(x_i,m_j)$ // Project $x_i$ to $m_p$
   d) $\mathrm{V_p}=V_p\cup\{x_i\}$ // Add $x_i$ to the Voronoi set of $m_p$
   e) $\mathrm{s_{ip}}=\begin{cases}1 & if\ Class(m_p)=Class(x_i)\\ -1 & otherwise\end{cases}$
5) For p = 1,…, m
   a) For j = 1,…, k
      ● If $F_j$ is numeric then
```

$$\mathrm{m_{pj}} = \frac{\sum_{x_i \in V_p} \omega(x_i) s_{ip} x_{ij}}{\sum_{x_i \in V_p} \omega(x_i) s_{ip}}$$

```
        // Update numeric attribute
    •   If $F_j$ is nominal then
        For r = 1,…, $n_j$
```

$$\mathrm{F}(\alpha_j^r, m_{pj}) = \sum_{x_i \in V_p} v(\omega(x_i) s_{ip} \mid x_{ij} = \alpha_j^r)$$

$$c = \max_{r=1}^{n_j} \mathrm{F}(\alpha_j^r, m_{pj})$$

$$\mathrm{m_{pj}} = \begin{cases} c & if\ F(c, m_{pj}) > 0 \\ m_{pj} & otherwise \end{cases}$$

```
        // Update nominal attribute
    b) For each Class  i in $V_p$
    •   $P(i \mid m_p) = |x \in V_p \wedge Class(x) = i| / |V_p|$
    •   $R(i \mid m_p) = \sum_{i \neq j} P(j \mid m_p) C(j,i)$
    c) $Class(m_p) = \min_i R(i \mid m_p)$
    d)
6)  t = t + 1
7)  Goto (3) until t = $\lambda$
```

IV. Empirical Analysis

TABLE II Data set description (No: nominal features, Nu: numeric features).

Data set	#instance	# No	# Nu	# class	distribution
diane	1200	0	30	2	600/600
transfusion	748	0	4	2	570/178
wpbc	198	0	33	2	157/47
hepatitis	155	19	6	2	85/70
haberman	336	0	3	2	255/81
ionosphere	351	0	34	2	126/225
diabetes	768	0	8	2	500/268
horse	368	7	15	2	232/136
german	1000	21	3	2	700/300
echocardiogram	131	1	6	2	88 /43
hypothyroid	3163	18	7	2	151/3012
lungcancer	32	56	0	2	9/23
credit	690	8	5	2	307/383
abalone	4177	1	7	3	147/1323/1447
iris	150	0	4	3	50/50/50
wine	178	0	13	3	59/71/48
soybean	47	35	0	4	10/10/10/17
xaa	94	0	18	4	28/20/26/20
glass	241	0	9	6	70/76/17/13/9/29
segmentation	210	0	19	7	30/30/30/30/30/30/30
zoo	101	15	1	7	41/20/5/13/4/8/10
ecoli	336	0	7	8	143/77/2/35/20/5/52/2
yeast	1484	0	8	10	244/429/463/163/51/44/35/30/20/5

A. Data Sets

In the empirical study, we select 22 data sets from UCI Machine Learning Repository [31], in which 12 data sets have binary-class, and the rest have multi-class. The properties of the data sets are characterized in Table II. These data sets are chosen from different domains varying in number of classes, features, and instances. Additionally, a real-world data set *diane* is used, containing information on a wide set of financial ratios shown in Table III. The data used in this study is a balanced sample composed of 600 distressed companies and 600 healthy ones.

B. Performance Evaluation

The performance of classification is usually evaluated by contingency matrix, in which each row represents the real class and each column represents the predicted class. The overall error rate (Err) is the most commonly used criterion for performance evaluation. As the cost should be taken into consideration in the evaluation, we also use the error rate of the highest costly class ($\mathrm{Err_{HC}}$) and the expected misclassification cost (EMC) on all instances. Let M(i,j) be the value of the row i and column j in the contingency matrix. The aforementioned criteria are defined as follows.

TABLE III Financial ratios of Diane companies.

Variable	Description
x1	Number of Employees Last available year
x2	Capital Employed / Fixed Assets
x3	Financial Debt / Capital Employed
x4	Depreciation of Tangible Assets
x5	Working Capital / Current Assets
x6	Current ratio
x7	Liquidity Ratio
x8	Stock Turnover days
x9	Collection Period days
x10	Credit Period days
x11	Turnover per Employee in EUR
x12	Interest / Turnover
x13	Debt Period days
x14	Financial Debt / Equity
x15	Financial Debt / Cashflow
x16	Cashflow / Turnover
x17	Working Capital / Turnover days
x18	Net Current Assets/Turnover days
x19	Working Capital Needs / Turnover
X20	Export
x21	Added Value per Employee in EUR
x22	Total Assets Turnover
x23	Operating Profit Margin
x24	Net Profit Margin
x25	Added Value Margin
x26	Part of Employees
x27	Return on Capital Employed
x28	Return on Total Assets
x29	EBIT Margin
x30	EBITDA Margin

$$\mathrm{Err}_{\mathrm{HC}} = \frac{\sum_{j,i\neq j} M(i,j)}{\sum_{j} M(i,j)}, where\ i = \arg\max_{j} S_j \tag{9}$$

$$Err = \frac{\sum_{i,j,i\neq j} M(i,j)}{N} \tag{10}$$

$$EMC = \sum_{i,j,i\neq j} M(i,j)C(i,j)/N \tag{11}$$

The robustness property denotes the performance of an algorithm on different data sets [16]. In this study, we investigate the robustness of LVQ algorithms in terms of the error of highest costly class and expected misclassification cost. Let D_1,..., D_n be the data sets used in experiments, A_1,..., A_m the compared algorithms, $v_{i,j}$ denote the performance of algorithm A_i on data D_j. For a particular data set, the maximal value among all algorithms represents the worst performance, then the relative performance $b_{i,j}$ is the absolute value divided by the maximum. The robustness of the algorithm is represented by summing up the relative performance over all data sets. A smaller value of indicates the more robust of a particular algorithm.

$$b_{ij} = \frac{v_{i,j}}{\max_{i=1}^{m} v_{i,j}}, r_i = \sum_{j=1}^{n} b_{i,j} \tag{12}$$

TABLE IV BINARY CLASS DATA SETS: PERFORMANCE COMPARISON AND SIGNIFICANCE TEST AT 5% LEVEL (*: VS BNCLVQ, +: VS CL-BNCLVQ).

Data set	BNCLVQ	cl-BNCLVQ	w BNCLVQ	MetaCost-LVQ
		$\mathrm{Err}_{\mathrm{HC}}$		
diane	0.110 (0.046)	0.099 (0.043)	**0.078** (0.046)*+	0.108 (0.091)
transfusion	0.236 (0.235)	**0.144** (0.165)*	**0.144** (0.184)*	0.151 (0.296)
wpbc	0.499 (0.247)	**0.380** (0.244)	0.405 (0.244)*	0.576 (0.386)
hepatitis	0.301 (0.176)	0.211 (0.186)	**0.197** (0.169)*	0.274 (0.177)
haberman	0.466 (0.335)	**0.362** (0.317)*	0.409 (0.389)	0.434 (0.433)
ionosphere	0.133 (0.136)	**0.110** (0.119)*	0.114 (0.129)*	0.153 (0.161)
diabetes	0.303 (0.152)	0.198 (0.159)*	**0.193** (0.143)*	0.284 (0.260)
horse	0.272 (0.112)	0.217 (0.097)*	**0.186** (0.088)*+	0.266 (0.139)
german	0.374 (0.377)	0.312 (0.306)*	**0.252** (0.252)*+	0.259 (0.251)
echocardiogram	0.294 (0.135)	0.234 (0.157)*	**0.200** (0.158)*	0.183 (0.232)
hypothyoid	0.353 (0.303)	0.344 (0.296)	**0.297** (0.264)+	0.597 (0.514)
lungcancer	0.396 (0.300)	0.369 (0.323)	**0.298** (0.322)	0.231 (0.107)
credit	0.148 (0.010)	**0.100** (0.023)*	0.114 (0.024) *	0.196 (0.048)
mean	**0.2989**	**0.2369**	**0.2223**	**0.2857**
		Err		
diane	0.120 (0.003)	0.134 (0.036)	0.122 (0.010)	0.151 (0.010)
transfusion	0.258 (0.011)	0.257 (0.038)	0.248 (0.024)	0.232 (0.007)
wpbc	0.267 (0.026)	0.328 (0.088)	0.275 (0.060)	0.250 (0.010)
hepatitis	0.321 (0.028)	0.345 (0.070)	0.345 (0.039)	0.314 (0.021)
haberman	0.310 (0.015)	0.316 (0.037)	0.289 (0.025)	0.263 (0.006)
ionosphere	0.121 (0.017)	0.145 (0.026)	0.125 (0.021)	0.144 (0.007)
diabetes	0.265 (0.008)	0.314 (0.077)	0.297 (0.046)	0.274 (0.013)
horse	0.196 (0.008)	0.214 (0.043)	0.231 (0.030)	0.215 (0.016)
german	0.300 (0.011)	0.298 (0.008)	0.303 (0.018)	0.264 (0.007)
echocardiogram	0.354 (0.037)	0.368 (0.025)	0.348 (0.028)	0.315 (0.012)
hypothyoid	0.030 (0.001)	0.031 (0.002)	0.034 (0.007)	0.048 (0.000)
lungcancer	0.263 (0.060)	0.273 (0.072)	0.287 (0.076)	0.244 (0.051)
credit	0.147 (0.005)	0.162 (0.016)	0.162 (0.014)	0.260 (0.023)
mean	**0.2272**	**0.2450**	**0.2359**	**0.2289**
		EMC		
diane	0.679 (0.216)	0.669 (0.206)	**0.617**(0.216)*+	0.808 (0.271)
transfusion	1.353 (0.432)	1.139 (0.420)*	1.096 (0.396)*	**1.044**(0.415)
wpbc	1.533 (0.417)	1.527 (0.617)	**1.361** (0.451)+	1.517 (0.606)
hepatitis	1.573 (0.480)	**1.386** (0.493)	1.407 (0.503)	1.619 (0.661)
haberman	1.704 (0.639)	1.562 (0.841)	1.447 (0.777)*+	**1.350** (0.779)
ionosphere	0.634 (0.288)	0.646 (0.275)	**0.599** (0.251)	0.754 (0.330)
diabetes	1.402 (0.520)	1.178 (0.604)*	**1.152** (0.563)*	1.334 (0.671)
horse	1.032 (0.312)	0.935 (0.265)*	**0.926** (0.284)*	1.060 (0.346)
german	1.782 (0.652)	1.662 (0.731)*	1.626 (0.735)*	**1.478** (0.596)
echocardiogram	1.804 (0.693)	1.626 (0.849)	1.525 (0.640)*	**1.279** (0.732)
hypothyoid	0.165 (0.082)	0.165 (0.079)	**0.154** (0.069)	0.266 (0.146)
lungcancer	1.288 (0.414)	1.203 (0.422)	**1.072** (0.484)	1.135 (0.171)
credit	0.746 (0.248)	**0.680** (0.238)*	0.707 (0.255)	1.176 (0.414)
mean	**1.2073**	**1.1059**	**1.0530**	**1.14**

C. *Experimental Strategy*

In the comparable study, we use the BNCLVQ as the baseline. One competing algorithm is cl-BNCLVQ (BNCLVQ in common with cost-based labeling) is used to detect the effectiveness of weighted prototype learning. Another is MetaCost [29], a well-known cost-sensitive method which provides a general framework to any classifier learning method by changing the label of each training example to its optimal label and then learning a classifier that predicts these new labels. We use the Weka implementation of MetaCost combined with LVQ (MetaCost-LVQ) [32] or the purpose of comparison. The experiments are performed in the following strategy:

1) The cost matrix is generated so that the diagonal values are 0 and others are random number taken from [1, 10], then the weight of instances is calculated.
2) The data set is divided randomly into ten folds for cross-validation. In each trial one fold is used as test data, and the remaining is used for training.
3) For each generated training data set, the LVQ approaches are applied resulting a learned, labeled map.
4) The test data is input to the resultant map, and the class is predicted via the nearest principle.
5) After the cross-validation is finished, the contingency matrix is obtained by summarizing the real class and predicted class for the entire data. Then the highest costly class error, overall error and expected misclassification cost are calculated from the contingency matrix and cost matrix.
6) The process is repeated 10 times with randomly generated cost matrix, and the average results are calculated.
7) The results are summarized over all data sets and different algorithms.

D. Experimental Results

The performance of aforementioned four algorithms, namely BNCLVQ, cl-BNCLVQ, wBNCLVQ and MetaCost-LVQ is shown. For each data set, the average results and standard deviation over different cost matrix configurations are given. The significance test is performed on 5% level between cl-BNCLVQ and BNCLVQ, wBNCLVQ and BNCLVQ, wBNCLVQ and cl-BNCLVQ respectively.

Table IV summarizes the results on binary class data sets. As expected, the three cost-related variants are indeed capable to lower the highest costly class error rate with a slight degradation on the overall error rate, and decrease the expected cost. Concerning the average performance, cl-BNCLVQ achieves a 20% reduction on Err_{HC} and a 8% reduction on EMC, wBNCLVQ achieves a 26% and 13% reduction on Err_{HC} and EMC respectively. The CostMeta-LVQ results in a small improvement with 4% and 6%

TABLE V MULTI-CLASS DATA SETS: PERFORMANCE COMPARISON AND SIGNIFICANCE TEST AT 5% LEVEL (*: VS BNCLVQ, +: VS CL-BNCLVQ)
.

Data set	BNCLVQ	cl-BNCLVQ	w BNCLVQ	MetaCost-LVQ
		Err_{HC}		
abalone	0.376 (0.113)	0.357 (0.194)	0.343 (0.209)	**0.338** (0.196)
glass	0.589 (0.329)	**0.509** (0.234)	0.532 (0.334)	0.727 (0.380)
segmentation	0.077 (0.088)	**0.053** (0.053)	0.063 (0.064)	0.254 (0.216)
iris	0.066 (0.041)	0.072 (0.044)	0.064 (0.039)+	**0.050** (0.044)
xaa	0.661 (0.140)	0.621 (0.147)	0.641 (0.206)	**0.526** (0.308)
wine	0.066 (0.052)	0.056 (0.052)	0.063 (0.056)	**0.043** (0.042)
zoo	0.092 (0.129)	**0.082** (0.124)	0.092 (0.129)	0.178 (0.188)
soybean	0.000 (0.000)	**0.000** (0.000)	**0.000** (0.000)	0.040 (0.052)
ecoli	0.401 (0.425)	**0.402** (0.424)	0.406 (0.422)	0.406 (0.381)
yeast	0.523 (0.184)	**0.542** (0.249)*	0.553 (0.236)	0.581 (0.324)
mean	**0.2850**	**0.2696**	**0.2757**	**0.3143**
		Err		
abalone	0.390 (0.003)	0.429 (0.040)	0.429 (0.037)	0.440 (0.013)
glass	0.332 (0.018)	0.367 (0.028)	0.377 (0.030)	0.427 (0.048)
segmentation	0.152 (0.015)	0.154 (0.024)	0.151 (0.021)	0.261 (0.025)
iris	0.053 (0.013)	0.059 (0.009)	0.054 (0.009)	0.042 (0.011)
xaa	0.484 (0.047)	0.542 (0.057)	0.550 (0.049)	0.532 (0.036)
wine	0.054 (0.013)	0.058 (0.019)	0.056 (0.011)	0.042 (0.006)
zoo	0.049 (0.012)	0.048 (0.011)	0.056 (0.015)	0.121 (0.024)
soybean	0.007 (0.024)	0.000 (0.000)	0.000 (0.000)	0.023 (0.007)
ecoli	0.165 (0.018)	0.195 (0.025)	0.194 (0.031)	0.211 (0.015)
yeast	0.459 (0.010)	0.523 (0.025)	0.521 (0.029)	0.527 (0.025)
mean	**0.2164**	**0.2377**	**0.2388**	**0.2628**
		EMC		
abalone	2.074 (0.558)	1.797 (0.620)*	**1.775** (0.673)*	2.017 (0.819)
glass	1.750 (0.316)	1.613 (0.335)	**1.597** (0.289)*+	1.820 (0.194)
segmentation	0.815 (0.127)	0.749 (0.173)	**0.698** (0.156)	1.291 (0.237)
iris	0.284 (0.135)	0.292 (0.115)	0.263 (0.100)	**0.188** (0.069)
xaa	2.593 (0.551)	2.529 (0.611)	**2.464** (0.529)	2.598 (0.496)
wine	0.284 (0.105)	0.258 (0.106)	0.256 (0.091)	**0.199** (0.061)
zoo	0.244 (0.095)	**0.221** (0.082)	0.271 (0.114)	0.524 (0.102)
soybean	0.045 (0.141)	**0.000** (0.000)	**0.000** (0.000)	0.102 (0.063)
ecoli	0.795 (0.154)	0.805 (0.155)	**0.787** (0.169)	0.893 (0.248)
yeast	2.449 (0.311)	2.227 (0.331)	**2.200** (0.322)*	2.658 (0.477)
mean	**1.1333**	**1.049**	**1.0312**	**1.2289**

The superiority of wBNCLVQ is apparent compared to the competing algorithms. It achieves the best results on 9 out of 13 data sets (10 are significantly better than BNCLVQ) in terms of Err_{HC}, and on 7 data sets in terms of EMC (7 are significantly better than BNCLVQ). Compared with cl-NCLVQ, wBNCLVQ performs better on almost all binary data sets except hepatitis and credit (Even on the two data sets, the inferiority is not significant). The inferiority is probably due to the inadequate iteration number or inappropriate initial point of prototypes. Nevertheless, it conforms that the weighted prototype learning methodology is effective on improving the performance in the case of non-uniform cost matrix. Although MetaCost-LVQ obtains the best EMC on transfusion, german, and echocardiogram, the average performance is the worst out of three cost-related variants due to the high standard deviation.

Table V compares the results on multi-class data sets. The wBNCLVQ algorithm is still the best one. It achieves the best value of EMC on 7 out of 10 data sets. However, the superiority is not so significant as that on binary-class data. The mean reduction is only 3% in terms of Err_{HC} and 9% in terms of EMC. The relative poor performance is probably due to the information loss during the conversion from cost matrix to class weights. This result is consistent with what was reported in [19].

The robustness property of these LVQ variants is shown in Fig. 1 and Fig. 2. The distribution of relative performance on each data set is plotted in stack. Regarding binary class data sets, wBNCLVQ has the best robustness on both Err_{HC} and EMC. Regarding multi-class data, wBNCLVQ have comparable robustness as cl-BNCLVQ.

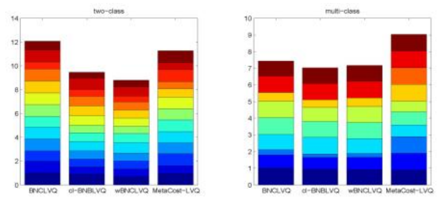

Fig. 1 Robustness comparison on Err_{HC}.

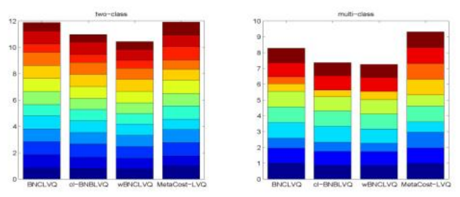

Fig. 2 Robustness comparison on EMC.

V. Conclusion

In this paper, we implement a weighted BNCLVQ algorithm embedding instance weight to induce a cost-sensitive LVQ classifier. With well-defined weight assignment, the cost matrix can be incorporated in the training and labeling of the LVQ model. The main contribution of the proposed research is to extend the algorithms of LVQ family to classify data with hybrid types when the costs of errors are different. Results on a number of real-world data confirm the presented weighted LVQ algorithm is effective on both binary-class and multi-class data. The weighted methodology is easily applied to other LVQ variants, such as GLVQ and GRLVQ, which is one of the research topic in future work. As disclosed in the experiments, performance on multi-class data sets is inferior to that on binary class data sets. Future work will be focused on better incorporating the cost matrix in the representation of instance weight. In the present work, only the constant misclassification cost is considered, however, other more complicated types of cost is of interest to classification. Besides, the comparative study with other cost-sensitive learning methods is needed to validate the effectiveness of the proposed approaches.

References

[1] H. Pao, S. Chang, and Y. Lee, "Model trees for classification of hybrid data types," Intelligent Data Engineering and Automated Learning (IDEAL), LNCS, 2005, pp. 32–39.

[2] N. Chen, and N. C. Marques, "Extending learning vector quantization for classifying data with categorical values," In J. Filipe, A. Fred, and B. Sharp Eds. Communications in Computer and Information Science (CCIS), 67, Springer, 2010, pp. 124–136.

[3] T. Kohonen, *Self-Organizing Maps*. Springer Verlag, Third edition, 2001.

[4] A. Sato and K. Yamada, "Generalized learning vector quantization," In G. Tesauro, D. Touretzky, T. Leen Eds. Advances in Neural Information Processing Systems, 7, MIT Press, 1995, pp. 423–429.

[5] T. Bojer, and B. Hammer, "Relevance determination in learning vector quantization," In European Symposium on Artificial Neural Networks, 2001, pp. 271–276.

[6] B. Hammer, and T. Villmann, "Generalized relevance learning vector quantization, " Neural Networks, 15, 2002, pp. 1059–1068.

[7] A. Qin, and P. Suganthan, "Initialization insensitive LVQ algorithm based on cost-function adaptation, " Pattern Recognition, 38(5), 2005, pp. 773–776.

[8] C. E. Pedreira, "Learning vector quantization with training data selection," IEEE Transaction on Pattern Analysis and Machine Intelligence, 28(1), 2006, pp. 157–162.

[9] J. C. Neves, and A. Vieira, "Improving bankruptcy prediction with hidden layer learning vector quantization," European Accounting Review, 15(2), 2006, pp. 253–271.

[10] M. Strickert, U. Seiffert, N. Sreenivasulu, W. Weschke, T. Villmann, and B. Hammer, "Generalized relevance LVQ (GRLVQ) with correlation measures for gene expression analysis," Neurocomputing, 69(7-9), 2006, pp. 651–659.

[11] P. Hajek, and V. Olej, "Municipal creditworthiness modelling by Kohonen's self-organizing feature maps and LVQ neural networks," In L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, J. M. Zurada Eds. Artificial Intelligence and Soft Computing (ICAISC), LNCS, 5097, 2008, pp. 52–61.

[12] E. Cuevas, D. Zaldivar, and M. Perez, "LVQ neural networks applied to face segmentation," Intelligent Automation and Soft Computing, 15(3), 2009, pp. 439–450.

[13] M. A. Boyacioglu, Y. Kara, and O. K. Baykan, "Predicting bank financial failures using neural networks, support vector machines and multivariate statistical methods: a comparative analysis in the sample of savings deposit insurance fund (SDIF) transferred banks in Turkey," Expert Systems with Applications, 36, 2009, pp. 3355–3366.

[14] P. D. Turney, "Types of cost in inductive concept leaning," Workshop on Cost-Sensitive Learning at 7th International Conference on Machine Learning, California, 2000.

[15] T. Eitrich, A. Kless, C. Druska, W. Meyer, and J. Grotendorst, "Classification of highly unbalanced CYP450 data of drugs using cost sensitive machine learning techniques," J. Chem. Inf. Model. 47, 2007, pp. 92–103.

[16] Z. H. Zhou, and 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, pp. 63–77.

[17] Y. M. Sun, M. S. Kamela, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," Pattern Recognition, 40, 2007, pp. 3358–3378.

[18] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J Stone, *Classification and regression trees*, Belmont, CA: Wadsworth, 1984.

[19] K. M. Ting, "An instance-weighting method to induce cost-sensitive trees," IEEE Transactions on Knowledge and Data Engineering, 14(3), 2002, pp. 659–665.

[20] S. Nanda, and P. Pendharkar, "Linear models for minimizing misclassification costs in bankruptcy prediction," Int J Intell Syst Account Finance Manage, 10, 2001, pp. 155–168.

[21] N. H. Vo, and Y. Won, "Classification of unbalanced medical data with weighted regularized least squares," Frontiers in the Convergence of Bioscience and Information Technologies, 2007, pp. 347–352.

[22] N. Chen, A. Vieira, J. Duarte, B. Ribeiro, and J. C. Neves, "Weighted learning vector quantization to cost-sensitive learning," In K. Diamantaras, W. Duch, and L. S. Iliadis Eds. ICANN, Part III, LNCS, 6354, 2010, pp. 277–281.

[23] T. Lee, and I. Chen, "A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines," Expert Systems with Applications, 28(4), 2005, pp. 743–752.

[24] P. Pendharkar, and S. Nanda, "A misclassification cost-minimizing evolutionary–-neural classification approach," Naval Research Logistics, 53(5), 2006, pp. 432–447.

[25] B. Zadrozny, and C. Elkan, "Learning and making decisions when costs and probabilities are both unknown," In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 2001, pp. 204–213.

[26] J. Zheng, "Cost-sensitive boosting neural networks for software defect prediction," Expert Systems with Applications, 37(6), 2010, pp. 4537–4543.

[27] N. Chen, A. Vieira, and J. Duarte, "Cost-sensitive LVQ for bankruptcy prediction: an empirical study," In W. Li, J. Zhou Eds. 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, 2009, pp. 115–119.

[28] P. C. Pendharkar, "A threshold varying bisection method for cost sensitive learning in neural networks," Expert Systems with Applications 34, 2008, pp. 1456–1464.

[29] P. Domingos, "MetaCost: a general method for making classifiers cost-sensitive," In Proceedings of 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, United States, 1999, pp. 155–164.

[30] J. S. Hamaker, and L. Boggess, "Non-euclidean distance measures in AIRS, an artificial immune classification system," In Proceedings of the 2004 Congress of Evolutionary Computation, 2004, pp. 1067–1073.

[31] A. Asuncion, and D. J. Newman. UCI Machine Learning Repository. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[32] WEKA Classification Algorithms. Available: http://wekaclassalgos.sourceforge.net