Efficient pairwise classification using Local Cross Off strategy

Mohammad ali Bagheri[1], Qigang Gao[1], and Sergio Escalera[2]

[1] Faculty of Computer Science, Dalhousie University, Halifax, Canada
[2] Computer Vision Center, Campus UAB, Edifici O, 08193, Bellaterra, Spain
Dept. Matemtica Aplicada i Anlisi, Universitat de Barcelona, Gran Via de les Corts Catalanes 585, 08007, Barcelona, Spain.

Abstract. The pairwise classification approach tends to perform better than other well-known approaches when dealing with multiclass classification problems. In the pairwise approach, however, the nuisance votes of many irrelevant classifiers may result in a wrong prediction class. To overcome this problem, a novel method, Local Crossing Off (LCO), is presented and evaluated in this paper. The proposed LCO system takes advantage of nearest neighbor classification algorithm because of its simplicity and speed, as well as the strength of other two powerful binary classifiers to discriminate between two classes. This paper provides a set of experimental results on 20 datasets using two base learners: Neural Networks and Support Vector Machines. The results show that the proposed technique not only achieves better classification accuracy, but also is computationally more efficient for tackling classification problems which have a relatively large number of target classes.

Keywords: Multiclass, Pairwise classification, Neural Networks, Support Vector Machines

1 Introduction

A common task in many real world pattern recognition applications is to discriminate between instances that belong to multiple classes. In contrast to this, most of the established classification algorithms, such as Support Vector Machine (SVM) [3] and Multi-Layer Perceptron (MLP), work better facing two-class problems. The predominant approach to overcome this problem is to recast the multi-class problem into a series of smaller binary classification tasks, which is referred to as "class binarization" [13]. In this way, two-class problems can be solved by binary classifiers and the results can then be combined so as to provide a solution to the original multiclass problem. Among the proposed methods for approaching class binarization, three techniques are well-known including one-versus-one[14], one-versus-all [6, 2], and Error Correcting Output Codes (ECOC) [8]. In one-versus-all, the multiclass problem is decomposed into several binary

[3] Indeed, the SVM algorithm is specifically designed for problems with only two target classes

problems in the following way: for each class a binary classifier is trained to discriminate among the patterns of the class and the patterns of the remaining classes. In the one-versus-one approach, one classifier is trained for each possible pair of classes. In both approaches, the final classification prediction is obtained by means of a voting or committee procedure. Dietterich and Bakiri [8] presented a general framework for class binarization approaches in order to enhance generalization ability of binary classifiers, which is known as Error Correcting Output Codes (ECOC). The ECOC scheme is split in two main steps: coding and decoding. At the coding step a set of binary classifiers splitting groups of classes are defined and codified in an ECOC coding matrix, where each row represents the code for a particular class. At the decoding step, the outputs of the individual binary classifiers are computed for a test pattern, and the sample is classified by the class with the code at minimum distance given a particular decoding measurement.

Among the three approaches, one-versus-one and one-versus-all are the two most commonly used, mainly because of their clarity in comparison with the ECOC approach. However, both the one-versus-one and one-versus-all methods have their drawbacks. In the one-versus-all method, each binary classifier is trained on an unbalanced training set,f tending to produce a negative output for all classifiers [5]. Some recent studies have also shown that this method generally performs worse than the other methods [13][15].

Concerning the one-versus-one method, one of its main drawbacks is the problem of incompetent classifiers, which seems to be ”inherent to the one-versus-one approach” and may result in an incorrect prediction class [12] [19]. That is, many of the binary classifiers are forced to give nuisance votes for many instances because each classifier must assign every instance to one of the two classes used in its training set [12] [19]. Suppose that a new instance belongs to class c_i. To classify this instance, it is presented to all the pairwise classifiers. Therefore, all the classifiers that are not trained with the data from class c_i will cast wrong votes. Consequently, using irrelevant classifiers to determine the target class is very likely to deteriorate the classification accuracy and confidence. The problem is that the actual class of the instance is obviously unknown a priori, and thus the meaningful classifiers cannot be selected a priori. The experimental results of [12] demonstrate that the percentage of times when an instance is misclassified and the classifiers for the correct class give accurate answers is relatively high.

Some strategies have been proposed in the past in order to improve the simple aggregation of base binary classifiers in the one-versus-one structure, such as weighted voting strategy [16], decision directed acyclic graph (DDAG) [23], QWeighted [22], classification by pairwise coupling [14] , the combination of one-versus-all and one-versus-one [21] [12] [19], binary tree of classifiers [10], and probability estimates by pairwise coupling approach (PE) [26].

In this paper, we present a simple, but efficient, technique to enhance the classification performance of the one-versus-one method. The strategy is to only choose the relevant binary classifiers based on the target classes using the nearest neighbor rule. In this way, we select only a few binary classifiers to classify a new test pattern. Experimental results show that the proposed approach receives better performance among all considered methods and is computationally more efficient than one-versus-one strategy.

The rest of this paper is organized as follows: Section 2 briefly reviews the three main class binarization methods. The proposed method for binary classifier selection is explained in detail in Section 3. Section 4 reports and analyses the experimental results. Finally, Section 5 concludes the paper.

2 Related work

The following briefly describes some notations used in this paper:

- $T = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_m}, y_m)\}$. A training set; where $\mathbf{x_i} \in R^n$; and each label,y_i, is an integer belongs to $Y = \{\omega_1, \omega_2, \ldots, \omega_c\}$, where c is the number of classes
- $h = \{h_1, h_2, \ldots, h_L\}$: A set of L binary classifiers.

The goal of class binarization methods it to get a feature vector,$\mathbf{x}$, as its input, and to assign it to a class label from Y. As we mentioned before, the methods for multiclass problems can be generally categorized into three approaches:
One-versus-all(OVA): The one-versus-all method constructs c binary classifiers, one for each class. The ith classifier, h_i, is trained with data from class i as positive instances and all data from the other classes as negative instances. A new instance is classified in the class whose corresponding classifier output has the largest value.
One-versus-one (OVO): The one-vs-one method, also called pairwise classification, constructs $c(c-1)/2$ classifiers [18]. Classifier ij, h_{ij}, is trained using all data from class i as positive instances and all data from class j as negative instances, and disregarding the remaining data. To classify a new instance, $\mathbf{x}$, each of the base classifiers cast a vote for one of the two classes used in its training. Then, the one-vs.-one method applies the majority voting scheme for labeling $\mathbf{x}$ to the class with the most votes. Ties are usually broken arbitrarily for the larger class. More complicated combination methods have also been proposed [22] [26].

Ko and Byun proposed a method based on combination of the one-versus-all method and a modification of the one-versus-one method using SVM as a base learner [19]. This method first obtain the top two classes whose corresponding classifiers have the highest confident based on the outputs of all one-versus-all classifiers. In a recent paper [12], very similar idea is presented and named A&O. However, in both these methods, the learning algorithm which finds the two most likely classes is the same as the final classification algorithm. Consequently, it is very likely that some classification errors will be common, arising from the limitation of base learner on certain patterns. Furthermore, it has been shown theoretically that the one-versus-all approach is more complex than the pairwise approach [11] because even though it has a linear number of binary classifiers,

the individual problems that are needed to train on are significantly larger. In addition, the problems are usually more difficult to learn in one-versus-all, as the classes have more overlapping instances. Thus, the one-versus-all approach and the overall system use to be computationally more expensive, especially for classifiers such as neural networks and SVMs.
Error Correcting Output Codes (ECOC):The basis of the ECOC framework consists of designing a codeword for each of the classes. This method uses a matrix M of $\{1, -1\}$ values of size $c \times L$, where L is the number of codewords codifying each class. This matrix is interpreted as a set of L binary learning problems, one for each column. That is, each column corresponds to a binary classifier, called *dichotomizer* h_j, which separates the set of classes into two meta-classes. Instance $\mathbf{x}$, belonging to class i, is a positive instance for the jth classifier if and only if $M_{ij} = 1$ and is a negative instance if and only if $M_{ij} = -1$. When testing an unlabeled pattern, $\mathbf{x}$, each classifier outputs a "0" or "1", creating a L long output code vector. This output vector is compared to each codeword in the matrix, and the class whose codeword has the closest distance to the output vector is chosen as the predicted class. The process of merging the outputs of individual binary classifiers is usually called decoding. The most commonly decoding methods are the Hamming distance. This method looks for the minimum distance between the prediction vector and codewords. The ECOC method was then extended by Allwein et al. [1] using a coding matrix with three values, $1, 0, -1$, where the zero value means that a given class is not considered in the training phase of a particular classifier. In this way, a class can be omitted in the training of a particular binary classifier. This extended codeword is denominated sparse random code and the standard codes (binary ECOC) were named dense random codes.

Hsu and Lin [15] compared different approaches for multiclass SVM problems, including one-versus-one, one-versus-all, and DDAG. Using ten benchmark datasets, the authors claimed that the one-versus-one method is superior to the other approaches. Pedrajas and Boyer's prominent paper [13] later presented an in-depth critical assessment of the three basic multiclass methods. One of the main paper's conclusions states that ECOC and one-versus-one are the best choices for powerful learners and for simpler learners, respectively. However, in the ECOC approach, since the ensemble system usually has more individual classifiers than one-versus-one and one-versus-all, it usually requires more computation, especially for the training phase. Regarding the one-versus-one method, as mentioned before, many of the binary base classifiers are forced to give wrong votes for a given test pattern. Consequently, using irrelevant classifiers to determine the target class is very likely to deteriorate the classification accuracy. Given this problem, some authors tried to include extra information to the binary classifiers of the one-versus-one scheme in the ECOC framework without retraining the binary problems [9]. The main idea of [9] is to include information about classes not included in each pairwise partition by using the confusion in the training data. However, performance improvement is not always guarantee, and the computationally complexity of the method is slightly increased. Some works have also used the idea of irrelevant classifiers to prune classifiers in ensemble schemes based on the low confidence output of binary classifiers [25].

Taking these facts into account, this paper proposes an effective strategy to the class binarization problem. The method is based on an idea that omitting the irrelevant classifiers and only using the votes of meaningful classifiers will outperform the one-versus-one method. The proposed classification technique is named "Local Crossing off (LCO)", as it excludes some classes and focuses on the most probable classes in the neighborhood space.

3 The proposed LCO method

In this section, the new technique for multiclass classification problems of c classes,$c > 2$, is proposed. The LCO technique works as follows:

In the training phase, for each pair of classes, an individual classifier is trained using the training data of the corresponding two classes (as in OVO). Therefore, we have built classifiers that produce a better class separation for a specific pair of classes in comparison with the one-versus-all approach.

In the test phase, the main aim is to avoid the wrong votes of irrelevant binary classifiers. To do this, the concept of local neighborhood is applied. That is, for any test pattern, the LCO method simply finds its nearest K neighbors in the training set, figures out which classes are the most frequent in those neighbors, and then uses these classes as a guide to choose the related classifiers for classifying a given pattern. In this step, two versions of the LCO method have been proposed:

LCO-Version 1: In the first version, the two most frequent classes of the nearest K neighbors in the training set of each test pattern are found. These two classes are considered as the two most probable classes for each test pattern. Given a test pattern,$\mathbf{x}$, supposing that the two most probable classes are c_i and c_j, $i, j : 1, \ldots, c$, $i \neq j$, classifier $h_{i,j}$ is nominated to predict the final target class of $\mathbf{x}$. In this scheme, only one binary classifier is selected to classify each test pattern.

LCO-Version 2: In this phase, all target classes of the nearest K neighbors in the training set of each test pattern are found. All these classes are considered as the probable classes for the given test pattern. Classifiers that correspond to all pairwise combinations of these classes are then nominated to predict the target class. Each nominated binary classifier casts a vote for one of the two classes used in its training and the majority voting strategy is applied for labeling $\mathbf{x}$ to the class with the most votes. More formally, if the set of classes of the nearest k neighbors of $\mathbf{x}$ is $P = \{\omega_1, \omega_2, \ldots, \omega_p\}$, $c_t \in \{1, \ldots, c\}$, the nominated classifiers would be $\{h_{ij} | i, j \in P, i \neq j\}$. If $P = \{\omega_t\}, \omega_t \in \{1, \ldots, c\}$ the predicted class would be ω_t. As an example, consider a 5-class classification problem ($c = 5$) and the nearest k neighbors of a given test pattern, $\mathbf{x}$, are from ω_2, ω_3, and ω_5. So, h_{23} ,h_{25}, and h_{35} are nominated to label the $\mathbf{x}$.

3.1 Modified K-Nearest Neighbor method:

The conventional K-nearest neighbor (KNN) is among the simplest classification techniques. In addition to its simplicity, it can generate a good, highly nonlinear classification boundary. KNN classifies each unlabeled pattern into the most frequent class among k nearest training patterns based on a distance measure. This classification method can be considered as a local estimation of the posterior probabilities of classes based on the relative frequency of the class labels in a neighborhood (defined by the k-nearest training samples). Let $Y = \{\omega_1, \omega 2, \ldots, \omega_c\}$ be a set of c class labels. Given an unlabelled test pattern, suppose that $\{k_1, k_2, , k_c\}$ denotes the numbers of nearest neighbors for the c classes. The estimate of the posterior probabilities is obtained as:

$$P(\omega_i|\mathbf{x}) \simeq \frac{k_i}{K} \tag{1}$$

In the classic KNN algorithm, $\mathbf{x}$ is classified in the class m if its posterior probability is the largest, i.e.

$$\omega_m = \arg\max_{\omega_i} P(\omega_i|\mathbf{x}) \tag{2}$$

In the case of LCO-ver1, the two most probable classes are found by choosing the two largest $P(\omega_i)$, which are equivalent to the two most frequent classes among the neighbor instances. For the second version of LCO, all $P(\omega_i) > 0$, $i \in \{1...c\}$ are chosen as probable classes.

4 Experimental comparison

4.1 Experimental settings

In order to present the results, first, we discuss the experimental settings of the experiments. In order to investigate the relative performance of our proposed method, an empirical study was conducted. We compared our proposed method with OVO, OVA, A&O, and ECOC methods on 20 multiclass datasets from the UCI machine learning repository [3], which are summarized in Table 1. We considered random codes of $10log_2(c)$ and $15log_2(c)$ bits for dense and sparse ECOC, respectively [1].The class of an instance in the ECOC schemes is chosen using the Hamming distance.

As mentioned before, a modified k-nearest neighbor algorithm was chosen to determine the most probable classes. Based on a preliminary set of experiments, the value of k was set to $K = 5$. The next decision was which base classifier to use. In this study, two base learners were chosen: Support Vector Machines (SVMs) using linear kernel and a Multilayer Perceptron (MLP). The SVM, MLP, and KNN classifiers cannot handle the missing values, so the instances with missing values were removed. For the MLP neural network, we chose 10 hidden nodes and the hyperbolic tangent transfer function and for linear SVM, we set $C = 10$. The experiments were all implemented in MATLAB software. For SVM

Table 1. Summary of the used datasets

	Dataset	# instances	# features	# classes
1	Abalone	4177	8	3
2	Balance	625	4	3
3	Car	1728	6	4
4	Cmc	1473	9	3
5	Derm	366	34	6
6	Ecoli	336	7	8
7	Glass	214	10	7
8	Iris	150	4	3
9	Isolet	7797	34	26
10	Lymph	148	18	4
11	Optdigits	5620	64	10
12	Page	5473	10	5
13	Pendigits	10992	16	10
14	Sat	6435	36	6
15	Vehicle	846	18	3
16	Vowel	990	10	11
17	Waveforms	5000	40	3
18	Wine	178	13	3
19	Yeast	1484	8	10
20	Zoo	101	16	7

implementation, we used the LIBSVM package (version 3.1) developed by Chang and Lin [4]. For performance evaluation, we utilized 10-fold cross-validation to improve the reliability of the results. In order to have a fair comparison, the training and test sets of all methods were the same for each repetition of the experiments.

4.2 Experimental results

The average accuracy of the six methods as well as the original KNN classifier for the 20 datasets is presented in Table 2 and Table 3. In theses tables, the means of prediction accuracy over 10 runs (expressed in %) are reported for each classification method on the considered datasets, where the values following $\pm$ are their respective standard deviations. Comparing the two versions of LCO, we can see that both achieve a similar performance, whereas LCO.ver1 is slightly inferior to LCO.ver2 on the current datasets. Due to the advantage of the second version over the first version, we usually compare other rival methods with LCO.ver2 in the following analysis.

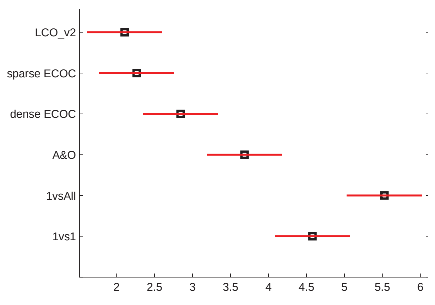
In order to see whether the proposed method is significantly better or worse than other methods, statistical analysis is necessary. According to the recommendations of Demsar [7], we consider the use of non-parametric tests. Non-parametric tests are safer than parametric tests, such as ANOVA and t-test, since they do not assume normal distribution or homogeneity of variance. In this study, we employ the Iman-Davenport test. If there are statistically significant differences in the classification performance, then we can proceed with the Nemenyi test as a post hoc test, which is used to compare six methods with each other.

To do that, we first rank competing methods for each dataset. The best performing method getting the rank of 1, the second best ranked 2, so on and so forth. A method's mean rank is obtained by averaging its ranks across all datasets. Then, we use the Friedman test [7] to compare these mean ranks to decide whether to reject the null hypothesis, which states that all considered methods have equivalent performance. Iman and Davenport [17] found that this statistic is undesirably conservative, and proposed a corrected measure. Applying this method, we can reject the null hypothesis, that is, there exists significant statistical difference among the rival methods.

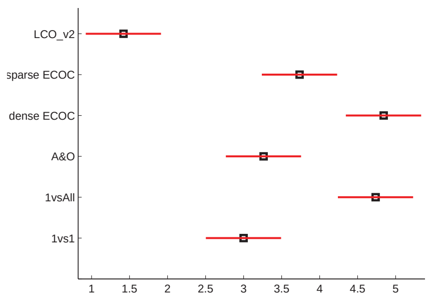
Further, to compare rival methods with each other, we apply the Nemenyi test, as illustrated in Fig. 1. In this figure, the mean rank of each method is indicated by a square. The horizontal bar across each square shows the critical difference. Two methods are significantly different if their corresponding average ranks differ by at least the critical difference value. That is, their horizontal bars are not overlapping.

In addition, to compare each pair of methods across multiple datasets, we show the win/lose/tie comparison record, as reported in Table 5 and Table 6 using MLP and SVM, respectively. Each record represents the number of datasets in which a method in the cloumn, respectively, wins over, loses to, or ties with the method of the corresponding row. To do that, we performed the non-parametric Wilcoxon signed rank test at 95% confidence level.

The results in Tables 2 and Table 3, along with the statistical tests presented in Table 4 and Table 5 and Fig. 1 indicate that overall, LCO.v2 receives the best performance among all six methods. We analyze these results in the next section using the commented statistical analyses.



(a) MLP



(b) SVM

Fig. 1. Comparison results of rival methods using the Nemenyi test (a) and (b)

Table 2. Classification accuracies of different methods with MLP Neural Network

	OVO	**OVA1**	**AO**	**dense ECOC**	**sparse ECOC**	**LCO1**	**LCO2**	**KNN**
Abalone	67.87 ± 1.47	63.76 ± 1.30	67.73 ± 1.39	66.94 ± 1.12	67.54 ± 1.08	67.27 ± 1.27	67.97 ± 1.27	62.03
Balance	91.59 ± 0.87	89.37 ± 0.97	92.86 ± 0.92	92.38 ± 0.47	91.43 ± 1.06	89.84 ± 0.50	91.90 ± 0.83	83.38
Car	96.42 ± 0.62	94.05 ± 0.25	96.53 ± 0.44	95.26 ± 0.24	96.88 ± 0.23	97.11 ± 0.52	97.11 ± 0.52	93.60
Cmc	49.59 ± 1.35	50.47 ± 0.64	49.80 ± 1.00	52.16 ± 0.74	52.30 ± 0.57	49.59 ± 0.81	50.68 ± 0.81	46.83
Derm	90.00 ± 1.02	85.83 ± 0.94	91.39 ± 0.98	96.11 ± 0.40	97.22 ± 0.29	93.33 ± 0.48	94.17 ± 0.48	94.36
Ecoli	85.00 ± 1.33	84.41 ± 0.74	85.59 ± 1.04	88.82 ± 0.99	89.41 ± 0.80	87.94 ± 1.12	87.65 ± 1.04	86.28
Glass	56.82 ± 2.82	47.73 ± 2.45	62.27 ± 2.64	65.00 ± 2.56	63.64 ± 1.44	62.73 ± 2.54	66.82 ± 2.16	61.39
Iris	94.00 ± 1.29	94.67 ± 1.44	94.00 ± 1.37	95.33 ± 0.91	94.67 ± 0.64	94.00 ± 1.30	94.00 ± 0.97	93.05
Isolet	96.18 ± 0.24	96.50 ± 0.18	96.35 ± 0.21	97.13 ± 0.29	97.62 ± 0.15	95.88 ± 0.26	95.11 ± 0.25	83.52
Lymph	72.67 ± 2.88	63.33 ± 1.62	74.00 ± 2.25	80.67 ± 1.55	77.33 ± 1.51	76.67 ± 2.12	77.33 ± 2.09	77.95
Optdigits	96.98 ± 0.49	92.28 ± 0.65	96.74 ± 0.57	98.01 ± 0.43	97.63 ± 0.39	98.75 ± 0.37	99.02 ± 0.31	98.27
Page	96.72 ± 0.49	96.48 ± 0.73	96.93 ± 0.61	96.93 ± 0.63	96.90 ± 0.64	96.66 ± 0.45	96.79 ± 0.44	95.54
Pendigits	99.01 ± 0.25	97.06 ± 0.22	98.96 ± 0.24	99.17 ± 0.19	99.28 ± 0.16	99.49 ± 0.17	99.53 ± 0.17	99.18
Sat	88.49 ± 1.19	86.01 ± 1.17	88.73 ± 1.18	89.57 ± 0.86	89.66 ± 0.96	90.96 ± 0.69	91.32 ± 0.72	90.22
Vehicle	83.41 ± 1.05	78.00 ± 1.20	83.88 ± 1.13	81.76 ± 0.67	83.18 ± 0.42	83.53 ± 1.17	83.76 ± 0.97	72.72
Vowel	92.83 ± 1.33	75.09 ± 1.18	92.26 ± 1.26	94.91 ± 0.85	96.04 ± 1.07	97.92 ± 0.88	98.11 ± 0.76	80.22
Waveform	85.90 ± 1.01	85.34 ± 1.27	85.98 ± 1.14	84.20 ± 0.83	86.16 ± 1.02	85.98 ± 1.14	85.98 ± 1.05	78.97
Wine	91.67 ± 1.39	94.44 ± 1.01	93.33 ± 1.20	95.00 ± 0.82	96.67 ± 0.80	95.00 ± 1.31	95.00 ± 0.96	95.32
Yeast	58.86 ± 0.90	42.75 ± 0.62	59.93 ± 0.76	58.39 ± 0.41	60.87 ± 0.57	59.80 ± 0.98	60.13 ± 0.57	57.04
Zoo	86.36 ± 3.40	88.18 ± 2.66	90.00 ± 3.03	96.36 ± 2.32	94.55 ± 1.15	93.64 ± 1.93	93.64 ± 1.50	88.96
Average	**84.02 ± 1.27**	**80.29 ± 1.06**	**84.86** ± 1.17	**86.21 ± 0.86**	**86.45 ± 0.75**	**85.81 ± 1.01**	**86.30 ± 0.90**	**81.94**

Table 3. Classification accuracies of different methods with linear SVM

	OVO	**OVA**	**AO**	**dense ECOC**	**sparse ECOC**	**LCO.v1**	**LCO.v2**	**KNN**
Abalone	65.65 ± 1.58	65.10 ± 1.44	65.50 ± 1.51	65.74 ± 1.20	65.60 ± 1.20	65.62 ± 1.21	66.22 ± 1.39	61.75
Balance	92.38 ± 0.03	89.21 ± 0.24	89.05 ± 0.14	92.38 ± 0.03	92.38 ± 0.03	89.52 ± 0.29	92.22 ± 0.59	87.10
Car	85.43 ± 0.25	80.06 ± 0.31	84.28 ± 0.28	81.68 ± 0.37	83.01 ± 0.84	88.67 ± 0.34	92.95 ± 0.26	93.34
Cmc	52.16 ± 0.45	49.12 ± 0.83	51.96 ± 0.64	47.30 ± 0.26	52.09 ± 0.26	51.28 ± 0.66	53.58 ± 0.84	48.35
Derm	98.61 ± 0.35	98.06 ± 0.24	98.61 ± 0.30	98.33 ± 0.26	98.33 ± 0.39	98.61 ± 0.38	98.61 ± 0.42	96.19
Ecoli	85.88 ± 0.81	86.47 ± 0.57	86.76 ± 0.69	83.53 ± 0.77	86.47 ± 0.63	87.06 ± 0.66	86.76 ± 0.63	85.17
Glass	68.18 ± 1.45	57.73 ± 1.51	64.55 ± 1.48	60.45 ± 2.67	66.82 ± 2.05	72.27 ± 1.16	73.64 ± 1.39	63.83
Iris	97.33 ± 0.34	96.00 ± 0.61	97.33 ± 0.48	97.33 ± 0.64	97.33 ± 0.64	97.33 ± 0.34	97.33 ± 0.45	95.50
Isolet	96.03 ± 0.17	93.40 ± 0.25	95.32 ± 0.21	82.71 ± 0.60	90.26 ± 0.52	91.54 ± 0.30	96.15 ± 0.23	79.21
Lymph	80.67 ± 1.45	82.00 ± 1.68	80.67 ± 1.57	80.67 ± 1.96	80.67 ± 1.96	82.00 ± 1.05	82.00 ± 1.07	78.76
Optdigits	97.98 ± 0.54	92.27 ± 0.55	94.18 ± 0.55	90.35 ± 0.76	95.05 ± 0.70	97.99 ± 0.55	98.69 ± 0.23	97.57
Page	95.89 ± 0.52	95.24 ± 0.51	95.58 ± 0.52	94.95 ± 0.51	94.74 ± 0.51	95.60 ± 0.47	96.28 ± 0.56	95.54
Pendigits	98.11 ± 0.17	92.88 ± 0.39	96.45 ± 0.28	84.89 ± 0.50	88.28 ± 0.52	99.14 ± 0.15	99.56 ± 0.13	99.06
Sat	86.75 ± 0.66	83.12 ± 0.72	85.98 ± 0.69	78.42 ± 0.80	83.29 ± 0.79	90.53 ± 0.46	90.87 ± 0.54	89.81
Vehicle	78.24 ± 0.60	76.12 ± 0.51	78.59 ± 0.56	78.12 ± 0.77	78.00 ± 0.77	78.47 ± 0.52	78.59 ± 0.68	71.75
Vowel	82.08 ± 0.70	43.77 ± 1.22	54.15 ± 0.96	35.28 ± 1.01	52.26 ± 1.35	91.89 ± 0.39	95.85 ± 0.42	80.49
Waveform	86.92 ± 1.00	86.88 ± 1.17	86.92 ± 1.09	81.16 ± 1.16	87.14 ± 0.95	87.00 ± 0.96	86.96 ± 1.03	79.70
Wine	93.89 ± 0.38	96.11 ± 0.23	93.89 ± 0.31	95.56 ± 0.97	93.89 ± 0.81	94.44 ± 0.38	95.00 ± 0.15	94.52
Yeast	58.39 ± 0.24	56.64 ± 0.54	58.12 ± 0.39	54.03 ± 0.52	57.65 ± 0.35	59.80 ± 0.48	60.27 ± 0.32	56.33
Zoo	95.45 ± 0.74	93.64 ± 1.55	95.45 ± 1.15	94.55 ± 1.75	94.55 ± 1.72	95.45 ± 0.74	95.45 ± 0.74	88.83
Average	**84.80 ± 0.62**	**80.69 ± 0.75**	**82.67** ± 0.69	**78.87 ± 0.88**	**81.89 ± 0.85**	**85.71 ± 0.57**	**86.85 ± 0.60**	**82.14**

Table 4. Rival methods' win/lose/tie records using MLP Neural Network

	OVO	OVA	A&O	dense ECOC	sparse ECOC	LCO.v2
OVO		0 /12 /8	2 /0 /18	6/2/12	9 /0 /11	9 /0 /11
OVA			14 /0 /6	14 /0 /6	15 /0 /5	14 /0 /6
A&O				5/2/13	7 /0 /13	6 /0 /14
dense ECOC					3 /0 /17	6 /0 /14
Sparse ECOC						4 /0 /16

Table 5. Rival methods' win/lose/tie records using Linear SVM

	OVO	OVA	A&O	dense ECOC	sparse ECOC	LCO.v2
OVO		0 /10 /10	0 /8 /12	0 /9 /11	1/6/13	6 /0 /14
OVA			9 /0 /11	3/7/10	4/3/13	12 /0 /8
A&O				1/9/10	2/5/13	11 /0 /9
dense ECOC					9 /0 /11	10 /0 /10
Sparse ECOC						9/1/10

4.3 Experimental result analysis

As can be seen in Tables 2-5, the proposed approach is generally able to outperform all the other methods for the two types of base learners. As a general conclusion, the advanced performance of the LCO method does not differ much depending on the base classifier. Compared to the one-versus-one method, the average accuracy improvement is 2.28% and 2.06% for MLP and linear SVM, respectively. The other finding is that the one-versus-all method generally performs poorly in the present experiments, especially for MLP neural network.

An analysis of the results shows a somewhat clearer picture. Using SVMs as the base learners, LCO.ver2 can indeed outperform the other methods. In addition, the win/lose/tie results show that LCO.v2 also outperforms the rival methods more often than not. The Nemenyi test in Fig. 1 also demonstrates that the rank of LCO.v2 is much better than that of the other method using SVM learner.

However, the case for MLP is somewhat different. Using MLP neural network as the base learner, LCO.v2 works significantly better than one-versus-one, one-versus-all, and A&O methods. Comparing the results of the LCO and ECOC methods, however, we can see that the classification accuracy of the LCO method tends to be slightly better than that of both ECOC methods, but does not significantly outperform ECOC methods. These results are consistent with the observations from Pedrajas and Boyer's paper [12]. Their results show that the performance of the ECOC method using a neural network as a base learner will significantly increase as the number of classifiers increases. The reason behind this improvement is that the effectiveness of the ECOC approach strongly depends on the independency of the binary classifiers, a term which is known as classifier diversity in ensemble classification literature . This literature proves that if each classifier makes different errors, then the total errors can be reduced by an appropriate combination of these classifiers. Due to the instability of neural networks, they are suitable candidates to be used in the ECOC approach. However, the ECOC approach is more computationally expensive than the LCO method. On the other hand, stable classifiers such as support vector machines cannot take advantage of the ECOC approach. Consequently, the results of the ECOC method are even worse than that of one-versus-one and LCO. This is consistent with the extensive comparison results of [24] [20], which show that a single SVM classifier performs better than SVM ensemble methods in many cases.

5 Conclusions

In this paper, we presented a novel strategy for pairwise classification approach to deal with multiclass classification problems. The proposed technique is based on omitting the votes of irrelevant binary classifiers, in order to improve final classification accuracy. For this task, the proposed LCO method finds its nearest K neighbors in the training set, figures out which classes are the most frequent in those neighbors, and then uses these classes as a guide to choose the related classifiers for classifying a given pattern. The experimental evaluation over several UCI Machine Learning repository datasets shows that performance improvements can be obtained compared to the one-versus-one, one-versus-all, A&O, and ECOC methods. The main reason behind this improvement is that the LCO approach can benefit from efficient nearest neighbor rule as a preprocessing step in pairwise structure and the strength of powerful binary classifiers (Neural Networks and Support Vector Machines in our present experiments).

Acknowledgment

The authors would like to thank Dr. Ehsan Nikbakhsh for his kind help with conducting experiments.

References

1. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: a unifying approach for margin classifiers. Journal of Machine Learning Research 1, 113–141 (2001)
2. Anand, R., Mehrotra, K., Mohan, C.K., Ranka, S.: Efficient classification for multiclass problems using modular neural networks. Neural Networks, IEEE Transactions on 6(1), 117–124 (1995)
3. Blake, C., Merz, C.: Uci repository of machine learning databases, department of information and computer sciences, university of california, irvine (1998)
4. Chang, C.C., Lin, C.J.: Libsvm: A library for support vector machines (2001)
5. Chang, C.C., Chien, L.J., Lee, Y.J.: A novel framework for multi-class classification via ternary smooth support vector machine. Pattern Recognition 44(6), 1235–1244 (2011)
6. Clark, P., Boswell, R.: Rule induction with cn2: Some recent improvements. In: Kodratoff, Y. (ed.) Machine Learning EWSL-91, Lecture Notes in Computer Science, vol. 482, pp. 151–163. Springer Berlin / Heidelberg (1991)
7. Demsar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7, 1–30 (2006)
8. Dietterich, T., Bakiri, G.: Solving multiclass learning problems via error-correcting output codes. Journal of Artificial Intelligence Research 2, 263286 (1995)
9. Escalera, S., Pujol, O., Radeva, P.: Re-coding ecocs without re-training. Pattern Recognition Letters 31, 555562 (2010)
10. Fei, B., Liu, J.: Binary tree of svm: A new fast multiclass training and cassification algorithm. IEEE Transactions on Neural Networks 17(696-704) (2006)

11. Frnkranz, J.: Round robin classification. Journal of Machine Learning Research 2, 721747 (2002)
12. Garcia-Pedrajas, N., Ortiz-Boyer, D.: Improving multiclass pattern recognition by the combination of two strategies. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(6), 1001–1006 (2006)
13. Garcia-Pedrajas, N., Ortiz-Boyer, D.: An empirical study of binary classifier fusion methods for multiclass classification. Information Fusion 12(2), 111–130 (2011)
14. Hastie, T., Tibshirani, R.: Classification by pairwise coupling. In: Proceedings of the 1997 conference on Advances in neural information processing systems 10. pp. 507–513. NIPS '97, MIT Press, Cambridge, MA, USA (1998)
15. Hsu, C.W., Lin, C.J.: A comparison of methods for multiclass support vector machines. Neural Networks, IEEE Transactions on 13(2), 415–425 (2002)
16. Hullermeier, E., S.Vanderlooy: Combining predictions in pairwise classication: An optimal adaptive voting strategy and its relation to weighted voting. Pattern Recognition 43(1), 128–142 (2010)
17. Iman, R., Davenport, J.: Approximations of the critical regions of the friedman statistic. Communications in Statistics 6, 571–595 (1980)
18. Knerr, S., Personnaz, L., Dreyfus, G.: Single-layer learning revisited: a stepwise procedure for building and training a neural network. In: Fogelman, J. (ed.) Neurocomputing: Algorithms, Architectures and Applications. Springer-Verlag (1990)
19. Ko, J., Byun, H.: Binary classifier fusion based on the basic decomposition methods. In: Windeatt, T., Roli, F. (eds.) Multiple Classifier Systems, Lecture Notes in Computer Science, vol. 2709, pp. 159–159. Springer Berlin / Heidelberg (2003)
20. Meyer, D., Leisch, F., Hornik, K.: The support vector machine under test. Neurocomputing 55(1-2), 169–186 (2003)
21. Moreira, M., Mayoraz, E.: Improved pairwise coupling classification with correcting classifiers. In: 10th European Conference on Machine Learning. vol. 1398, pp. 160–171. Lecture Notes in Computer Science, Springer, Chemnitz, Germany (1998)
22. Park, S.H., Fürnkranz, J.: Efficient pairwise classification. In: Kok, J.N., Koronacki, J., López de Mántaras, R., Matwin, S., Mladenić, D., Skowron, A. (eds.) Proceedings of the 18th European Conference on Machine Learning (ECML 2007, Warsaw, Poland). pp. 658–665. Springer-Verlag (2007)
23. Platt, J.C., Cristianini, N., Shawe-taylor, J.: Large margin dags for multiclass classification. In: Advances in Neural Information Processing Systems 12. pp. 547–553 (2000)
24. Wang, S.j., Mathew, A., Chen, Y., Xi, L.f., Ma, L., Lee, J.: Empirical analysis of support vector machine ensemble classifiers. Expert Systems with Applications 36(3, Part 2), 6466–6476 (2009)
25. Windeatt, T., Ghaderi, R.: Binary labelling and decision-level fusion. Information Fusion 2(2), 103–112 (2001)
26. Wu, T.F., Lin, C.J., Weng, R.C.: Probability estimates for multi-class classification by pairwise coupling. J. Mach. Learn. Res. 5, 975–1005 (2004)