Comparison of a novel Combined ECOC Strategy with Different Multiclass Algorithms together with Parameter Optimization Methods

Marco Hülsmann[1,2] and Christoph M. Friedrich[2]

[1] Universität zu Köln, Germany
[2] Fraunhofer-Institute for Algorithms and Scientific Computing (SCAI), Schloß Birlinghoven, 53754 Sankt Augustin, Germany

Abstract. In this paper we consider multiclass learning tasks based on Support Vector Machines (SVMs). In this regard, currently used methods are *One-Against-All* or *One-Against-One*, but there is much need for improvements in the field of multiclass learning. We developed a novel combination algorithm called *Comb-ECOC*, which is based on posterior class probabilities. It assigns, according to the Bayesian rule, the respective instance to the class with the highest posterior probability. A problem with the usage of a multiclass method is the proper choice of parameters. Many users only take the default parameters of the respective learning algorithms (e.g. the regularization parameter C and the kernel parameter γ). We tested different parameter optimization methods on different learning algorithms and confirmed the better performance of *One-Against-One* versus *One-Against-All*, which can be explained by the maximum margin approach of SVMs.

1 Introduction

All multiclass learning methods considered here are based on Support Vector Machines, which are presented for example by Schölkopf and Smola [22] and Vapnik [25]. Mostly, several binary classifications are resolved by an SVM, which are then combined to a multiclass solution. Our goal is to present improved methods in the open-research field of multiclass learning.

In section 2 we start with the presentation of different state-of-the-art multiclass algorithms. We consider the standard methods *One-Against-All (OAA)* and *One-Against-One (OAO)* using implementations of the *libsvm* [11] and *SVMlight* [15]. We continue with two direct approaches, which are not based on several binary optimization problems: the algorithm by Crammer and Singer [4] and *SVMmulticlass* based on the theory of [24]. Furthermore we use the exhaustive ECOC algorithm introduced by Dietterich and Bakiri [5] and present a novel combination approach of ECOC, OAA and probability predictions. This method ist called *Comb-ECOC* and has been developed in [14]. The probability predictions are based on Bradley-Terry models described in [13].

The next principal subject of this paper will be the parameter optimization. In many applications, Support Vector Machines are used with their default

parameters. Optimizing the parameters improves the classification performance drastically, which will be shown in section 3. We consider different optimization methods, such as the common grid search and the *SVMpath* algorithm introduced in [10].

Finally, we give results obtained from different test runs with all considered multiclass algorithms and parameter optimization methods in section 3, together with the practical confirmation of the maximum margin explanation in the case of *One-Against-One* and *One-Against-All*. A final discussion is carried out in section 4.

2 Theoretical Background

In this section we consider the multiclass algorithms from a theoretical point of view. We shortly describe the principal ideas of both the learning methods and the parameter optimization. In order to avoid later misunderstandings, we here already enumerate the used algorithms in Table 1. Detailed experiments and comparisons can also be found in [9] and [12]. In the latter, especially *One-Against-One* is suggested for real-world applications, which will be confirmed by our analysis.

We start with a general description of the multiclass learning task, in order to appoint the notations used in this paper: We consider an input space $\mathcal{X} = \{x_1, ..., x_m\}$ and assign $k > 2$ classes to this set, so that each element in $\mathcal{X}$ belongs to exactly one class. The goal is to find a decision function $f : \mathcal{X} \to \{1, ..., k\}$ to get a pair $(x_i, f(x_i))$ for all $i = 1, ..., m$. The assigned class $f(x_i) = r \in \{1, ..., k\}$ is also called *label*. We furthermore distinguish between the *input space* and the *feature space*. The input space $\mathcal{X}$ can be anything. It is not necessary that it consists of vectors or numerical values. In contrast, the feature space $\mathcal{F}$ is a high dimensional vector space. In this paper, let its dimension be defined as n. A map $\Phi : \mathcal{X} \to \mathcal{F}$ is used to assign an element of the input space to a vector in the feature space. In order to avoid the computation of Φ, we simply use a kernel function $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ for all $x, x' \in \mathcal{X}$. Here we only give results obtained by using Gaussian kernels $k(x, x') = \exp(-\gamma ||x - x'||^2)$, $\gamma \in \mathbb{R}$, because pursuant to [14], it delivers the best results and the corresponding SVM algorithm works faster in comparison to the polynomial and the sigmoid kernel. Instead of predicting classes we also use methods which predict posterior class probabilities.

2.1 Standard Multiclass Algorithms

One-Against-All (OAA) A well-known simple approach for the assignment of instances to several classes is to separate each class from all the other classes. This method is called *One-Versus-the-Rest* or *One-Against-All* and bears on k

Table 1. Description of the different multiclass learning algorithms.

Algorithm	Description	Section
OAA	One-Against-All classification by *libsvm* using binary class probabilities	2.1
OAO	One-Against-One classification by *libsvm*, voting	2.1
SVMpath	SVMpath algorithm described in [10] using the appropriate predict function	2.6
SVMlight-OAA	Interface between *R* and *SVMlight*, One-Against-All classification	2.1
SVMlight-OAO	Interface between *R* and *SVMlight*, One-Against-One classification	2.1
Crammer-Singer	Multiclass Algorithm by Crammer and Singer [4]	2.2
SVMmulticlass	Direct Algorithm by Thorsten Joachims [24]	2.2
ECOC	Standard Exhaustive ECOC Algorithm by Dietterich and Bakiri [5] using the *libsvm* for binary classification	2.3
ECOC-SVMpath	ECOC with SVMpath as binary predictor	2.3, 2.6
Comb-ECOC	Combined ECOC Algorithm, combination with OAA predicting posterior class probabilities	2.4

binary classificators $f^1, ..., f^k$, where k is the number of classes. In order to classify a test point x, one computes the decision function

$$f(x) = \arg \max_{j=1,...,k} \sum_{i=1}^{m} y_i \alpha_i^j k(x, x_i) + b^j, \tag{1}$$

where the index j refers to the binary separation of class j from the rest. The coefficients α_i and b stem from the dual optimization problem, which is set up in the context of binary Support Vector Machine classification, see e.g. [3]. We use this method in two implementations: First, we consider the method named *OAA* (see Table 1), which uses binary class probabilities computed – pursuant to [19] – by:

$$P(j|x) = \frac{1}{1 + \exp(Af^j(x) + B)}, \tag{2}$$

where f^j is the binary classificator that separates class j from the rest. A and B are parameters obtained by the minimization of a negative log-likelihood function. Then the class is computed by:

$$f(x) = \arg \max_{j=1,...,k} P(j|x). \tag{3}$$

The second implementation called *SVMlight-OAA* is based on the interface between *R* [21] and *SVMlight* [15] (website: `http://svmlight.joachims.org`) provided by the *R*-library *klaR* (see [20]). It calls *SVMlight* k times for all k binary classifications and computes the class by equation (1).

One-Against-One (OAO) The idea of this method is to extract all pairs of classes and accomplish a binary classification between the two classes in each pair. Altogether, there are $\binom{k}{2} = \frac{k(k-1)}{2}$ binary classifications. The training set contains only elements of two classes. The other training instances are eliminated from the set. This results in a smaller complexity in comparison to the *One-Against-All* method, but the number of classes is $\mathcal{O}(k^2)$ instead of $\mathcal{O}(k)$. The assignment of a class to a test point occurs by voting. Pursuant to [16], an advantage of the pairwise classification is that in general, the margin is larger than in the OAA case, as we can see from Figure 1. Moreover, the difference of the margin sizes is bigger in the OAO case. Very large margins are possible to appear.

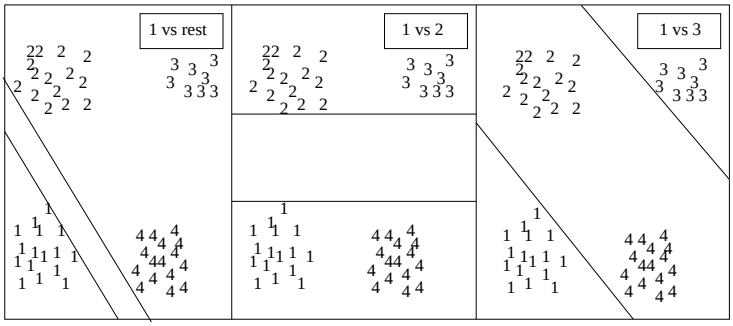

Fig. 1. Comparison of the margin size in the case of OAA and OAO (4 classes): In the left picture class 1 is separated from the rest. The margin is small. In the middle class 1 is separated from class 2 and in the right picture class 1 is separated from class 3. The margin is larger in general. The margin size differs more in the OAO case. The right picture shows a very large margin.

We use OAO from the *libsvm* (see [17]) as a function called *svm* from the *R*-package *e1071* (see [6]).

2.2 Direct Multiclass Algorithms

In our evaluation, we use two direct methods. Direct means that there is only one optimization problem to solve instead of multiple binary ones.

The Algorithm by Crammer and Singer The direct approach consists in formulating one quadratic program with constraints for each class. The algorithm developed by Crammer and Singer [4] is based on the same idea as the OAA-approach, namely to look for a function $H_M : \mathcal{F} \to \{1, ..., k\}$ of the form

$$H_M(x) = \arg\max_{r=1}^{k} \{M_r \cdot x\}, \tag{4}$$

where M is a matrix with k rows and n columns and M_r the rth row of M. The aim is to determine M, so that the training error is minimized.

We take the implementation of this algorithm from the *R*-library *kernlab*. The used function is named *ksvm*.

The Algorithm by Thorsten Joachims This method is based on [24], where the composition of the optimization problem and its solution are described in detail. The first reason for finding a more general direct multiclass algorithm is the fact that in real world examples, one does not only have labels in $\mathbb{N}$. The label set can be arbitrary. For instance, it can consist of structured output trees. The output set $\mathcal{Y}$ does not have to be a vector space. The second reason is allowing different loss functions. Most multiclass algorithms only minimize the zero-one-loss.

We use the software $SVM^{multiclass}$ implemented by Thorsten Joachims (see `http://www.cs.cornell.edu/People/tj/svm_light/svm_multiclass.html`).

2.3 Error Correcting Output Codes

A very simple but efficient method is based on **E**rror **C**orrecting **O**utput **C**odes (the so-called *ECOC* method) and has been developed by Dietterich and Bakiri [5]. ECOC is not a direct application but uses several binary classifications like OAA and OAO. The main difference between OAA, OAO and ECOC is that these are not predisposed. They can be chosen arbitrarily and are determined by a coding matrix. The advantage of the usage of error correcting output codes is that several binary classification errors can be handled, so that no error occurs in the multiclass problem. We use an *exhaustive* code matrix, as proposed in [5]. The *ECOC* algorithm is part of our own implementation.

2.4 Combined ECOC Method

The novel *Comb-ECOC* algorithm (see also [14]) is a mixture of OAA, ECOC and probability prediction. The results of three different partial algorithms are combined by a combination method. We use three code matrices, one defined by the user himself and two generated at random. The random code matrices are composed, so that the minimum Hamming distance is equal to $\lceil \frac{\ell}{2} \rceil$. The number of binary classificators ℓ is a random integer between $2k$ and 2^k, so the algorithm can be NP-complete, like the *Exhaustive-ECOC* algorithm of section 2.3. This disadvantage can be reduced by precalculating the code matrices for a given k. Furthermore we use the binary class probabilities computed by equation (2) for each of the ℓ binary classificators and apply Bradley-Terry methods to estimate the multiclass probabilities following [13]. The optimization problem results in a fixpoint algorithm. The complete procedure of *Comb-ECOC* is described in [14].

All three initial code matrices contain OAA columns, that means columns which define an OAA classification (one entry 1 and the rest 0). By a small modification of the convergence proof in [13], one can show that the fixpoint

algorithm converges with this assumption. A test point is assigned to the class with the highest posterior probability.

The fact that we consider three different code matrices delivers three different multiclass models $m_i,\ i = 1, ..., 3$. These models can be combined in order to be able to compensate the weaknesses of particular models, as described in [8]. We illustrate the idea of combination by the following example:

Example 1 (Combination of multiclass models) *Consider k classes and μ multiclass models. Let $p_i(x_{j,r})$ be the posterior probability with which a test point x_j belongs to a class $r \in \{1, ..., k\}$ predicted by the multiclass model $m_i,\ i \in \{1, ..., \mu\}$. Let furthermore $p_{comb}(x_{j,r})$ be the resulting combination classificator. Then x_j is assigned to a class as follows:*

$$class(x_j) = \arg \max_{r=1,...,k} p_{comb}(x_{j,r}). \tag{5}$$

We consider different combination models for p_{comb}:

1. The maximum: $p_{comb}(x_{j,r}) = \max_{i=1,...,\mu} p_i(x_{j,r})$
2. The minimum: $p_{comb}(x_{j,r}) = \min_{i=1,...,\mu} p_i(x_{j,r})$
3. The average: $p_{comb}(x_{j,r}) = \frac{1}{\mu} \sum_{i=1}^{\mu} p_i(x_{j,r})$
4. The median:

$$p_{comb}(x_{j,r}) = \begin{cases} \frac{p_{\frac{\mu}{2}}(x_{j,r}) + p_{\frac{\mu}{2}+1}(x_{j,r})}{2} & : \ \mu \text{ is even} \\ p_{\frac{\mu+1}{2}}(x_{j,r}) & : \ \mu \text{ is odd} \end{cases} \tag{6}$$

5. The entropy:

$$p_{comb}(x_{j,r}) = H_r(x_j) = \frac{1}{\mu} \sum_{i=1}^{\mu} \frac{p_i(x_{j,r})}{-\sum_{s=1}^{k} p_i(x_{j,s}) \log p_i(x_{j,s})} \tag{7}$$

6. The product:

$$p_{comb}(x_{j,r}) = \prod_{i=1}^{\mu} p_i(x_{j,r}) \tag{8}$$

2.5 Parameter Optimization via Grid Search

In order to show that it is not sufficient to use the default parameters of the respective methods, we performed a parameter optimization via grid search. The parameters we consider are the SVM regularization parameter C and the Gaussian kernel parameter γ. We first defined a training set and a test set of the original dataset. To warrant the comparability of the results we used the same training set and the same test set for all algorithms and optimization methods. For the parameter optimization itself, we determined 10 bootstrap replications on the training set (for details concerning the bootstrap see [7]). Each of them is evaluated with different parameter pairs. The parameter pair with the smallest mean error over the 10 bootstrap replications is taken for predictions on the test set.

The parameters are defined on a grid, that means that we only allow a final number of parameters for the optimization. For computational reasons, we only use $C \in \mathcal{C} = \{2^{-2}, 2^{-1}, ..., 2^3, 2^4\}$ and $\gamma \in \mathcal{G} = \{2^{-3}, 2^{-2}, ..., 2^2, 2^3\}$. In total, have $7 \cdot 7 \cdot 10 = 490$ bootstrap replications in our optimization process. If we have two or more pairs (C, γ) with the same mean error rate over the 10 bootstrap replications, we take the pair with the largest C to facilitate the choice of the maximum margin classification. We differenciate between two methods: *global* and *local*. The *global* method tunes the whole multiclass algorithm globally, that means it takes one pair (C, γ) for the entire algorithm, i.e. the same pair for all contained binary classifications. However, the *local* method tunes all binary classificators, making a grid search for each binary classification. Therefore, in the global case we have a *for*-loop over C and over γ and evaluate the multiclass algorithm with C and γ. The result is one parameter pair (C, γ). In the local case we implemented a *for*-loop over all ℓ binary classificators, which in turn contains the C- and the γ-*for*-loops. Then the binary classification $i \in \{1, ..., \ell\}$ is made with C and γ. The result is a vector $(C_i, \gamma_i)_{i=1,...,\ell}$. The computational complexity of the global optimization is $|\mathcal{C}| \cdot |\mathcal{G}| \cdot \#$(op. in the multiclass algorithm). The complexity of the local optimization is $\ell \cdot |\mathcal{C}| \cdot |\mathcal{G}| \cdot \#$(op. in the binary algorithm). But especially in the case of the *ECOC* algorithms: #(op. in the multiclass algorithm) $\geq$ $\ell \cdot \#$(op. in the binary algorithm). Therefore the local optimization method is faster in general.

2.6 Parameter Optimization via SVMpath

In [10], a different method to optimize the cost parameter C is suggested. They compute the entire regularization path for a binary Support Vector Machine algorithm. The advantage of the calculation of this path is the fact that the complexity is as large as the one of a usual SVM algorithm. Consider the decision function $f(x) = \beta_0 + g(x)$, where

$$\forall_{i=1,...,m}\ g(x_i) = \frac{1}{\lambda} \sum_{j=1}^{m} \alpha_j y_j k(x_i, x_j) \tag{9}$$

with α_j the Lagrangian multipliers of the SVM optimization problem and $\lambda = \frac{1}{C}$. Then the following definitions are made:

- $\mathcal{E} := \{i | y_i f(x_i) = 1, 0 \leq \alpha_i \leq 1\}$ is called *Elbow*
- $\mathcal{L} := \{i | y_i f(x_i) < 1, \alpha_i = 1\}$ is called *Left of the Elbow*
- $\mathcal{R} := \{i | y_i f(x_i) > 1, \alpha_i = 0\}$ is called *Right of the Elbow.*

For the calculation of the optimal λ a Linear Equation System has to be solved. The respective matrix can be singular, which is a problem of the *SVMpath* algorithm.

3 Results

Finally, we show some results from the evaluation of the algorithms indicated in Table 1. We took the *Glass* dataset for the evaluation. It consists of 241 instances, 9 features and 6 classes and stems from the *UCI* repository of machine learning databases [18]. The task of the *Glass* problem is to discriminate between different glass types to support criminological and forensic research. As mentioned in section 2.5, we divided the dataset into a training and a test set. Table 2 shows the classification error rates (= 1 − test accuracy) for the application of the model established by the *Glass* training set on the independent test set with the optimal parameters computed by the methods described in sections 2.5 and 2.6. The training set contains about 70% of the original dataset. It is guaranteed that all classes are contained both in the training set and in the test set.

The results are comparable with error rates obtained by authors of other reviews: Szedmak and Shawe-Taylor [23] got results in the range of 0.3-0.4 with standard OAA and OAO algorithms, García-Pedrajas and Ortiz-Boyer [9] a result of 0.28. Friedrich [8] used 50 bootstrap replications which evoked 50 error rates. Their mean was 0.39 with a standard deviation of 0.04, using a *k-Nearest-Neighbor* algorithm, 0.40 with a standard deviation of 0.07 using a *Decision Tree* based method and 0.41 with a standard deviation of 0.07 using a *Linear Discriminant Analysis*. The best result obtained by [8] was 0.23 with evolutionarily generated architectures of neural networks and combination models originated from bagging (see [2]). The best result of 0.23 in Table 2 is obtained by *Comb-ECOC* with a local parameter optimization. In general, the local optimization delivers the best results but also from *svmpath*, quite good results are achieved, especially with the simple multiclass algorithms that use the *libsvm*.

Table 2 also shows the results obtained with the default parameters of the methods. Note that *ECOC* and *Comb-ECOC* use the *libsvm* and *SVMlight*, respectively, as binary classificators. So we only took the binary default parameters. We also got a result from the method by Crammer and Singer described in section 2.2. Due to high runtimes during the parameter optimization, we did not indicate any results for it in the lower tabulars of Table 2.
If we compare these results with the error rates in the tabular below, we see that the results are much worse than with parameter optimization, except in the case of *OAA* and *ECOC* (global and svmpath optimization). Mainly in the case of the ECOC algorithms, a good parameter optimization is indispensable. To explain this, we differenciate between two kinds of classification errors:

1. Errors occuring because of overlapping classes
2. Errors occuring because of classes that are too far away from each other with other classes between them.

Especially errors of the second type can be serious. Depending on how the classes are distributed in the input space and which classes have to be merged by a binary classificator, the Gaussian kernel may not accomplish several classifications anymore. Therefore, the choice of better parameters is necessary.

Table 2. Error Rates, runtime in CPU seconds (on an Intel Pentium IV processor with 3.06 GHz and 1 GB RAM) and number of support vectors for *Glass* dataset with default and optimal parameters for each algorithm and each optimization method. The mean of the support vectors (over all binary classifications) with the standard deviations in parentheses are mostly indicated. Exceptions: *OAO-global* (total number of support vectors, specific output of *libsvm*), *Crammer-Singer/SVMmulticlass* (Note that these are direct algorithms. See [4] and [24] for more details.) Note that in the case of *Comb-ECOC* we have three different code matrices: the user defined matrix (U) and two randomly defined matrices (R1, R2).

Results with Default Parameters of each Multiclass Algorithm

Algorithm	Error Rate	Runtime	Number of SVs
OAA	**0.27**	0.53	57.17 (38.85)
OAO	0.34	**0.09**	132.00
SVMlight-OAA	0.34	0.98	85.17 (29.53)
SVMlight-OAO	0.38	2.27	32.27 (17.14)
Crammer-Singer	0.28	0.46	480.00
SVMmulticlass	**0.27**	3.33	430.00
ECOC	0.45	1.81	84.42 (33.51)
Comb-ECOC	0.48	189.92	U: 96.07 (28.95)
			R1: 101.00 (24.92)
			R2: 101.00 (24.91)

Error Rates and Runtime with Parameter Optimization

	Optimization Method					
Algorithm	**Global**		**Local**		**SVMpath**	
OAA	0.30	947.64	0.38	1695.26	0.30	23.88
OAO	0.31	282.63	0.30	1062.92	**0.28**	5.63
SVMpath		–		–	0.30	**4.47**
SVMlight-OAA	0.33	**95.5**	0.30	1253.93	0.34	19.72
SVMlight-OAO	0.33	217.62	0.33	2522.45	0.30	10.34
SVMmulticlass	**0.25**	817.79		–		–
ECOC	0.47	996.54	0.44	**944.09**	0.52	81.29
ECOC-SVMpath		–		–	0.48	87.36
Comb-ECOC	0.28	57822.65	**0.23**	5279.15	0.31	458.38

Number of Support Vectors with Parameter Optimization

	Optimization Method								
Algorithm	**Global**			**Local**			**SVMpath**		
OAA		70.00	(37.36)		64.00	(40.63)		48.00	(34.04)
OAO		135.00			28.47	(18.58)		21.13	(15.22)
SVMpath		–			–			14.8	(8.67)
SVMlight-OAA		82.00	(24.47)		72.17	(30.31)		34.00	(21.73)
SVMlight-OAO		32.6	(13.83)		24.27	(15.90)		14.8	(9.50)
SVMmulticlass		570.00			–			–	
ECOC		101.84	(22.31)		88.87	(28.34)		94.03	(15.81)
ECOC-SVMpath		–			–			61.81	(21.87)
Comb-ECOC	U:	20.35	(4.66)	U:	33.10	(2.45)	U:	42.47	(22.05)
	R1:	21.67	(3.62)	R1:	33.82	(2.55)	R1:	53.48	(22.65)
	R2:	21.65	(3.70)	R2:	33.64	(2.44)	R2:	53.48	(22.65)

It is not excluded that the default parameters are better than the optimized ones. This can have two reasons:

1. They are not taken from a grid but defined by another way, so their domain is different.
2. As we use bootstrapping for finding the optimal parameters, the parameters can differ from run to run, as the bootstrap replicates are defined at random.

However, the higher runtime of a parameter optimization is worth getting better models for classification tasks.

Beside the test accuracies, one also has to account the number of support vectors, in order to exclude overfitting. They are also shown in Table 2. If we confront the *OAO* algorithms with the *OAA* algorithms, we see that in the *OAO* case, the number of support vectors is much lower than in the *OAA* case, as in general, the margin is much larger in the first case (see Figure 1). Another explanation might be the larger size of the training sets in the OAA case. Secondly, if we compare *ECOC* and *Comb-ECOC*, we see that the bad test accuracies of *ECOC* in Table 2 are related to the high number of support vectors, whereas the good results of *Comb-ECOC* are connected with a small number of support vectors.

Note that the complexities of the standard algorithms *OAA* and *OAO* are polynomial in the number of classes. Also the algorithms by Crammer and Singer [4] and by Thorsten Joachims [24] can be run in polynomial time. The methods using error correcting output codes are NP-complete. As *Comb-ECOC* consists of three runs defined by three different code matrices, each of which can have an exponential number of columns, its elapsed runtime is the largest by far, followed by *ECOC* and *SVMmulticlass*. *OAA* is not always faster than *OAO*, even if its complexity is $\mathcal{O}(k)$ instead of $\mathcal{O}(k^2)$. Note that the number of data points is smaller in the case of pairwise classification.

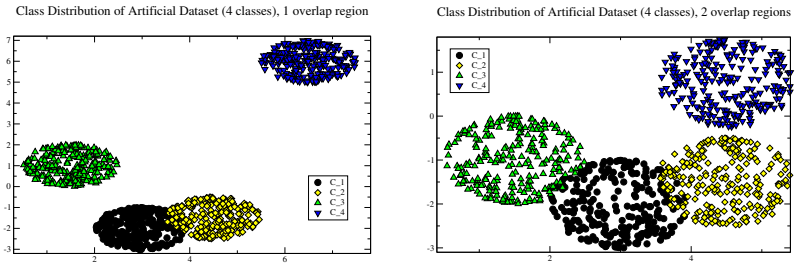

Fig. 2. Artificial Datasets with different margin sizes for OAA and OAO. Each class is defined by a unit circle. The first dataset has one, the second has two overlap regions.

Table 3. Error Rates, runtime and number of support vectors for the artificial dataset in Figure 2 with 1 overlap region with optimal parameters for each algorithm and each optimization method. The mean of the support vectors (over all binary classifications) with the standard deviations in parentheses are mostly indicated. Exception: *OAO-global* (total number of support vectors, specific output of *libsvm*).

Error Rates and Elapsed Runtime in CPU seconds						
	Optimization Method					
Algorithm	**Global**		**Local**		**SVMpath**	
OAA	0.0233	477.08	**0.0233**	**136.37**	0.0233	177.52
OAO	**0.0200**	**40.11**	0.0267	157.93	**0.0167**	**40.52**
SVMlight-OAA	0.0267	290.11	**0.0233**	2980.90	**0.0167**	381.90
SVMlight-OAO	0.0267	368.49	**0.0233**	3999.49	**0.0167**	108.60

Number of Support Vectors						
	Optimization Method					
Algorithm	**Global**		**Local**		**SVMpath**	
OAA	58.25	(41.92)	70.75	(40.05)	22.5	(23.04)
OAO	53.00		41.17	(38.94)	13.67	(19.56)
SVMlight-OAA	82.00	(61.16)	131.25	(76.68)	33.75	(35.53)
SVMlight-OAO	29.50	(18.43)	85.67	(14.25)	13.83	(13.17)

Table 4. Error Rates, runtime and number of support vectors for the artificial dataset in Figure 2 with 2 overlap regions with optimal parameters for each algorithm and each optimization method. The mean of the support vectors (over all binary classifications) with the standard deviations in parentheses are mostly indicated. Exception: *OAO-global* (total number of support vectors, specific output of *libsvm*). Note that in the OAA case, the matrix of the resulting Linear Equation System is singular.

Error Rates and Elapsed Runtime in CPU seconds						
	Optimization Method					
Algorithm	**Global**		**Local**		**SVMpath**	
OAA	0.0267	509.11	0.0233	166.41	singular system	
OAO	0.0200	**48.33**	0.0267	**157.09**	**0.0267**	**60.39**
SVMlight-OAA	0.0233	10748.38	**0.0200**	3081.58	singular system	
SVMlight-OAO	**0.0133**	450.48	**0.0200**	3848.86	**0.0267**	142.03

Error Rates and Elapsed Runtime in CPU seconds						
	Optimization Method					
Algorithm	**Global**		**Local**		**SVMpath**	
OAA	184.25	(5.85)	133.5	(34.85)	singular system	
OAO	113.00		58.83	(31.69)	23.83	(32.96)
SVMlight-OAA	111.00	(8.29)	133.25	(69.01)	singular system	
SVMlight-OAO	37.50	(19.69)	89.83	(24.84)	17.00	(23.68)

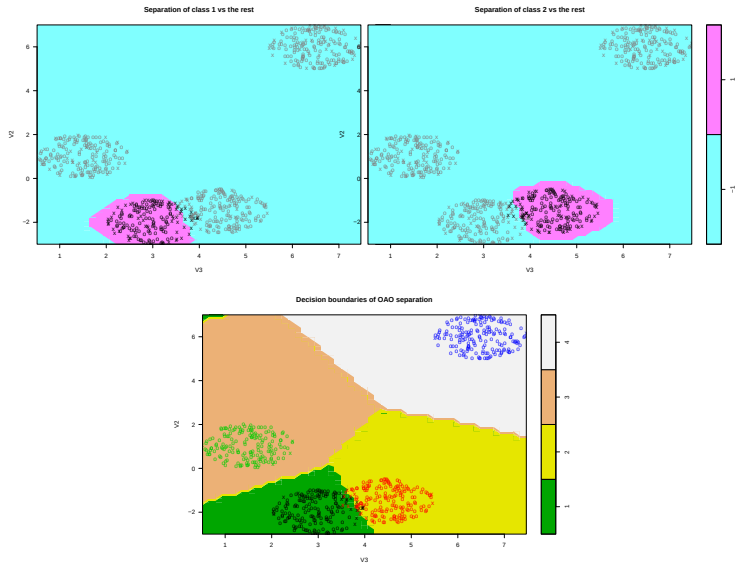

Fig. 3. Decision boundaries for the separations 'Class 1 vs rest', 'Class 2 vs rest' and OAO decision boundaries for the first dataset in Figure 2. We see that the OAA margins are much narrower than the OAO margins. The support vectors are highlighted by crosses.

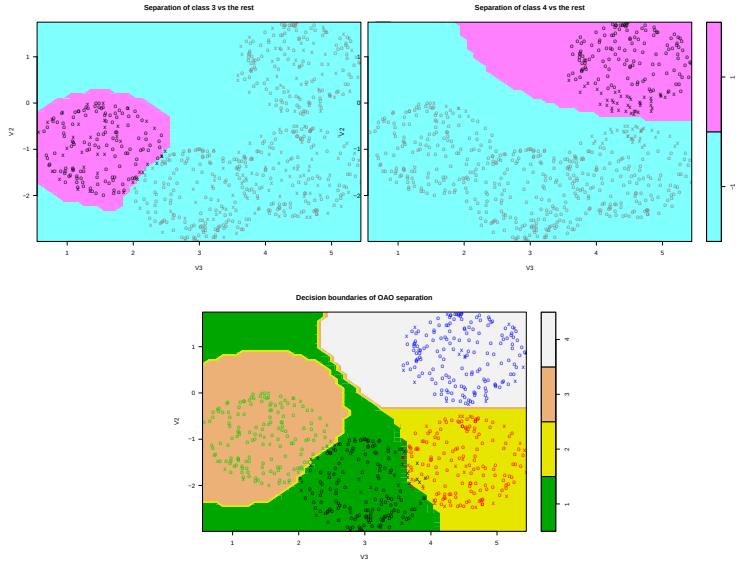

Fig. 4. Decision boundaries for the separations 'Class 3 vs rest', 'Class 4 vs rest' and OAO decision boundaries for the second dataset in Figure 2. We see that the OAA separations are much narrower than the OAO margins. The support vectors are highlighted by crosses.

Finally, we compared the OAA and OAO algorithms on two artificially composed datasets, which are plotted in Figure 2. It is a realization of the theoretical idea formulated by Figure 1. The margin sizes of the OAA algorithm are much smaller than the ones of the OAO algorithm. Furthermore, we have different margin sizes in the OAO case.

Each of the two datasets is two-dimensional and consists of 1000 instances. The 4 classes are equally distributed. They are divided into a training set with 700 and a test set with 300 instances. Table 3 shows the results for the first and Table 4 for the second dataset. In the case of one overlap region, we cannot see any difference between OAA and OAO yet, except the fact that in the OAO case, the number of support vectors is much smaller. The separations 'Class 1 vs rest', 'Class 2 vs rest' and 'Class 1 vs Class 2' will produce training errors. All the other cases are separable. Figure 3 shows the binary decision boundaries of the first two OAA separations and the multiclass decision boundaries of the OAO algorithm. In the OAO case, the margins are much larger than in the OAA case.

The results for the second dataset with two overlap regions are summarized in Table 4. Here, we see a difference between *SVMlight-OAA* and *SVMlight-OAO*: As in Table 3, the number of support vectors is always smaller in the OAO case. Furthermore the error rate is smaller using *SVMlight-OAO*. Also the runtime is much higher in the OAA case. That is because the optimization problems are more difficult to solve. The individual binary separations are more complicated. The corresponding feature spaces will have very high dimensions. For the second dataset, the separations 'Class 1 versus the rest', 'Class 2 versus the rest', 'Class 3 versus the rest', 'Class 1 versus Class 2' and 'Class 1 versus Class 3' cause positive training errors. Figure 4 shows the binary decision boundaries of the last two OAA separations and the multiclass decision boundaries of the OAO algorithm. The margin is smaller in the OAA case, as expected.

In order to complete the experimental proof that the OAO algorithm performs better than the OAA algorithm in the aforesaid special geometric case, we executed pairwise t-tests on 30 bootstrap replications, using a confidence level $\alpha = 0.05$. Especially in the global optimization case the t-tests were significant (p-value: $p \ll \alpha$), in favor of the OAO algorithm.

4 Discussion and Conclusion

In this paper, we present a multiclass combination method named *Comb-ECOC* and compare it statistically with other existing algorithms enlisted in Table 1. Furthermore, we oppose different parameter optimization methods.

Comb-ECOC delivers good results. It performed best for the *Glass* dataset with a special kind of parameter optimization. Its advantages are

1. the usage of error correcting output codes,
2. the prediction of posterior class probabilities,
3. its robustness and

4. the combination of three multiclass runs defined by three different code matrices.

Of course, there are also some disadvantages. The result for the *Glass* dataset is the best, but from this fact, one cannot interprete that it always outperforms the other algorithms. Pursuant to [26], this also depends on the respective dataset. The main problem of *Comb-ECOC* lies in the precoding. In order to maintain the robustness against several binary classification errors, a large minimum Hamming distance between the rows of the code matrices must always be guaranteed. Some approaches can be found in [1], but it is indispensable to use algebraic methods from Galois theory. Especially if the number of k is large (e.g. $k = 15$), a large minimum Hamming distance is not procurable in an easy way, and therefore *Comb-ECOC* will fail. But with a good precoding strategy its performance will be enhanced.

The second disadvantage is its high runtime: For the *local* parameter optimization it takes 1.5 hours and for the *global* optimization even 16 hours. However, from Table 2 (first tabular) we can see that one modelling and classification process without parameter optimization before only takes about 3 minutes.

The second subject of this paper is the parameter optimization. We suggest three methods: *global, local* and *svmpath*. Intuitively, the *local* optimization should perform better than the *global* optimization, because each classificator gets its own optimal parameters. The danger of overfitting is excluded, because we do not minimize training errors but test errors occuring during several bootstrap replicates. As we can see from Table 2, the results are mostly better in the local case.

Another way is to take into account the contrast between a small training error and a large margin instead of only considering several test errors. *SVMpath* does so. Furthermore, it has the advantage of a complexity that is not higher than a usual Support Vector Machine. But as we see in this paper, it does not perform well on *foreign* classification algorithms. It should be used for algorithms implemented in the *libsvm*. The best solution would be to use its own prediction function. A disadvantage of *SVMpath* is the fact that it may compute negative regularization constants which are useless for Support Vector Machines. We found this problematic using other datasets.

Despite the advantages that SVMs show for binary classifications, other methods like *Neural Networks*, *Linear Discriminant Analysis* or *Decision Trees* should be considered in multiclass scenarios.

References

1. Bose, R.C., Ray-Chaudhuri, D.K.: On A Class of Error Correcting Binary Group Codes. Information and Control **3** (1960)
2. Breiman, L.: Bagging Predictors. In: Machine Learning **24** (1996) 123–140
3. Christianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
4. Crammer, K., Singer, Y.: On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. Journal of Machine Learning Reseach **2** (2001) 265–292

5. Dietterich, T., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. Journal of Artificial Intelligence Research **2** (1995) 263–286
6. Dimitriadou, E., Hornik, K., Leisch, F., Meyer, D., Weingessel, A.: The `e1071` package. Manual (2006)
7. Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Chapman & Hall/CRC (1993)
8. Friedrich, C.: Kombinationen evolutionär optimierter Klassifikatoren. PhD thesis, Universität Witten/Herdecke (2005)
9. García-Pedrajas, N., Ortiz-Boyer, D.: Improving Multiclass Pattern Recognition by the Combination of Two Strategies IEEE Transactions on Pattern Analysis and Machine Intelligence **28** (2006)
10. Hastie, T., Rosset, S., Tibshirani, R., Zhu, J.: The Entire Regularization Path for the Support Vector Machine. Technical Report, Statistics Department, Stanford University (2004)
11. Hsu, C.-W., Chang, C.-C., Lin, C.-J.: A Practical Guide to Support Vector Classification. Department of Computer Science and Information Engineering, National Taiwan University (2006)
12. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multi-class Support Vector Machines. IEEE Transactions on Neural Networks **13** (2002) 415–425
13. Huang, T.-J., Weng, R.C., Lin, C.-J.: Generalized Bradley-Terry Models and Multiclass Probability Estimates. Journal of Machine Learning Research **7** (2006) 85–115
14. Hülsmann, M.: Vergleich verschiedener kernbasierter Methoden zur Realisierung eines effizienten Multiclass-Algorithmus des Maschinellen Lernens. Master's thesis, Universität zu Köln (2006)
15. Joachims, T.: Making large-Scale SVM learning practical. Advances in Kernel Methods – Support Vector Learning, MIT Press (1999) 41–56
16. Mencía, E.L.: Paarweises Lernen von Multilabel-Klassifikatoren mit dem Perzeptron-Algorithmus. Master's thesis, Technische Universität Darmstadt (2006)
17. Meyer, D.: Support Vector Machines, the Interface to `libsvm` in package `e1071`. Vignette (2006)
18. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI Repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRespository.html` (1998)
19. Platt, J.C.: Probabilistic Ouputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. Proceedings of Advances in Large-Margin Classifiers, MIT Press (1999) 61–74
20. Roever, C., Raabe, N., Luebke, K., Ligges, U., Szepanek, G., Zentgraf, M.: The `klaR` package. Manual (2006)
21. Ihaka, R., Gentleman, R.: R: A Language for Data Analysis and Graphics. Journal of Computational and Graphical Statistics **5** (1996) 299–314
22. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. MIT Press (2002)
23. Szedmak, S., Shawe-Taylor, J.: Multiclass Learning at One-Class Complexity. Information-Signals, Images, Systems (ISIS Group), Electronics and Computer Science. Technical Report (2005)
24. Tsochantaridis, I., Hofmann, T., Joachims, T., Altun, Y.: Support Vector Machine Learning for Interdependent and Structured Output Spaces. Proceedings of the 21th International Conference on Machine Learning. Banff, Canada (2004)
25. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1995)
26. Wolpert, D.H.: No Free Lunch Theorems for Optimization. Proceedings of IEEE Transactions on Evolutionary Computation **1** (1997) 67–82