H-0241 (H0602-012) February 4, 2006
Computer Science

IBM Research Report

On Multiclass Classification via a Single Binary Classifier

Ran El-Yaniv, Tomer Kotek, Dmitry Pechyony
Department of Computer Science
Technion

Elad Yom-Tov
IBM Research Division
Haifa Research Laboratory
Mt. Carmel 31905
Haifa, Israel

On Multiclass Classification via a Single Binary Classifier

Ran El-Yaniv RANI@CS.TECHNION.AC.IL
Tomer Kotek KOTEK@CS.TECHNION.AC.IL
Dmitry Pechyony PECHYONY@CS.TECHNION.AC.IL
Dept. of Computer Science, Technion

Elad Yom-Tov YOMTOV@IL.IBM.COM
IBM Research Labs, Israel

Abstract

We study multiclass classification methods, whereby the problem is reduced to a *single binary classifier (SBC)*. Such SBC reductions are obtained by embedding the original problem in a higher dimensional space consisting of the original features, as well as several other dimensions determined by a set of (error correcting) codewords. The outstanding features of these methods are their operational simplicity and competitive classification performance. We examine several known and new SBC reductions and provide a comprehensive study of their empirical performance. We also consider a subsampling heuristic that can decrease the computational cost of SBC methods, without significantly reducing classification accuracy. We conclude that SBC approaches are an attractive alternative to standard multiclass decompositions.

1. Introduction

There are many ways in which supervised learning can be employed for classification of data into multiple classes. Our work involves the reduction of multiclass classification problems to binary ones so as to enable the use of binary classifiers for multiclass problems. The widespread practice of employing support vector machines (SVMs) in applications is the main incentive for the ongoing study of this basic problem. However, despite numerous attempts to resolve the issue of how SVMs can be applied to multiclass classification, the understanding of this problem appears to be rather limited, both theoretically and empirically. The confusion surrounding this problem has increased with the availability of increasingly clever and sophisticated solutions, whose authors indicate that there is much to gain by using their approaches. Practitioners who need to solve multiclass problems and choose to use SVMs must face this reduction issue and select one of the available methods, without conclusive literature support for one of the available methods.

Currently, the simplest multiclass decomposition method is ‘one-vs-all’ (a.k.a. ‘one-vs-rest’). Other well known decompositions are the ‘all-pairs’ approach (a.k.a. ‘one-vs-one’, Friedman, 1996) and the ‘error-correcting output coding (ECOC) framework (Sejnowski & Rosenberg, 1987; Dietterich & Bakiri, 1995). We elaborate on some of these methods and other ideas in the sequel.

A related difficulty that only recently started to gain sufficient recognition, and was overlooked in previous studies is the problem of *hyper-parameter tuning*. This issue is of crucial importance when aiming at high performance classification. For example, when using SVM with the RBF kernel, the hyper-parameters are C and γ (σ^2), which must be appropriately selected based on the data to achieve the best possible out-of-sample performance.

A recent paper by Rifkin and Klautau (2004) is among the first to emphasize multiclass reductions in conjunction with hyper-parameter selection. This prominent paper presents an in-depth critical assessment of many previous multiclass papers; its main message is that the simple-minded one-vs-all (OVA) approach is not inferior to many of the other fancier methods, provided that adequate efforts are devoted to hyper-parameter tuning. The compelling case made in this paper for OVA is accomplished through extensive evaluations. We therefore chose to use this paper as a contextual anchor point for our work.

In this paper we study multiclass methods that reduce the problem to a Single Binary Classifier (*SBC*). From our point of view, any multiclass method that relies on only *one application* of a standard binary (soft) classifier is an *SBC method.* SBC reductions are obtained by embedding the original problem in a higher dimensional space consisting of the original features as well as several other dimensions determined by a fixed set of r feature vectors (*codewords*) of some length ℓ. This embedding is implemented by *replicating* the training set points so that each original point is concatenated with each of the r feature vectors. The *binary* labels of the replicated points are set so as to maintain a particular structure in the extended space. The result of this construction is a binary training set given to the learning algorithm that outputs a single soft binary classifier. To classify a new point, the point is replicated r times, once with each of the fixed feature vectors; these r extended points are fed to the soft binary classifier that generates r "signals" and the class is determined as a function of these signals.

The idea of SBC reductions through dimensionality expansion and replication has existed in several rudimentary forms for quite a while. As far as we know, the first documented SBC reduction is the so called *Kesler Construction* (see Duda & Hart, 1973, Sec. 5.12.1). A different type of SBC reduction is the *single call* method presented by Allwein et al. (2000), where the embedding is determined via an error correcting matrix as in error correcting output coding (ECOC). Here we propose a general SBC reduction that is a natural generalization of a recent SBC reduction studied in Anguita et al. (2004). The new SBC reduction also relies on a coding matrix.

Our goal in this work is to examine the various SBC reductions (including ours) and compare them to standard approaches. Overall, we considered five different reductions and compared them to OVA and ECOC. We present extensive empirical evaluations identifying the better SBC methods and show that performance-wise, the best SBC methods are not inferior to both OVA and ECOC. Relying on the paper of Rifkin and Klautau (2004), which compares OVA against other methods, we deduce that SBC is not inferior to standard multiclass methods but is considerably simpler. In particular, SBC methods truly use the underlying *binary* classifier (e.g., SVM) as a *black box*, including the optimization of hyper-parameters. This combination of high performance and operational simplicity makes the SBC approach an attractive alternative to standard multiclass methods.

2. Definitions and Related Work

Let $S = \{(x_i, y_i)\}_{i=1}^m$ be a training set of m examples, where x_i are points in some d-dimensional space $\mathcal{X}$ and each y_i is a label in $\mathcal{Y} = \{1, \dots, k\}$. A *multiclass classifier* h is any function $h : \mathcal{X} \to \mathcal{Y}$. Our goal in multiclass classification is to generate a good multiclass classifier, based on the training set. We measure performance via the standard 0/1-loss function and are interested in average low error over out-of-sample examples.

The 'one-vs-all' (OVA) is one of the simplest methods for multiclass decomposition. It uses the original data relabeled in a binary form to train k soft classifiers. Each classifier is trained to distinguish one of the classes from the rest. The multiclass label of a new data point is predicted by first having each of the binary classifiers classify the point. The index of the classifier with the maximal response is chosen as the predicted label. It is hard to trace the exact origin of OVA, but early references date back to Duda and Hart (1973).

In the basic *error-correcting output coding* (ECOC) framework each of the k given classes is assigned a unique binary vector (called a *codeword*) over $\{\pm 1\}$ of length ℓ. This collection of k codewords forms a $k \times \ell$ *coding matrix* M, whose ℓ columns define ℓ binary partitions of the k classes. Given a training set $S = \{(x_i, y_i)\}$, ℓ binary classifiers are trained. The jth classifier f_j is assigned a unique binary partition defined by the jth column of M and is trained using a training set $\{(x_i, M(i, j))\}$. After the learning process is complete, whenever an unseen point x is given, it is classified by all binary classifiers. This results in a vector $\overline{f}(x) = (f_1(x), \dots, f_\ell(x))$ with $f_j(x)$ being the output of the jth classifier. The point x is assigned to the class whose matrix row is closest to $\overline{f}(x)$. This class assignment mechanism is called *decoding.* In the basic ECOC scheme, a Hamming-based decoding is used where the distance between $\overline{f}(x)$ and the rows of the matrix is computed using the Hamming distance. This general technique was pioneered by Sejnowski and Rosenberg (1987; 1995) and further developed by Allwein et al. (2000).

Single Binary Classifier (SBC) reductions have been discussed already by Duda and Hart (1973), where the *Kesler construction* is presented. This construction is defined only for linear discriminant rules and training sets that are realizable in the OVA sense. The Kesler construction is formally equivalent to learning k independent classifiers simultaneously. Hence, it does not decrease the number of unknown parameters. The construction replicates each d-dimensional training point to $(k-1)$ new points, each of dimension kd. Section 3 is devoted for presenting other known and one new SBC reductions.

Single-machine SVM constructions that were introduced by Vapnik (1998) are related to SBC reductions, but unlike the SBC approach these constructions typically modify the standard SVM optimization problem to include k soft classifiers simultaneously, one for each class. Each example is labeled according to the classifier giving maximal soft classification. In this sense, such constructions can be viewed as a single-machine implementation of OVA and the extended multiclass SVM classifier essentially consists of a number of binary classifiers. See Section 3.1 in (Rifkin & Klautau, 2004) for a more detailed survey.

3. Single Binary Classifier (SBC) Reductions

We propose the following SBC reduction, which is a straightforward generalization of the method studied by Anguita et al. (2004). Let M be a $k \times \ell$ *coding matrix*. We mainly focus on methods where the matrix entries $M(i,j)$ are in $\{0,1\}$. Denote by M_i the ith row of M. We consider SBC reductions that work as follows. Given as input the set of training examples, a coding matrix M, and a learning algorithm $\mathcal{A}$ for soft binary classifiers, the SBC method consists of two stages: *preprocessing* and *binary learning*.

In the preprocessing stage, we construct k different "copies" of each training example x_i, where the rth copy of x_i is $z_{i,r} = x_i \circ M_r$, the concatenation of the row vector x_i with the row vector M_r. The resulting set of new instances $\{z_{i,r}\}$, $i = 1, \ldots, m$, $r = 1, \ldots, k$, are assigned binary labels as follows: for each i and r the instance $z_{i,r}$ is labeled by $y_{i,r} = +1$ iff $y_i = r$ (i.e., the original label y_i of x_i is r). Otherwise, $z_{i,r}$ is labeled by $y_{i,r} = -1$. The resulting binary labeled set $S' = \{(z_{i,r}, y_{i,r})\}$ is of size km and each instance (excluding the label) has $d + \ell$ dimensions.

In the second stage of binary learning, we train the learning algorithm $\mathcal{A}$ with the training set S' and the outcome is a soft binary classifier h_2. To determine a label (in $\mathcal{Y}$) of a new instance x, we generate k copies of x, where the rth copy is $z_r = x \circ M_r$. The label we predict is $\mathrm{argmax}_r\, h_2(z_r)$.

For example, suppose that $\mathcal{Y} = \{1, 2, 3\}$ and the training set consists of the following three labeled examples: $\{(x_1, 1), (x_2, 2), (x_3, 3)\}$. Then, if the coding matrix is

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix},$$

the resulting labeled training set for the binary problem is

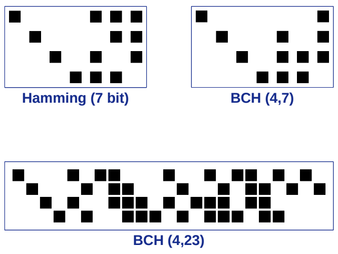
$$\begin{array}{ll}
z_{1,1} = x_1 \circ (1,0,1,1) & y_{1,1} = +1 \\
z_{1,2} = x_2 \circ (1,1,0,0) & y_{1,2} = -1 \\
z_{1,3} = x_3 \circ (0,1,1,0) & y_{1,3} = -1 \\
z_{2,1} = x_1 \circ (1,0,1,1) & y_{2,1} = -1 \\
z_{2,2} = x_2 \circ (1,1,0,0) & y_{2,2} = +1 \\
\\
z_{2,3} = x_3 \circ (0,1,1,0) & y_{2,3} = -1 \\
z_{3,1} = x_1 \circ (1,0,1,1) & y_{3,1} = -1 \\
z_{3,2} = x_2 \circ (1,1,0,0) & y_{3,2} = -1 \\
z_{3,3} = x_3 \circ (0,1,1,0) & y_{3,3} = +1.
\end{array}$$

A special case of our method is the SBC reduction proposed by Anguita et al. (2004). In this reduction, the matrix M is taken to be the $k \times k$ identity matrix. We therefore refer to this reduction as SBC-Identity. We consider several other coding matrices. In SBC-Hamming we take k rows of a Hamming coding matrix and in SBC-BCH we take k codewords of the BCH (Bose, Ray-Chaudhuri, Hocquenghem) code (see Section 4.2 for details). Note that there are many other possibilities for error-correcting coding matrices. For a treatment of Hamming, BCH and other codes see (Roman, 1992). We have not experimented with other codes.

Another method we consider is SBC-Single. It is obtained by taking the column $(1, 2, \ldots, k)^T$ as the coding matrix. In other words, a single feature is concatenated to the data such that the rth "replication" of x_i, is $z_{i,r} = x_i \circ r$. Binary labels are assigned to this data exactly as described above. SBC-Single is thus another special case of our approach. However, it is also a special case of the general 'single-call' SBC reduction of Allwein et al. (2000), which we now describe.

Given a $k \times \ell$ coding matrix M, in the *single call* method of Allwein et al. (2000) each training example (x_i, y_i) is "replicated" ℓ times to create ℓ new training examples of the form $((x_i, s), M(y_i, s))$, where $M(y_i, s)$ is a binary label. Using this training set we induce a binary classifier denoted by h_2. For classifying a new point x we similarly replicate it ℓ times, $z_i = x \circ i$, $i = 1, \ldots, \ell$, and apply h_2 on each of the ℓ instances. As in ECOC, the resulting vector of (soft) classifications $(h_2(z_1), \ldots, h_2(z_\ell))$ is matched to the closest codeword (row) in M to determine the label. The

Figure 1. Three coding matrices. Black squares represent ones and spaces represent zeros.



matching can be done using a Euclidian norm (if h_2 is a soft classifier) or a Hamming distance (if h_2 is a hard classifier). We term this SBC reduction SBC-ECOC. Notice that SBC-Single is a special case of SBC-ECOC applied with a matrix M that is the $k \times k$ identity matrix.

In contrast to our reductions that also use a coding matrix (e.g., SBC-BCH) and extend each example by ℓ additional binary features and replicate each example k times, the SBC-ECOC construction adds a single attribute to each example and replicates it ℓ times.

An immediate question after introducing the new SBC reductions is what is the effect of the coding matrix M. The following experiment over the 4-class UCI `car` dataset (see Table 1) examines the performance of six SBC applications: SBC-Single, SBC-Identity, SBC-Hamming and three SBC-BCH instances corresponding to $(4, 7)$, $(4, 23)$, and $(4, 55)$ BCH coding matrices.[1] The Hamming and two BCH coding matrices are depicted in Figure 1. The 10xCV average test errors of these methods are shown in Figure 2 (see Section 4 for a description of our experimental protocol). With a 5.64% average error, the worst performer is SBC-Single. The best performers are the SBC-BCH reductions with the longer codes. Their error rate is 1.1%, which matches the error obtained by OVA (see Table 2). These results indicate that the choice of the coding matrix can make a significant difference.

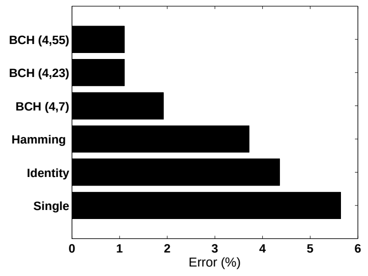
4. Experiments

4.1. Experimental Procedure

Since one of our goals was to use the empirical results of Rifkin and Klautau (2004) as a contextual

[1]The $(4, 23)$ and $(4, 55)$ matrices were generated from $(11, 31)$- and $(64, 127)$-BCH codes, respectively, as described in Section 4.2.

Figure 2. 10xCV average test error rates of several SBC algorithms on the `car` dataset.



	# Train	# Test	# Features	# Classes
`Glass`	214		9	7
`Soybean`	307	376	35	19
`Satimage`	4435	2000	36	6
`Abalone`	3133	1044	8	29
`Optdigits`	3823	1797	64	10
`Car`	1728		6	4
`Spectrometer`	531		101	48
`Yeast`	1484		8	10
`Page blocks`	5473		10	5

Table 1. Datasets summary.

anchor point, we attempted to replicate their experimental procedure. Our replication includes the choice of datasets, the hyper-parameter selection method (through cross-validation), and statistical tests to derive statistical conclusions. This replication was accomplished in its entirety up to variations due to random choices of cross validation folds and our exclusion of one large dataset (see below). Thus, our results for OVA are not identical but differences are on average negligible. Here we present the results of our own runs.

The datasets we used, all from the UCI repository, are summarized in Table 1. Note that Rifkin and Klautau also experimented with the UCI `letter` dataset, but we omitted it due to its overwhelming size that precluded obtaining results in time for the ICML deadline. It is interesting to note that over this collection of datasets, Fürnkranz (2002) found a substantial difference between OVA and all-pairs when the underlying classifier was Ripper. However, Rifkin and Klautau found that with well-tuned SVMs, there is no difference in the performance of these decompositions.

For all sets that include a fixed train/test split (indicated in Table 1 by a number in the 'Test' column) we used the given split. Otherwise, ten-fold cross-validation (10xCV) was used; namely, in each fold, the union of nine out of ten equally sized subsets were used for training, and the tenth for testing.

Nominal attributes with n possible values were sub-

	Multiple call classifiers		Single-call classifiers				
	OVA	ECOC	SBC-Single	SBC-Identity	SBC-ECOC	SBC-Hamming	SBC-BCH
`Glass`	32.38	32.38	36.67	32.38	49.52	**30.95**	**30.95**
`Soybean`	**6.91**	7.18	25.00	7.18	88.56	7.18	**6.91**
`Satimage`	**8.60**	8.75	12.05	8.75	8.80	8.70	8.75
`Abalone`	79.89	78.16	79.89	79.60	**76.82**	79.60	79.60
`Optdigits`	**2.56**	2.84	5.29	2.89	99.61	2.84	2.84
`Car`	**1.10**	4.36	5.64	4.36	3.90	3.72	1.92
`Spectrometer`	53.02	53.07	89.43	53.58	53.02	53.02	**52.83**
`Yeast`	**39.53**	40.47	40.07	40.07	40.88	40.07	40.54
`Page blocks`	3.33	3.20	3.51	3.18	3.42	3.16	**3.13**
Average rank	**2.67**	4.06	6.17	4.28	5.33	2.83	**2.67**

Table 2. 10xCV average test errors rates (%) of seven algorithms on nine datasets. Best results for the dataset appear in boldface. Average ranks of the algorithms appear in last row.

stituted by n binary features, where the ith binary feature was set to 1 iff the corresponding nominal attribute took the ith possible value. For each feature its average and standard deviation over the training set was computed, and these were used for normalizing the data (training and testing) by subtracting the average and dividing by the standard deviation.

In all our experiments, we used the *SVMTorch* implementation of a binary SVM inducer (Collobert & Bengio, 2001) and applied it with a radial-basis function (RBF) kernel. Following Rifkin and Klautau (2004), the hyper-parameters σ and C of this kernel were optimized using a simple greedy search via 10xCV *on the training set* as follows. Initial values of σ and C were set to 1. The value of σ was then increased or decreased by a factor of 2 until no improvements were seen for three consecutive attempts. Then, σ was held fixed at the best value found, and an identical optimization was performed over C.[2]

4.2. Methods for comparison

We compared five SBC reductions: SBC-Single, SBC-Identity, SBC-ECOC, SBC-Hamming, and SBC-BCH (discussed in Section 3). These SBC reductions were compared to OVA and ECOC. For each dataset, all algorithms which use a BCH coding matrix (SBC-BCH, ECOC, and SBC-ECOC) were applied with the same BCH coding matrix as described below. Overall, including OVA, we tested seven algorithms. Our initial experiments with SBC-ECOC showed catastrophic errors. We therefore applied this reduction while treating the single added feature as nominal. This change somewhat improved its results.

The BCH coding matrices were generated as follows. First we encoded the columns of an identity matrix of size k using a BCH polynomial of length n.[3] Since a BCH polynomial over the binary field $GF(2)$ has length $n = 2^r - 1$ (bits), we used the *shortest* code with a length larger than or equal to the number of classes, concatenating zeros to the codewords to attain the necessary length for the encoded vector. We dropped any locations in the resulting code words that were identical across all code words. We used a similar procedure for Hamming codes.

4.3. Results

Table 2 shows the errors (%) obtained for each of the seven methods. The best results (lowest errors) in each row appear in boldface. The average *ranks* of the various algorithms appear in the last row of the table. Following Demšar (2006), these ranks were computed as averages of row ranks.[4] The best performers, in terms of ranks, are jointly OVA and SBC-BCH, and the runner-up is SBC-Hamming. The worst performer is SBC-Single. Interestingly, SBC-ECOC is a best performer on one dataset (`abalone`), but achieves terrible results on others (e.g., `optdigits`). We note that our error rates for OVA are very close to those reported in Rifkin and Klautau (2004). Specifically, the average error difference over the nine datasets is about 0.2% (and the maximal difference of 1.9% is obtained on the `glass` dataset).

Since it was tempting to use the average ranks as overall rating of the algorithms, we conducted a statistical test to assess the significance of these ranks. We used the comparison methodology for *multiple algo-*

[2]One can consider various ways to improve the optimization routine suggested by Rifkin and Klautau. For example, it is potentially better to jointly optimize over C and σ, but computationally, this would be rather expensive.

[3]The Matlab command (communication toolbox) for generating such (k, n)-BCH encoding is `bchenc(gf(eye(`k`)),`n`,`k`)`.

[4]For each row, if the errors of all algorithms are distinct they are assigned the ranks in $\{1, \ldots, 7\}$. When algorithms share exactly the same error they are all assigned the same average rank. For example the rank vector for `Glass` dataset is $(4, 4, 6, 4, 7, 1.5, 1.5)$.

	SBC-BCH vs. SBC-IDENTITY	SBC-BCH vs. OVA
`Glass`	$\leftarrow [-9.52, 4.76]$	$\leftarrow [-4.23, 1.59]$
`Soybean`	$\leftarrow [-0.80, 0.00]$	$[-1.60, 1.60]$
`Satimage`	$[0.00, 0.00]$	$[-0.05, 0.35]$
`Abalone`	$[-0.26, 0.26]$	$\leftarrow [-1.02, 0.41]$
`Optdigits`	$[-0.13, 0.00]$	$[0.00, 1.11] \rightarrow$
`Car`	$\leftarrow [-4.65, -0.58]$	$[0.00, 2.33] \rightarrow$
`Spectrometer`	$\leftarrow [-3.77, 1.89]$	$[-5.66, 5.66]$
`Yeast`	$[-2.03, 3.38] \rightarrow$	$[-2.03, 4.05] \rightarrow$
`Page blocks`	$[-0.37, 0.18]$	$[-0.91, 0.55]$

Table 3. 90% bootstrap confidence intervals for the difference in performance between SBC-BCH and SBC-IDENTITY (left column), and SBC-BCH and OVA (right column).

rithms described in Demšar (2006). Specifically, we computed the F_F statistics of Iman and Davenport (1980) (see Demšar, 2006) with a confidence level of 90% to determine if all compared algorithms exhibited the same performance. According to this test, the error rates of the algorithms (with respect to the entire collection of datasets) were statistically different with a confidence level of 90%. We then applied Holm's procedure (Holm, 1979; see Demšar, 2006) with a confidence level of 90% and found that the performances of SBC-SINGLE and SBC-ECOC are the worst among the algorithms. Finally, we compared the performance of OVA, SBC-IDENTITY, ECOC, SBC-HAMMING and SBC-BCH using the F_F statistics and found that with a confidence level of 90% the differences between these five methods are statistically insignificant.

Clearly, this result is highly dependent on the composition of datasets in our collection. Certain subsets of this collection show more pronounced (and statistically significant, according to some tests) differences between various methods. For example, with respect to the `car` dataset alone we do observed interesting differences, as shown in Figure 2.

To systematically identify potentially interesting differences between the methods, with respect to individual datasets, we followed Rifkin and Klautau (2004) and calculated a 90% bootstrap confidence interval for the differences in performance between *pairs* of algorithms *with respect to individual datasets.* For a dataset with a given train/test partition and two given classifiers c_1 and c_2 (induced on the train set), we drew 10,000 bootstrap samples from the test set. When a train/test partition was unavailable, each bootstrap sample consisted of 10% of the points in the dataset, drawn with equal probability from each of the ten folds. For each sample we calculated the performance difference of the two classifiers. This difference is a number in $[-1, 1]$, where a negative difference reflects an advantage of c_1 and a positive difference an advantage of c_2. Zero means that the classifiers performed identically over the test set. For a desired confidence level δ (e.g., 90%) we output the confidence interval $[a, b]$ where a is the $(\frac{1-\delta}{2})$-quantile of the 10,000 differences and b is the $(\frac{1+\delta}{2})$-quantile of these differences.[5]

Table 3 shows confidence intervals corresponding to $\delta = 90\%$ for two pairwise comparisons: SBC-BCH vs. SBC-IDENTITY and SBC-BCH vs. OVA. An entry where both a and b are negative reflects a statistically significant advantage to the first algorithm (which is SBC-BCH in both columns of the table). Specifically, such an entry indicates a probability of over 95% that the first algorithm is better (note that b is a 95%-quantile). Entries of the form $[b, 0]$, where b is negative, suggest a probability of less than 5% that the second algorithm is better. The symmetrically reversed statements (for entries of the form $[0, a]$) hold as well. If 0 is properly included in the interval, then no algorithm is significantly better than the other but the magnitudes of a and b can indicate which algorithm has an advantage . Note that this statistical procedure does not account for the Bonferroni correction for multiple testing. Nevertheless, the presented statistics are based on the actual classifications of individual test points, and therefore provide another useful perspective. Note that the previous (F_F) test we used is only based on the *relative ranks* of the algorithms.

We interpret the results of Table 3 as follows. We consider a confidence interval $[a, b]$ as 'interesting' if $\big||a| - |b|\big| > 0.5$ (i.e., the skew in the advantage of one algorithm is greater than 0.5%); or (ii) $a < b < 0$; or (iii) $0 < a < b$. Other cases are considered 'not interesting'. We mark with '$\leftarrow$' interesting cases where SBC-BCH is significantly better than the other algorithm (SBC-IDENTITY or OVA). Entries marked with '$\rightarrow$' indicate cases where the other algorithm is better.

[5]This test was proposed in Rifkin and Klautau (2004) as an alternative to McNemar's statistical test that accounts for the size of the differences between the algorithms' errors (which is ignored by McNemar's test).

Unmarked entries do not exhibit a statistically sig icant interesting event. We see that in four datas SBC-BCH is better than SBC-Identity, which p forms better over one dataset. On the four rema ing sets, there are no interesting differences betw the algorithms (where they are identical specifically `satimage`). A less pronounced difference is obser between SBC-BCH and OVA.

Based on these observations our goal now is to f a large subset of the nine datasets that "separat the SBC algorithms with high significance. It tu out that the exclusion of `Abalone`, `Yeast` and `P` `blocks` leaves a collection of six datasets on which we have the following result using the Demšar (2006) routine for testing the difference between the algorithms: With confidence 90% OVA, SBC-BCH and SBC-Hamming were significantly better than SBC-ECOC, SBC-Single, SBC-Identity and ECOC. While this result can be accused of "data snooping" it may still indicate that there are quite a few datasets that will be better handled by OVA, SBC-BCH or SBC-Hamming (rather than by the other SBC approaches or ECOC).

Finally, we considered the question of whether there were any differences in the computational efforts required to optimize the SVM hyper-parameters for the different methods. To this end, we counted the total number of optimization *search steps* needed to reach the best set of parameters (according to our optimization routine). We compared the average number of steps required by the algorithms using the F_F statistics of Iman and Davenport (1980) (see Demšar, 2006) with a confidence level of 90%. The result is that there is no significant difference in the number of optimization steps needed among algorithms.[6]

5. Reducing the Computational Load

When considering SBC as an alternative to standard methods such as OVA, one should be aware of the overall time complexity of SBC methods, which depend on a number of factors. As discussed in Section 4, the number of optimization steps required to tune the hyper-parameters is not larger than OVA or ECOC. The main penalty we pay when using SBC is caused by the blow up in the number of training examples due to replication. The original multiclass problem had m examples, and SBC constructs one binary classifier based on km examples. Note that OVA also replicates the problem and solves k binary problems each of size m. However, because standard SVM inducers do not scale linearly with the training set size, the replication performed by SBC methods can raise a computational burden when the original problem is large.[7]

[6]Note, however, that each of these search steps required different efforts from the binary SVMs (see discussion in Section 5).

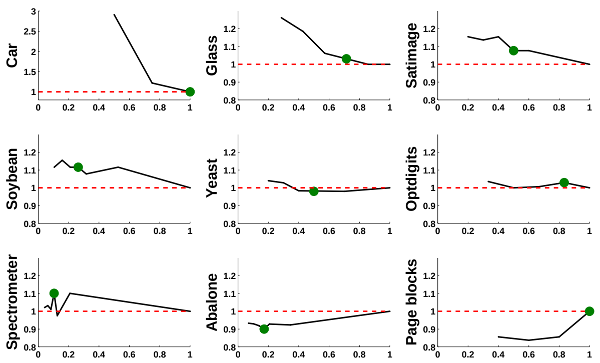


Figure 3. Relative errors of SBC-BCH with subsampling compared to SBC-BCH trained on the full dataset, as a function of the fraction of training data sampled.

However, in many cases it may not be necessary to replicate the data k times in order to generate a successful SBC classifier. Following Allwein et al. (2000) we examined possibilities for subsampling the replicated training data and tested the effect of this sampling on the classifier performance. Specifically, after replicating each point k times, as in the original SBC reduction, the data was sampled so that for each data point the replica with the correct class label was kept, in addition to $s < k$ other replications that were chosen uniformly at random without replacement. All other steps of the training and testing remained as in the original setting.

Figure 3 shows the effect of subsampling on each of the nine datasets for varying values of s when using the SBC-BCH algorithm. In each of the nine graphs, the x-axis is the ratio $(s+1)/k$ for $1 \leq s \leq k-1$. The y-axis is the ratio of test error with s-subsampling to test error without subsampling. Interestingly, for three of the datasets subsampling can actually improve performance. In the `car` dataset, performance significantly deteriorates. For all other datasets, the relative error is increased by no more than 25%, even for very small values of s.

Thus, it is plausible that one could train a classifier that is almost as accurate as that trained on the whole dataset using much smaller resources. Consider taking always $s = \min(4, k-1)$ (so that the data is never

[7]There are approximations to SVM that do scale linearly (Tsang et al., 2005).

replicated more than five times). The values of s selected by this heuristic (and the resulting relative error increase) are indicated by (green) dots in the graphs of Figure 3. This subsampling leads to an error deterioration of $1\% \pm 2\%$, averaged over the datasets.

This idea of diluting the training set in multiclass reductions was proposed by Allwein et al. (2000), where the standard ECOC coding matrix was extended to ternary codes (i.e., each entry $\in \{-1, 0, 1\}$). For a given matrix M, the SBC-ECOC discussed above (and termed the 'single call' scheme by Allwein et al.) ignores the (replicated) example $((x_i, s), M(y_i, s))$ if its label $M(y_i, s) = 0$. Allwein et al. showed that with roughly half of the replicated examples it is still possible to sustain the same error level.

6. Concluding Remarks

This paper presents an extensive study of SBC multiclass reductions and proposes a new reduction scheme based on coding matrices. A benchmark of nine UCI problems (that was assembled by other papers) showed that the best SBC methods achieve a classification performance that is competitive with standard OVA and with ECOC (applied with BCH codes). The main advantage of SBC is its operational simplicity, which allows practitioners to use standard, off-the-shelf, binary SVM packages and utilize them in a black box fashion, including the optimization of hyper-parameters. Other multiclass reductions do not offer this convenience.

A major new application of SBC reductions is in the context of active learning with SVMs (see, for example, Tong and Koller (2001)). So far, most (if not all) studies of such methods were restricted to binary problems. In active learning, the inherent difficulty when attempting to utilize a multiclass reduction to multiple binary problems is that the active queries should arrive from an ensemble of binary learners. How to aggregate these queries into one in each active learning trial is an open question. See one attempt to solve this problem by Tong, 2001. In contrast, when using SBC reductions, the single binary problem automatically determines the relative importance of its "subproblems". Our initial experiments with this approach for multiclass active learning were very successful.

Many questions remain open. The main issue that deserves careful study is: What makes a coding matrix effective in SBC reductions? This question is related to but different from the parallel question in ECOC decompositions (and the understanding of this issue in ECOC appears to be limited). Also, it would be desirable to find a subsampling method that can intentionally (rather than randomly) eliminate unnecessary data replications whenever they can be identified.

While the understanding of SBC reductions is incomplete, our unequivocal conclusion is that when the dataset contains small to moderate training samples and number of classes, SBC reductions can conveniently replace standard SVM multiclass methods.

References

Allwein, E., Schapire, R., & Singer, Y. (2000). Reducing multiclass to binary: a unifying approach for margin classifiers. *JMLR*, *1*, 113–141.

Anguita, D., Ridella, S., & Sterpi, D. (2004). A new method for multiclass support vector machines. *IJCNN* (pp. 407–412).

Collobert, R., & Bengio, S. (2001). SVMTorch: support vector machines for large-scale regression problems. *J. Mach. Learn. Res.*, *1*, 143–160.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *JMLR*, *7*, 1–30.

Dietterich, T., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *JAIR*, *2*, 263–286.

Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Friedman, J. (1996). *Another approach to polychotomous classification* (Technical Report). Stanford University.

Fürnkranz, J. (2002). Round robin classification. *Journal of Machine Learning Research*, *2*, 721–747.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, *6*, 65–70.

Iman, R., & Davenport, J. (1980). Approximations of the critical regions of the Friedman statistic. *Communications in Statistics*, *A9*, 571–595.

Rifkin, R., & Klautau, A. (2004). In defense of One-vs-All classification. *JMLR*, *5*, 101–141.

Roman, S. (1992). *Coding and information theory*. Springer.

Sejnowski, T., & Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Journal of Complex Systems*, *1*, 145–168.

Tong, S. (2001). *Active learning: Theory and applications*. Doctoral dissertation, Stanford University.

Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *JMLR*, *2*, 45–66.

Tsang, I., Kwok, J., & Cheung, P. (2005). Core vector machines: Fast SVM training on very large data sets. *J. Mach. Learn. Res.*, *6*, 363–392.

Vapnik, V. (1998). *Statistical Learning Theory*. Wiley Interscience.