From Margins to Probabilities in Multiclass Learning Problems

Andrea Passerini[1] and **Massimiliano Pontil**[2] and **Paolo Frasconi**[3]

Abstract. We study the problem of multiclass classification within the framework of error correcting output codes (ECOC) using margin-based binary classifiers. An important open problem in this context is how to measure the distance between class codewords and the outputs of the classifiers. In this paper we propose a new decoding function that combines the margins through an estimate of their class conditional probabilities. We report experiments using support vector machines as the base binary classifiers, showing the advantage of the proposed decoding function over other functions of the margin commonly used in practice. We also present new theoretical results bounding the leave-one-out error of ECOC of kernel machines, which can be used to tune kernel parameters. An empirical validation indicates that the bound leads to good estimates of kernel parameters and the corresponding classifiers attain high accuracy.

Keywords: Machine Learning, Error Correcting Output Codes, Support Vector Machines, Statistical Learning Theory.

1 Introduction and Notation

Many machine learning algorithms are intrinsically conceived for binary classification. However, many real world learning problems require that inputs are mapped into one of several possible categories. The extension of a binary algorithm to its multiclass counterpart is not always possible or easy to conceive. An alternative consists in *reducing* a multiclass problem into several binary sub-problems. Perhaps the most general reduction scheme is the information theoretic method based on error correcting output codes (ECOC), introduced by Dietterich and Bakiri [5] and more recently extended in [1]. ECOC work in two steps: training and classification. During the first step, S binary classifiers are trained on S dichotomies of the instance space, formed by joining non overlapping subsets of classes. Assuming Q classes, let us introduce a "coding matrix" $M \in \{-1, 0, 1\}^{Q \times S}$ which specifies a relation between classes and dichotomies. $m_{qs} = 1$ ($m_{qs} = -1$) means that examples belonging to class q are used as positive (negative) examples to train the $s-$th classifier f_s. When $m_{qs} = 0$, points in class q are not used to train the $s-$th classifier. Thus each class q is encoded by the $q-$th row of matrix M which we also denoted by $\mathbf{M}_q$. Typically the classifiers are trained independently. This approach is known as *multi-call* approach. Recent works [8, 2] considered also the case where classifiers are trained simultaneously (*single-call* approach). In this paper we focus on the former approach although some of the results may be extended to the latter one. In the second step a new input $\mathbf{x}$ is classified by computing the vector formed by the outputs of the classifiers, $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_S(\mathbf{x}))$ and choosing the class whose corresponding row is closest to $\mathbf{f}(\mathbf{x})$. In so doing, classification can be seen as a decoding operation and the class of input $\mathbf{x}$ is computed as

$$\arg\min_{q=1}^{Q} d(\mathbf{M}_q, \mathbf{f}(\mathbf{x})),$$

where d is the decoding function. In [5] the entries of matrix M were restricted to take only binary values and the d was chosen to be the Hamming distance:

$$L(\mathbf{M}_q, \mathbf{f}) = \sum_{s=1}^{S} \frac{|m_{qs} - \operatorname{sign}(f_s)|}{2} \tag{1}$$

In the case that the binary learners are margin-based classifiers, [1] shown the advantage of using a loss-based function of the margin

$$d_L(\mathbf{M}_q, \mathbf{f}) = \sum_{s=1}^{S} L(m_{qs} f_s)$$

where L is a loss function. Such a function has the advantage of weighting the confidence of each classifier thus allowing a more robust classification criterion. The simplest loss function one can use is the linear loss for which $L(m_{qs} f_s) = -m_{qs} f_s$ but several other choices are possible and it is not clear which one should work the best. In the case that all binary classifiers are computed by the same learning algorithm Allwein, Shapire and Singer [1] proposed to set L to be the same loss function used by that algorithm. In this paper we suggest a different approach which is based on decoding via conditional probabilities of the outputs of the classifiers. The advantages offered by our approach is twofold. First, the use of conditional probabilities allows to combine the margins of each classifier in a principled way. Second, the decoding function is itself a class conditional probability which can give an estimate of multiclassification confidence. This approach is discussed in Section 2. In Section 3 we present experiments, using SVM as the underlying binary classifiers, which enlighten the advantage over other proposed decoding schemes. In Section 4 we study the generalization error of ECOC. We present a general bound on the leave one out error in the case of ECOC of kernel machines. The bound can be used for estimating optimal kernel parameters. The novelty of this analysis is that it allows multiclass parameters optimization even though the binary classifiers are trained independently. We report experiments showing that the bound leads to good estimates of kernel parameters and the corresponding classifiers attain high accuracy.

[1] Dept. of Systems and Computer Science, University of Florence, Firenze, Italy
[2] Dept. of Computer Science, University of Siena, Siena, Italy
[3] Dept. of Systems and Computer Science, University of Florence, Firenze, Italy

2 Decoding Functions Based on Conditional Probabilities

We have seen that a loss function of the margin presents some advantage over the standard Hamming distance because it can encode the confidence of each classifier in the ECOC. This confidence is, however, a relative quantity, i.e. the range of the values of the margin may vary with the classifier used. Thus, just using a linear loss function may introduce some bias in the final classification in the sense that classifiers with a larger output range will receive a higher weight. Not surprisingly, we will see in the experiments below that the Hamming distance can work better than the linear and soft-margin losses in the case of pairwise schemes. A straightforward normalization in some interval, e.g. $[-1, 1]$, can also introduce bias since it does not fully take into account the margin distribution. A more principled approach is to estimate the conditional probability of each output codebit O_s given the input vector $\mathbf{x}$. Assuming that all the information about $\mathbf{x}$ that is relevant for determining O_s is contained in the margin $f_s(\mathbf{x})$ (or f_s for short) the above probability for each classifier is $P(O_s|f_s, s)$. We can now assume a simple model for the probability of Y given the codebits $O_1, \ldots, O_S$. The conditional probability that $Y = q$ should be 1 if the configuration on $O_1, \ldots, O_S$ is the code of q, should be zero if it is the code of some other class $q' \neq q$, and uniform (i.e. $1/Q$ if the configuration is not a valid class code. Under this model,

$$P(Y = q|\mathbf{f}) = P(O_1 = m_{q1}, \ldots, O_S = m_{qS}|\mathbf{f}) + \alpha$$

being α a constant that collects the probability mass dispersed on the $2^S - Q$ invalid codes. Assuming that O_s and $O_{s'}$ are conditionally independent given $\mathbf{x}$ for each s, s', we can write the likelihood that the resulting output codeword is q as

$$P(Y = q\,|\mathbf{f}) = \prod_{s=1}^{S} P(O_s = m_{qs}\,|f_s) + \alpha. \qquad (2)$$

In this case, if α is small, the decoding function will be:

$$d(\mathbf{M}_q, \mathbf{f}) \approx -\log P(Y = q\,|\mathbf{f}). \qquad (3)$$

The problem boils down to estimating the individual conditional probabilities in Eq. (2). To do so, one can try to fit a parametric model on the output produced by a n-fold cross validation procedure. In our experiments we choose this model to be a sigmoid computed on a 3-fold cross validation as suggested in [10]:

$$P(O_s = m_{qs}\,|f_s, s) = \frac{1}{1 + \exp\{A_s f_s + B_s\}}.$$

It is interesting to notice that an additional advantage of the proposed decoding algorithm is that the multiclass classifier outputs a conditional probability rather than a mere class decision.

3 Experimental Comparison Between Different Decoding Functions

The proposed decoding method is validated on ten datasets from UCI repository. Their characteristics are shortly summarized in Table 1.

We trained multiclass classifiers using SVM as the base binary classifier. In our experiments we compared our decoding strategy to Hamming and other common loss-based decoding schemes (linear, and the soft margin loss used to train SVM) for different types of ECOC schemes: one-vs-all, all-pairs, and dense matrices consisting of $3Q$ columns of $\{-1, 1\}$ entries. SVM were trained on a Gaussian kernel with the same value of the variance for all the experiments. For datasets with less than 2,000 instances we used ten random splits of the available data (picking up 2/3 of examples for training and 1/3 for testing) and averaged results over the ten trials. For the remaining datasets we used the original split defined in the UCI repository. Results are summarized in Tables 2–4.

Table 1. Characteristics of the Datasets used

Name	Classes	Train	Test	Inputs
Anneal	5	898	-	38
Ecoli	8	336	-	7
Glass	6	214	-	9
Letter	26	15000	5000	16
Optdigits	10	3823	1797	64
Pendigits	10	7494	3498	16
Satimage	6	4435	2000	36
Segment	7	1540	770	19
Soybean	19	683	-	35
Yeast	10	1484	-	8

Table 2. One-vs-All

Dataset	Hamming	Linear	Soft-margin	Likelihood
Anneal	94.4	94.8	94.8	**96.4**
Ecoli	64.9	**79.0**	**79.0**	76.8
Glass	27.6	54.9	54.9	**59.6**
Letter	30.9	69.8	69.8	**76.8**
Optdigits	94.8	**97.2**	**97.2**	**97.1**
Pendigits	90.8	**94.5**	**94.5**	**94.8**
Satimage	81.0	**82.9**	**82.9**	**83.1**
Segment	56.5	82.2	82.2	**88.2**
Soybean	75.7	92.1	92.1	**92.7**
Yeast	14.5	54.0	54.0	**57.9**

Table 3. All-Pairs

Dataset	Hamming	Linear	Soft-margin	Likelihood
Anneal	94.7	93.6	95.1	**95.9**
Ecoli	77.0	76.6	77.0	**82.8**
Glass	50.0	47.9	50.7	**61.0**
Letter	80.1	54.3	80.3	**81.1**
Optdigits	**97.3**	93.8	96.3	**97.4**
Pendigits	96.1	89.5	95.0	**96.5**
Satimage	**85.3**	75.2	**84.9**	**85.0**
Segment	84.9	70.5	85.5	**86.1**
Soybean	90.1	90.7	90.6	**92.3**
Yeast	52.7	53.1	52.6	**58.6**

Our likelihood decoding works significantly better for all ECOC schemes. This result is particularly clear in the case of pairwise classifiers. In the case of dense ECOC some dichotomies can be particularly hard to learn and in this situation the sigmoid may be a too simple model for the conditional probability. We suspect that by using a better estimate of the conditional probability one could obtain

Table 4. Dense Codes

Dataset	Hamming	Linear	Soft-margin	Likelihood
Anneal	94.5	94.9	94.9	**95.9**
Ecoli	78.4	**79.2**	**79.1**	78.0
Glass	43.9	47.9	48.1	**53.6**
Letter	61.7	62.9	63.2	**64.3**
Optdigits	**97.3**	**97.3**	**97.3**	**97.2**
Pendigits	90.9	92.1	92.1	**92.9**
Satimage	81.3	**82.6**	**82.5**	**82.9**
Segment	82.7	84.3	84.3	**85.7**
Soybean	91.8	**92.2**	**92.2**	**92.4**
Yeast	50.4	54.2	54.2	**56.8**

better results with the likelihood decoding. Notice that the Hamming distance works well in the case of pairwise classification, while it performs poorly with one-vs-all classifiers. Both results are not surprising: the Hamming distance corresponds to the majority vote, which is known to work well for pairwise classifiers [7] but does not make much sense for one-vs-all because in this case ties may occur often.

4 Tuning Kernel Parameters

In this section we study the problem of model selection in the case of ECOC of Kernel Machines [12, 11, 6, 3]. The analysis uses a leave-one-out error estimate as the key quantity for selecting the best model. We first recall the main features of kernel machines for binary classification. For a more detailed account consistent with the notations in this section see [6].

4.1 Background on Kernel Machines

Kernel machines are the minimizers of functionals of the form:

$$H[f; D_\ell] = \frac{1}{\ell}\sum_{i=1}^{\ell} V(c_i f(\mathbf{x}_i)) + \lambda \|f\|_K^2, \tag{4}$$

where we use the following notation:

- $\{(\mathbf{x}_i, c_i) \in X \times \{-1,1\}\}_{i=1}^{\ell}$ is the training set.
- f is a function $\mathbb{R}^n \to \mathbb{R}$ belonging to a reproducing kernel Hilbert space $\mathcal{H}$ defined by a symmetric and positive definite kernel K, and $\|f\|_K^2$ is the norm of f in this space. See [12, 13] for a number of kernels. The classification is done by taking the sign of this function.
- $V(cf(\mathbf{x}))$ is a loss function whose choice determines different learning techniques, each leading to a different learning algorithm (for computing the coefficients α_i - see below). In this paper we assume that V is monotonic non increasing.
- λ is called the regularization parameter and is a positive constant.

Machines of the form in Eq. (4) have been motivated in the framework of statistical learning theory. Under rather general conditions the solution of Equation (4) is of the form[4]

$$f(\mathbf{x}) = \sum_{i=1}^{\ell} \alpha_i c_i K(\mathbf{x}_i, \mathbf{x}). \tag{5}$$

[4] We assume that the bias term is incorporated in the kernel K.

Let us introduce the matrix G defined by $G_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. The coefficients α_i in Equation (5) are learned by solving the following optimization problem:

$$\begin{array}{l} \max_\alpha H(\alpha) = \sum_{i=1}^{m} S(\alpha_i) - \frac{1}{2}\sum_{i,j=1}^{\ell} \alpha_i\alpha_j c_i c_i G_{ij} \\ \text{subject to}: \ 0 \le \alpha_i \le C, \ i = 1, \ldots, \ell, \end{array} \tag{6}$$

where $S(\cdot)$ is a concave function and $C = \frac{1}{2\ell\lambda}$ a constant. Support Vector Machines (SVMs) are a particular case of these machines for $S(\alpha) = \alpha$. This corresponds to a loss function V in (4) that is of the form $|1 - cf(\mathbf{x})|_+$, where $|x|_+ = x$ if $x > 0$ and zero otherwise. The points for which $\alpha_i > 0$ are called support vectors.

An important property of kernel machines is that, since we assumed K is symmetric and positive definite, the kernel function can be re-written as

$$K(\mathbf{x}, \mathbf{t}) = \sum_{n=1}^{\infty} \lambda_n \phi_n(\mathbf{x})\phi_n(\mathbf{t}). \tag{7}$$

where $\phi_n(\mathbf{x})$ are the eigenfunctions of the integral operator associated to kernel K and $\lambda_n \ge 0$ their eigenvalues [4]. By setting $\mathbf{\Phi}(\mathbf{x})$ to be the sequence $\left(\sqrt{\lambda_n}\phi_n(\mathbf{x})\right)_n$, we see that K is a dot product in a Hilbert space of sequences (called the feature space), that is, $K(\mathbf{x}, \mathbf{t}) = \mathbf{\Phi}(\mathbf{x}) \cdot \mathbf{\Phi}(\mathbf{t})$. Thus, function in Eq. (5) is a linear combination of features, $f(x) = \sum_{n=1}^{\infty} w_n \phi_n(x)$. The great advantage of working with the compact representation in Eq. (5) is that we avoid directly estimating the the infinite parameters w_n.

Finally, note that kernel K can also be defined/built by choosing the ϕs and λs but, in general, those do not need to be known and can be implicitly defined by K.

4.2 Bounds on the Leave-One-Out Error

We first introduce some more notation. We define the multiclass margin (see also [1]) of point $(\mathbf{x}, y)$ to be

$$g(\mathbf{x}, y) = -d(\mathbf{M}_y, \mathbf{f}(\mathbf{x})) + d(\mathbf{M}_{r(\mathbf{x},y)}, \mathbf{f}(\mathbf{x}))$$

with

$$r(\mathbf{x}, y) = \mathrm{argmin}_{r \ne y} d(\mathbf{M}_r, \mathbf{f}(\mathbf{x})).$$

When $g(\mathbf{x}, y)$ is negative, point $\mathbf{x}$ is misclassified. Notice that when $Q = 2$ and and L is the linear loss, $g(\mathbf{x}, y)$ reduces to the definition of margin for a binary real valued classification function. The empirical misclassification error can be written as:

$$\frac{1}{\ell}\sum_{i=1}^{\ell} \theta\left(-g(\mathbf{x}_i, y_i)\right).$$

The leave-one-out (LOO) error is defined by

$$\frac{1}{\ell}\sum_{i=1}^{\ell} \theta\left(-g^i(\mathbf{x}_i, y_i)\right)$$

where we have denoted by $g^i(\mathbf{x}_i, y_i)$ the margin of example $\mathbf{x}_i$ when the ECOC is trained on the dataset $D_\ell \backslash \{(\mathbf{x}_i, y_i)\}$. The LOO error is an interesting quantity to look at when we want to select/find the optimum (hyper)parameters of a classification function, as is an almost unbiased estimator for the test error and we may expect that his minimum will be close to the minimum of the test error. Unfortunately, computing the LOO error is time demanding when ℓ is large. This becomes practically impossible in the case that we need to know the LOO error for several values of the parameters of the machine used.

In the following theorem we give a bound on the LOO error of ECOC of kernel machines. An interesting property of this bound is that it only depends on the solution of the machine trained on the full dataset (so training the machine once will suffice). Below we denote by f_s the $s-$machine, $f_s(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i^s m_{y_i s} K^s(\mathbf{x}_i, \mathbf{x})$, and let $G_{ij}^s = K^s(\mathbf{x}_i, \mathbf{x}_j)$.

Theorem 1 *Suppose the decoding function uses the linear loss function. Then, the LOO error of the ECOC of kernel machines is bounded by*

$$\frac{1}{\ell}\sum_{i=1}^{\ell} \theta\left(-g(\mathbf{x}_i, y_i) + \max_{t \neq y_i} U_t(\mathbf{x}_i)\right) \qquad (8)$$

where we have defined the function

$$U_t(\mathbf{x}_i) = (\mathbf{M}_t - \mathbf{M}_r)\cdot \mathbf{f}(\mathbf{x}_i) + \sum_{s=1}^{S} m_{y_i s}(m_{y_i s} - m_{ts})\alpha_i^s G_{ii}^s$$

To prove this theorem we first need the following Lemma for binary kernel machines.

Lemma 4.1 *Let $f(\mathbf{x})$ be the kernel machine as defined in Equations (5) and (6). Let $f^i(\mathbf{x})$ be the solution of (4) found when the data point $(\mathbf{x}_i, c_i)$ is removed from the training set. We have*

$$c_i f(\mathbf{x}_i) - \alpha_i G_{ii} \leq c_i f^i(\mathbf{x}_i) \leq c_i f(\mathbf{x}_i). \qquad (9)$$

Proof: The l.h.s part of Inequality (9) was proved in [9]. Note that, if $\mathbf{x}_i$ is not a support vector, $f = f^i$ and we have a trivial inequality. Thus suppose that $\mathbf{x}_i$ is a support vector. To prove the r.h.s. inequality we observe that: $H[f; D_\ell] \leq H[f^i; D_\ell]$, and $H[f; D_\ell^i] \geq H[f^i; D_\ell^i]$. By subtracting the second bound from the first one and using the definition of H we obtain that $V(c_i f(\mathbf{x}_i)) \leq V(c_i f^i(\mathbf{x}_i))$. Then, the result follows by the monotonically property of V. $\square$

We now turn to the proof of Theorem 1. Our goal is to bound the difference of the multiclass margin of $\mathbf{f}$ and $\mathbf{f}^i$, $I = g(\mathbf{x}_i, y_i) - g^i(\mathbf{x}_i, y_i)$. Let us apply Lemma 4.1 to each SVM trained in the ECOC procedure. Inequality (9) can be rewritten as

$$m_{ts} f_s^i(\mathbf{x}_i) = m_{ts} f_s(\mathbf{x}_i) - \lambda_s m_{y_i s} m_{ts} \qquad (10)$$

where λ_s is a parameter in $[0, \alpha_i^s G_{ii}^s]$. Using this equation we can transform quantity I to the desired result through the following steps, where we have defined $r^i = \mathrm{argmin}_{t \neq y_i} d(\mathbf{M}_q, \mathbf{f}^i(\mathbf{x}))$:

$$\begin{aligned} I &= \mathbf{M}_{y_i}\cdot \mathbf{f}(\mathbf{x}_i) - \mathbf{M}_r \cdot \mathbf{f}(\mathbf{x}_i) - \left[\mathbf{M}_{y_i}\cdot \mathbf{f}^i(\mathbf{x}_i) - \mathbf{M}_{r^i}\cdot \mathbf{f}^i(\mathbf{x}_i)\right] \\ &= \mathbf{M}_{y_i}\cdot \mathbf{f}(\mathbf{x}_i) - \mathbf{M}_{y_i}\cdot \mathbf{f}^i(\mathbf{x}_i) + \left[\mathbf{M}_{r^i}\cdot \mathbf{f}^i(\mathbf{x}_i) - \mathbf{M}_r \cdot \mathbf{f}(\mathbf{x}_i)\right] \\ &= \sum_{s=1}^{S}(M_{y_i s})^2 \lambda_s + (\mathbf{M}_t - \mathbf{M}_r)\cdot \mathbf{f}(\mathbf{x}_i) - \sum_{s=1}^{S} m_{y_i s} M_{r^i s}\lambda_s \\ &= (\mathbf{M}_{r^i} - \mathbf{M}_r)\cdot \mathbf{f}(\mathbf{x}_i) + \sum_{s=1}^{S} m_{y_i s}(m_{y_i s} - m_{r^i s})\lambda_s \end{aligned}$$

Then, bound in Eq. (8) follows by using the definition of r^i and by setting $\lambda_s = \alpha_i^s G_{ii}^s$. $\square$

This theorem enlightens some interesting properties of the ECOC of kernel machines. First, observe that the larger the margin of an example is, the less likely that this point will be counted as a LOO error. Second, if the number of support vectors of each kernel machine is small, say less than ν, the LOO error will be at most $S\frac{\nu}{\ell}$. Therefore, if $S\nu \ll \ell$ the LOO error will be small and so will be the test error.

We also notice that, although bound in Eq. (8) only uses the knowledge about the machine trained on the full dataset, it gives an estimate close to the LOO error when the parameter C used to train the kernel machine is "small". Small in this case means that $C \sup_{i=1}^{\ell} K(\mathbf{x}_i, \mathbf{x}_i) < 1$.

4.3 Model Selection Experiments

We now show experiments where we use the bound provided by Theorem 1 to select optimal kernel parameters. We focused on four of the largest datasets and searched the best value of the variance σ of the Gaussian kernel. To simplify the problem we searched a common value for all the binary classifiers. Figures 1–3 show the test error and our LOO estimate for different values of $\gamma = \frac{1}{\sqrt{2\sigma^2}}$ for the three ECOC schemes considered.

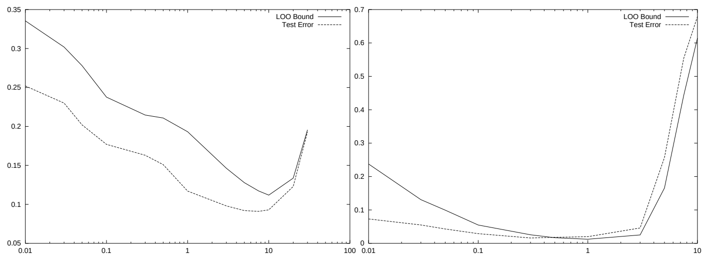

Figure 1. *One-vs-all: Test error and LOO error bound as a function of the logarithm of the parameter γ for satimage (left) and optdigits (right) datasets.*

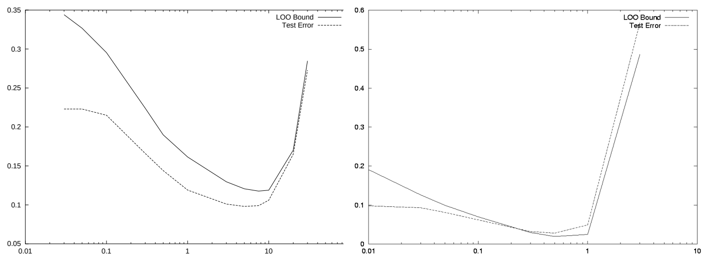

Figure 2. *All-pairs: Test error and LOO error bound as a function of the logarithm of the parameter γ for satimage (left) and optdigits (right) datasets.*

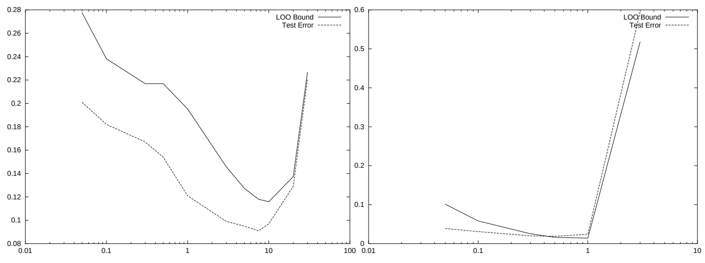

Figure 3. *Dense-codes: Test error and LOO error bound as a function of the logarithm of the parameter γ for satimage (left) and optdigits (right) datasets.*

Results indicate that the minimum of our LOO estimate is very close to the minimum of the test error, although we often observed a slight bias towards smaller values of the variance. Experiments with all ECOC schemes were repeated with such optimal parameters. Tables 5–7 show that we can improve performance significantly. Notice that in the case of one-vs-all and dense codes the advantage of using likelihood decoding is less evident than in experiments in section 3. We conjecture that this may be due to the fact that likelihood estimates are computed on a 3-fold cross validation, thus the optimal value of the variance should be computed separately for each fold. Moreover the bias issues discussed above may further affect such estimates. We will investigate such problems in future experiments.

Table 5. One-vs-all: Optimizing the variance of the Gaussian kernel

Dataset	Hamming	Linear	Soft-margin	Likelihood
Optdigits	79.1	**98.1**	**98.1**	**98.2**
Pendigits	95.5	**98.1**	**98.1**	**98.0**
Satimage	82.9	**90.8**	**90.8**	**91.0**
Segment	89.1	**94.8**	**94.8**	**94.8**

Table 6. All-pairs: Optimizing the variance of the Gaussian kernel

Dataset	Hamming	Linear	Soft-margin	Likelihood
Optdigits	97.9	97.2	97.2	**98.0**
Pendigits	97.7	96.9	97.2	**98.1**
Satimage	**90.6**	90.2	**90.7**	**90.9**
Segment	94.2	94.3	94.3	**95.3**

Table 7. Dense-codes: Optimizing the variance of the Gaussian kernel

Dataset	Hamming	Linear	Soft-margin	Likelihood
Optdigits	97.3	97.6	97.6	**98.1**
Pendigits	**98.0**	**97.9**	**97.9**	**98.1**
Satimage	89.6	**90.4**	**90.4**	**90.7**
Segment	93.8	**94.8**	**94.8**	**94.9**

5 Conclusions

We studied ECOC constructed on margin based binary classifiers under two complementary perspectives: the use of conditional probabilities for building a decoding function, and the use of a theoretically estimated bound on the leave-one-out error for optimizing kernel parameters. Our experiments show that transforming margins into conditional probabilities is very effective and improves the overall accuracy of multiclass classification in comparison to standard loss-based decoding schemes. Moreover, fitting Gaussian kernel parameters by means of our theoretical bound further improves classification accuracy of ECOC machines.

REFERENCES

[1] E. L. Allwein, R. E. Schapire, and Y. Singer, 'Reducing multiclass to binary: A unifying approach for margin classifiers', in *Proc. 17th International Conf. on Machine Learning*, pp. 9–16. Morgan Kaufmann, San Francisco, CA, (2000).

[2] K. Crammer and Y. Singer, 'the learnability and design of output codes for multiclass problems', in *In Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, 2000.*, (2000).

[3] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[4] F. Cuker and S. Smale, 'On the mathematical foundations of learning', *Bulletin of American Mathematical Society*, **39**(1), 1–49, (2001).

[5] T. G. Dietterich and G. Bakiri, 'Solving multiclass learning problems via error-correcting output codes', *Journal of Artificial Intelligence Research*, (1995).

[6] T. Evgeniou, M. Pontil, and T. Poggio, 'Regularization networks and support vector machines', *Advances in Computational Mathematics*, **13**, 1–50, (2000).

[7] Jerome H. Friedman, 'Another approach to polychotomous classification', Technical report, Department of Statistics, Stanford University, (1997).

[8] Y. Guermeur, A. Elisseeff, and H. Paugam-Mousy, 'A new multi-class svm based on a uniform convergence result', in *Proceedings of IJCNN - International Joint Conference on Neural Networks*. IEEE, (2000).

[9] T. Jaakkola and D. Haussler, 'Exploiting generative models in discriminative classifiers', in *Proc. of Neural Information Processing Conference*, (1998).

[10] J. Platt, 'Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods', in *Advances in Large Margin Classifiers, A. Smola et al. Eds*, MIT Press, (1999).

[11] B. Scholkopf, C. Burges, and A. Smola, *Advances in Kernel Methods – Support Vector Learning*, MIT Press, 1998.

[12] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[13] G. Wahba, *Splines Models for Observational Data*, Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.