On Tuning Hyper-Parameters of Multiclass Margin Classifiers

Andrea Passerini[1] Massimiliano Pontil[2]
Paolo Frasconi[1]
[1] *DSI, Università di Firenze, via di Santa Marta 3, 50139 Firenze*
[2] *DII, Università di Siena, via Roma 56, 53100 Siena*

Abstract. The choice of hyper-parameters (e.g. kernel parameters) can significantly affect generalization performance of large margin classifiers. In this paper we are concerned with the problem of tuning these values in the case of multi-class problems that have been recast into a set of binary problems. We report several experimental results comparing independent and joint tuning of the hyper-parameters of the binary classifiers. Several different encoding strategies are explored, including error correcting output codes. Tuning was carried out by using a validation set and a newly introduced bound on the leave-one-out error.

Keywords: Machine Learning, Kernel Machines, Tuning Hyper-parameters, Error Correcting Codes.

1 Introduction and Notation

Often a multiclass categorization learning problem is reduced to a set of binary learning problems. This approach enables the application of learning tools that are intrinsically conceived for binary classification. Binary reduction can be practically effected by associating codes to the classes of interest. In this context, variable-length codes lead to architectures based on decision trees. Fixed-length codes are more commonly used and are typically based on a redundant number of bits in order to gain robustness with respect to errors made by single classifiers. In particular, error correcting output codes (ECOC) have been introduced by Dietterich and Bakiri [6] and more recently extended in [1]. In this paper we focus on fixed-length codes and binary classifiers that can be characterized through hyper-parameters, i.e. parameters that affect the solution of the learning problem but that are normally considered as constants if the solution is obtained through the numerical optimization of some criterion [2]. Let us assume $Q > 2$ classes and S binary classifiers, each trained on one of S dichotomies of the instance space, formed by joining non overlapping subsets of classes. A "coding matrix" $M \in \{-1, 0, 1\}^{Q \times S}$ specifies the relation between classes and dichotomies. Here, $m_{qs} = 1$ ($m_{qs} = -1$) 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 raw of matrix M which we also denoted by $\mathbf{M}_q$. 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 $\mathbf{x}$'s class is computed as

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

where d a decoding function (some examples of decoding functions are given later). Suppose the output of the s-th classifier depends on some hyper-parameters θ_s (for example the radius in a radial basis function kernel), and let $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_S)$. Let $e_s(\theta_s)$ denote an estimate of the true error of the s-th classifier, and $e(\boldsymbol{\theta})$ denote an estimate of the true error of the multiclass classifier that combines $f_1(\cdot;\theta_1), \ldots, f_S(\cdot;\theta_S)$. The main question in this paper is whether tuning hyper-parameters by globally optimizing e with respect to $\boldsymbol{\theta}$ gives better results than optimizing each single e_s with respect to θ_s.

Our study is focused on support vector machine classifiers and the use of different alternative loss functions for decoding. In [6] the entries of matrix M were restricted to take only binary values and the d was chosen to be the Hamming distance:

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

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 [11] we have suggested 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 classification confidence. The approach is summarized in the next section.

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 showed in experiments reported in [11] 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

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

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

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

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 [12]:

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

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 Tuning Kernel Parameters

In this section we study the problem of model selection in the case of ECOC of Kernel Machines [14, 13, 7, 4]. 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 [7].

3.1 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 [14, 15] 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[1]

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

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:

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

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 $\lambda_n \geq 0$ their eigenvalues [5]. 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 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.

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

3.2 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

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

When $g(\mathbf{x}, y)$ is negative, point $\mathbf{x}$ is misclassified. Notice that when $Q = 2$ and 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 $G_{ij}^s = K^s(\mathbf{x}_i, \mathbf{x}_j)$. The following result was proved in [11]:

Theorem 3.1. *Suppose the decoding function uses the linear loss function. Then, the LOO error of the ECOC of kernel machines is bounded by*

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

where we have defined the function

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

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

Table 1: Characteristics of the Datasets used

Name	Classes	Train	Test	Inputs
Optdigits	10	3823	1797	64
Satimage	6	4435	2000	36
Segment	7	1540	770	19

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 Comparison of local and global tuning

In order to compare local and global tuning of hyper-parameters we realized a set of experiments using three datasets from the UCI repository. Their characteristics are shortly summarized in Table 1.

We trained multiclass classifiers using Gaussian Kernel SVM as the base binary classifier, with Gaussian variance as the hyper-parameter to be tuned. The C parameter was fixed to the same value for all the experiments. In order to reduce the search space we looked for a common value of the variance for all SVMs of the multiclass classifier. We found such a value by dividing the training set in two thirds for training and one third for validating and choosing the best value of the variance according to some estimate of the generalization error computed on the validation set. In local tuning such estimate was the average of the accuracies of each binary classifier, while in global tuning we used the multiclass accuracy. Comparison between local and global tuning was done via accuracy computed on the test set. Furthermore, in order to compare the two methods for smaller sizes of the training set, we studied the learning curves obtained by randomly choosing 10/30/50/70/90 percent of the training set and using it for the training (and validation) phase and testing on the full test set. We repeated such a procedure five times and averaged the results. Figures 1–4 show global (solid line) and local (dashed line) tuning learning curves for different coding matrix (one against all, all pairs, dense codes) and decoding function (hamming[2], linear loss, SVM soft margin, our likelihood loss). Results do not offer a clear evidence of which tuning method is better. However, global tuning seems to have some advantages in the cases of one-vs-all and all-pairs. In addition the behavior of the learning curves appears to be quite insensitive to the decoding function employed.

A natural question is whether two multiclassifiers having similar accuracy also have strongly correlated errors. To investigate this issue we used an oracle that can always choose the best option between the predictions of the two multiclassifiers. Such a classifier, represented by the dotted line in figures 1–4, obtains great improvements over the single methods for all coding matrices, thus showing that local and global errors are often not correlated and a weighted mixture of the two approaches could lead to classification improvements.

[2]Not reported for one against all coding matrix because ties often occur in this case

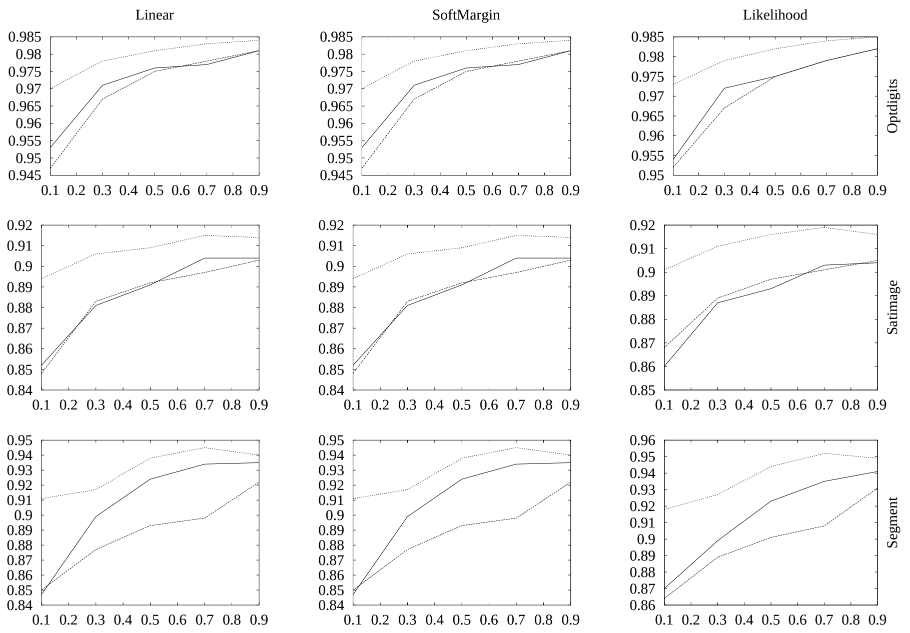

Figure 1: Tuning via validation set with one-against-all coding matrix for different datasets (rows) and decoding functions (columns). Learning curves represent global tuning (solid line), local tuning (dashed line) and oracle (dotted line).

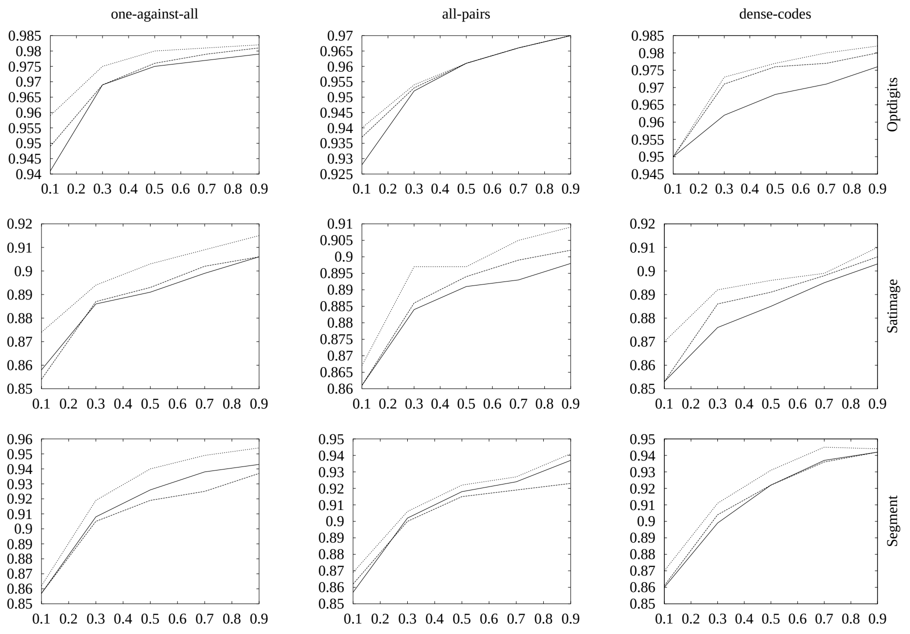

Figure 2: Tuning via LOO error estimate for different datasets (rows) and coding matrices (columns) with linear decoding function. Learning curves represent global tuning (solid line), local tuning (dashed line) and oracle (dotted line).

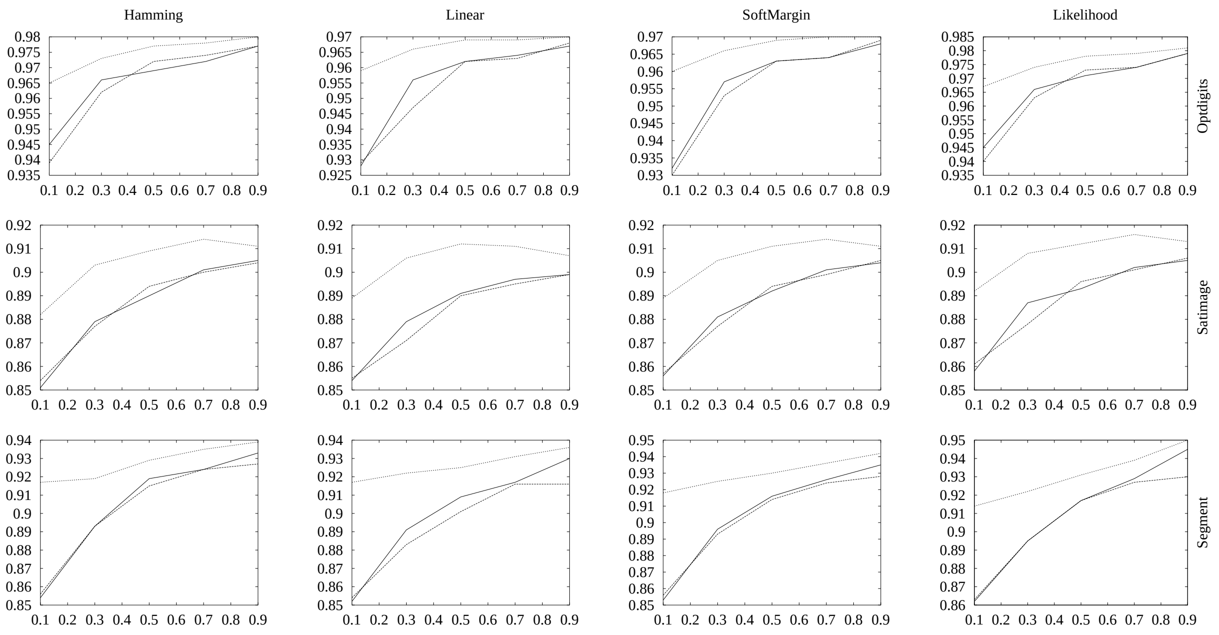

Figure 3: Tuning via validation set with all-pairs coding matrix for different datasets (rows) and decoding functions (columns). Learning curves represent global tuning (solid line), local tuning (dashed line) and oracle (dotted line).

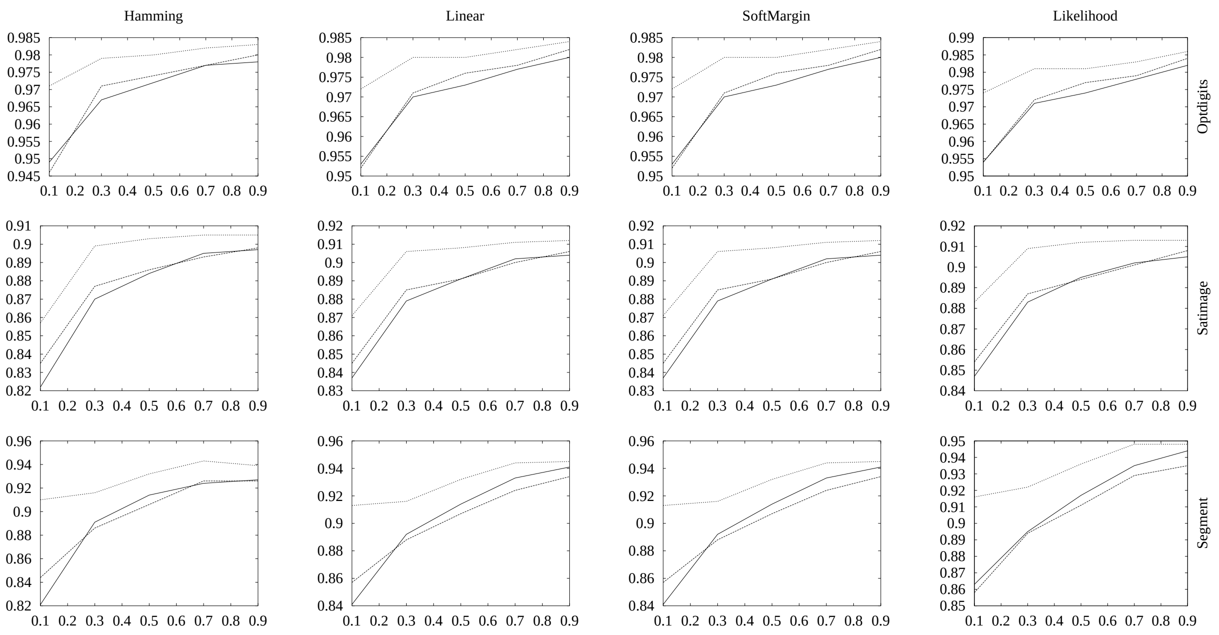

Figure 4: Tuning via validation set with dense-codes coding matrix for different datasets (rows) and decoding functions (columns). Learning curves represent global tuning (solid line), local tuning (dashed line) and oracle (dotted line).

We repeated the whole set of experiments tuning the parameters through a LOO error estimate. Similarly to the validation case, in local tuning we used the average of the LOO error estimates of each binary classifier, while in global tuning we used the multiclass LOO error estimate in equation 8. Figure 2 show these learning curves for different coding matrices for the linear decoding distance. Results indicate that the averaged binary bound is usually better than the multiclass one. We argue this might be due to the complexity of the multiclass bound that could make it a worse estimate of the true LOO error.

5 Conclusions

We presented experimental comparisons of hyperparameter tuning for multiclass margin classifiers. Results show that global tuning can offer advantages when a validation set is used and simple schemes like one-against-all and all-pairs are used. Our experiments also show that transforming margins into conditional probability improves the overall accuracy of multiclass classification and should be preferred to standard loss-based decoding schemes. We have also discussed a newly introduced analysis on the leave-one-out error of ECOC of kernel machine classifiers. In this case experiments show that a simple average of binary LOO error estimate is usually more effective than multiclass LOO error estimate. We conjecture this is due to the greater complexity of the multiclass bound, and we are currently investigating how to refine such estimate.

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] Y. Bengio, ‘Gradient-Based Optimization of Hyper-Parameters’, *Neural Computation* **12**:8, pp. 1889–1900, (2000).

[3] 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).

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

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

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

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

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

[9] 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).

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

[11] A. Passerini, M. Pontil and P. Frasconi, ‘From Margins to Probabilities in Multiclass Learning Problems’, in F. van Harmelen (ed.), *Proc. 15th European Conf. on Artificial Intelligence*, (2002).

[12] 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).

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

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

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