Multiclass LS-SVMs: Moderated Outputs and Coding-Decoding Schemes

T. VAN GESTEL, J.A.K. SUYKENS, G. LANCKRIET,
A. LAMBRECHTS, B. DE MOOR, J. VANDEWALLE
Katholieke Universiteit Leuven, Dept. of Electrical Engineering - ESAT/SISTA, Kard. Mercierlaan 94, B-3001 Leuven, Belgium.
E-mail: {tony.vangestel,johan.suykens}@esat.kuleuven.ac.be

Abstract. A common way of solving the multiclass categorization problem is to reformulate the problem into a set of binary classification problems. Discriminative binary classifiers like, e.g., Support Vector Machines (SVMs), directly optimize the decision boundary with respect to a certain cost function. In a pragmatic and computationally simple approach, Least Squares SVMs (LS-SVMs) are inferred by minimizing a related regression least squares cost function. The moderated outputs of the binary classifiers are obtained in a second step within the evidence framework. In this paper, Bayes' rule is repeatedly applied to infer the posterior multiclass probabilities, using the moderated outputs of the binary plug-in classifiers and the prior multiclass probabilities. This Bayesian decoding motivates the use of loss function based decoding instead of Hamming decoding. For SVMs and LS-SVMs with linear kernel, experimental evidence suggests the use of one-versus-one coding. With a Radial Basis Function kernel one-versus-one and error correcting output codes yield the best performances, but simpler codings may still yield satisfactory results.

Keywords: Bayesian Decoding, Discriminant Analysis, Evidence Framework, Hamming Decoding, Multiclass Classification, Regression, Support Vector Machines

1. Introduction

Multiclass categorization is a general task in pattern recognition with important applications like, e.g., text and speech recognition. The task of an M-class classifier is to predict the class label $\mathcal{C}_m$, $m = 1, ..., M$, given a new input vector $x \in \mathbb{R}^n$. A popular way to solve the M-class problems is to reformulate the problem into a set of L binary classification problems [1, 7, 9, 14, 16, 18, 19]. A first possibility is to construct $M(M-1)/2$ one-versus-one (1vs1) binary classifiers, each classifier discriminating between each pair of 2 classes [10, 14]. An alternative approach [16] is to represent each class $\mathcal{C}_m$, $m = 1, ..., M$, by a unique binary output codeword $c_m \in \{-1, +1\}^L$ of L bits. Then L binary classifiers are trained to discriminate between two opposing subsets with different output bits. In [18] a minimal output coding (MOC) has been applied to solve the multiclass problem with binary Least Squares Support Vector Machines (LS-SVMs), using L bits to

© 2001 *Kluwer Academic Publishers. Printed in the Netherlands.*

encode up to 2^L classes. One-versus-all (1vsA) coding [1, 6] uses $L = M$ bits by putting the mth bit of the codeword c_m equal to $+1$, while all other bits of c_m are equal to 0 or -1, depending on the type of coding. Error correcting output codes (ECOC) [7] use more bits than MOC in order to increase the Hamming distance between the M codewords, allowing for one or more misclassifications of the binary classifiers. Finally, notice that one can represent the 1vs1 output coding by L-bit codewords when one uses a 'don't care' symbol ($\times$) for the classes that are not considered.

Several decoding schemes exist in order to assign the multiclass label to a new input x. Hamming decoding [7, 16] computes the output vector $y \in \{-1, +1\}^L$ as the sign of the outputs of the L binary discriminants. The class label $\mathcal{C}_m$ is then assigned to the corresponding codeword c_m with minimal Hamming distance to the output vector y. In [1], it is argued that this method ignores the used loss function and the related confidences. Instead, a loss function based decoding has been proposed, selecting the codeword with minimal loss function. Other decoding schemes are related to a specific output coding. In 1vsA output coding, the class label is assigned to the class $\mathcal{C}_m$, where m is the binary classifier generating the largest output. For 1vs1 coding, one uses a max-wins criterion or a directed acyclic graph [14].

In this paper, we use binary LS-SVM plug-in classifiers with linear and Radial Basis Function (RBF) kernels [17, 18] within the evidence framework [3, 8, 11, 12, 20, 21]. Each binary discriminant function is optimized using a least squares regression cost function, while its binary class probabilities are inferred in a second step within the related probabilistic framework. Bayes' rule is then applied L times to infer posterior multiclass probabilities, using the prior multiclass probabilities and the posterior binary class probabilities. The Bayesian decoding is related to loss based decoding [1] and is compared with Hamming decoding on ten multiclass datasets. The evidence framework can also be applied to other classifiers [3, 8, 11] in a straightforward way.

This paper is organized as follows. In Section 2, the probabilistic interpretation of LS-SVM classifiers is reviewed. The Bayesian decoding scheme is derived in Section 3. Design and decoding algorithms are discussed in Section 4. Experimental results are given in Section 5.

2. Moderated Outputs of Binary LS-SVMs

In this Section, the binary LS-SVM classifier [17] is reviewed within the evidence framework [3, 11, 12, 20, 21]. Each output coding uses L binary plug-in classifiers to discriminate between the opposing subsets.

2.1. LS-SVM Classifier

In SVMs [5, 17, 22] the binary classifier takes the form

$$y = \text{sign}[w^T\varphi(x) + b], \tag{1}$$

where the nonlinear function $\varphi(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n_f}$ maps the input space to a higher n_f dimensional feature space, which possibly may be infinite dimensional. However, the SVM classifier is never evaluated in this form. As explained in [5, 22], the standard SVM classifier is obtained by solving a quadratic programming problem in the dual space.

In order to obtain a linear set of equations in the dual space, the SVM classifier formulation was modified basically as follows in [17]:

$$\min_{w,b,e} \mathcal{J}(w,e) = \tfrac{\mu}{2} w^T w + \tfrac{\zeta}{2} \textstyle\sum_{i=1}^N e_i^2 \tag{2}$$

subject to the equality constraints

$$y_i[w^T\varphi(x_i) + b] = 1 - e_i, \qquad i = 1, ..., N. \tag{3}$$

Besides the quadratic cost function, an important difference with standard SVMs is that the formulation consists now of equality instead of inequality constraints. The hyperparameters μ and ζ are related to the regularization term $E_W = \frac{1}{2} w^T w$ and the error term $E_D = \frac{1}{2}\sum_{i=1}^N e_i^2$, respectively. As in standard SVMs, the Lagrangian is constructed and one obtains the following set of linear equations in the Lagrange multipliers $\alpha_i \in \mathbb{R}$ $(i = 1, ..., N)$ and the bias term $b \in \mathbb{R}$:

$$\left[\begin{array}{c|c} 0 & Y^T \\ \hline Y & \Omega + \frac{\mu}{\zeta} I_N \end{array}\right] \left[\begin{array}{c} b \\ \hline \alpha \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 1_v \end{array}\right] \tag{4}$$

with $Y = [y_1; ...; y_N]$, $1_v = [1; ...; 1]$, $e = [e_1; ...; e_N]$, $\alpha = [\alpha_1; ...; \alpha_N]$ and where Mercer's condition is applied within the Ω matrix: $\Omega_{ij} = y_i y_j\, \varphi(x_i)^T \varphi(x_j) = y_i y_j\, K(x_i, x_j)$. For the kernel function $K(\cdot,\cdot)$ one has, e.g., the following choices: $K(x,x_i) = x_i^T x$ (linear kernel), $K(x, x_i) = (x_i^T x + 1)^d$ (polynomial kernel of degree d), $K(x, x_i) = \exp\{-\|x - x_i\|_2^2/\sigma^2\}$ (RBF-kernel), where $d \in \mathbb{N}$ and $\sigma \in \mathbb{R}^+$. Notice that the Mercer condition holds for all σ and d values in the RBF, resp. the polynomial case, see [5, 17, 22] for details. The LS-SVM classifier is then constructed as follows:

$$y(x) \; = \; \text{sign}[\textstyle\sum_{i=1}^N \alpha_i y_i K(x, x_i) + b], \tag{5}$$

where $z_{MP} = \sum_{i=1}^N \alpha_i y_i K(x, x_i) + b$ by definition.

Observe that the LS-SVM formulation (2)-(3) implicitly corresponds to a regression formulation [3, 8, 15, 17]. Indeed, by multiplying the error e_i with $y_i \in \{-1, +1\}$, the error term E_D becomes

$$E_D = \tfrac{1}{2}\textstyle\sum_{i=1}^N e_i^2 = \tfrac{1}{2}\sum_{i=1}^N (y_i e_i)^2 = \tfrac{1}{2}\sum_{i=1}^N (y_i - (w^T\varphi(x_i) + b))^2. \tag{6}$$

This relates the LS-SVM formulation to Gaussian Processes [23] and to Kernel Fisher Discriminant Analysis [2, 13].

2.2. Bayesian Interpretation of LS-SVMs

We will now relate a probabilistic framework to the binary LS-SVM classifier. Applying Bayes' rule at the first level of inference [3, 11, 12], we obtain the posterior probability for $[w;\, b]$:

$$P(w,b|D,\mu,\zeta,\mathcal{H}) \;=\; \tfrac{P(D|w,b,\mu,\zeta,\mathcal{H})P(w,b|\mu,\zeta,\mathcal{H})}{P(D|\mu,\zeta,\mathcal{H})}. \tag{7}$$

The prior $P(w,b|\mu,\zeta,\mathcal{H}) = P(w,b|\mu,\mathcal{H})$ and likelihood $P(D|w,b,\mu,\zeta,\mathcal{H}) = P(D|w,b,\zeta,\mathcal{H})$ are independent of ζ and μ, respectively. The evidence $P(D|\mu,\zeta,\mathcal{H})$ is a normalizing constant.

It is assumed that the prior is equal to $P(w,b|\mu,\mathcal{H}) = P(w|\mu,\mathcal{H})P(b|\mathcal{H})$, with an uninformative prior $P(b)$ on the bias term b and where

$$P(w|\mu,\mathcal{H}) = \left(\tfrac{\mu}{2\pi}\right)^{\frac{n_f}{2}} \exp(-\tfrac{\mu}{2}||w||_2^2), \tag{8}$$

which corresponds to the regularization term $\frac{\mu}{2}w^Tw$ in (2). The error term $\frac{\zeta}{2}\sum_{i=1}^N e_i^2$ corresponds to the likelihood

$$\begin{aligned} P(D|w,b,\zeta,\mathcal{H}) &= \textstyle\prod_{i=1}^N P(x_i,y_i|w,b,\zeta,\mathcal{H}) \\ &= \textstyle\prod_{i=1}^N P(x_i|y_i,w,b,\zeta,\mathcal{H})P(y_i|w,b,\zeta,\mathcal{H}) \\ &= \textstyle\prod_{i=1}^N P(y_i|w,b,\mathcal{H}) \left(\tfrac{\zeta}{2\pi}\right)^{1/2} \exp(-\tfrac{\zeta}{2}e_i^2), \end{aligned} \tag{9}$$

with $P(x_i|y_i,w,b,\zeta,\mathcal{H}) = (\frac{\zeta}{2\pi})^{1/2}\exp(-\frac{\zeta}{2}e_i^2)$ and where $P(y_i|w,b,\mathcal{H}) = P(y_i)$ is the prior class distribution of y_i. The maximum posterior weights w_{MP} and bias term b_{MP} are obtained by minimizing the negative logarithm of (7). Substituting (8) and (9) into (7) and neglecting all constants, this corresponds to minimizing (2).

The posterior class probability is assigned to a new input x within the above defined probabilistic framework by applying Bayes' rule [3, 12, 20, 21]:

$$P(y|x,D,\mu,\zeta,\mathcal{H}) = \tfrac{P(x|y,D,\mu,\zeta,\mathcal{H})P(y)}{\sum_{y\in\{-1,+1\}} P(x|y,D,\mu,\zeta,\mathcal{H})P(y)}. \tag{10}$$

The probability $P(x|y,D,\mu,\zeta,\mathcal{H})$ corresponds to the likelihood (9), adjusted for an extra variance σ_z^2 [3, 11, 12, 20, 21]:

$$P(x|y,D,\mu,\zeta,\mathcal{H}) = \left(2\pi(\tfrac{1}{\zeta}+\sigma_z^2)\right)^{-1/2} \exp(-\tfrac{(1-yz_{MP}))^2}{2(\frac{1}{\zeta}+\sigma_z^2)}). \tag{11}$$

The variance σ_z^2 is due to the posterior uncertainty on $[w_{MP};\, b_{MP}]$ and is equal to [3, 11, 12] $\sigma_z^2 = g^TH^{-1}g$, where $g = \frac{\partial z(x;[w;b])}{\partial [w;b]}|_{x,[w_{MP};b_{MP}]} =$

$[\varphi(x);\, 1]$ is the sensitivity of the output z to the parameters $[w;b]$ and where H is the Hessian $H = \frac{\partial^2 \mathcal{J}}{(\partial[w;b])^2}$ of the cost function (2). The probabilistic output $P(y|x, D, \mu, \zeta, \mathcal{H})$ is called the moderated output of the LS-SVM classifier. It is easily observed that when $P(y{=}{+}1) = P(y{=}{-}1)$, the class label is assigned to the most likely class as $y = \text{sign}[z_{MP}]$.

A key idea of SVMs is that the mapping $\varphi(x)$ is never explicitly calculated. By applying the Mercer condition, the following expression for the variance σ_z^2 in the dual space is obtained [20, 21]:

$$\begin{aligned} \sigma_z^2 \;=\; & \theta(x)^T H_D \theta(x) + \tfrac{1}{\mu} K(x,x) + \tfrac{1}{N\zeta} - \tfrac{2}{N}\theta(x)^T H_D \Omega Y \\ & + \tfrac{2}{N}\mu^{-1}\theta(x)^T Y + \tfrac{1}{N^2} Y^T \Omega H_D \Omega Y + \tfrac{1}{\mu N^2} Y^T \Omega Y. \end{aligned} \tag{12}$$

The vector $\theta(x) \in \mathbb{R}^N$ and the matrices $H_D \in \mathbb{R}^{N_{eff} \times N_{eff}}$, $U_G \in \mathbb{R}^{N \times N_{eff}}$ and $D_G \in \mathbb{R}^{N_{eff} \times N_{eff}}$ are defined as follows: $\theta_i(x) = y_i K(x, x_i)$, $i = 1, ..., N$, $H_D = U_G[(\mu I_{N_{eff}} + \zeta D_G)^{-1} - \mu^{-1} I_{N_{eff}}]U_G^T$, $U_G(:,i) = (\nu_{G,i}\Omega\nu_{G,i})^{-1/2}\,\nu_{G,i}$, $i = 1, ..., N_{eff} \leq N-1$, and $D_G = \text{diag}([\lambda_{G,1}, ..., \lambda_{G,N_{eff}}])$, where $\nu_{G,i}$ and $\lambda_{G,i}$ are the solutions to the eigenvalue problem

$$(I_N - \tfrac{1}{N} Y Y^T)\Omega \nu_{G,i} \;=\; \lambda_{G,i}\nu_{G,i}, \quad i = 1, ..., N_{eff} \leq N-1. \tag{13}$$

The number of non-zero eigenvalues is denoted by $N_{eff} < N$, see [20, 21] for more details.

The above problem formulation corresponds to the first level of Bayesian inference [11, 12, 20, 21]. By applying Bayes' rule on the second level, the optimal hyperparameters μ_{MP} and ζ_{MP} are inferred from the data D. Model comparison is performed on the third level, e.g., to tune the parameter σ of an RBF-kernel [11, 12, 20, 21].

3. Bayesian Decoding

We will now repeatedly apply Bayes' rule to estimate the posterior probability of the codeword $c_m = [y_m^{(1)}; ...; y_m^{(l)}; ...; y_m^{(L)}]$ of the corresponding class $\mathcal{C}_m$, $m = 1, ..., M$. It is assumed that the moderated outputs of the L binary classifiers $P(y^{(l)}|x, D^{(l)}, \mu^{(l)}, \zeta^{(l)}, \mathcal{H}^{(l)})$, $l = 1, ..., L$, are known, e.g., from (10) and (11) [3, 8, 12, 11, 20, 21]. The superscript (l) is used to denote all parameters related to the lth binary classifier. For convenience, we will use the notation of $\mathcal{H}_{\mu,\zeta}^{(l)}$ to denote the binary classifier of output bit $y^{(l)}$ with model structure $\mathcal{H}^{(l)}$ and with corresponding hyper- and model parameters: $P(y^{(l)}|x, \mathcal{H}_{\mu,\zeta}^{(l)}) := P(y^{(l)}|x, D^{(l)}, \mu^{(l)}, \zeta^{(l)}, \mathcal{H}^{(l)})$. We also use the short-hand notations $y^{(i:j)} =$

$[y^{(i)};...;y^{(j)}]$ and $\mathcal{H}_{\mu,\zeta}^{(i:j)} = \{\mathcal{H}_{\mu,\zeta}^{(i)}, ..., \mathcal{H}_{\mu,\zeta}^{(j)}\}$. A key assumption of the Bayesian (and also Hamming [7, 16]) decoding scheme is that the output of the lth binary classifier is independent of the other classifiers $\mathcal{H}_{\mu,\zeta}^{(...,l-1,l+1,...)}$, as will be exploited in the next Subsection.

3.1. Bayesian Decoding with Binary Targets $\{-1,+1\}$

The key step in the Bayesian decoding scheme is the updating of the multiclass probabilities $P(c_m|x,\mathcal{H}_{\mu,\zeta}^{(0:l-1)})$ of the codewords c_m, $m = 1, ..., M$ by applying Bayes' rule for each output bit $y_m^{(l)}$, $l = 1, ..., L$.

The updating starts from the prior multiclass probabilities $P(c_m) = P(\mathcal{C}_m) = P(c_m|x, \mathcal{H}_{\mu,\zeta}^{(0)})$, $m = 1, ..., M$. When no prior class probabilities are known, one typically assumes equal prior class distributions, i.e., $P(\mathcal{C}_m) = 1/M$. The notation $P(c_m|x, \mathcal{H}_{\mu,\zeta}^{(0)})$ is introduced to indicate that the Bayesian updating scheme starts from the prior. The introduction of the dummy model $\mathcal{H}_{\mu,\zeta}^{(0)}$ also allows for a uniform notation for all bits $l = 1, ..., L$. We also use the dummy $y_m^{(0)} = \times$, where $y_m^{(0)}$ is introduced for convenience of notation only; e.g., we have $P(y_m^{(0:l)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l)}) = P(y_m^{(1:l)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l)})$ and $P(y_m^{(0)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l)}) = 1$.

Given the likelihood $P(y_m^{(0:L)}|x,\mathcal{H}_{\mu,\zeta}^{(0:l-1)})$, we infer $P(y_m^{(0:L)}|x,\mathcal{H}_{\mu,\zeta}^{(0:l)})$ by first applying the chain rule:

$$\begin{aligned} P(y_m^{(0:L)}|x,\mathcal{H}_{\mu,\zeta}^{(0:l)}) = P(y_m^{(0:l-1)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l)}) \cdot \\ P(y_m^{(l)}|x,y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)}) P(y_m^{(l+1:L)}|x,y_m^{(0:l)}, \mathcal{H}_{\mu,\zeta}^{(0:l)}), \end{aligned} \tag{14}$$

where we assume that the first $l-1$ outputs bits are independent of the model $\mathcal{H}_{\mu,\zeta}^{(l)}$, i.e., $P(y_m^{(0:l-1)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l)}) = P(y_m^{(0:l-1)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l-1)})$. It is also assumed that $P(y_m^{(l+1:L)}|x, y_m^{(0:l)}, \mathcal{H}_{\mu,\zeta}^{(0:l)}) = P(y_m^{(l+1:L)}|x, y_m^{(0:l)}, \mathcal{H}_{\mu,\zeta}^{(0:l-1)})$. The likelihood of the lth bit is computed by applying Bayes' rule as in (10):

$$\begin{aligned} P(y_m^{(l)}|x,y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)}) &= \frac{P(x|y_m^{(l)},y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)}) P(y_m^{(l)}|y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)})}{P(x|y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)})} \\ &= \frac{P(x|y_m^{(l)}, \mathcal{H}_{\mu,\zeta}^{(l)}) P(y_m^{(l)}|y_m^{(0:l-1)}, x, \mathcal{H}_{\mu,\zeta}^{(0:l-1)})}{P(x|y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)})}, \end{aligned} \tag{15}$$

using $P(x|y_m^{(l)},y_m^{(0:l-1)}) = P(x|y_m^{(l)}, \mathcal{H}_{\mu,\zeta}^{(l)})$ and $P(y_m^{(l)}|y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l)}) = P(y_m^{(l)} |y_m^{(0:l-1)}, \mathcal{H}_{\mu,\zeta}^{(0:l-1)}) = P(y_m^{(l)}|y_m^{(0:l-1)}, x, \mathcal{H}_{\mu,\zeta}^{(0:l)})$. The denominator $P(x|y_m^{(0:l-1)}; \mathcal{H}_{\mu,\zeta}^{(0:l)})$ is a normalizing constant as in (10) such that the sum of both probabilities for $y^{(l)} \in \{-1,+1\}$ is equal to one.

Applying the chain rule in the opposite direction, we can write

$$\begin{aligned} P(y_m^{(l+1:L)}|y_m^{(0:l)}, x, \mathcal{H}_{\mu,\zeta}^{(0:l-1)}) P(y_m^{(l)}|y_m^{(0:l-1)}, x, \mathcal{H}_{\mu,\zeta}^{(0:l-1)}) \cdot \\ P(y_m^{(0:l-1)}|x, \mathcal{H}_{\mu,\zeta}^{(0:l-1)}) = P(c_m|x,\mathcal{H}_{\mu,\zeta}^{(0:l-1)}). \end{aligned} \tag{16}$$

By substituting (15) into (14) and using (16), we obtain

$$P(c_m|x,\mathcal{H}_{\mu,\zeta}^{(0:l)}) = \frac{P(x|y_m^{(l)},\mathcal{H}_{\mu,\zeta}^{(l)})P(c_m|x,\mathcal{H}_{\mu,\zeta}^{(0:l-1)})}{P(x|y_m^{(0:l-1)};\mathcal{H}_{\mu,\zeta}^{(0:l)})}, \tag{17}$$

where the normalizing constant $P(x|y_m^{(0:l-1)};\mathcal{H}_{\mu,\zeta}^{(0:l)})$ can be calculated by imposing that $\sum_{c_m} P(c_m|x,\mathcal{H}_{\mu,\zeta}^{(0:l)}) = 1$.

By applying the updating rule (17) for $l = 1,...,L$, the unnormalized posterior multiclass probabilities are obtained as

$$P(c_m|x,\mathcal{H}_{\mu,\zeta}^{(0:L)}) \propto P(c_m)\prod_{l=1}^{L} P(x|y_m^{(l)},\mathcal{H}_{\mu,\zeta}^{(l)}), \tag{18}$$

for $m = 1,...,M$. Taking the negative logarithm of (18), the most probable class corresponds to the class with minimum weighted loss function, adjusted for the prior information $-\log P(c_m)$. Neglecting all constants, this corresponds to selecting the class m for which the weighted squared error

$$-\log P(c_m) + \sum_{l=1}^{L} \frac{(1-y_m^{(l)}z_{MP}^{(l)})^2}{2(\frac{1}{\zeta^{(l)}}+\sigma_{z^{(l)}}^2)} \tag{19}$$

is minimal. When all classes are a priori equally likely and when one considers only codewords related to defined classes $\mathcal{C}_m$, the first term in (19) may be omitted and the decision criterion corresponds to a weighted squared error criterion, taking the accuracy of the different classifiers into account. When all classes have the same prior class probability and when the all weightings $1/\zeta^{(l)} + \sigma_{z^{(l)}}^2$ are equal, the Bayesian decoding scheme corresponds to the loss based decoding proposed in [1]. While it may occur in Hamming decoding that no class label can be assigned when multiple codewords have the same minimal Hamming distance from a given output, this problem will rarely occur in practice in the Bayesian decoding, since (19) is a real-valued criterion. Furthermore, the Bayesian decoding allows to obtain the posterior multiclass probabilities as in the case of the binary classification scheme [11, 12, 20, 21].

3.2. Bayesian Decoding with Don't Cares $\{-1, \times, +1\}$

Bayes' rule is now applied to update the probabilities of the codewords c_m, when the probability $P(x|y^{(l)},\mathcal{H}_{\mu,\zeta}^{(l)})$ of a new bit $y^{(l)} \in \{-1,+1\}$, $l = 1,...,L$ is calculated. The difference with the previous Subsection is that there may be a nonempty set $\mathcal{M}_{\times}^{(l)}$ of classes m for which the lth bit of the corresponding codeword c_m is a don't care: $y_m^{(l)} = \times$. This don't care will cause that the probability of these codewords is not updated[1]. Starting from the initial probability $P(c_m|x,\mathcal{H}^{(0)}_{\mu,\zeta}) = P(c_m)$, the updating rule (17) now becomes

$$P(c_m|x,\mathcal{H}^{(0:l)}_{\mu,\zeta}) = \begin{cases} P(c_m|x,\mathcal{H}^{(0:l-1)}_{\mu,\zeta}), & \text{for } m \in \mathcal{M}^{(l)}_{\times} \\ \frac{P(x|y^{(l)}_m,\mathcal{H}^{(l)}_{\mu,\zeta})P(c_m|x,\mathcal{H}^{(0:l-1)}_{\mu,\zeta})}{\sum_{m\notin\mathcal{M}^{(l)}} P(x|y^{(l)}_m,\mathcal{H}^{(l)}_{\mu,\zeta})P(c_m|x,\mathcal{H}^{(0:l-1)}_{\mu,\zeta})} & \text{for } m \notin \mathcal{M}^{(l)}_{\times}. \end{cases} \quad (20)$$

By applying this updating rule for $l = 1,...,L$, the posterior multiclass probabilities $P(c_m|x,\mathcal{H}^{(0:L)}_{\mu,\zeta})$ are obtained.

4. Algorithms

The M-class classifier design within the evidence framework basically consists of the following steps:

1. Normalize the inputs of the training data to zero mean and unit variance [3]. Assign the codewords $c_m \in \{-1,\times,+1\}^L$ $(m = 1,...,M)$ to the outputs according to the selected output coding.

2. For each output bit $y^{(l)}$ $(l = 1,...,L)$ one constructs the binary classifier, which is done as follows:

   a) For each binary classifier, select the model $\mathcal{H}^{(l)}_j$ with a certain kernel function, possibly with kernel parameter (e.g., σ_j for an RBF-kernel); the index j is used to denote the different kernels and parameters [17, 20, 21].
   b) The optimal hyperparameters $\mu^{(l)}_{MP}, \zeta^{(l)}_{MP}$ of the model $\mathcal{H}^{(l)}_j$ are inferred from the data $D^{(l)} = \{x_i, y^{(l)}_i\}^{N^{(l)}}_{i=1}$, with $y^{(l)}_i \in \{-1,+1\}$, on the second level of inference as described in [12, 20, 21]. The corresponding level 2 cost function is optimized in an iterative way. In each iteration one basically calculates the following steps: i) for given $\mu^{(l)}$ and $\zeta^{(l)}$, the support values $\alpha^{(l)}$ and bias $b^{(l)}$ of the lth binary LS-SVM are obtained on level 1 from (4); ii) calculate $E^{(l)}_D$ and $E^{(l)}_W$. Use these terms in the evaluation of the level 2 cost function in $\mu^{(l)}$ and $\zeta^{(l)}$.
   c) Eventually, one may refine the kernel parameter (e.g., $\sigma^{(l)}_j$) in order to improve the model evidence [12, 20, 21]. Go to step a), unless the best model $\mathcal{H}^{(l)}_j$ has been selected.

Given the output coding c_m, the prior multiclass distribution $P(c_m) = P(\mathcal{C}_m)$ and the L binary classifiers $\mathcal{H}^{(1:L)}_{\mu,\zeta}$, the posterior multiclass probabilities are inferred as follows:

[1] Notice that this scheme differs from the approach suggested in [1], where one uses a 0 instead of a don't care for the output coding.

1. Normalize the input in exactly the same way as the training data.

2. For each bit $y^{(l)}$ ($l = 1, ..., L$) calculate the likelihood $P(x|y^{(l)}, \mathcal{H}_{\mu,\zeta}^{(l)})$ for $y^{(l)} \in \{-1, +1\}$ from (11) and (12), given x, the model parameters $\alpha^{(l)}$, $b^{(l)}$, the hyperparameters $\mu_{MP}^{(l)}, \zeta_{MP}^{(l)}$ and the kernel function (with parameter, e.g., $\sigma^{(l)}$ for an RBF-kernel) of the best model $\mathcal{H}^{(l)}$.

3. Update the posterior probability of each codeword c_m, $m = 1, ..., M$ according to the updating rules (17) or (20) starting from the prior probability $P(c_m)$ for $l = 1, ..., L$.

4. Select the codeword c_m with the maximum posterior probability and assign the class label to the corresponding class $\mathcal{C}_m$.

When no $\times$ are used, one can also use directly (19) to determine the codeword with minimal loss.

5. Experiments

The test set performances of the output codings (1vs1, 1vsA, MOC and ECOC) using Hamming and Bayesian decoding were assessed on ten benchmark datasets using binary LS-SVM plug-in classifiers with linear and RBF-kernels. We also used SVM classifiers [4, 5, 6, 22] in combination with 10-fold cross-validation on the training set in order to select the regularization parameter C and the parameter σ of the RBF-kernel. The following multiclass datasets were retrieved from the UCI benchmark repository[2]: the balance scale (`bal`), dermatology (`der`), glass identification (`gla`), iris plant (`iri`), LED display (`led`), new thyroid (`nth`), small soybean (`ssd`), the Statlog vehicle silhouettes (`veh`), the wine recognintion (`win`) and the zoo (`zoo`) database. The main characteristics of these datasets are summarized in Table 1. Each dataset was randomized 10 times. For each randomization, we used 2/3 of the data points for the training and the remaining 1/3 for testing. All experiments were conducted within the matlab environment.

The outputs were assigned according to the 4 output codings mentioned above. The codebook $[c_1, ..., c_m]$ for the ECOC was constructed as follows. We used maximally $L = 10\lceil \log_2 M \rceil$ bits, where $\lceil \cdot \rceil$ rounds up to the nearest integer. Rows only differing up to the sign were only used once. We generated maximally 10000 times randomly constructed codebooks and selected the codebook for which the minimal Hamming

[2] UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. University of California, Dept. of Information and Computer Science, Irvine, CA.

distance between its codewords c_m was maximal [1]. The L binary LS-SVM classifiers with linear kernel and RBF-kernel are designed within the evidence framework [11, 12, 20, 21] as explained in Section 4. For the 1vs1 coding scheme, we mention that the Hamming distance to a $\times$ was taken to be 1/2 [1]. Hamming decoding was also used for the 1vsA output coding, but when more than one codeword had minimal Hamming distance, the class label was assigned corresponding to the classifier with the largest output. All classes were assumed to have equal prior multiclass probabilities in the Bayesian decoding scheme.

The average test set performances and their standard deviations (between parantheses) of the LS-SVM and SVM classifiers are reported in Tables 2 and 3, respectively. Both SVM and LS-SVM classifiers yield comparable results. For the LS-SVM and SVM with linear kernels, one notices that the performances of the different output coding schemes may vary significantly. The 1vs1 output generally yields the best test set performance, combined with Bayesian output coding. This is intuitively understood since this coding scheme is the generate simple classification problems that are more likely to be solvable with a low capacity classifier, like, e.g., the linear kernel [10]. The average test set performances for the RBF-kernel are reported in the second part of Table 2 and 3. The difference in performance between the different output codings is less significant than for the linear kernel, which is explained by the possibility of a nonlinear decision boundary with the RBF-kernel (see also the example in [17]). The ECOC and 1vs1 coding generally yield the best performance. The difference with the computationally simple MOC is much smaller than with the linear kernel.

We also applied the 1vs1 output coding on the 10-class US Postal dataset (`usps`) [5, 10], consisting of a training set and test set of 7291 and 2007 instances, respectively. The training the binary SVM and LS-SVM classifier was done in the same way for each of the 45 binary plug-in classifiers. On a Pentium III 733 MHz machine with 128 Mb, the design of a binary SVM [4] and LS-SVM [20] took about 47680 and 7670 seconds using Matlab cmex implementations, respectively. The final training and evaluation of a binary SVM took 389 and 28 seconds, respectively, while 31 and 59 seconds were needed for the training and evaluation of a binary LS-SVM classifier. Binary SVM plug-in classifiers in combination with Hamming and loss function decoding yield test set performances of 95.1% and 94.0%, respectively. Using LS-SVMs in combination with Hamming and Bayesian decoding yielded comparable test set performances of 95.2% and 94.7%, respectively.

6. Conclusions

Multiclass categorization problems can be reformulated as a set of binary classification problems. In this paper, we considered the use of binary Support Vector Machine (SVM) and Least Squares SVM (LS-SVM) plug-in classifiers. Each binary LS-SVM classifier was optimized with respect to a least squares regression cost function. The moderated outputs of these binary classifiers are used together with the prior multiclass probabilities in order to infer the posterior multiclass probabilities of the codewords by repeatedly applying Bayes' formula. Loss function based decoding can be obtained as a special case of Bayesian decoding. Empirical evidence motivates the use of one-versus-one coding when using a linear kernel in combination with Bayesian or Hamming decoding. For a radial basis function (RBF) kernel, which allows to construct nonlinear decision boundaries, error correcting and one-versus-one coding yield the best performances, but satisfactory results are also obtained with the simpler minimum output coding.

Acknowledgements

T. Van Gestel and J.A.K. Suykens are a Research Assistant and a Postdoctoral Researcher with the FWO-Flanders, resp. This work was carried out at the ESAT laboratory, the Interdisciplinary Center of Neural Networks ICNN and supported by grants and projects from the Flemish Gov.: (Res. Council KULeuven: GOA-Mefisto 666; FWO-Flanders: proj. G.0262.97, G.0135.95 and comm. ICCoS; AWI: Bil. Int. Coll.; IWT: IMPACT); from the Belgian Fed. Gov. (IUAP-P4/02, P4/24); from the Eur. Comm.: (Niconet, ERNSI).

References

1. Allwein, E., Schapire, R. and Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers, *Journal of Machine Learning Research* **1** (2000), 113-141.
2. Baudat, G. and Anouar, F.: Generalized Discriminant Analysis Using a Kernel Approach, *Neural Computation* **12** (2000), 2385-2404.
3. Bishop, C.M.: *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.
4. Cawley, G.C.: MATLAB Support Vector Machine Toolbox (v0.54β), University of East Anglia, School of Information Systems, Norwich, Norfolk, U.K., 2000.
5. Cristianini, N. and Shawe-Taylor, J.: *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

6. Cortes, C. and Vapnik, V.: Support vector networks. *Machine Learning* **20** (1995), 273-297.
7. Dietterich, T.G. and Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes, *Journal of Artificial Intelligence Research* **2** (1995), 263-286.
8. Duda, R.O. and Hart, P.E.: *Pattern Classification and Scene Analysis*, John Wiley, New York, 1973.
9. Hastie, T., Tibshirani, R. and Buja, A.: Flexible Discriminant Analysis by Optimal Scoring, *Journal of the American Statistical Association* **98** (1994), 1255-1270.
10. Kressel, U.H.-G.: Pairwise Classification and Support Vector Machines, In: B. Schölkopf, C.J.C. Burges and A.J. Smola (Eds.), *Advances in Kernel Methods, Support Vector Learning*, MIT Press, 1999.
11. Kwok, J.T.: The evidence framework applied to Support Vector Machines, *IEEE Transactions on Neural Networks* **10** (2000), 1018-1031.
12. MacKay, D.J.C.: Probable Networks and Plausible Predictions - A Review of Practical Bayesian Methods for Supervised Neural Networks, *Network: Computation in Neural Systems* **6** (1995), 469-505.
13. Mika, S., Rätsch, G., Weston, J., Schölkopf, B. and Müller, K.-R.: Fisher Discriminant Analysis with Kernels, In: Y.-H. Hu, J. Larsen, E. Wilson & S. Douglas (Eds.), *Proc. Neural Networks for Signal Processing IX*, IEEE, pp. 41-48, 1999.
14. Platt, J.C., Cristianini, N. and Shawe-Taylor, J.: Large Margin DAGs for Multiclass Classification, In: S.A. Solla, T.K. Leen and K.-R. Muller (Eds.), *Advances in Neural Information Processing Systems, 12*, MIT Press, 2000.
15. Saunders, C., Gammerman, A. and Vovk, V.: Ridge regression learning algorithm in dual variables, In *Proc. 15th Int. Conf. on Machine Learning ICML'98*, Morgan Kaufmann, pp. 515-521, 1998.
16. Sejnowski, T.J. and Rosenberg, C.R.: Parallel networks that learn to pronounce english text, *Journal of Complex Systems* **1** (1987), 145-168.
17. Suykens, J.A.K. and Vandewalle, J.: Least squares support vector machine classifiers, *Neural Processing Letters* **9** (1999), 293-300.
18. Suykens, J.A.K. and Vandewalle, J.: Multiclass Least Squares Support Vector Machines, In: *Proc. International Joint Conference on Neural Networks (IJCNN'99)*, Washington DC, 1999.
19. Utschick, W.: A regularization method for non-trivial codes in polychotomous classification, *International Journal of Pattern Recognition and Artificial Intelligence* **12** (1998), 453-474.
20. Van Gestel, T., Suykens, J.A.K., Lanckriet, G., Lambrechts, A., De Moor, B. and Vandewalle, J.: *A Bayesian Framework for Least Squares Support Vector Machine Classifiers*, Technical Report 00-65, ESAT-SISTA, K.U.Leuven (Leuven, Belgium), 2000, submitted for publication.
21. Van Gestel, T., Suykens, J.A.K., Baestaens, D.-E., Lambrechts, A., Lanckriet, G., Vandaele, B., De Moor, B. and Vandewalle, J.: Predicting Financial Time Series using Least Squares Support Vector Machines within the Evidence Framework, *IEEE Transactions on Neural Networks* **12** (2001), to appear.
22. Vapnik, V.: *Statistical learning theory*, John Wiley, New-York, 1998.
23. Williams, C.K.I.: Prediction with Gaussian Processes: from Linear Regression to Linear Prediction and Beyond, In: M.I. Jordan (Ed.), *Learning and Inference in Graphical Models*, Kluwer Academic Press, 1998.

Table I. Characteristics of the UCI multiclass datasets with n inputs and M classes. The number of data points used for training, testing and the total number of data points is denoted by N_{tr}, N_{test} and N_{tot}, respectively.

	`bal`	`der`	`gla`	`iri`	`led`	`nth`	`ssd`	`veh`	`win`	`zoo`
n	4	34	9	4	7	5	21	18	13	16
N_{tr}	416	244	142	100	400	143	31	564	118	67
N_{test}	209	122	72	50	200	72	16	282	60	34
N_{tot}	625	366	214	150	600	215	47	846	178	101
M	3	6	6	3	10	3	4	4	3	7

Table II. Test set performances of LS-SVMs with linear and RBF-kernel for different output codings using Hamming and Bayesian decoding, the corresponding sample standard deviations are put between parantheses.

	LS-SVM with Linear kernel							
	Hamming Decoding				Bayesian Decoding			
	1vs1	1vsA	MOC	ECOC	1vs1	1vsA	MOC	ECOC
`bal`	87.5(2.7)	87.4(2.5)	**92.0**(2.2)	84.9(2.8)	87.0(2.5)	87.5(2.6)	**92.0**(2.2)	87.5(2.6)
`der`	96.3(1.1)	97.0(1.6)	94.3(2.1)	97.1(1.4)	96.6(1.0)	96.8(1.6)	95.5(1.6)	**97.3**(1.4)
`gla`	**61.5**(4.9)	57.4(7.1)	47.2(3.9)	56.9(6.6)	58.3(5.0)	58.2(6.4)	50.1(3.8)	59.0(7.6)
`iri`	97.0(1.7)	81.4(5.3)	72.2(4.7)	72.0(4.8)	**97.2**(2.3)	85.2(5.1)	72.8(4.8)	85.2(5.1)
`led`	70.7(3.7)	70.9(4.4)	54.5(2.1)	68.2(3.8)	**72.0**(2.8)	71.5(3.3)	54.5(2.1)	71.2(4.2)
`nth`	91.1(3.9)	87.8(3.9)	86.7(4.5)	86.7(4.5)	**92.5**(3.8)	88.9(4.4)	87.4(4.7)	87.4(4.7)
`ssd`	96.2(3.2)	95.0(2.6)	95.0(2.6)	95.0(2.6)	**96.9**(3.3)	95.0(2.6)	95.0(2.6)	95.0(2.6)
`veh`	80.6(1.8)	77.0(1.6)	69.3(2.1)	68.0(1.9)	**80.7**(1.3)	78.0(2.2)	69.3(2.1)	78.4(2.2)
`win`	98.2(1.5)	98.7(1.7)	96.5(2.0)	98.7(1.7)	98.0(1.7)	**99.0**(1.2)	97.3(1.6)	98.7(1.7)
`zoo`	91.5(3.8)	91.2(5.4)	88.2(6.0)	90.6(5.8)	91.2(3.9)	**92.6**(5.0)	88.2(6.0)	**92.6**(5.2)
	LS-SVM with RBF-kernel							
	Hamming Decoding				Bayesian Decoding			
	1vs1	1vsA	MOC	ECOC	1vs1	1vsA	MOC	ECOC
`bal`	92.3(1.7)	88.6(2.1)	93.1(2.9)	88.6(2.1)	91.4(1.8)	90.3(2.1)	**93.3**(2.8)	90.3(2.1)
`der`	97.4(0.8)	96.0(0.7)	96.7(1.1)	**97.7**(0.8)	97.3(1.0)	97.4(0.8)	97.0(1.2)	97.6(0.8)
`gla`	62.6(7.4)	54.4(4.4)	50.7(5.6)	63.2(9.5)	58.5(6.1)	64.3(7.3)	51.2(5.1)	**66.1**(8.5)
`iri`	**95.8**(3.5)	94.4(2.8)	94.6(2.8)	94.4(2.8)	95.6(3.4)	95.0(3.2)	94.6(2.8)	95.0(3.2)
`led`	71.9(3.1)	68.8(1.6)	70.7(2.9)	71.3(2.4)	**72.2**(2.8)	71.9(3.0)	70.7(2.9)	72.0(3.1)
`nth`	96.0(1.7)	96.1(1.9)	**96.5**(2.0)	96.1(1.9)	96.2(1.9)	96.1(1.9)	**96.5**(2.0)	96.1(1.9)
`ssd`	**97.5**(3.2)	93.8(4.2)	95.0(2.6)	95.0(2.6)	**97.5**(3.2)	95.0(2.6)	95.0(2.6)	95.0(2.6)
`veh`	82.3(1.5)	76.0(4.8)	80.7(2.4)	76.2(2.2)	**82.8**(1.5)	79.5(3.9)	80.7(2.4)	82.7(2.3)
`win`	**98.3**(1.6)	97.7(1.8)	97.8(1.9)	97.7(1.8)	**98.3**(1.6)	98.2(1.7)	97.8(1.9)	98.2(1.7)
`zoo`	92.6(3.7)	89.4(5.9)	91.2(3.9)	92.6(4.2)	**92.9**(3.7)	**92.9**(4.2)	91.2(3.9)	**92.9**(3.7)

Table III. Test set performances of SVMs with linear and RBF-kernel for different output codings using Hamming and loss function decoding, the corresponding sample standard deviations are put between parantheses.

SVM with Linear kernel								
	Hamming Decoding				Loss Function Decoding			
	1vs1	1vsA	MOC	ECOC	1vs1	1vsA	MOC	ECOC
bal	92.3(2.0)	87.5(2.4)	**92.9**(1.8)	84.9(2.8)	92.9(1.8)	92.0(2.0)	**92.9**(1.8)	92.0(2.0)
der	96.5(1.0)	96.1(1.0)	94.8(1.7)	**97.4**(0.8)	94.9(1.5)	96.6(1.4)	95.2(1.6)	96.1(1.3)
gla	**61.0**(4.4)	50.0(6.1)	49.6(8.0)	57.2(4.4)	54.4(7.8)	53.2(7.0)	50.7(7.7)	56.1(6.8)
iri	**95.6**(1.8)	72.4(4.7)	72.2(4.6)	70.8(3.8)	85.4(5.2)	76.6(6.0)	72.2(4.6)	76.6(6.0)
led	**73.5**(2.8)	67.4(4.0)	55.5(3.7)	67.1(4.6)	65.9(4.9)	71.0(4.5)	56.1(4.3)	55.2(8.4)
nth	**96.0**(2.6)	90.0(4.3)	88.9(3.6)	88.2(3.3)	94.6(4.2)	92.8(2.4)	92.1(2.0)	92.8(2.4)
ssd	**98.8**(2.6)	97.5(3.2)	97.5(3.2)	97.5(3.2)	**98.8**(2.6)	98.1(3.0)	97.5(3.2)	97.5(3.2)
veh	**80.0**(2.2)	69.4(6.9)	69.8(2.5)	77.5(2.2)	**80.0**(1.8)	76.1(2.8)	69.8(2.5)	76.5(3.0)
win	**98.0**(1.9)	**98.0**(2.0)	97.2(1.9)	96.3(2.3)	93.5(4.1)	97.7(2.1)	97.3(2.0)	97.7(2.1)
zoo	90.0(6.4)	87.9(5.6)	88.2(6.4)	**92.6**(4.2)	89.7(8.2)	88.5(5.3)	88.2(6.4)	88.2(6.4)
SVM with RBF-kernel								
	Hamming Decoding				Loss Function Decoding			
	1vs1	1vsA	MOC	ECOC	1vs1	1vsA	MOC	ECOC
bal	96.5(1.2)	90.2(2.3)	**96.9**(1.9)	89.0(2.6)	92.9(1.8)	92.9(1.8)	**96.9**(1.9)	92.9(1.8)
der	96.6(1.0)	94.8(1.0)	96.1(1.6)	**97.3**(1.0)	95.8(1.8)	94.3(1.5)	96.5(1.4)	96.6(1.2)
gla	65.1(6.1)	62.2(6.2)	61.4(6.0)	65.7(6.3)	60.8(5.2)	46.1(9.0)	**76.4**(6.3)	65.3(7.6)
iri	94.4(2.6)	94.8(2.7)	94.4(3.4)	93.6(2.8)	77.8(11.3)	80.2(7.5)	94.4(3.4)	**95.4**(2.5)
led	**73.0**(3.1)	67.5(3.5)	71.0(3.0)	71.4(2.8)	68.1(6.4)	70.7(3.1)	71.0(3.0)	71.9(3.3)
nth	95.6(2.0)	94.4(2.4)	95.1(1.6)	94.2(2.2)	94.3(3.8)	94.0(3.7)	95.7(1.5)	**96.0**(2.3)
ssd	**98.8**(2.6)	96.2(5.3)	**98.8**(2.6)	**98.8**(2.6)	**98.8**(2.6)	**98.8**(2.6)	**98.8**(2.6)	**98.8**(2.6)
veh	83.8(1.4)	82.3(1.5)	79.4(2.3)	83.0(1.5)	78.9(3.7)	74.7(3.9)	84.1(2.3)	**84.4**(2.6)
win	**98.0**(2.3)	97.3(1.8)	97.3(1.8)	96.8(2.4)	96.3(2.8)	97.3(2.2)	97.3(1.8)	97.8(2.1)
zoo	89.4(4.4)	86.2(8.4)	89.7(4.0)	**92.6**(3.7)	89.4(5.6)	90.9(5.3)	89.7(4.0)	**92.6**(4.0)