PROBABILITY ESTIMATION FOR MULTICLASS PROBLEMS COMBINING SVMS AND NEURAL NETWORKS

Cristián Bravo[*], *Gastón L'Huillier*[†], *Jose Luis Lobato*[*], *Richard Weber*[*]

Abstract: This paper addresses the problem of probability estimation in Multiclass classification tasks combining two well-known data mining techniques: Support Vector Machines and Neural Networks. We present an algorithm which uses both techniques in a two-step procedure. The first step employs Support Vector Machines within a One-vs-All reduction from multiclass to binary approach to obtain the distances between each observation and the Support Vectors representing the classes. The second step uses these distances as inputs for a Neural Network, built with an entropy cost function and softmax transfer function for the output layer where class membership is used for training. Consequently, this network estimates probabilities of class membership for new observations. A benchmark using different databases demonstrates that the proposed algorithm is highly competitive with the most recent techniques for multiclass probability estimation.

Key words: *Probability estimation, multiclass classification, support vector machines, artificial neural networks*

Received: August 14, 2009
Revised and accepted: March 11, 2010

1. Introduction

Support Vector Machines (SVMs) [20, 22] are one of the best-known machine learning algorithms for binary classification. In the case of binary classification, Platt developed a method [16] to obtain an analytical expression for probability estimation using SVMs. In multiclass classification problems, several binary SVMs can be employed to classify observations, using the continuous output to estimate the probability of belonging to a given class. Furthermore, there have been few approaches which estimate those probabilities using numerical approximations and optimization methods [13, 15, 24].

[*]Cristián Bravo, José Luis Lobato, Richard Weber
Department of Industrial Engineering, University of Chile, Republica 701, Santiago, Chile, E-mail: `cbravo@dii.uchile.cl, jlobato@ing.uchile.cl, rweber@dii.uchile.cl`

[†]Gastón L'Huillier
Department of Computer Science, University of Chile, Blanco Encalada 2120, Santiago, Chile, E-mail: `glhuilli@dcc.uchile.cl`

Artificial Neural Networks (ANN) have been used successfully for probability estimation in multiclass problems. In this paper, we propose to combine SVMs and ANNs in the following hybrid approach: First, using all available attributes, a multiclass extension of binary SVMs determines for each object its distances to all hyperplanes. Then, the ANN takes these continuous values as input to calculate the corresponding class probabilities. Here, the ANN is defined with a softmax transfer function for the output neurons and cross-entropy error function, a well-known structure to obtain output probabilities.

The proposed approach extends Platt's idea to the multiclass case, allowing the estimation of probabilities from the SVMs outputs and providing an analytical function that is more simple to handle than existing, non-analytical, methods. An analytical function eliminates the need to re-run an algorithm each time a new estimation is needed, granting an advantage over other approaches because it greatly reduces the time taken to needed the model in new data. The algorithm takes a longer time to be trained, mainly due to the complexity associated to train both ANN and multiple SVMs. In spite of this potential drawback, the training step is done just once, which is contrasted with the multiple times it can be applied faster and more reliably to new data.

This paper is structured as follows: Section 2 introduces SVMs and alternative methods for probability estimation in multiclass classification. The proposed hybrid algorithm is presented in Section 3. Section 4 provides the error measurement criteria and numerical results. Finally the conclusions are stated in Section 5.

2. SVM Classification and Probability Estimation

In this section, binary and multiclass SVMs are presented, along with previously introduced methods for estimating *a posteriori* probabilities.

2.1 Binary classification with SVMs

The main idea of SVMs is to find the optimal hyperplane that separates observations belonging to two classes in a *Feature Space* F, maximizing the margin between those classes. The *Feature Space* is a Hilbert Space defined by a dot product, known as the Kernel function, $k(x,x') := (\phi(x)\cdot\phi(x'))$, where $\phi:\chi\rightarrow F$, is the mapping defined to translate an input vector into the *Feature Space.*

The general formulation of SVMs is defined over a data set $\{(x_1,y_1),\ldots(x_N,y_N)\}$, $(x_i,y_j)\in\chi\times\{+1,-1\}$, where x_i is the A-dimensional feature vector for the i^{th} observation, and y_i, is its label. The objective of the SVM algorithm is to find the optimal hyperplane $w^T\cdot x+b$ defined by the following optimization problem,

$$\begin{aligned} \min \quad & \frac{1}{2}\sum_{i=1}^{A} w_i^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i\left(w^T x_i + b\right) \geq 1-\xi_i \ \forall i\in\{1,\ldots,N\} \\ & \xi_i \geq 0 \ \forall i\in\{1,\ldots,N\}. \end{aligned} \tag{1}$$

The objective function includes training errors ξ_i while obtaining the maximum margin hyperplane, adjusted by parameter C. Its dual formulation is defined by the following expression, known as the Wolfe dual formulation.

$$\begin{aligned}
\max \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \cdot \kappa(x_i, x_j) \\
\text{subject to} \quad & \alpha_i \geq 0, \forall i \in \{1, \dots, N\} \\
& \sum_{i=1}^{N} \alpha_i y_i = 0
\end{aligned} \tag{2}$$

Finally, after determining the optimal parameters α, the continuous outputs are represented by

$$g\left(x_j\right) = \sum_{i=1}^{N} \alpha_i y_i \cdot \kappa(x_i, x_j) + b. \tag{3}$$

The resulting classification for the j^{th} element is given by $f(x_j) = sign\left(g(x_j)\right)$.

2.2 Probability estimation for binary SVM

Some approaches proposed by [13, 16, 25] extend the original output (3) to a probability estimation, defined by the expression $p_{ij} = P(y = i|x_j)$, $i = \{1, 2, \dots, K\}$, $j = \{1, 2, \dots, N\}$. A well-known and extensively used approach proposed by Platt [16], establishes an analytical expression for the output probability, based on a sigmoid function given by equation (4).

$$\hat{P}(y = i|g(x_j)) := p_{ij} = \frac{1}{1 + e^{M \cdot g(x_j) + B}}. \tag{4}$$

This function maps the $g(x)$ scores from the SVM, the distances from each object to the hyperplane that divides them, into $\hat{P}(c|g(x))$. There are two more constants to estimate, M and B, which can be obtained by maximum likelihood method.

Platt argues the assumption of sigmoid shape for the datasets is consistent with empirical results, and this is sustained by the many different shapes the function can take depending on the parameters. As an example, Fig. 1 shows different functions varying parameter M.

2.3 Multiclass classification with SVMs

For multiclass classification, according to [11, 14], there are five principal approaches to extend binary SVMs to solve multiclass problems:

- One-vs-All (OVA) [18].
- One-vs-One (OVO) [23].

- Half-vs-Half (HVH) [14].
- Multiclass Objective Function [23].
- Error-Correcting Output Coding (ECOC) [13].

The main idea of the OVA approach is to train K SVMs, one for each class defined by the label of the j^{th} object $y_j \in \{1, \ldots, K\}$, as shown in Fig. 2(a).

In the case of the OVO approach, $\frac{K(K-1)}{2}$ SVMs must be trained in order to separate every class from each other, where a final output labeling is needed to be determined, as shown in Fig. 2(b). Many alternatives are proposed in [11, 15], where the simplest is combining the output of the SVMs as a "majority voting" solution. The HVH methodology is built recursively dividing a training data set of K classes into two subsets, which can be represented as a hierarchy clustering problem. The ECOC approach, proposed in [13], defines from a given binary classification a special data structure to solve the multiclass classification.

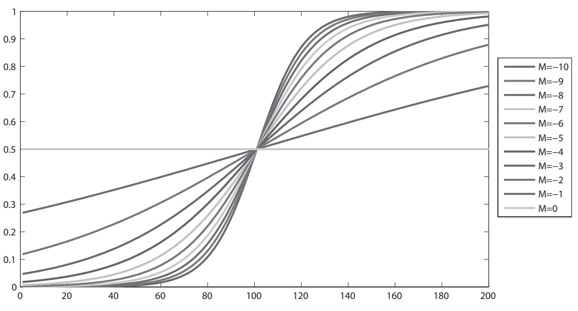
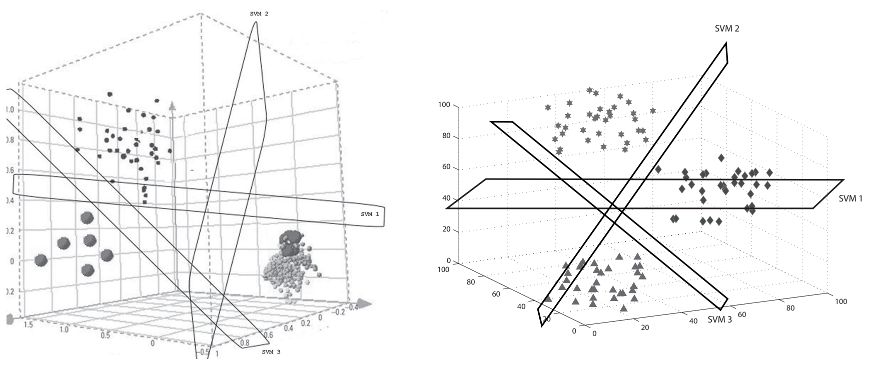


Fig. 1 *Different sigmoid functions depending on parameter M. Parameter B is fixed to zero.*



(a) SVMs in OVA approach

(b) SVMs in OVO Approach

Fig. 2 *Schemes for Multiclass SVMs.*

2.4 Probability estimation for multiclass SVMs

In the following, let K be the number of classes, x_j the attribute vector of the j^{th} observation and $y_j \in \{1, \dots, K\}$ its class label. The expression $p_{ij} = P(y_j = i|x_j)$, $i \in \{1, 2, \dots, K\}$ represents the observation's probability of belonging to class i which must satisfy $\sum_i p_{ij} = 1$. In the following, whithout loss of generality and for the sake of simplicity a generic observation $\mathbf{x}$ and the class label y will be considered. Likewise, the observation's probability of beloging to class i will be considered as $p_i = P(y = i|\mathbf{x})$.

Several approaches have been developed to estimate these probabilities, where the first approach, developed by Wu, Lin, and Weng [24], estimates p_{ik} proposing two different methods, as outlined below. Both assume that the pairwise class probability estimation r_{ik} of $P(y = i|y = i \text{ or } k, \mathbf{x})$ which is obtained from the binary SVM [22].

The first method (WLW1), based on the Markov Chain theory, improves previous studies of Hastie and Tibshirani [10] and determines probabilities $\mathbf{p} = (p_1, \dots, p_K)$ according to the following linear problem:

$$\begin{aligned} p_i &= \sum_{k:k\neq i} \left(\frac{p_i + p_k}{K-1}\right) r_{ik}, \forall i \\ \text{subject to } &\sum_{i=1}^{K} p_i = 1, p_i \geq 0, \forall i. \end{aligned} \tag{5}$$

The decision rule is given by:

$$\delta_1 = arg \max_i [p_i]. \tag{6}$$

Whose solution is unique under conditions which are satisfied in this problem by construction. Furthermore, Wu et al. in [24], demonstrated that problem (5) is equivalent to the following

$$Q\mathbf{p} = \mathbf{p} \ , \ \sum_{i=1}^{K} p_i = 1, p_i \geq 0, \forall i, Q = \begin{cases} r_{ik}/(K+1) & \text{if } i \neq k \\ \sum_{s:s\neq i} r_{is}/(K+1) & \text{otherwise} \end{cases} . \tag{7}$$

For which the solution for $\mathbf{p}$ can be easily calculated with methods, such as Gaussian elimination or Markov Chains' stationary solution.

The second method (WLW2) changes the linear problem (5) to an optimization one, as follows:

$$\begin{aligned} \min_{\mathbf{p}} \quad & \sum_{i=1}^{K} \sum_{k:k\neq i} (r_{ki}p_i - r_{ik}p_k)^2 \\ \text{subject to} \quad & \sum_{i=1}^{K} p_i = 1, p_i \geq 0, \forall i \end{aligned} \tag{8}$$

for which the decision rule for p_i is the same as in (6).

Another method to determine the multi-class probability p_i, proposed by Hastie and Tibshirani algorithm (HT) [10], minimizes the Kullback-Leibler distance ($L(\mathbf{p})$) between r_{ik} and ν_{ik}, given by:

$$L(\mathbf{p}) = \sum_{i<k} q_{ik} \left(log \frac{r_{ik}}{\nu_{ik}} + (1 - r_{ik}) log \frac{1 - r_{ik}}{1 - \nu_{ik}} \right), \tag{9}$$

where $\nu_{ik} = p_i/(p_i + p_k)$ and q_{ik} is the quantity of observations that belong to the i^{th} or k^{th} class. In this case, values for $\mathbf{p}$ are determined by an iterative algorithm for which the Kullback-Leibler distance is minimized.

Furthermore, the method proposed by Price, Kner, Personnaz, and Dreyfus (PKPD) [17], consists of a simple formula that approximates the probability p_i using the coefficients r_{ik} in the following expression,

$$p_i = \frac{1}{\sum_{k:k \neq i} \frac{1}{r_{ik}} - (K - 2)}. \tag{10}$$

Other approaches proposed by [9, 21] based on multiclass SVM that estimate posterior probabilities instead of hard labels are described in [22, 23]. Also, the idea presented by Petrovsky [15] uses SVMs together with game theory to estimate class probabilities using mixed strategies to formulate a linear programming model. However, both methods escape the scope of this paper due to their objectives and implementation methods that greatly differ from the benchmarks and models presented in this work.

2.5 Probability estimation based on neural networks

An alternative method [8, 19] uses ANN to approximate *a posteriori* class probabilities. This requires that the network's outputs are consistent with probabilities, i.e. $\sum_i y_i = 1$ and the cost (error) function used to train the network must be *strict sense Bayesian* or SSB [6].

Let $S = \{y \in \mathbb{R}^K | 0 \leq y_i \leq 1, \sum y_i = 1\}$, then a cost function $C(y, t)$, where t is the vector of targets, is SSB if $\mathrm{E}[C(y, t)]$ has a unique minimum in S when the outputs y_i are the *a posteriori* probabilities of the classes. In particular, the entropy cost function given by

$$C(y, t) = -\sum_i t_i \log(y_i) \tag{11}$$

is SSB, and so is the Mean Square Error (MSE). According to [7] (11) outperforms MSE in approximating probabilities, because (11) diverges when $y_i \to 0$. Thus, if $t_i = 1$ and $y_i \approx 0$ the error is severely penalized, fact that is not considered when using MSE. This criterion is closely related to the Kullback-Leibler information measure.

The *softmax* output function may be used to take into account the second condition ($\sum_i y_i = 1$). For K different classes, and coefficients $\beta_{k,j}$ associated to each class and variable, the generic *softmax* function corresponds to

$$p_k(x) = \frac{exp(\sum_{j=1}^{N} \beta_{k,j} x_j)}{\sum_{l=1}^{K} exp(\sum_{j=1}^{J} \beta_{l,j} x_j)} \tag{12}$$

In case this definition is applied to binary classification problems, a sigmoid function can be used, leading to the original proposition by Platt introduced in Section 2.

In terms of network specification, it is stated in [1] that this particular way to find probabilities requires large datasets. However, Vapnik [22] demonstrated that in this case if the approximation function is consistent with the dataset, it will converge to the "best possible guess" of the probability.

3. Multiclass Probability Estimation Combining ANN and SVMs

In this section, we present a hybrid method for multiclass probability estimation combining SVMs and Neural Networks (MCP-SVM-ANN). Next, we provide the general model structure of the approach. This is followed by the detailed algorithm and complexity analysis.

3.1 General structure of proposed approach

The first step obtains classifications of the observations using SVM within the OVA approach, defined in Section 2. Rifkin and Klautau [18] state that this approach is "as accurate as any other approach" when the underlying binary classifiers are well-tuned regularized classifiers, such as SVMs, on the basis of thoroughly controlled experiments. This conclusion, and the fact that this type of multiclass classifier has the smallest number of binary classifiers (K, one for each class) between the studied ones, supports its use for our method. The main idea of the second step is that now there are new observations characterized by the K distances of an instance $\mathbf{x}$ to the K hyperplanes created. In general, for most problems, $K < A$ so there is a gain related to dimension reduction. Given this, an object j is now defined by a vector $f(x_j) = (f_1, \ldots, f_K)$ of distances to the hyperplanes as an extension of (3).

$$f_k(x_j) = \sum_{l \in SV_s} \alpha_{l,k} y_{l,k} \cdot \kappa(x_l, x_j) + b_k, \tag{13}$$

where $\alpha_{l,k}$ is the lagrangian multiplier defined for the dual problem of the k^{th} SVM and the l^{th} Support Vector. Each input $f_k(x_j)$ represents the distance between the point and the separating hyperplane associated with class k, multiplied by -1 in case the point is in the negative side of the hyperplane, and by $+1$ in case it is in the positive side, representing a score of the credibility that observation j belongs to class k.

The neural network designed for this task consists of:

- K neurons in the input layer, one for each SVM.

- Any number of hidden neurons in the hidden layer. Each dataset will require a different number of neurons in the input layer, as this is correlated with the complexity of the patterns in the data. There is no consensus on the standard number of neurons in this layer, but Zhang in [26] points out that this number should be between $K/2$ and $2K$ neurons, based on an extensive survey of the literature.

- Linear transfer functions for all layers, except the output layer. This is done to reduce the models' complexity and maintain the similarity between our approach and Platt's. Any other transfer function will greatly augment training times and add unnecessary complexity to the final expression, whereas linear transfer functions create regular sigmoid-like outputs.

- K output neurons, with *softmax* transfer function in this layer (equation 12). In case it is a binary classification problem, this function should be replaced by a sigmoid function.

- Cross entropy error function is recommended because, as explained in Section 2.5, it brings several benefits when approximating probabilities. In any case, the error function that is used must be at least strict-sense Bayesian.

Using this architecture, and assuming M neurons in the hidden layer, the input layer has parameters $t_{k_i,m}$, $k_i \in \{1,\dots,K\}$ associated to each input variable (we will obviate biases for clarity) and each neuron in the hidden layer multiplies this parameters with an internal one given by u_{m,k_o}, $k_o \in \{1,\dots,K\}$. The final expression that enters the output layer is a linear combination of the variables $f_k(x_j)$ and the parameters, given by $\sum_{m,k_i} t_{k_i,m} u_{m,k_o} f_{k_i}(x_j)$.

Reordering the previous equation, we get that each output neuron k_o receives $\sum_{k_i} f_{k_i}(x_j) \sum_m t_{m,k_i} u_{m,k_o}$ as input. Renaming $k_i = l$, $k_o = k$ and $\sum_m t_{m,k_i} u_{m,k_o} = \beta_{k,l}$, we get the input for the softmax function (12):

$$p_k(x_j) = \frac{e^{\sum_{l=1}^{K} \beta_{k,l} f_l(x_j)}}{\sum_{j=1}^{K} e^{\sum_{l=1}^{K} \beta_{j,l} f_l(x_j)}}. \tag{14}$$

It is not recommended to adjust the transfer function in neural networks, but to alter the kernel function in SVMs, as this adds little complexity to the algorithm. A review of general architectures for finding *a posteriori* probabilities and density functions, not restricted for use in conjunction with hyperplanes, can be found in [2].

In general, equation (14) is of interest because it closely resembles Platt's method [16], allowing SVMs to be combined to form an exponential function that can approximate probabilities, considering the properties stated in Section 2.5. This equation offers the possibility to determine analytically the probabilities in a multiclass problem, which has several managerial implications and benefits, such as a fast evaluation time for new observations, and even use it in other applications for which an analytical expression is necessary.

Finally, there is another remark to be made: SVMs work as feature extractors when used as input for the neural network model, because they transform the input x_j into factors f_j (distances to hyperplanes) that are then used as input for the neural network model. The f scores have an inherent meaning, but mathematically they are references to whether an element is "closer" to a given class when used as continuous inputs, hence constructing a feature extractor for the original inputs. It has been stated that SVMs incorporate feature extraction characteristics intrinsically into their structure through the use of kernels [12], which makes them ideal for this application because they reduce the load on neural network's training time as well as acting as pattern refiners.

3.2 MCP-SVM-ANN algorithm and complexity

In the following, the MCP-SVM-ANN algorithm is formally exposed. Consider $\mathcal{T}$ as the training set composed by $\mathcal{T} = \{(x_1, y_1), \dots, (x_N, y_N)\} \in \{\chi \times \{+1, -1\}\}$. Let $\textsc{mc}$SVM$(\mathcal{T}, K, C, \sigma)$ be a model which trains an OVA multiclass SVM algorithm with the C parameter and σ, a set of parameters related to the kernel function used for training this model. Consider ANN$(\mathcal{T}, K, M)$ a object which trains the ANN algorithm using the dataset $\mathcal{T}$, with their respective input parameters and architecture previously explained. Finally, assume that all computed parameters can be retrieved from SVM and ANN, respectively. Then, the algorithm for the proposed hybrid method is stated as follows,

Algorithm 1: MCP-SVM-ANN

```
   Data: $\mathcal{T}, K, C, \sigma, M$
   Result: Analytical expression for $p_k(\cdot)$
 1 Initialize $\mathcal{T}_p = \{\}$;
 2 svm ← mcSVM($\mathcal{T}, K, C, \sigma$).solve();
 3 $SV_s$ ← svm.getSupportVectors();
 4 foreach $x_j \in \mathcal{T}$ do
 5     $\mathcal{T}_p \leftarrow \mathcal{T}_p \cup (\{f_1(x_j), \dots, f_K(x_j)\} \cup y_j)$, using $SV_s$ in equation 13;
 6 ann ← ANN($\mathcal{T}_p, K, M$).solve();
 7 foreach $l, k = \{1, \dots, K\}$ do
 8     $\beta_{l,k}$ ← ann.getBeta(l,k);
 9 Build $p_k(\cdot)$ (equation 14) using $\beta_{l,k}$;
10 return $p_k(\cdot)$;
```

As presented in algorithm 1, the interaction between the SVM and the ANN algorithms can be used to determine the *a posteriori* probabilities for a multiclass classification problem. In terms of computational complexity, as previously stated by [5] for SVMs, and [3] for ANNs, and considering that algorithm 1 is a hybrid method, its order is defined by the following expression,

$$O(NK^2M) + O(SV_s^3 + SV_s^2N + SV_sNA) + O(KN + K^2). \tag{15}$$

Recalling that K is the number of classes, M is the number of neurons in the hidden layer, SV_s is the number of support vectors, N is the number of observations to be evaluated, A is the number of features. This expression states that the proposed methodology's computational complexity is less than $O(KN + K^2)$, in the worst case scenario. Furthermore, the overall complexity is determined by $\max(O(NK^2M), O(SV_s^3+SV_s^2N+SV_sNA))$, which is not worse than any methodology that uses ANNs and SVMs sequentially.

4. Experimental Results and Discussion

In order to test the results of the proposed algorithm, different data sets obtained from the UCI Machine Learning repository [4] are used. As a benchmark, the results are compared against the HT, PKPD, WLW1 and WLW2 algorithms described in Section 2. The data sets used and their characteristics are presented in Tab. I.

Data set	Classes	Features
Waveform	3	21
Satimage	6	36
Segment	7	19
Letter	26	16

Tab. I *Data sets used and their characteristics.*

4.1 Experimental setup

To obtain the parameters for the SVMs, a grid search is conducted for σ and C for values in $(\sigma, C) \in [2^{-5}, \dots, 2^5] \times [2^{-5}, \dots, 2^5]$ and then the results are refined, conducting a new search in $(\sigma, C) \in [\sigma^*2^{-1}, \dots, \sigma^*2^1] \times [C^*2^{-1}, \dots, C^*2^1]$, where (σ^*, C^*) are the parameters obtained in the previous search. The numbers of epochs and neurons in the hidden layer are defined performing a new grid search for $(Epochs, Neurons) \in [5, 20] \times [5, 15]$ and using the combination with minimum error. The parameters used for each dataset are presented in Tab. II.

Data Set	K	C	σ	Epochs	Hidden Layer Size
Waveform	3	1,02	35,61	10	10
Satimage	6	2	1,5157	5	10
Segment	7	2,6390	1,149	19	9
Letter	26	6,0629	1,149	16	10

Tab. II *Parameters used for each model, per dataset.*

The previous parameters are used to train the SVMs and ANNs in twenty different partitions, each consisting of 300 observations for training and 500 samples for testing.

4.2 Error measurement

To compare our results, a common and unbiased error measure is necessary, one that is not used by any of the procedures to train the algorithm or to obtain the results. MSE and Cross Entropy do not accomplish this, because some of the algorithms exposed here are trained by minimizing one of these functions, whether directly or not. The use of any of them would give those algorithms an unfair advantage over the rest, and alter the results presented.

We use *Kurtosis*, a non-parametric measure of how close (on average) the probabilities are from the true results. In case that each of the samples in a data set is correctly classified, the error distribution plot of the true class (i.e. $e = 1 - p$ with p the predicted probability for the object's true class) would represent a *Dirac's delta* function with 100% of the results with error zero, whereas usually a curve with certain misclassified cases or with probabilities not equal to one is observed.

The Kurtosis of a normal distribution has an expected value of 3; lower values represent more peaked distributions. This measure can be used to compare between different algorithms' results, where more accurate algorithms have a smaller Kurtosis. The general formula for this error measure is:

$$\text{kurtosis} = \left(\frac{\frac{\sum_i e_i}{N}}{\sqrt{\frac{\sum_i e_i^2}{N-1}}} \right)^4 . \tag{16}$$

The formula above represents the Kurtosis of a distribution centered in zero. Such assumption is made because the measure needed is the dispersion of the errors against a perfect match, represented by mean zeroSuch assumption is made because the measure needed is the dispersion of the errors against a perfect match, represented by mean zero.

4.3 Experimental results

The following results are the average of the Kurtosis for the test samples:

Data Set	**HT**	**WLW 1**	**WLW 2**	**PKPD**	**MCP-SVM-ANN**
Waveform	0,0937	0,0922	0,0922	$\underline{0{,}0913}$	0,1398
Satimage	0,1442	0,1344	$\underline{0{,}1175}$	$\underline{0{,}1214}$	0,1189
Segment	0,1625	0,1479	$\underline{0{,}1174}$	0,1227	$\underline{0{,}0632}$
Letter	0,7944	0,7844	0,7111	0,6965	$\underline{0{,}4669}$

Tab. III *Kurtosis averaged over 20 test samples, 500 observations each. (Best values are underlined)*

Another complementary result is the class loss, which is the class associated to the j^{th} object.

The method is highly competitive in all datasets and shows the advantages of using a more complex model (neural networks) against other simpler methods.

Data Set	HT	WLW 1	WLW 2	PKPD	MCP-SVM-ANN
Waveform	15,35%	15,35%	15,37%	15,37%	$\underline{13{,}06\%}$
Satimage	15,11%	14,93%	$\underline{14{,}67\%}$	14,82%	15,18%
Segment	10,48%	10,34%	10,25%	10,29%	$\underline{9{,}76\%}$
Letter	47,56%	46,09%	$\underline{44{,}24\%}$	44,87%	48,00%

Tab. IV *Class loss averaged over 20 test samples, 500 observations each. (Best values are underlined).*

A smaller kurtosis implies that errors are concentrated around zero (Section 4.2), so the probabilities obtained are close to the real result. Class loss indicates accuracy for one class, the real class of the object, and is calculated by measuring whether the estimated probability is the maximum value against all other per-class probabilities. The second measure does not take into account dispersion, considering that it is sufficient for it to be the maximum value by a small quantity for it to be correct, and this may not be a good estimation because of severe dispersion of the result. Both quantities give a joint diagnosis of the performance of the models, with lower kurtosis meaning small dispersion of the probabilities (less uncertainty) and lower class losses implying better relative accuracy across classes.

In terms of each particular dataset, we observe that letter dataset presents a high number of classes, and our method gives the lowest kurtosis of all tested methods. This indicates a smaller dispersion in the probabilities and more reliable estimations for all classes. The method tries to better estimate probabilities for all the dataset, reflected in a drastically lower kurtosis, and by doing so it sacrifices some class accuracy, with a slightly lower class loss.

The inverse phenomena can be observed in waveform dataset, which is characterized by the lowest number of classes. Our model now is willing to be more disperse (higher kurtosis) but with a great gain in class accuracy against all other methods tested. Given the smaller number of classes, it favors a reduction of the per-class error instead of the overall dispersion.

In segment dataset the method is able to reproduce both a low dispersion and a high class accuracy, being the best overall model. A similar behavior is observed in satimage dataset, with all methods leading to a similar class accuracy and kurtosis. The same interesting result can be seen in all datasets: error is divided between classes so results associated to unclear patterns are improved by adjusting the probabilities in order to maximize overall accuracy.

Another interesting feature of the MCP-SVM-ANN algorithm is its flexibility given by kernel and parameter selection, which is not limited to SVMs, but can also be performed related to ANNs, granting additional degrees of freedom.

5. Conclusions

We present a hybrid approach for probability estimation in multiclass problems, where SVMs are used as feature extractor for an ANN for probability estimation in the case of multiclass classification.

First, the curse of dimensionality in the most complex step (ANNs) is generally avoided since the number of classes is in most cases smaller than the number of attributes.

The degrees of freedom of the algorithm can be both in favor or a disadvantage in a given application, because they increase fine tuning time, but can substantially improve results when properly used. Related to this, the computational complexity is equal to the complexity of the most costly step in the algorithm, which is divided into $K + 1$ independent steps so this does not become a major issue.

The method accomplishes the desired goal, with a performance highly competitive with other algorithms. By using neural networks it is possible to capture complex patterns, reflected in results that reduce dispersion, class loss, or both, if possible, against all other methods tested.

One of the major advantages of this method is its analytical expression given by a linear combination of the K outputs of the SVMs in an exponential function, as given by equation (14). This configuration resembles Platt's proposal for binary problems and can be considered an extension to multiclass problems. An analytical expression allows for lower testing and applications times, associated with time gains in the long run, because a costly training cycle once is contrasted with lower application times each time the model is applied.

Another benefit of the method comes from the advantages of both techniques used. On the one hand, using hyperplane classifiers such as SVMs incorporates robust statistical results into the model giving confidence regarding the risk of the classifier. On the other hand, neural networks' capability when approximating functions is used to estimate probabilities.

Future work has to be done comparing alternative approaches for feature extraction combined with ANN, and more profound algorithm complexity studies.

Acknowledgements

The first author would like to acknowledge the CONICYT grant that finances this research. Support from the Chilean "Instituto Sistemas Complejos de Ingeniera" (ICM: P-05-004-F, CONICYT: FBO16; *www.sistemasdeingenieria.cl*), from the PhD in Engineering Systems (for the first author) and Master in Operations Management (second and third author) of the University of Chile is greatly acknowledged.

References

[1] Arribas J. I., Cid-Suero J., Adali T.: Neural networks to estimate ml multi-class constrained conditional probability density functions. In: International Joint Conference on Neural Networks, Washington, D.C., USA, 1999, pp. 1429–1432.

[2] Arribas J. I., Cid-Suero J., Adali T., Figueiras-Vidal A. R.: Neural architectures for parametric estimation of a posteriori probabilities by constrained conditional density functions. In: Proceedings of the IEEE Workshop on Neural Networks for Signal Processing, Madison, WI, USA, 1999, pp. 263–272.

[3] Back A. D., Tsoi A. C.: On the backpropagation algorithm: paralysis in multilayer perceptrons. In: ACNN94 Proceedings Fifth Australian Conference on Neural Networks, Brisbane, Australia, 1994, pp. 102–104.

[4] Blake C. L., Merz C. J.: UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[5] Burges C. J. C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Discov., **2**, 2, 1998, pp. 121–167.

[6] Cid-Sueiro J., Arribas J. I., Urban-Munoz S., Figueiras-Vidal A. R.: Cost functions to estimate a posteriori probabilities in multiclass problems. IEEE Transactions on Neural Networks, **10**, 3, 1999, pp. 645–656.

[7] El-Jaroudi A., Makhoul J.: A new error criterion for posterior probability estimation with neural nets. In: International Joint Conference on Neural Networks, Washington, D.C., USA, 1990, pp. 185–192.

[8] Gish H.: A probabilistic approach to the understanding and training of neural network classifiers. In: Proceedings of ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Albuquerque, New Mexico, USA, 1990, pp. 1361–1364.

[9] Gönen M., Tanugur A. G., Alpaydin E.: Multiclass posterior probability support vector machines. IEEE Transactions on Neural Networks, **19**, 1, 2008, pp. 130–139.

[10] Hastie T., Tibshirani R.: Classification by pairwise coupling. In: NIPS '97: Proceedings of the 1997 conference on Advances in neural information processing systems 10, Cambridge, MA, USA, 1998. MIT Press, pp. 507–513.

[11] Hsu C.-W., Lin C.-J.: A comparison of methods for multiclass support vector machines. IEEE Transactions on Neural Networks, **13**, 2, August 2002, pp. 415–425.

[12] Kim K. I., Jung K., Park S. H., Kim H. J.: Support vector machines for texture classification. IEEE Trans. Pattern Anal. Mach. Intell., **24**, 11, 2002, pp. 1542–1550.

[13] Kong E. B., Dietterich T.: Probability estimation via error-correcting output coding. In: IASTED International Conference: Artificial Intelligence and Soft Computing, Banff, Canada, 1997.

[14] Lei H., Govindaraju V.: Half-against-half multi-class support vector machines. In: Nikunj C. Oza, Robi Polikar, Josef Kittler, and Fabio Roli, editors, Multiple Classifier Systems, Lecture Notes in Computer Science, Springer, 2005, **3541** , pp. 156–164.

[15] Petrovskiy M.: A game theory approach to pairwise classification with support vector machines. In: Proceedings of 2004 International Conference on Machine Learning and Applications, Louisville, Kentucky, USA, 2004, pp. 115–122.

[16] Platt J.: Probabilistic outputs for support vector machines and comparison to regularize likelihood methods. In: A. J. Smola, P. Bartlett, B. Schoelkopf, and D. Schuurmans, editors, Advances in Large Margin Classifiers, 2000, pp. 61–74.

[17] Price D., Knerr S., Personnaz L., Dreyfus G.: Pairwise neural network classifiers with probabilistic outputs. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, NIPS, MIT Press, 1994, pp. 1109–1116.

[18] Rifkin R., Klautau A.: In defense of one-vs-all classification. J. Mach. Learn. Res., 5, 2004, pp. 101–141.

[19] Ruck D. W., Rogers S. K., Kabrisky M., Oxley M. E., Suter B. W.: The multilayer perceptron as an approximation to a bayes optimal discriminant function. IEEE Transactions on Neural Networks, **1**, 4, 1990, pp. 296–298.

[20] Schölkopf B., Smola A. J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA, 2001.

[21] Tao Q., Wu G.-W., Wang F.-Y., Wang J.: Posterior probability support vector machines for unbalanced data. IEEE Transactions on Neural Networks, 2005, pp. 1561–1573.

[22] Vapnik V. N.: The Nature of Statistical Learning Theory (Information Science and Statistics). Springer, 1999.

[23] Weston J., Watkins C.: Multi-class support vector machines. Technical Report CSD-TR-9800-04, Department of Computer Science, Royal Holloway, University of London, 1998.

[24] Wu T.-F., Lin C.-J., Weng R. C.: Probability estimates for multi-class classification by pairwise coupling. Journal of Machine Learning Research, 5, 2004, pp. 975–1005.

[25] Zadrozny B., Elkan C.: Transforming classifier scores into accurate multiclass probability estimates. In: KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, 2002, ACM, pp. 694–699.

[26] Zhang G. P.: Avoiding pitfalls in neural network research. IEEE Transactions on Systems, Man, and Cybernetics, Part C, **37**, 1, 2007, pp. 3–16.