"One Against One" or "One Against All": Which One is Better for Handwriting Recognition with SVMs?

Jonathan Milgram *Mohamed Cheriet* *Robert Sabourin*
École de Technologie Supérieure, Montréal, Canada
milgram@livia.etsmtl.ca mohamed.cheriet@etsmtl.ca robert.sabourin@etsmtl.ca

Abstract

The "one against one" and the "one against all" are the two most popular strategies for multi-class SVM; however, according to the literature review, it seems impossible to conclude which one is better for handwriting recognition. Thus, we compared these two classical strategies on two different handwritten character recognition problems. Several post-processing methods for estimating posterior probability were also evaluated and the results were compared with the ones obtained using MLP. Finally, the "one against all" strategy appears significantly more accurate for digit recognition, while the difference between the two strategies is much less obvious with upper-case letters. Besides, the "one against one" strategy is substantially faster to train and seems preferable for problems with a very large number of classes. To conclude, SVMs allow significantly better estimation of probabilities than MLP, which is promising from the point of view of their incorporation into handwriting recognition systems.

Keywords: Support Vector Machine (SVM), Multi-class Classification, Posterior Probability Estimation.

1. Introduction

In the 1990s, a new type of learning algorithm was developed: the Support Vector Machine (SVM). As shown in the Burges' tutorial [1], SVM has several interesting properties for pattern recognition. Moreover, thanks to improved computing power and the development of fast learning algorithms, it is now possible to train SVM in real-world applications.

However, although SVM has attracted a great deal of attention in the machine learning community, the multi-class SVM is still an ongoing research issue. The existing methods can roughly be divided between two different approaches: the "single machine" approach, which attempts to construct a multi-class SVM by solving a single optimization problem, and the "divide and conquer" approach, which decomposes the multi-class problem into several binary sub-problems, and builds a standard SVM for each. The most popular decomposing strategy is probably the "one against all", which consists of building one SVM per class, trained to distinguish the samples in a single class from the samples in all remaining classes. Another popular strategy is the "one against one", which builds one SVM for each pair of classes. On the other hand, more complex decomposition schemes based on error correcting output codes (ECOC) have been introduced by Diettrich & Bakiri [2] and more recently extended by Allwein *et al.* [3]. A comparison of several multi-class SVM methods (2 "single machine" and 3 "divide and conquer") has been realized by Hsu & Lin [4]. The results observed are very similar; however, the authors conclude that "one against one" is more practical, because the training process is quicker. Moreover, as to the claim put forward by Allwein *et al.* [3] that "one against one" and other ECOC are more accurate than the "one against all" strategy, Rifkin & Klautau [5] disagree, arguing that the "one against all" strategy is as accurate as any other approach, assuming that the SVMs are well tuned. Thus, according to the literature review, it seems impossible to conclude which multi-class SVM is better for handwriting recognition. For this reason, we chose to compare the two most popular strategies, which are "one against all" and "one against one".

On the other hand, generally in handwriting recognition applications, the classifier only contributes to a small part of the final decision. It is essential, then, that the output of the classifier is a calibrated confidence measure, like posterior probability. However, although standard SVMs do not provide such probabilities, a simple post-processing method for mapping the outputs of a single SVM into posterior probabilities has been proposed by Platt [6] and improved by Lin *et al.* [7]. Moreover, while many methods for estimating probabilities with the "one against one" strategy have been proposed [8, 9, 10], probability estimation with the "one against all" strategy has not, to the best of our knowledge, been studied. The likely reason for this is that, with this strategy, mapping the outputs of each SVM separately seems sufficient for estimating multi-class probabilities; but, as we will see, this is not necessarily the best solution.

The remainder of this paper is organized as follows. Section 2 presents our experimental protocol (database, baseline classifier, and comparison criteria). Section 3 describes the "one against all" strategy, and compares two post-processing methods to estimate posterior probability. Section 4 describes the "one against one" strategy, and compares three methods for combining the probabilities by each SVM. Finally, section 5 compares the two strategies in terms of complexity and accuracy, and section 6 concludes with some practical suggestions.

2. Experimental Protocol

The experiments were conducted on a personal computer with 1.9 GHz CPU and 1 Go of RAM. All the SVMs were trained with the LIBSVM software [11]. We used the C-SVM with a Gaussian kernel. The kernel parameter γ and the regularization parameter C were empirically optimized by minimizing the error rate on the validation dataset.

2.1. Database

We used the NIST-SD19 database [12], which contains the full-page binary images of Handprinted Sample Forms (HSF) from 3,600 writers. In our experiments, we used only the images of isolated handwritten digits and uppercase letters. The number of samples in each dataset is reported in Table 1. The training datasets contain exactly the same number of examples in each class, which are the first images from the hsf_{0,1,2,3} corpus. The validation datasets are composed of the remaining images from hsf_{0,1,2,3} for the digit database and of all images from the hsf_4 corpus for the letter database. Finally, the testing datasets are composed of all the images from the hsf_7 corpus.

Table 1. Number of samples in each dataset.

	Digit	Letter
Training	195,000	43,160
Validation	28,123	11,941
Testing	60,089	12,092

We chose to use the same feature extraction procedure as Oliveira *et al*. [13]. Indeed, this feature space has been used on the same digit dataset and made it possible to obtain an accurate classification. According to this method, each image is divided into six zones: 3 rows and 2 columns. In each zone, 22 components are extracted: 13 concavity measures, 8 corresponding to the histogram of the contour directions and one related to the surface of the character. Finally, we obtained 132 discriminative features, normalized to between 0 and 1.

2.2. Baseline Classifier

We have elected to use a Multi-Layer Perceptron (MLP) as the baseline classifier because this type of artificial neural network makes it possible to estimate accurate posterior probabilities and has been widely incorporated into handwriting recognition systems.

For our experimentation, we used the same type of topology as Oliveira *et al*. [13], in which an MLP is used for recognizing handwritten numerical strings. The network used has one hidden layer. The neurons of the input and the output layers are fully connected to the neurons of the hidden layer, and the transfer function is the sigmoid function. Furthermore, the network is trained with a sequential gradient descent with momentum applied to a sum-of-squares error function.

The error rates obtained with MLP are reported in Table 2 and compared with another classical classifier: the *k*-Nearest Neighbor.

Table 2. Error rate obtained on the testing dataset.

	Digit	Letter
k-NN	1.35%	7.60%
MLP	0.80%	3.81%

Let us note that the number of hidden neurons ($h = 80$ for digit, and $h = 100$ for letter) and the number of neighbors ($k = 1$ for digit, and $k = 3$ for letter) are fixed using the validation dataset.

2.3. Comparison Criteria

To compare the different approaches in terms of accuracy, the simplest way would be to evaluate the error rate on the testing dataset, but this value is often not accurate enough. For this reason, an error function is generally used for comparing the various probability estimates. We chose to use the negative log-likelihood:

$$-\sum_{k=1}^{n} \log\left(\hat{P}\left(\omega_k \mid x_k\right)\right), \qquad (1)$$

where ω_k denotes the label of the sample x_k.

In addition, we propose to use a third measure based on the reject option. Indeed, if the posterior probabilities of the data classes are known exactly, then, as Chow demonstrated in [14], the optimal reject option is to reject a sample *x* if:

$$\max_{j=1,\ldots,c} \left(P(\omega_j \mid x)\right) < T\,. \qquad (2)$$

Then, the threshold *T* defines the rate of samples rejected and consequently the error rate among the samples accepted. Thus, a complete description of recognition performance is given by the error-reject tradeoff, which is obtained by varying *T*. An example is shown in Figure 1. However, in real applications, such probabilities are affected by significant estimate errors, and the better the probabilities estimate is, the better the error-reject tradeoff is. Thus, we propose to evaluate the rejection rate necessary to decrease the error rate to a specific value (0.1% for digits and 0.5% for letters).

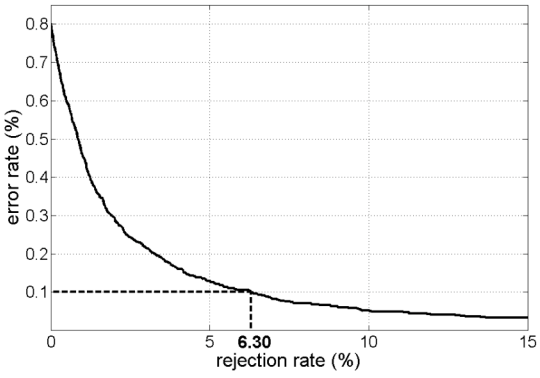

Figure 1: Error-reject tradeoff obtained with the baseline classifier on the digit dataset.

3. The "One Against All" Strategy

3.1. Description

The "one against all" strategy consists of constructing one SVM per class, which is trained to distinguish the samples of one class from the samples of all remaining classes. Usually, classification of an unknown pattern is done according to the maximum output among all SVMs.

3.2. Probability Estimation

The most intuitive approach to estimate posterior probability with the "one against all" strategy is to separately map the outputs of each SVM into probability using the method proposed by Platt [6], which consist of using an additional sigmoid:

$$\hat{P}\left(\omega_j \mid f_j(x)\right) = \frac{1}{1+\exp\left(A_j\, f_j(x) + B_j\right)}, \qquad (3)$$

where $f_j(x)$ denotes the output of the SVM trained to separate the class ω_j from all the others. Then, for each sigmoid the parameters A_j and B_j are optimized by minimizing the local negative log-likelihood:

$$-\sum_{k=1}^{n} t_k \log\left(p_k\right) + \left(1-t_k\right)\log\left(1-p_k\right), \qquad (4)$$

where, p_k denotes the output of the sigmoid and t_k the probability target. To solve this optimization problem, Platt [6] proposes using a model-trust minimization algorithm based on the Levenberg-Marquardt algorithm. However, Lin *et al.* [7] showed that there are some problems with this algorithm and propose using another minimization algorithm based on Newton's method with backtracking line search.

However, nothing guarantees that:

$$\sum_{j=1}^{c} \hat{P}\left(\omega_j \mid f_j(x)\right) = 1. \qquad (5)$$

For this reason, it seems preferable to normalize the probabilities as follows:

$$\hat{P}\left(\omega_j \mid x\right) = \frac{\hat{P}\left(\omega_j \mid f_j(x)\right)}{\sum_{j'=1}^{c} \hat{P}\left(\omega_{j'} \mid f_{j'}(x)\right)}. \qquad (6)$$

Another approach to estimate posterior probability with the "one against all" strategy would be to exploit the outputs of all SVMs to estimate overall probabilities. In order to do this, we propose using the softmax function, which can be regarded as a generalization of the sigmoid function for the multi-class case. Thus, in the same spirit as Platt's algorithm, we use a parametric form of the softmax function:

$$\hat{P}(\omega_j \mid x) = \frac{\exp\left(A_j f_j\left(x\right) + B_j\right)}{\sum_{j'=1}^{c} \exp\left(A_{j'} f_{j'}\left(x\right) + B_{j'}\right)}, \qquad (7)$$

and derive the parameters A_j and B_j by minimizing the global negative log-likelihood (Eq. 1).

Thus, it is necessary to construct a dataset of SVM outputs, which will be used to fix the parameters of sigmoid and softmax functions. The easiest way to do this is to use the same training samples used to fit SVMs; but, as pointed out by Platt [6], using the same data twice, can sometimes lead to a disastrously biased estimate. Therefore, it is preferable to derive an unbiased training set of the SVM outputs. A first solution would be to use a validation dataset; but, in our case, the number of samples in each class is not proportional to the prior probability. For this reason, it seems preferable to use cross-validation. Then, the training dataset was split into four parts. Each of four SVMs is trained on permutations of three out of four parts, and the SVM outputs are evaluated on the remaining fourth part. Finally, the union of all four sets of SVM outputs forms an unbiased dataset, which can be used to fix the parameters of functions. Furthermore, once the parameters are fixed, the final SVM is trained on the entire training set.

3.3. Experimental Results

Firstly, we tested the "one against all" strategy with the classical decision making, which directly exploits the SVM outputs. The error rates obtained on the testing datasets are 0.63% with digits and 3.24% with letters. Thereafter, we implemented the two approaches for mapping the SVM outputs into probabilities. The results obtained on the testing dataset are reported in Table 3 and Table 4. Considering these results, a number of remarks can be derived. First, the two post-processing methods allow a slight reduction of the error rate on the letter dataset, while no improvement is observed on the digit dataset. Second, as we thought, the error rate is not accurate enough in comparing the various estimates. Indeed, while the error rates obtained with the two methods are similar, the rejection rates and the negative log-likelihood (NLL) are significantly different. Finally, it appears that it is better for posterior probability estimation to optimize globally a softmax function than locally several sigmoid functions.

Table 3. Results obtained with the "one against all" strategy on the digit dataset.

	error rate *- no rejection -*	rejection rate *- 0.1% of error -*	NLL
sigmoid	0.64%	3.73%	1,517
softmax	0.63%	2.30%	1,310

Table 4. Results obtained with the "one against all" strategy on the letter dataset.

	error rate *- no rejection -*	rejection rate *- 0.5% of error -*	NLL
sigmoid	3.17%	13.17%	1,570
softmax	3.18%	10.07%	1,375

4. The "One Against One" Strategy

4.1. Description

The "one against one" strategy, also known as "pairwise coupling", "all pairs" or "round robin", consists in constructing one SVM for each pair of classes. Thus, for a problem with c classes, $c(c\text{-}1)/2$ SVMs are trained to distinguish the samples of one class from the samples of another class. Usually, classification of an unknown pattern is done according to the maximum voting , where each SVM votes for one class.

4.2. Probability Estimation

After mapping the output of each SVM into probability with a sigmoid function, the task is to express the global posterior probabilities $\hat{P}(\omega_j \mid x)$ as functions of the local posterior probabilities $\hat{P}\left(\omega_j \mid f_{j,j'}(x)\right)$, where $f_{j,j'}(x)$ denotes the output of the SVM trained to distinguish class ω_j from class $\omega_{j'}$. Various methods have been proposed in the literature. We chose to compare three of these:

- **Method 1**

Price *et al.* [8] considered that for all classes ω_j:

$$\sum_{j',j'\neq j} P(\omega_{j,j'} \mid x) - (c-2)P(\omega_j \mid x) = 1, \tag{8}$$

where $\omega_{j,j'}$ denotes the union of classes ω_j and $\omega_{j'}$. Then, using:

$$\hat{P}\left(\omega_j \mid f_{j,j'}(x)\right) \approx \frac{P(\omega_j \mid x)}{P(\omega_{j,j'} \mid x)}, \tag{9}$$

it is possible to derive the following expression:

$$\hat{P}(\omega_j \mid x) = \frac{1}{\sum_{j',j'\neq j} \frac{1}{\hat{P}\left(\omega_j \mid f_{j,j'}(x)\right)} - \left(c-2\right)}. \tag{10}$$

However, since nothing guarantees that the sum of all the probabilities is 1, we must divide each estimate $\hat{P}(\omega_j \mid x)$ by $\sum_{j=1}^{c} \hat{P}(\omega_j \mid x)$.

- **Method 2**

In a different way, Hastie & Tibshirani [9] proposed using an iterative algorithm to estimate the posterior probabilities $p_j = \hat{P}(\omega_j \mid x)$, which minimizes the Kullback-Leibler distance between $r_{jj'} = \hat{P}\left(\omega_j \mid f_{j,j'}(x)\right)$ and $\mu_{jj'} = \dfrac{p_j}{p_j + p_{j'}}$, that is:

$$\sum_{j<j'} n_{jj'} \left(r_{jj'} \log \frac{r_{jj'}}{\mu_{jj'}} + (1 - r_{jj'}) \log \frac{1 - r_{jj'}}{1 - \mu_{jj'}} \right), \tag{11}$$

where $n_{jj'}$ denotes the number of training samples in the classes ω_j and $\omega_{j'}$. To this end, they start with simple non-iterative estimates:

$$p_j = \frac{2}{c(c-1)} \sum_{j',j'\neq j} r_{jj'}, \tag{12}$$

and repeat $(j = 1, 2, \ldots, c, 1, \ldots)$ until convergence:

1. $\mu_{jj'} \leftarrow \dfrac{p_j}{p_j + p_{j'}}, \ \forall j' \neq j$

2. $p_j \leftarrow p_j \cdot \dfrac{\sum_{j'\neq j} n_{jj'} r_{jj'}}{\sum_{j'\neq j} n_{jj'} \mu_{jj'}}$

3. $p_j \leftarrow \dfrac{p_j}{\sum_{j'=1}^{c} p_{j'}}$

In practice, we used the following stopping condition:

$$\sum_{j=1}^{c} \left(p_j(t) - p_j(t-1) \right)^2 \leq 10^{-12}, \tag{13}$$

where $p_j(t)$ denotes the actual values of p_j and $p_j(t-1)$ the previous values of p_j.

- **Method 3**

More recently, Hamamura *et al.* [10] proposed a combination based on the assumption that discriminant functions are independent of one another. Then, since prior probabilities are all the same, posterior probabilities can be estimated by:

$$\hat{P}(\omega_j \mid x) = \frac{\prod_{j',j'\neq j} \hat{P}\left(\omega_j \mid f_{j,j'}(x)\right)}{\sum_{j''=1}^{c} \prod_{j',j'\neq j''} \hat{P}\left(\omega_{j''} \mid f_{j'',j'}(x)\right)}. \tag{14}$$

4.3. Experimental Results

Firstly, we tested the "one against one" strategy with the classical voting rule. The error rates obtained on the testing datasets are 0.71% with digits and 3.29% with letters. Thereafter, we implemented the three methods for combining probabilities. The results obtained on the testing dataset are reported in Table 5 and Table 6.

Considering these results, two remarks can be made. First, the last method is less accurate than the first two, which yield significantly better rejection rates and negative log-likelihoods on the two datasets. Second, although the results obtained with the first two methods are comparable, the first method seems slightly more accurate than the second. Indeed, the first method yields better error rate on the letter dataset and better negative log-likelihoods on the two datasets. Moreover, the first method has the advantage of being non-iterative and is thus faster than the second method.

Table 5. Results obtained with the "one against one" strategy on the digit dataset.

	error rate *- no rejection -*	rejection rate *- 0.1% of error -*	NLL
method 1	0.70%	3.49%	1,483
method 2	0.69%	3.37%	1,604
method 3	0.70%	4.31%	1,825

Table 6. Results obtained with the "one against one" strategy on the letter dataset.

	error rate *- no rejection -*	rejection rate *- 0.5% of error -*	NLL
method 1	3.22%	11.22%	1,421
method 2	3.37%	11.36%	1,548
method 3	3.29%	13.89%	2,197

5. Comparison of the two strategies

We can now try to answer to the question raised in our title. With this in mind, we compared the two strategies in terms of accuracy, but also in terms of complexity.

5.1. Comparison in terms of accuracy

The results obtained on the testing dataset are reported in Table 7 and Table 8, and the error-reject tradeoffs are shown in Figure 2 and Figure 3. Concerning the "one against all" strategy (OAA) the softmax function is used for probability estimation, while the method 1 is used for combining the probabilities of the "one against one" strategy (OAO).

Table 7. Comparison in terms of accuracy on the digit dataset.

	error rate *- no rejection -*	rejection rate *- 0.1% of error -*	NLL
MLP	0.80%	6.30%	2,591
SVM - OAO	0.70%	3.49%	1,483
SVM - OAA	0.63%	2.30%	1,310

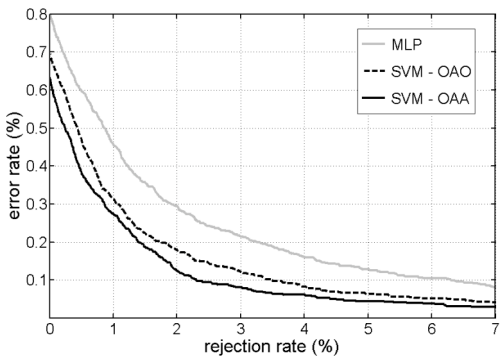

Figure 2: Comparison of the error-reject tradeoff obtained on the digit dataset.

Table 8. Comparison in terms of accuracy on the letter dataset.

	error rate *- no rejection -*	rejection rate *- 0.5% of error -*	NLL
MLP	3.81%	22.83%	2,923
SVM - OAO	3.22%	11.22%	1,421
SVM - OAA	3.18%	10.07%	1,375

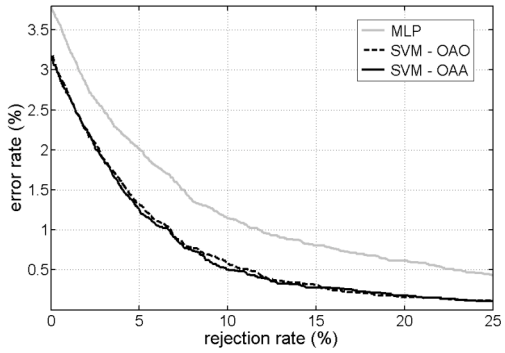

Figure 3: Comparison of the error-reject tradeoff obtained on the letter dataset.

Considering these results, two conclusions can be made. First, in agreement with the literature, SVMs allow more accurate classification than a classical MLP. Second, in disagreement with Allwein *et al.* [3], in our experiments, the "one against all" strategy is more accurate than the "one against one". Indeed, if the difference between the two strategies is small on the letter dataset, it is significant on the digit dataset.

5.2. Comparison in terms of complexity

Two types of complexity must be considered:

- **The complexity of the training process**

It can seem logical that the total training time with the "one against one" strategy is larger than with the "one against all", because it is necessary to train more binary classifiers; but it is not necessarily true when the binary classifiers are SVMs. Indeed, the training time of an SVM increases considerably with the number of training samples. Thus, since each sub-problem involves a small number of training samples and is easier to solve, it is generally quicker to train the $c(c\text{-}1)/2$ SVMs of the "one against one" strategy than the c SVMs of the "one against all" strategy. In ours experiments, the total training time is approximately divided by 12 with letters (see Table 9) and by 50 with digits (see Table 10).

- **The complexity of the decision making process**

Again, it can seem logical that the decision making with the "one against one" strategy is more complex than with the "one against all", because it is necessary to evaluate more decision functions; but, as previously, it is not necessarily true with SVMs. Indeed, the complexity of an SVM's decision making is directly linked to the

number of support vectors (SVs), and although the decision making is more complicated in the multi-class case, it is reasonable to consider that the complexity is proportional to the total number of support vectors[1]. However, in our experiments the "one against all" strategy uses more support vectors than the "one against one" (48% more for digits, and 21% more for letters).

Table 9. Comparison of the two strategies in terms of complexity on the digit dataset.

	OAO	OAA
number of SVMs	45	10
total training time	36 min.	32 h 17
number of SVs	5,753	8,514

Table 10. Comparison of the two strategies in terms of complexity on the letter dataset.

	OAO	OAA
number of SVMs	325	26
total training time	4 min.	51 min.
number of SVs	9,152	11,109

6. Conclusion

Finally, our answer to the title question will depend on what the problem is! Indeed, it is not reasonable to claim that one strategy is always better that the other; but according to the application constraints, the number of classes, and the number of training samples, it will be one or the other of the two strategies that will be more suitable to solve the classification problem. Thus, considering the conclusions of the previous section, some suggestions can be made as to which strategy is best for a specific problem. For problems with few classes, like digit recognition, the "one against all" strategy seems significantly more accurate; while for problems with more classes, like Latin letter recognition, the difference of accuracy between the two strategies seems much less significant. Lastly, for problems with a very large number of classes, like Chinese or Japanese ideogram recognition, we suspect that the unbalance of the number of the samples causes problem with the "one against all" strategy, especially when it has few training samples per class. Moreover, the "one against one" strategy, which is more modular, is more suitable for speeding up the decision making process by combining with other classifiers [15]. Furthermore, if the number of training samples is very large, the training time can become problematic, and then the "one against one" strategy appears more suitable for practical use. To conclude, we have shown in this paper that appropriate post-processing make it possible to estimate accurate posterior probabilities with SVMs. Thus, these promising results open the way to new perspectives with respect to incorporating SVMs into handwriting recognition systems.

[1] Notes that in the multi-class case, the total number of support vectors is not necessarily equal to the sum of the number of support vectors of each SVM, because a training sample can be a support vector for several SVMs.

References

[1] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition", *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998.

[2] T.G. Dietterich and G. Bakiri, "Solving Multiclass Learning Problems via Error-Correcting Output Codes", *Journal of Artificial Intelligence Research*, vol. 2, pp. 263-286, 1995.

[3] E.L. Allwein, R.E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers", *Journal of Machine Learning Research*, vol. 1, pp. 113-141, 2000.

[4] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines", *IEEE transactions on Neural Networks*, vol. 13, pp. 415-425, 2002.

[5] R. Rifkin and A. Klautau, "In defence of one-vs-all classification", *Journal of Machine Learning Research*, vol. 5, pp. 101-141, 2004.

[6] J.C. Platt, "Probabilities for SV Machines", in *Advances in Large Margin Classifiers*, A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, Eds., MIT Press, pp. 61-74, 1999.

[7] H.-T. Lin, C.-J. Lin, and R.C. Weng, "A note on Platt's probabilistic outputs for support vector machines", Technical Report, National Taiwan University, 2003.

[8] D. Price, S. Knerr, L. Personnaz, and G. Dreyfus, "Pairwise Neural Network Classifiers with Probabilistic Outputs", in *Neural Information Processing Systems*, MIT Press, pp. 1109-1116, 1995.

[9] T. Hastie and R. Tibshirani, "Classification by pairwise coupling", *The Annals of Statistics*, vol. 26, pp. 451-471, 1998.

[10] T. Hamamura, H. Mizutani, and B. Irie, "A multiclass classification method based on multiple pairwise classifiers", *International Conference on Document Analysis and Recognition*, pp. 809-813, Edinburgh, Scotland, August 3-6, 2003.

[11] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines", Technical Report, National Taiwan University, 2001.

[12] P.J. Grother, "NIST Special Database 19-Handprinted Forms and Characters Database", Technical Report, National Institute of Standards and Technology, 1995.

[13] L.S. Oliveira, R. Sabourin, F. Bortolozzi, and C.Y. Suen, "Automatic recognition of handwritten numerical strings: a recognition and verification strategy", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1438-1454, 2002.

[14] C.K. Chow, "On optimum recognition error and reject tradeoff", *IEEE Transactions on Information Theory*, vol. 16, pp. 41-46, 1970.

[15] J. Milgram, M. Cheriet, and R. Sabourin, "Speeding Up the Decision Making of Support Vector Classifiers", *International Workshop on Frontiers in Handwriting Recognition*, pp. 57-62, Tokyo, Japan, October 26-29, 2004.