# Support Vector Machines for Handwritten Numerical String Recognition

Luiz S. Oliveira and Robert Sabourin
Pontifícia Universidade Católica do Paraná, Curitiba, Brazil
Ecole de Technologie Supérieure - Montreal, Canada
soares@ppgia.pucpr.br, robert.sabourin@etsmtl.ca

## Abstract

*In this paper we discuss the use of SVMs to recognize handwritten numerical strings. Such a problem is more complex than recognizing isolated digits since one must deal with problems such as segmentation, overlapping, unknown number of digits, etc. In order to perform our experiments, we have used a segmentation-based recognition system using heuristic over-segmentation. The contribution of this paper is twofold. Firstly, we demonstrate by experimentation that SVMs improve the overall recognition rates. Secondly, we observe that SVMs deal with outliers such as over- and under-segmentation better than multi-layer perceptron neural networks.*

**Keywords:** Handwritten numerical string recognition, heuristic over-segmentation, Support Vector Machines.

## 1 Introduction

In the last years, Support Vector Machines (SVMs) have gained a lot of attention of machine learning and pattern recognition communities. They have been successfully applied to several different areas ranging from face verification and recognition, speaker verification, text categorization, prediction, image retrieval, and handwriting recognition. For a recent review, please see [7]. Those who advocate in favor of SVMs argue that they generalize well even in high dimensional spaces under small training set conditions and have shown to be superior to traditional empirical risk minimization principle employed by most of neural networks.

Those who advocate against SVMs, on the other hand, say that they are very expensive in learning and recognition [17]. Indeed, in terms of running time, SVMs are slower than neural networks for a similar generalization performance. In addition, some authors [5, 7] argue that the performance of SVMs largely depends of the choice of kernels and also that multi-class SVM classifier is still an open problem.

To overcome such problems, a lot of research have been done on computational issues such as speed [11, 19], large-scale problems [8], kernels [23, 22], multi-class SVMs [13], etc. In light of this, several authors have been taken advantage of these advances and applied SVMs to solve handwriting recognition problems, more specifically, the handwritten digit recognition problem. In this paper we discuss the use of SVMs to recognize handwritten numerical strings. Such a problem is more complex than recognizing isolated digits since one must deal with problems such as segmentation, overlapping, unknown number of digits, etc. We have used a segmentation-based recognition system using heuristic over-segmentation to perform our experiments. The contribution of this paper is twofold. Firstly, we demonstrate by experimentation that SVMs improve the overall recognition rates. Secondly, we observe that SVMs deal with outliers such as over- and under-segmentation better than multi-layer perceptron neural networks.

The remaining of this work is organized as follows: Section 2 presents a brief review about SVMs to recognize isolated digits. Section 3 introduces the handwritten string digit recognition problem and the concept of outlier as well. Section 4 presents an overview of SVMs. Section 5 summarizes our experimental results while Section 6 concludes this work.

## 2 A Review on SVMs for Handwritten Digit Recognition

As stated before, the problem of handwritten digit recognition has been used to assess SVM-based clas-

**Table 1. Performance of SVM-based classifiers on handwritten digit recognition.**

| Author | Database | Tr Size | Test Size | Error Rate |
|---|---|---|---|---|
| Krebel et al, 1998 [13] | NIST | 10000 | 10000 | 1.09 |
| Ayat et al, 2002[1] | NIST | 18000 | 10000 | 1.02 |
| Scholkopf et al, 1996[21] | USPS | 7291 | 2007 | 3.20 |
| Dong et al, 2002 [11] | USPS | 7291 | 2007 | 2.24 |
| LeCun et al, 1998 [14] | MNIST | 60000 | 10000 | 1.10 |
| Li et al, 2002 [15] | MNIST | 60000 | 10000 | 0.76 |
| DeCoste and Scholkopf, 2003 [10] | MNIST | 60000 | 10000 | 0.56 |
| Liu et al, 2002 [16] | MNIST | 60000 | 10000 | 0.42 |
| Liu et al, 2002 [16] | CEDAR | 18468 | 2711 | 0.63 |
| Liu et al, 2002 [16] | CENPARMI | 4000 | 2000 | 1.10 |

sifiers since the introduction of Vapnik's book [25]. By reviewing the literature, we can find several variations of SVMs as well as results on several different databases. Table 1 summarizes some works found in the literature.

Perhaps, the most used benchmark to evaluate SVMs is MNIST, which is a modified version of NIST database and was originally set up by the AT&T group [14]. This database contains 60,000 and 10,000 $28 \times 28$ images for training and testing, respectively, and have been used by machine learning and pattern recognition communities. The former, usually takes into account the raw grey-level image to feed the classifier, since their goal is to assess the technique being applied rather than improve the performance on a given database. The pattern recognition community, is more preoccupied in achieving performance. For this reason, they emphasizes the use of prior knowledged about symmetries of the problem (i.e., feature extraction) to reach better results. This explains the different results reported in Table 1 for MNIST.

Liu et al [16] show a comparative study on handwritten digit recognition using different classifiers and databases. They conclude that SVMs using Gaussian kernel outperform all traditional techniques such as neural networks (MLP and RBF), polynomial classifiers, and learning quadratic discriminant functions. Nevertheless, they point out that memory space and computational speed for classification still are important issues to be considered when discussing SVMs. In light of this, some authors have proposed using SVMs for verification rather than classification [2]. In such cases, SVMs are used just when the result of the classifier is not so reliable. This strategy is computationally cheaper once SVMs are called just to solve difficult cases.

## 3 Handwritten Digit String Recognition

The system used as baseline is depicted in Figure 1. It takes a segmentation-based recognition with an heuristic over-segmentation, where the classifier and verifiers are the well-known Multilayer Perceptrons (MLPs). The approach combines the outputs from different levels such as segmentation, recognition, and postprocessing in a probabilistic model, which allows a sound integration of all knowledge sources used to infer a plausible interpretation. For a complete description of this system, please see [18].

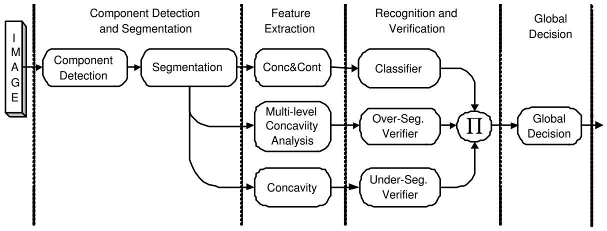
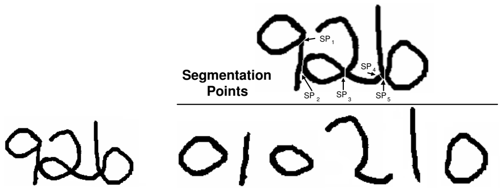


**Figure 1. Block diagram of the digit string recognition system.**

The literature shows that this kind of system produces good results, however, it has to deal with outliers such as over- and under-segmentation. Such outliers are by-product of the segmentation process and sometimes they are very similar to digits. Figure 2 shows an example of over-segmentation, where without any contextual information, some over-segmented pieces (Figure 2b) could be easily classified as digits.

It has been demonstrated that MLPs are not robust enough to deal with these outliers [12]. For this reason, several techniques have been investigated to improve the resistance of MLPs to outliers [17, 4]. The forego-



(a) (b)

**Figure 2. Example of over-segmentation: (a) Original string and (b) over-segmented pieces.**

ing system applies the concept of verifiers, which are "plugged" into the system to detect outliers. Table 2 reports the results produced by the system described in [18] on NIST SD19. We have used 12,802 strings of digits with lengths ranging from 2 to 10. It can be observed that the results achieved without the two verifiers are very poor, but they are considerably improved by the verifiers. We will demonstrate in the remaining of this paper that SVMs are more robust than MLP to recognize string of digits in the context of over-segmentation. It is worth of remark that, to the knowledge of the authors, these results are the state of the art for this database.

**Table 2. Recognition rates on NIST database.**

| String Length | Nb. of Strings | Rec. Rate (%) Without verifiers | Rec. Rate (%) With verifiers |
|---|---|---|---|
| 2 | 2370 | 91.56 | 96.88 |
| 3 | 2385 | 87.98 | 95.38 |
| 4 | 2345 | 84.91 | 93.38 |
| 5 | 2316 | 82.00 | 92.40 |
| 6 | 2169 | 86.66 | 93.12 |
| 10 | 1217 | 78.97 | 90.24 |

## 4 Overview of Support Vector Machines

In his book, Vapnik [25] proposed a method of finding a hyperplane optimally dividing two classes, which does not depend on a probability estimation. This optimal hyperplane is a linear decision boundary which separates the two classes and leaves the largest margin between the vectors of the two classes. In order to determine the optimal hyperplane, Vapnik´s method uses just a small fraction of the data points, the so-called "support vectors". It has been demonstrated that the probability of making errors depends only on the number of these support vectors (the complexity of the classifier) and the number of the training vectors. However, this method fits only for separable classes.

A extension to nonlinear decision surfaces is necessary since real-life classification problems are difficult to be solved by a linear classifier. This can be achieved using the kernel trick, where every time a linear algorithm uses a dot product, replace it with a non-linear kernel function. This causes the linear algorithm to operate in a different space. For SVMs, using the kernel trick makes the maximum margin hyperplane be fit in a feature space. The feature space is a non-linear map from the original input space, usually of much higher dimensionality than the original input space. In this way, non-linear SVMs can be created. The decision function derived by the SVM classifier for a two-class problem can be formulated, using a kernel function $K(x, x_i)$ of a new example $x$ (to classify) and a training example $x_i$, as follows:

$$f(x) = \sum_i \alpha_i y_i K(x, x_i) + b \tag{1}$$

where the parameters $\alpha_i$ and $b$ are found by maximizing a quadratic function (maximum margin algorithm [25]) while $y_i$ is the label of example $x_i$. Table 3 summarizes the most common kernels.

**Table 3. Summary of common kernels**

| Kernel | Inner Product Kernel |
|---|---|
| Linear | $K(x, y) = (x \cdot y)$ |
| Gaussian | $K(x, y) = \exp\left(-\frac{\|x-x_i\|^2}{2\sigma^2}\right)$ |
| Polynomial | $K(x, y) = (x \cdot y)^p$ |
| Tangent Hyperbolic | $K(x, y) = \tanh(x \cdot y - \Theta)$ |

Besides optimizing the kernel parameters (such as $\sigma$ in a Gaussian kernel), one should consider the trade-off parameter $C$. It indicates how severely errors have to be punished. The choice of $C$ may have a strong effect on the behavior of the classifier for difficult classification problems, e.g., if the errors are punished too much, the SVMs can overfit the training data.

Since SVM is primarily a binary classifier, it should be extended to deal with $q$-class (where $q > 2$) pattern recognition problems such as digit recognition. There are two basic approaches to solve $q$-class problems with SVMs: pairwise and one-against-others. In the former, the pairwise classifiers are arranged in trees, where each tree node represents a SVM. For a given test sample, it is compared with each two pairs, and the winner will be tested in an upper level until the top of the tree (see Figure 3). In this strategy, the number of classifiers we have to train is $q(q-1)/2$ (e.g., 45 in the case of digit recognition where $q = 10$).

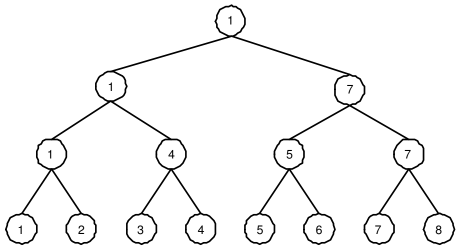


**Figure 3. Example of pairwise SVM. The numbers 1-8 encode the classes.**

The second strategy is the one-against-others decomposition, which works by constructing an SVM $\omega_i$ for each class $q$ that first separates that class from all the other classes and then uses an expert $F$ to arbitrate between each SVM output in order to produce the final decision. The most common arbitrator is the arg max. Let $h = (h_1, \ldots, h_Q)^T$ be the output of a system of $Q$ one-against-others SVMs, the arg max picks class $q$ for the input $x$, which then maximizes $h_q$ is defined as:

$$F = \arg\max(h) \tag{2}$$

However, this kind of decision strategy suffers from a scaling problem once it assumes that all the SVMs produce outputs on the same scale, which is not true. If the SVMs are trained to produce outputs for the support vectors as ±1, the scale is not robust since it only depends on a few data, often including outliers. Therefore, before comparing the outputs, they need to be normalized. In light of this, let $s(h)$ be the normalized output of a system of $Q$ one-against-others SVMs, the decision rule is defined as:

$$F = \arg\max(s(h)) \tag{3}$$

### 4.1 Estimating probabilities with SVM

As stated in the previous section, SVMs produce an uncalibrated value that is not a probability. There is several situations where would be very useful to have a classifier producing a posterior probability $P(class|input)$. In our case, particulary, we are interested in estimation of probabilities because the baseline system presented in Figure 1 was built on a probabilistic framework.

Due to the benefits of having classifiers estimating probabilities, many researchers have been working on the problem of estimating probabilities with SVM classifiers. Sollich in [24] proposes a Bayesian framework to obtain estimation of probabilities and to tune the hyper-parameters as well. His method interprets SVMs as maximum a posteriori solutions to inference problems with Gaussian process priors. Wahba et al [26] use a logistic function of the form

$$P(y = 1|f(x)) = \frac{1}{1 + \exp(-f(x))} \tag{4}$$

where $f(x)$ is the SVM output and $y = \pm 1$ stands for the target of the data sample $x$. In the same vein, Platt [20] suggests a slightly modified logistic function, defined as:

$$P(y = 1|f(x)) = \frac{1}{1 + \exp(Af(x) + B))} \tag{5}$$

The difference lies in the fact that it has two parameters trained discriminatively, rather one parameter estimated from a tied variance. The parameters A and B of Equation 5 are found by minimizing the negative log likelihood of the training data, which is a cross-entropy error function.

## 5 Experiments and Discussion

In order to show the robustness of SVMs to recognize strings of digits, we have used them into the system presented in Section 3. As we can see, the classification module of such a system is composed of three sub-modules: classifier, over-segmentation verifier, and under-segmentation verifier. The first is responsible for recognizing the ten numerical classes, while the other two are responsible for detecting outliers, such as over- and under-segmentation. Then, the results are combined in a probabilistic framework.

In a first moment, we have kept the MLP-based verifiers and replaced the main classifier by ten SVMs combined trough the one-against-others strategy. We have also tried a pairwise approach, but in our experiments we have got better results using one-against-others. We have also tried different kernel models, namely, Gaussian, Polynomial, and Tangent Hyperbolic. The first one produced better results in our experiments. The SVMs were trained by using TORCH [9], which is a machine-learning library developed at IDIAP.

In light of this, ten SVMs were trained on 195,000 samples of the NIST SD19. The feature set [18], which contains 132 components, is based on a mixture of concavity and contour measures. In order to estimate the parameters of the SVMs we have considered a validation set composed of 28,000 samples. The best parameters found were $\sigma = 1.15$ and $C = 1000$.

Thereafter, we have used the approach proposed by Platt [20] to transform the scores provides by the SVMs

**Table 4. Recognition rates on NIST database using SVMs (NV: Without verifiers, V: With verifiers.)**

| String | Number | MLP-based system | | SVM-based system | | Rec. Rate |
|---|---|---|---|---|---|---|
| Length | of | Rec. Rate | Rec. Rate | Rec. Rate | Rec. Rate | published |
| | Strings | NV | V | NV | V | in [3] |
| 2 | 2370 | 91.56 | 96.88 | 96.07 | 97.67 | 94.8 |
| 3 | 2385 | 87.98 | 95.38 | 93.19 | 96.26 | 91.6 |
| 4 | 2345 | 84.91 | 93.38 | 90.89 | 94.28 | 91.3 |
| 5 | 2316 | 82.00 | 92.40 | 90.50 | 94.00 | 88.3 |
| 6 | 2169 | 86.66 | 93.12 | 92.15 | 93.80 | 89.1 |
| 10 | 1217 | 78.97 | 90.24 | 89.87 | 91.38 | 86.9 |

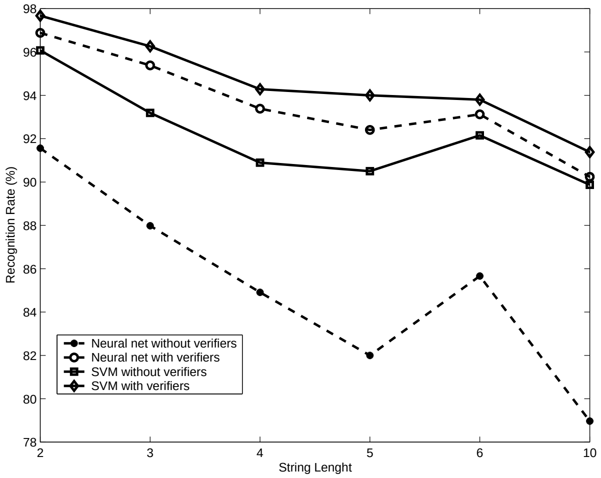


**Figure 4. Comparison between the SVM- and neural-net-based systems.**

into estimation of probabilities. In order to fit the sigmoid of Equation 5 we have used the same training set we have used to fit the SVMs. Platt has pointed out that using the same data twice, sometimes can lead to biased fits. However, we did not observe this phenomenon in our experiments.

The recognition rate achieved by the SVMs on the test set, which is composed of 60,089 samples of hsf_7, was 99,20%. This rate was very close to that reached by the original classifier, an MLP that got 99,13% on the same data set. The results on strings of digits are summarized in Table 4. Note that "SVM-based system" means that the main classifier is composed of ten SVMs while the two verifiers are MLP-based.

By comparing the results reported in Table 4, we can notice that the gap between the results is much smaller when considering the system with SVMs. This means that the SVM-based system can deal better with outliers such as over- and under-segmentation, i.e., it has more outlier resistance than the neural-net-based system. In spite of this better resistance, we can observe that the verifiers still are important pieces in the system, since they improve the results in about 3% (in average). Figure 4 depicts the results presented in the foregoing tables. We can see that the gap between the SVM-based systems is much smaller than the gap between the neural-net-based system.

On the other hand, the neural-net-based system is faster during the test phase. As pointed out by other authors [6, 16], speed for large data sets is still a issue for SVMs. However, a lot of efforts have been made in this direction, so that, we believe SVMs will be more viable in a near future. Table 4 also compares our results to the work published by Britto et al in [3]. The comparison here becomes interesting since both systems have been tested on the same database.

To conclude our experiments, we have replaced the MLP-based verifiers by SVMs as well. In such a case, both verifiers are binary classifiers, since they discriminate between digit and over-segmentation (over-segmentation verifier) and digit and under-segmentation (under-segmentation verifier). The results achieved by the MLP-based over-segmentation verifier and MLP-based under-segmentation verifier are 99.40% and 99.17%, respectively. The SVM-based verifiers reached very similar results. When using these new verifiers into the system, the results were practically the same.

## 6 Conclusion

So far, a lot of efforts have been published in the literature about SVMs, where the benchmarks very often are isolated handwritten digit recognition. In this paper, we have investigated the use of SVMs to recognize strings of digits, which is a more complicated problem. We demonstrated through experimentation that the proposed strategy (i.e., one-against-others SVMs estimating probabilities using Platt´s methods) can surpass the results produced by the baseline system, which is based on MLP classifiers. Other important contribution of this work, is to show that SVMs are suitable for systems based on explicit segmentation, since they can deal with outliers better than neural nets.

## Acknowledgements

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 150542/2003-8.

## References

[1] N. E. Ayat, M. Cheriet, and C. Y. Suen. Optimization of the svm kernels using an empirical error minimization scheme. In *Proc. of the International Workshop on Pattern Recognition with Support Vector Machine*, pages 354–369, 2002.

[2] A. Bellili, M. Gilloux, and P. Gallinari. An hybrid MLP-SVM handwritten digit recognizer. In *Proc. of $6^{th}$ International Conference on Document Analysis and Recognition*, pages 28–31, Seattle, USA, 2001.

[3] A. S. Britto, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Recognition of handwritten numeral strings using a two-stage HMM-Based method. *International Journal on Document Analysis and Recognition*, 5(2-3):102–117, 2003.

[4] J. Bromley and J. S. Denker. Improving rejection performance on handwritten digits by training with rubbish. *Neural Computation*, 5(3):367–370, 1993.

[5] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[6] H. Byun and S. W. Lee. Applications of support vector machines for pattern recognition. In *Proc. of the International Workshop on Pattern Recognition with Support Vector Machine*, pages 213–236, 2002.

[7] H. Byun and S. W. Lee. A survey on pattern recognition applications of support vector machines. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(3):459–486, 2003.

[8] R. Collobert, S. Bengio, and Y. Bengio. Parallel mixture of SVMs for very large scale problems. *Neural Computation*, 14(5):1105–1114, 2002.

[9] R. Collobert, S. Bengio, and J. Mariethoz. Torch: A modular machine learning software library. Technical Report 02-46, IDIAP-RR, 2002.

[10] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning Journal*, 46(1-3):161–190, 2002.

[11] J. X. Dong, A. Krzyzak, and C. Y. Suen. A practical SMO algorithm. In *Proc. of $16^{th}$ International Conference on Pattern Recognition (ICPR)*, Quebec City, Canada, 2002.

[12] M. Gori and F. Scarselli. Are multilayer perceptrons adequate for pattern recognition and verification? *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(11):1121–1132, 1998.

[13] U. Krebel. Parwise classification and support vector machines. In B. S. et al, editor, *Advances in Kernel Methods: Support Vector Machines*, pages 255–268. MIT Press, 1998.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Procs of IEEE*, 86(11):2278–2324, 1998.

[15] Z. Li, S. Tang, and S. Yan. Multi-class SVM classifier based on pairwise coupling. In *Proc. of the International Workshop on Pattern Recognition with Support Vector Machine*, pages 321–333, 2002.

[16] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten digit recognition using state-of-the-art techniques. In *Proc. of $8^{th}$ International Workshop on Frontiers of Handwriting Recognition (IWFHR-8)*, pages 320–325, 2002.

[17] C.-L. Liu, H. Sako, and H. Fujisawa. Performance evaluation of pattern classifiers for handwritten character recognition. *International Journal on Document Analysis and Recognition*, 4(3):191–204, 2002.

[18] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(11):1438–1454, 2002.

[19] E. E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. In B. S. et al, editor, *Advances in Kernel Methods: Support Vector Machines*, pages 271–283. MIT Press, 1998.

[20] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. S. et al, editor, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[21] B. Schölkopf, C. J. C. Burges, and V. Vapnik. Incorporating invariances in support vector learning machines. In *International Conference on Artificial Neural Networks (ICANN'96)*, pages 47–52, Berlin, 1996.

[22] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, G. Rätsch, , and A. Smola. Input space vs. feature space in kernel-based methods. *IEEE Trans. on Neural Networks*, 10(5):1000–1017, 1999.

[23] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In B. S. et al, editor, *Advances in Kernel Methods: Support Vector Machines*, pages 327–352. MIT Press, 1998.

[24] P. Sollich. Bayesian methods for support vecotr machines: Evidence and predictive class probabilities. *Machine Learning*, 46(1-3):21–52, 2002.

[25] V. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

[26] G. Wahba, X. Lin, F. Gao, D. Xiang, R. Klein, and B. Klein. The bias-variance trade-off and the randomized GACV. In *Proc. of the $13^{th}$ Conference on Neural Information Processing Systems*, pages 8–31, Vancouver, Canada, 2001.