An Efficient Way of Combining SVMs for Handwritten Digit Recognition

Renata F.P. Neves, Cleber Zanchettin, and Alberto N.G. Lopes Filho

Center of Informatics, Federal University of Pernambuco, Recife, Brazil
{rfpn,cz,anglf}@cin.ufpe.br

Abstract. This paper presents a method of combining SVMs (support vector machines) for multiclass problems that ensures a high recognition rate and a short processing time when compared to other classifiers. This hierarchical SVM combination considers the high recognition rate and short processing time as evaluation criteria. The used case study was the handwritten digit recognition problem with promising results.

Keywords: pattern recognition, handwriting digit classifier, support vector machine.

1 Introduction

Nowadays the world is digital. Technology has become ubiquitous in people´s lives, and some human tasks, such as handwriting recognition, voice recognition, face recognition and others are now machine tasks. The main recognition process [1][2] used in this kind of application requires the following steps: data acquisition; pre-process data to eliminate noise; segmentation, where the objects (text, numbers, face, etc) to be recognized are located and separated from the background; feature extraction, where the main features of each object are extracted; and finally there is the recognition, or classification, where the objects are labeled based on their features.

This paper is focused on the classification task and we used as case study the handwritten digit recognition problem because this task can represent some classification issues. For example, the patterns can be ambiguous or some features are similar in more than one group of classes. An example of this problem is presented in Fig. 1. In Fig. 1a and in Fig. 1c the correct value of the image is seven, and in Fig. 1b, four. But the Fig. 1a and b are similar and could be the same digit. Fig. 1c could be confused as a digit one. Because of this, to build a classifier that generalizes well is a hard task. In some cases the best choice is try to use the context information to differentiate one class from the other.

The Hidden Markov Models (HMM) [3] is a technique frequently used to analyze the context and improve the classifier recognition rate. But its main disadvantage is the processing time. Modeling context techniques also usually are slower. Thus, our research focuses on studying the optimization and combination of classical approaches and trying to introduce more knowledge in the classifier.

A brief overview of the handwritten digit recognition research in recent years shows that classical classifiers such as the multilayer perceptron (MLP) [5], k-nearest neighbor (kNN) [2] and support vector machine (SVM) [6] are extensively used. Some authors tried combinations of these classifiers to improve results [7][8][10][11][12]. The main problem of combining different techniques is that although we are combining the advantages, we are also joining the disadvantages of both.

The MLP [5] is a powerful classifier for multiclass problems, but there is a disadvantage when using back-propagation as the learning algorithm. The algorithm can stop training in a local minimum. It is possible to use the momentum strategy to escape from local minima, however, if we try to continue the training phase the network can overfit the weights, decreasing the generalization capability. kNN [2] classifies a sample based on the distance from the patterns in training set nearest to the sample. Thus, the more patterns there are in training set, equally distributed between the classes, higher is the recognition rate. But the time to classify a sample depends of the number of patterns in training database. Therefore, this technique is usually slow.

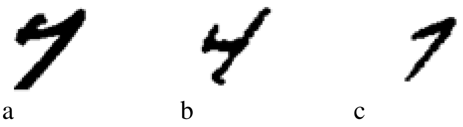

Fig. 1. Imagens from NIST database[3]. a) handwritten digit 7; b) handwritten digit 4; and c) handwritten digit 7 again.

The SVM [6] is considered the best binary classifier, because it finds the best margin of separation between two classes. The fact that SVM is a binary classifier is its greatest disadvantage, as most of the recognition tasks are multiclass problems. To solve this, some authors try to combine the SVMs [8] or use it as a decision maker classifier [9].

Based on these assumptions, this paper introduces a hierarchical SVM combination that provides a highly accurate recognition rate in a short time to answer when applied to handwritten digit recognition.

The present study is structured as follows: related works are presented in Section 2; the SVM combination architecture proposed in Section 3; the experiments and results are in Section 4; and the final conclusions of this paper are in Section 5.

2 Related Works

Support Vector Machine (SVM) [6][5] is a binary classification technique. The training phase consists of finding the support vectors for each class and creating a function that represents an optimal separation margin between the support vectors of different classes. Consequently, it is possible to obtain an optimal hyperplane for class separation.

Analyzing SVM and its previously presented characteristics, it seems similar to the perceptron [1] because it also tries to find a linear function to separate classes. But there are two main differences: the SVM discovers the optimal linear function while the perceptron seeks to discover any linear separation function; and the second difference is that the SVM can deal with non-linearly separable data. To do that, SVM uses a kernel function to increase the feature dimensionality and consequently turning the data linearly separable.

There are two classical ways to work with multiple classes using SVM: one-against-all and one-against-one [13]. In one-against-all approach, one SVM is created for each class. If we have 10 classes for example, as in digit recognition, we will have 10 SVMs, one for each digit. In this way we train the SVM(0) to differentiate the class 0 from the others classes labeling it as 1 and the others patterns as 0; the SVM(1) to differentiate the class 1 from the others classes in the same way, and so on. In the recognition phase, the pattern is submitted to the 10 SVMs, the SVM that replies label 1 indicates the class of the pattern [2]. The training set is the same database for all SVMs changing just the label of patterns. If the set is used to train the SVM(i), patterns classified as i will be replaced by 1 and the others patterns replaced by 0.

Neves et al. [8] presented the combination one-against-one applied to handwritten digit recognition. The authors used a different SVM for each possible pair of classes. In this approach, symmetric pairs, such as 0-1 and 1-0, are considered as the same pair, whereas pairs of the same class, such as 1-1 are not considered. In the case of handwriting digit recognition, there are 10 classes: 0 to 9. Pairing the classes, we have a grand total of 45 pairs. Each class will appear in 9 SVMs. For example, the class 0 will be present in the following pairs: 0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 0-7, 0-8, 0-9.

The training phase consists of two steps:

- Separate the database creating 45 data subsets for each pair of SVMs. Each subset contains only the patterns of its respective classes. For example, if the pair is responsible for differentiating 0 from 1, the subset (0,1) will contain only 0 and 1 patterns;
- Find the kernel function that better separate the classes;
- Train all 45 SVMs.

The classification phase consists in submitting the pattern to all 45 SVMs and identifying the most frequent class in the outputs of the SVMs.

The main advantage of this algorithm comes from the optimal hyperplane separation given by each SVM. It produces a highly accurate recognition rate. However, the number of SVMs combined depends of the number of classes. If the problem has a lot of classes, the system time processing will increase.

Another method of using SVM for handwriting digit recognition is as a decision classifier, as proposed by Bellili et al. [9]. In this work the authors observed that the correct recognition class is present in the highest MLP output in 97.45% of cases. However, if we consider the two highest MLP outputs, the percentage of correct class is increased to 99% of cases. Then, the authors verified which main pairs of classes were being confused. In their proposed method, when these pairs are present in the two highest MLP outputs, the authors use an SVM to decide which output corresponds to the correct class. For all other cases, they use the MLP output.

In [7] the main idea is to increase the kNN recognition rate using the SVM as a decision maker classifier, similar to Bellili et al. [9]. The adaptation in this case is to take the two most frequent classes in the k nearest neighbors and to use the SVM to decide between these two classes. It is a satisfactory technique to be used where a misclassification results in high and processing time or computational cost are not essentially.

In Ciresan et al. [17] the digit recognition task is performed using a MLP. The author's argument is that the main problem to find the correct MLP architecture is to produce a robust classifier from the training dataset. The proposal is to provide a MLP with hidden layers and neurons enough. Camastra [18] combines SVMs with neural gas. The method uses a neural gas network to verify in which case the characters are, capitalized or not, and then defines if the upper or lower case forms of a character can be included into a single class. The characters are then submitted to the SVM recognizer to obtain the final classification.

In [19] the authors create an ensemble classifier, using gating networks to assemble outputs given from three different neural networks. In [20] is proposed a method that uses a new feature extraction technique based on recursive subdivision of the character image. The combination of MLP and SVM are also used to recognize no western languages.

3 Proposed Algorithm

After analyzing the state of the art, we propose another simple SVM combination. The main idea is to create a hierarchical SVM structure. The first level is composed by a set of SVMs. There is one SVM for each class pair but if one class is in one pair it cannot be in other pairs. For example, in the case of digit recognition, we have 10 possible classes (outputs): 0 to 9. The first level will have five SVMs, one for each of these pairs: 0-1, 2-3, 4-5, 6-7 and 8-9.

The pattern will be classified by each SVM in the first level. It is expected that the SVM trained with the correct classification set correctly classifies the sample and the others can choose any class of its pair. The second level will combine the outputs obtained in the first level, using the same strategy of the previous level. The process will continue until there is only one output. An example of this hierarchical structure is shown in Fig. 2, where the letters a, b, x, y and i represents the outputs given by the SVMs and the number in parenthesis represents the pair that the SVM can differentiate.

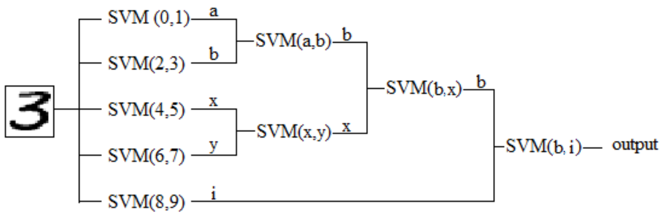

Fig. 2. Hierarchical SVM structure for digit recognition

4 Methods, Experiments and Results

The images used in the experiments were extracted from the NIST SD19 database [3], which is a numerical database available from the American National Institute of Standards and Technology. Each image in this database contains a varied number of digits, as presented in Fig. 3.

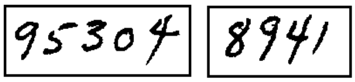

Fig. 3. Sample image of NIST SD19 database [4]

The images were separated into isolated digits using an algorithm which employed connected component labeling [10]. Each label corresponds to an isolated digit. After segmentation, vertical and horizontal projections [15] were used to centralize the digit in the image. Images larger than 20x25 were cropped, removing only the extra white borders. If the object in the image is larger than 20x25, the white border is eliminated and the digit resized. The size (20x25) was selected because the majority of digits are approximately of this size.

Each digit was manually separated and labeled into classes to be used in the supervised training of classifiers. The final database contains a total of 11,377 digits. Each class contains in average 1,150 digits. This digit database was separated into a training set and a test set. The training set contains 7,925 samples. It contains approximately 800 digits per class. The test set contains 3,452 samples, which is approximately 350 digits per class.

The feature vector is the same for all classifiers. It is the matrix image structured as a vector. However, before converting it into a vector the image was resized again to 12x15 in order to reduce the dimensionality of the feature vector, generating a vector with 180 binary features. This size was empirically defined based on previous experiments.

4.1 Algorithms: Training and Configurations

The methods used in the experiments were MLP, kNN, MLP-SVM, 45SVMs (one-against-one), kNN-SVM, SVM (one-against-all) and the proposed hierarchical SVM structure. During experiments we tried to find the best configuration of each classifier. The used configurations are described below:

a) Multilayer Perceptron

- Number of inputs: 180 (based on the feature vector);
- Number of hidden layers: one;
- Number of hidden layer nodes: 180 (during the experiments this number was reduced and increased, this is the best topology found).
- Number of outputs:10 (number of possible classes);
- Hyperbolic tangent as activation function for all nodes;
- Number of training epochs: 30,000;
- Back-propagation (gradient descent) as the training algorithm.

b) Support Vector Machine (pairs)

As previously defined, a set of training databases were created for each possible pair and forty-five pairs were used.

The polynomial and Gaussian radial basis functions (RBF) were used as the kernel function. The polynomial kernel presented the best results and was selected for SVMs pairs and SVMs one-against-all.

c) K Nearest Neighbor

The kNN main parameters are the *k* value and the distance equation used. The *k* value was varied between 3 and 11. The distances used in the experiments were Euclidian, Manhattan and Minkwoski. The best results are found with *k* equal to 3 and Euclidian distance.

All algorithms were implemented using Matlab™ [16] version R2010a. All algorithms were trained after the parameters selection. The same trained MLP and SVMs were used throughout the methods, as were the same kNN parameters.

4.2 Experiments and Results

Two criteria were used to compare the seven algorithms (MLP, kNN, MLP-SVM [9], 45 SVMs – one-against-one [8], hierarchical SVMs combination, kNN-SVM [7] and SVM – one-against-all). These criteria were processing time and recognition rate.

The Table 1 presents the recognition rates obtained by the algorithms on the test set. Table 2 presents the average and standard deviation of the processing time in seconds to classify one pattern in each algorithm.

In a statistical test considering the hypothesis that the algorithm with hierarchical SVMs is the fastest algorithm (Table 2), the results demonstrated that the hypothesis is true with a 98% of confidence.

Analyzing the SVM combinations, one-against-all and one-against-one, they presented promising results but it can be insufficient if the criteria to use the classifier are: processing time and high recognition rate. In one against-all, the main difficulty was to find a kernel function that increases the feature space turning the one class linearly separable from the others. If the number of classes increases, the complexity to find an effective kernel function increases as well. In some cases, this combination does not return a valid output. For example, after submitting one pattern to all SVMs, the outputs were labeled as 0 and the classification was not possible.

In one-against-one, the number of SVMs used is based on the number of classes. This number can be obtained by using the formula below (1). The architecture proposed in this paper also depends on the number of classes but decreases significantly because the number of SVMs is smaller, as is shown in (3).

$$\textit{Number of SVMs in one-against-one} = n * (n-1)/2 \tag{1}$$

$$\textit{Number of SVMs in one-against-all} = n \tag{2}$$

$$\textit{Number of SVMs in new approach} = n-1 \tag{3}$$

Where n is the number of classes.

The techniques based on kNN obtained a high recognition rate and a long processing time. Thus, these approaches are inadequate to use for online recognition tasks, for example. The techniques based on SVM achieve good results if we consider

Table 1. Recognition Rate in the Test Set

Algorithms	**Recognition rate**		**Algorithms**	**Recognition rate**	
	Number of samples	***(%)***		***Number of samples***	***(%)***
MLP	3,301	95.63%	MLP-SVM	3,362	97.39%
kNN	3,350	97.05%	kNN-SVM	3,382	**97.97%**
45 SVMs	3,375	**97.76%**	SVM(One-against-all)	2,818	81,63%
Hyerarchical SVMs	3,374	**97.74%**			

Table 2. Processing time results in the Test Set

Algorithms	**Processing Time (seconds)**		**Algorithms**	**Processing Time (seconds)**	
	Average	***Standard Deviation***		***Average***	***Standard Deviation***
MLP	0.0221	0.0049	MLP-SVM	0.0252	0.0102
kNN	0.0674	0.0088	kNN-SVM	**0.0684**	0.0104
45 SVMs	**0.0603**	0.0273	SVM(One-against-all)	0.0177	0.0174
Hyerarchical SVMs	**0.0115**	0.0065			

the recognition rate criterion. The one-against-all approach is the only one that does not present a high recognition rate among the SVM approaches. However, analyzing it carefully, we can see that among the 634 errors, 629 had not been labeled, when all SVMs returns the same classification and in this way it is not possible to classify the pattern (the sample was rejected by the classifier). Among the 634 errors just 15 were misclassified. This approach can be a good choice when rejection is rather than a classification error.

The proposed hierarchical SVM is the third best classifier for handwritten digits, but the difference between the first and the second is very small so in the statistical tests they are equivalent. In this case, the processing time is the main goal because the kNN-SVM and 45SVMs techniques are the slowest and the proposed method is the fastest. Thus, the hierarchical method can be highlighted by the high recognition rates in a short processing time. In Fig. 4 the processing time and error rate of the evaluated methods were normalized and plotted together in the graphic. In this case the best method is the method with results close to graphical origin. This analysis shows that the proposed method is the best evaluated method.

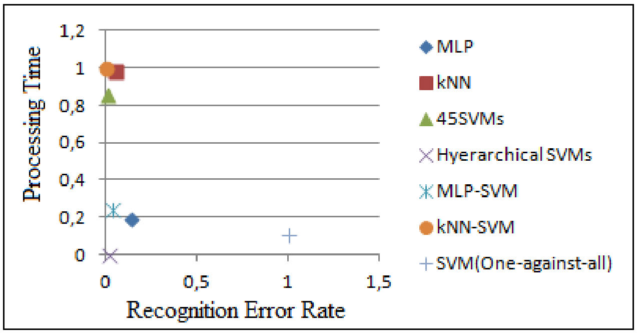

Fig. 4. Comparison graphic between the algorithms

5 Conclusion

This paper presents a method to combining SVMs applied to handwritten recognition problem considering a short processing time and a highly accurate recognition rate. After a brief study of the related works, it was found that classical classifiers are still used to recognize handwritten texts because of their low processing times and high recognition rates. New approaches, such as proposed by Neves et al. [8] and Zanchettin et al. [7] increase the recognition rates but also increase processing time and computational costs.

Based on these criteria, classical classifiers and classifiers built specifically to handwritten digits recognition were implemented and tested. The proposed new approach presented the best results considering the processing time and recognition rate. The techniques based on kNN had the longest processing time and the highest recognition rate. On the other hand, the hierarchical SVM combination obtained high recognition rates and the shortest processing time. As verified by experiments this SVM combination is the best choice when both of these criteria were important requirements.

The future works will consider handwritten characters and to attempt to combine the proposed method with word classification methods.

Acknowledgment. This work was supported by CAPES (Brazilian Research Agency) and Faculdade Boa Viagem (FBV).

References

1. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall Pearson Education Inc. (2003)
2. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley–Intersciance (2001)
3. Plötz, T., Fink, G.A.: Markov models for offline handwritting recognition: a survey. International Journal on Document Analysis and Recognition 12(4), 269–298 (2009)
4. NIST Special Database 19. Handprinted Forms and Characters Database, `http://www.nist.gov/srd/nistsd19.cfm`
5. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Prentice-Hall (1999)
6. Vapnik, V.N.: Statistical Learning Theory. John Wiley and Sons, New York (1998)
7. Zanchettin, C., Bezerra, B.L.D., Azevedo, W.W.: A KNN-SVM Hybrid Model for Cursive Handwriting Recognition. In: IEEE Int. Joint Con. on Neural Networks, Birsbane (2012)
8. Neves, R.F.P., Lopes-Filho, A.N.G., Mello, C.A.B., Zanchettin, C.: A SVM Based Off-Line Handwritten Digit Recognizer. In: IEEE International Conference on Systems, Man, and Cybernetics, vol. 1, pp. 510–515 (October 2011)
9. Bellili, A., Gilloux, M., Gallinari, P.: An MLP-SVM combination architecture for offline handwritten digit recognition. Reduction of recognition errors by Support Vector Machines rejection mechanisms. International Journal on Document Analysis and Recognition 5, 244–252 (2004)
10. Bhowmik, T.K., Ghanty, P., Roy, A., Parui, S.K.: SVM-based hierarchal architectures for handwritten Bangla character recognition. International Journal on Document Analysis and Recognition 12, 97–108 (2009)
11. Parsiavash, H., Mehran, R., Razzazi, F.: A robust free size OCR for omni-font Persian/Arabic document using combined MLP/SVM. In: Proceedings of Iberoamerican Congress on Pattern Recognition, pp. 601–610 (2005)
12. Camastra, F.: A SVM-based cursive character recognizer. Pattern Recognition 40, 3721–3727 (2007)

13. Hsu, C.W., Lin, C.J.: A Comparison of Methods for Multiclass Support Vector Machines. IEEE Transactions on Neural Networks 13(2), 415–425 (2002)
14. Parker, J.R.: Algorithms for Image Processing and Computer Vision. John Wiley and Sons (1997)
15. Gonzalez, R., Woods, C., Richard, E.: Digital Image Processing. Addison-Wesley (1992)
16. Mathworks MatlabTM – The language of technical computing, `http://www.mathworks.com/products/matlab/`
17. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. Neural Computation 22, 3207–3220 (2010)
18. Camastra, F.: A SVM-based cursive character recognizer. Pattern Recognition 40, 3721–3727 (2007)
19. Zhang, P., Bui, T.D., Suen, C.Y.: A novel cascade ensemble classifier system with a high recognition performance on handwritten digits. Pattern Recognition 40, 3415–3429 (2007)
20. Vamvakas, G., Gatos, B., Perantonis, S.J.: Handwritten character recognition through two-stage foreground sub-sampling. Pattern Recognition (43), 2807–2816 (2010)