Evaluation of a Training Method and of Various Rejection Criteria for a Neural Network Classifier Used for Off-line Signature Verification

Jean-Pierre Drouhard, Robert Sabourin and Mario Godbout

Laboratoire d'imagerie et de modélisation tridimensionnelle
École de technologie supérieure
Département de génie de la production automatisée
4750, avenue Henri-Julien, Montréal (Québec), H2T 2C8, CANADA

Abstract

This paper addresses the problems related to the design of a neural network classifier used in the first stage of an Automatic Handwritten Signature Verification System (AHSVS). We used the directional Probability Density Function (PDF) as a global shape vector, and its discriminating power was enhanced by a pretreatment. The training phase of the BackPropagation Network (BPN) was conducted by using the global classification error in memorization and in generalization. To improve the global performance of the BPN classifier, various rejection criteria were evaluated and the number of hidden neurons optimized by means of experimental protocols. The BPN classifier is better than the threshold classifier, and compares favourably with the k Nearest Neighbour classifier.

1. Introduction

The design of a complete Automatic Handwritten Signature Verification System (AHSVS) that will be able to take into account all classes of forgeries is a very difficult task [1]. Since random and simple forgeries represent almost 95 % of the cases usually encountered in practice [2], a better solution might be to subdivide the verification process in such a way as to rapidly eliminate gross forgeries. The first stage of the AHSVS would be responsible for this rapid elimination and the second stage used only in complicated cases. To eliminate only random and simple forgeries, a characteristic dealing with the overall shape of handwritten signatures seems appropriate. Accordingly, we have chosen to use the directional Probability Density Function (PDF) as a global shape vector [3]. Its discriminating power is not optimum because, even if it is invariant in translation and in scale, it is not invariant in rotation. To make a rapid decision, we have chosen to use a Neural Network (NN) as a signature classifier. Indeed, once trained, unlike conventional classifiers such as the k Nearest Neighbour (kNN) classifier, it has a very fast response time since it does not have to memorize all signature specimens. However, the training phase of these classifiers is a relatively difficult task in this application. As a matter of fact, we must know a priori all the false signatures and have many examples prior to training. In addition, due to the very high variability of handwritten signatures, the separation between true and false signatures is not a sharp one. This results in very long convergence times and the overall performance will never be perfect. Nevertheless, as shown in a feasibility study [4], these difficulties can be diminished if some precautions are taken.

2. Neural network classifier

Neural Networks (NN) present a computational paradigm for constructing classifiers that can perform as accurately as conventional techniques [5]. For the first stage of the AHSVS, we used a completely connected feed-forward neural network with the classical backpropagation learning algorithm, more simply known as the BackPropagation Network (BPN). However, with the BPN the training phase is critical, especially when the data to be classified are not clearly distinguishable and when there are not enough examples to conduct training. In this case, the training phase can be very long and it may even be impossible to obtain an acceptable performance. Since this is the case for our application (few signatures with high variability), we will define a criterion for stopping the training phase. After that, we will evaluate several rejection methods to improve the decision taken by this kind of classifier. Finally, we will adjust the number of neurons in the hidden layer of the BPN in order to increase the global performance of the first stage of the AHSVS.

2.1. Backpropagation network definition

The BPN is a multilayered neural network with an input layer that receives the data to be classified, an output layer that gives the result of the classification, and one or more hidden layers that are used to represent the domain's knowledge. Each layer contains

0-7803-1901-X/94 $4.00 ©1994 IEEE

several neurons which are fully interconnected to the neurons of the next layer via a weight. Since network weights are initially undetermined, a training process is needed to set their values. To do this, a learning law is used to modify weight values based on an output error signal propagated back through the network. From random initial values, the weights are changed according to this learning law that uses a learning rate and a smoothing rate which sometimes allows a faster convergence of the training phase. This learning law is applied for each of the training pairs (input pattern and desired output pattern) and stopped when the error (individual or global) is reduced below a preestablished limit.

To build a BPN, there are many parameters to choose from dealing with the network size or the learning law. Unfortunately, there is no way to determine them rigorously since they are strongly dependent on the application. The number of neurons on the input layer (N_i) is eighteen, which corresponds to the dimension of the PDF vector after applying a pretreatment. The goal of pretreatment on the full directional PDF is to enhance its discriminating power. A classifier will be more efficient if the dimension of the input vector is not too big (unnecessary data) and if the variations in the input vector are not too abrupt (noisy data). Unfortunately, the full directional PDF does not have these characteristics. So, pretreatment including filtering and compression, must be made. To find the pretreatment that improves the discriminating power of the directional PDF the most, we must use a classifier. We cannot use a NN classifier since its training and its performance depend strongly on the input vector (dimension and noise). So, we have to choose from among the conventional classifiers. Even though this requires a great deal of computer time when the reference set is enough large, the kNN classifier was selected because it permits the evaluation of a lower limit of the total error rate when the maximum available information is kept in memory [6]. To find the best pretreatment, an experimental protocol was developed and a rigorous analysis with statistical methods (ANOVA and LSD tests) was carried out [7]. From this analysis, we retained the integration filter with a compression step of ten. By applying this pretreatment on the full directional PDF, we increase its discriminating power roughly twofold and, overall, we reduce the input vector dimension from 180 to 18. The number of neurons on the output layer (N_o) is two since we have two classes (ω_1 and ω_2) and we want to study various rejection methods. The number of hidden layers was set to one since generally a single hidden layer is sufficient for most applications. Up to now, only rules of thumb have been proposed to determine the number of neurons on the hidden layer (N_h). We have arbitrarily set N_h = 12, a number included between the maximum and the minimum proposed by the various methods. Concerning the learning law, there are two parameters to choose: the learning rate and the smoothing rate. Again, there is no way to find a rigorous value for these parameters. Consequently, after a few preliminary trials, we arbitrarily decided to settle the learning rate = 0.6 and the smoothing rate = 0.0. In order to facilitate the start of the training phase, the weights was initialized to random values in the -0.1 and 0.1 range, and a bias term was used for the hidden and output layers to avoid a saturation of the output of the neurons [5]. We have normalized between 0 and 1 all training, test and validation sets used by the BPN classifier to improve the convergence time of the BPN during the training phase [8].

2.2. Data set definition

A standard signature database of 40 signatures written by 20 individuals (800 images) is used in this study. For the design of the NN classifier, the first 20 signatures were used to build the training set, and the last 20 signatures were divided into two groups to build the test set (first 10) and the validation set (last 10). The training set was composed of 280 examples: 140 related to the genuine signatures of an individual (class ω_1) and other 140 related to random forgeries defined as a subset of signatures from all the other individuals (class ω_2). For each writer, the 140 examples in class ω_1 were obtained by rotating all the full directional PDF issued from its 20 reference signatures in a circular fashion from -6° to +6° in 2° increments. This was done to increase the number of examples in class ω_1 that facilitate the training phase of the NN classifier, and to introduce a pseudo-invariance in rotation for the directional PDF which is, by definition, already invariant in translation and in scale. In the case of class ω_2, the 140 examples were obtained by choosing 7 or 8 reference signatures at random from 19 other individuals. The cardinality of classes ω_1 and ω_2 is equal in order not to favour one class over the other. The test and validation sets were built up in the same way. They were composed of 105 examples: 10 genuine signatures (class ω_1) and 95 signatures taken at random from all the other writers (class ω_2). The 10 examples of class ω_1 are the 10 signatures of one individual, and the 95 examples of class ω_2 are 5 random examples of 19 other individuals. The cardinality of class ω_2 were reduced to offset the diminution in the cardinality of class ω_1, but not too much in order not to affect the accuracy of the performance measure. Unlike the training and test sets, the validation set is not used to design the NN classifier, but only to compare the performances of each NN configuration.

2.3. Performance measures

The performance of each classifier is evaluated globally for the 20 writers and for 25 experiments for which the training and test sets are changed each time. Thus, in each experiment, class ω_1 will be always the same, but class ω_2 will be always different. In this way, it is possible to reduce the bias that may have been introduced by particular random forgeries. Classifier performance is measured by means of the total error rate ϵ_t expressed in terms of ϵ_1 (type I error rate, the false classification of genuine signatures) and ϵ_2 (type II error rate, the false classification of random forgeries) and $P[\omega_i]$ the a priori probability of classes ω_i, which is set at 0.5 in our case: $\epsilon_t = ((\epsilon_1 \times P[\omega_1]) + (\epsilon_2 \times P[\omega_2]))$. When appropriate, we use the total rejection rate R_t which is obtained with an equation similar to the latter one where error rates ϵ_i are substituted with rejection rates R_i (R_1 rejection of genuine signatures and R_2 rejection of random forgeries). Finally, in order to take into account in a single parameter the total error and rejection rates, a reliability factor [9] ($RF = [100 - \epsilon_t - R_t] / [100 - R_t]$) is used to find the best configuration of the NN classifier.

3. Training method

3.1. Definition

It is well known that the training phase is crucial in the design of a NN classifier, especially when we use a BPN. The major difficulty is to decide on what basis to stop training. For the BPN, training is conducted in the supervised mode, that is, during the training phase we give the input pattern and its desired output pattern, and it is the difference between this and the actual output pattern that is used to adjust the weights. When this difference is small enough, we can consider that the network has correctly learned this example. We can thus use this error value ϵ to stop training when it is lower than a preestablished limit (ideally 0) for all examples included in the training set. However, it is not always possible to reach this stopping criterion, especially when the input data are not clearly separated. A variant of this method, which takes this problem into account, is to stop training when the Root Mean Square (RMS) error on the training set is lower than a fixed threshold (ideally 0). This very popular method was not selected because it is not well suited to our application. Indeed, it is not absolutely necessary for us to have a small RMS error to obtain an acceptable performance of the classifier. In other words, we do not want the BPN to learn to give the exact output level but to make a good decision. This may be done well before the RMS error becomes weak. Consequently, we have based our stopping criterion on the performance measure ϵ_t defined in section 2.3. Thus, the network weights were adjusted for each example of the training set, and once all examples have been presented to the network (later referred to as a presentation), we freeze the weights and we evaluate ϵ_t on this training set (later referred to as ϵ_{tm}). In this way, we measure the memorization performance of the NN classifier. But the first objective of the NN classifier is to have a good generalization performance, that is, on examples not seen during the training phase. It is thus logical to stop training on the generalization performance measure of the NN classifier. To do this, we evaluate ϵ_t on the test set (later referred to as ϵ_{tg}). The drawback of the BPN which is called the "overtraining phenomenon"([10]) implies that we must stop training when ϵ_{tg} goes through a global minimum. We have not retained this criterion since preliminary results showed that for our data ϵ_{tg} was quite noisy and the curve was relatively flat after a few hundred presentations, and that the overtraining phenomenon was often imperceptible even after many thousands of presentations. Under these conditions, it would be best to stop training as soon as ϵ_{tm} and ϵ_{tg} are both almost stable. However, this method needs a great deal of computing time and is very unstable when the data are noisy. For these reasons, we decided to stop training when ϵ_{tg} is stable for all twenty writers. This consists of finding the number of presentations (NBP = T_p) after which the generalization performance for each writer is not significantly improved with longer training. Nevertheless, in some cases the training phase could be greatly reduced if we stopped it when the memorization and generalization performances are acceptable, that is, when $\epsilon_{tm} < T_m$ **and** $\epsilon_{tg} < T_g$. In summary, the stopping criterion adopted in this study is the following: "The training phase will be stopped when ($\epsilon_{tm} < T_m$ **and** $\epsilon_{tg} < T_g$) **or** when NBP $> T_p$."

3.2. Determination of T_m,T_g and T_p

Since we do not want to impose a limit to the NN classifier's performance either in memorization or in generalization, T_m and T_g must be set to 0. In this way, the NN classifier so obtained will always be the best one. Moreover, if this condition is met first, the NN classifier will be perfect both in memorization and in generalization. T_p, the maximum number of presentations used in the training phase, is determined by means of an experimental protocol [7]. Fig.1 illustrates the results found with this experimental protocol. We can see that the mean global error curves for ϵ_{tg} and ϵ_{tm} are very noisy, that the standard deviation (SD) of these curves is weak enough to justify a posteriori the 25 experiments and that ϵ_{tg} is more stable than ϵ_{tm} and, overall, sharply higher ($\epsilon_{tg} \approx 4\%$ and $\epsilon_{tm} \approx 0.3\%$), thus confirming the choice of ϵ_{tg} for finding T_p. To find T_p for which the slope of this curve is very weak, we

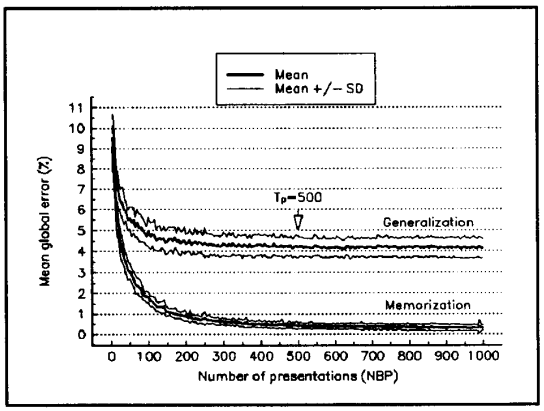

Fig. 1

have smoothed this curve by curve fitting. We have tried the two following slope values 10^{-3} and 10^{-4} and the resulting T_p values were 250 and 650, respectively. As we can see in Fig.1, the first value is quite low and the second one is a little too high. So, we have set T_p at 500 to obtain a sufficiently stable curve without excessive computation time.

4. Rejection criteria

4.1. Definition

For systems not requiring an immediate decision, the addition of a rejection criterion to the decision rule allows significant improvement in the classifier's performance by refusing to classify doubtful cases [11]-[12]. However, even if the cost of a rejection is lower than that of an error [11], the rejection rate must be as weak as possible, for two reasons. The first is that we can also reject good decisions and if the rejection rate of good decisions becomes higher than that of bad decisions, the classifier performance will be decreased instead of increased. The second concerns classifier utility. Indeed, in an extreme case, the classifier can have an error rate of 0% but a rejection rate of 100%. In this case, the classifier is useless. A way to take this phenomenon into account is to use the reliability factor (RF) as defined in section 2.3. With this measure, we consider the error rate as well as the rejection rate in evaluating the classifier's performance.

The definition of all rejection criteria used in this study is as follows: RC1 = (MAX $\{o_1, o_2\} < T_M$); RC2 = (ABS $\{o_1 - o_2\} < T_A$); RC3 = ([MAX $\{o_1, o_2\}$ - MIN $\{o_1, o_2\}$] / [MAX $\{o_1, o_2\}$] $< T_{R1}$); RC4 = ([MAX $\{o_1, o_2\}$ - MIN $\{o_1, o_2\}$] / [MIN $\{o_1, o_2\}$] $< T_{R2}$); RC5 = (RC1 **and** RC2), and RC6 = (RC1 **and** RC3) where o_1 and o_2 are the output levels for classes ω_1 and ω_2 respectively and where T_M, T_A, T_{R1} and T_{R2} are adjustable thresholds. The first two are very popular [11]-[12] and deal with the highest output level (RC1) and with the absolute difference between the output levels (RC2) respectively. The next two concern the relative difference between the output levels, either with respect to the maximum value (RC3) or with respect to the minimum value (RC4). Finally, the last two are a combination of the first one and one of the following two. For all rejection criteria, the decision rule used is of the following form: **IF** RC? = YES **THEN** the observation is not classified **ELSE** it is.

4.2. Evaluation

First, we have examined how the rejection area varies when we change the threshold of rejection criteria RC1 to RC4. Except for RC4, the rejection area is a linear function of the threshold value which varies between 0 and 1. This also shows that RC3 and RC4 are equivalent, since in each case we can find a threshold value that gives exactly the same rejection area. Given that we are more familiar with a relative variation included between 0 and 100%, we have used only RC3 in this study. Based on this examination, it is not possible to choose among the rejection criteria RC1 to RC3. Consequently, we have to evaluate them by means of an experimental protocol. We have also examined whether or not a combination of RC1 and RC2 (RC5) or RC3 (RC6) can improve classifier quality. The various rejection criteria were evaluated by using the mean value and standard deviation over the 500 files of 105 examples for ϵ_t, R_t and RF of the NN classifier obtained by varying the thresholds T_M, T_A and T_{R1} for the rejection criteria RC1, RC2, RC3, RC5 and RC6 respectively. All these results show a better performance of the NN classifier when the threshold values increase. This better performance, corresponding to a higher reliability (RF) and a lower total error rate (ϵ_t), is obtained at the expense of a total rejection rate (R_t) that is clearly higher. Unlike RC2 and RC3, RC1 becomes effective only when the threshold T_M is higher than 0.5 (Fig.2). We can also conclude that RC1, RC2

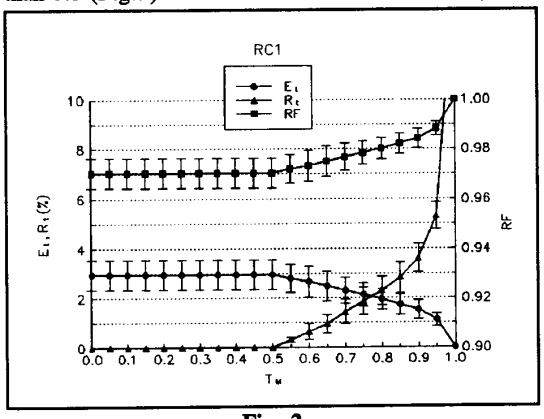

Fig. 2

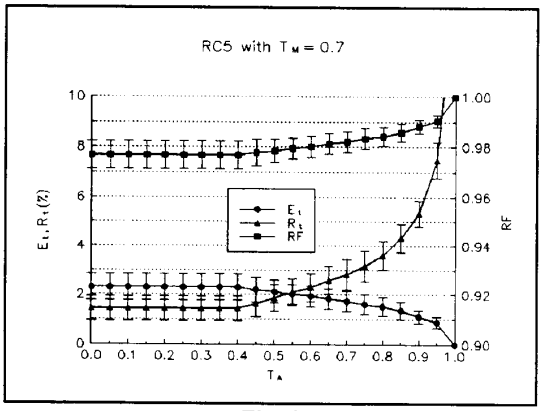

Fig. 3

and RC3 are equivalent when applied to our data. As for RC5 and RC6, they simply show that the addition of RC2 or RC3 to RC1 is only effective when the threshold T_A or T_{R1} is higher than a value for which it allows a better performance than that already obtained with the threshold T_M (Fig.3). This means that below this particular threshold T_A or T_{R1}, the rejection criteria RC1 and RC2 or RC3 reject exactly the same examples. All these results can easily be explained by the fact that our output levels are complementary. Indeed, under these conditions, the output level of one of the output neurons can never be lower than 0.5. This is why the threshold T_M must be higher than 0.5 to be effective. It is also for this reason that the rejection criteria RC1, RC2 and RC3 are equivalent. Finally, we can also explain the results of RC5 and RC6 with the complimentary phenomenon. In conclusion, we can say that, due to the output complementarity, the rejection criteria RC1, RC2 and RC3 are strictly equivalent and the rejection criteria RC5 and RC6 are only useful when we want to impose a lower limit on the NN classifier performance via RC1. Consequently, the rejection criterion RC1 has been chosen only because of its practical aspect. We now have to determine the threshold T_M, but this determination must be made with caution. Indeed, a careful examination of our results showed that if the threshold T_M was too small we could accept bad classifications. On the other hand, if we want to refuse these bad classifications with a higher threshold T_M this often leads to the elimination of good decisions. This proves that the application of a rejection criterion may degrade the NN classifier's performance, and also indicates that the threshold value must be high enough but not too high. Consequently, we chose the threshold T_M according to the following heuristic: "Obtain the best performance with an acceptable rejection rate". For our AHSVS application, a rejection rate of 5% could be considered acceptable. In this case, we find a value of 0.94 for the threshold T_M. In addition, the NN classifier's performance with the RC1 rejection criterion using a threshold T_M of 0.94 (ϵ_t = 1.22%, R_t = 4.83%, RF = 0.987) was also compared to that obtained without a rejection criterion (ϵ_t = 2.96%, RF = 0.97). The introduction of a rejection criterion improves the reliability (RF) of the AHSVS slightly but sharply decreases its total error rate (ϵ_t) via the type I error rate (ϵ_1). In other words, the rejection criterion makes it possible to reduce the false classification of genuine signatures greatly. Nevertheless, there are always genuine signatures that are more difficult to classify with our NN classifier since ϵ_1 is higher than ϵ_2 without a rejection criterion or R_1 is higher than R_2 with a rejection criterion.

5. Neural network optimization

Essentially, there are two ways to optimize a BPN: 1) we can adjust the learning algorithm [13] and 2) we can modify its structure before [14] or during [15] the training phase. In this study we only performed the BPN optimization on the structural aspect of the network. Unfortunately, there are no theoretical means for finding the optimum number of hidden neurons (N_{ho}). However, we know its upper limit, which is $N_{hm} = 2N_i+1$ [10] thus in our case N_{hm} = 37. Consequently, N_{ho} must be found by experimentation in which N_h varies gradually between 0 and N_{hm}. Moreover, to see if the rejection criterion had an effect on N_{ho}, we also conducted this experimentation without rejection. But in order to reduce the simulation time we have varied N_h between 0 and 36 with an increment of 4.

The results of this experimentation showed that the BPN performance clearly improves as soon as we have a hidden layer, and that, for our database, the effect of the number of hidden neurons is not very pronounced with or without a rejection criterion (Fig.4). In order to find the best configuration, a statistical analysis is necessary since the variations are weak. We therefore did an ANOVA test to show whether or not there was a significant difference between the configurations, and a LSD test to find the best configuration. The results of the ANOVA test emphasize two main points: first, with or without a rejection criterion, there is a significant difference between the configurations; second, the use of this rejection criterion reduced twofold the variability of the error rate and the reliability factor at the expense of a very large variability in the rejection rate. The results of the LSD test show that the regrouping obtained with the error rate (ϵ_t) and the reliability factor (RF) are identical with or without the rejection criterion. In addition, the use of a rejection criterion suppresses only the last configuration of the regrouping obtained without a rejection criterion. But, for the

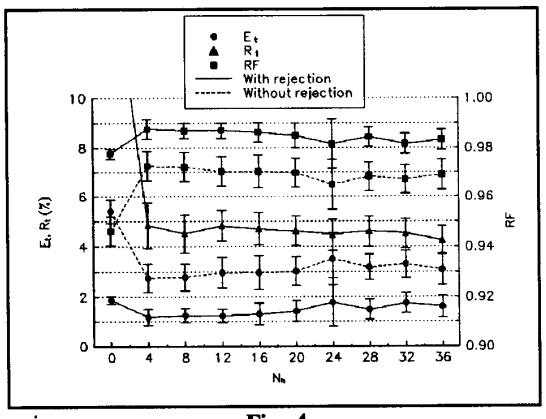

Fig. 4

rejection rate, the regrouping is very different. This result can be explained by the fact that the rejection rate has a higher variability, as indicated by the ANOVA test. Finally, the configuration with 8 hidden neurons has the best combination of the error (ϵ_t = 1.24%) and rejection (R_t = 4.5%) rates and reliability factor (RF = 0.987).

6. Conclusion

The main objective of this work was to determine whether or not a NN classifier could be used in the design of the first stage of a complete AHSVS. To do this, we have chosen the directional PDF as a global shape vector and the BackPropagation Network (BPN). In order to better assess the potential of the BPN classifier, we performed a comparative study [7] with the kNN classifier, which should give an upper limit for the performance, and the threshold classifier which is very popular in spite of its failings. Since these latter do not use a rejection criterion in making their decision, we used a BPN classifier without the rejection criterion previously defined. The conclusion of this comparative study is that the BPN classifier (ϵ_t = 3.22%) behaves better than the threshold classifier with its best reference set of 5 examples (ϵ_t = 5.61%) and worse than the kNN classifier (ϵ_t = 1.68%). In conclusion, we can say that, once trained, the BPN classifier compares favourably with the kNN classifier since it has almost the same performance but with a shorter response time at generalization, especially if we use a hardware implementation of the BPN classifier. The fact of having obtained a mean error rate of around 3% with the directional PDF, which is not the best global shape vector for a signature, clearly shows that the BPN classifier is a very good candidate for the design of the first stage of a complete AHSVS.

Acknowledgments

This work was supported in part by a PSIR grant from the École de technologie supérieure to Jean-Pierre Drouhard and Robert Sabourin, and by grant OGP010456 to Robert Sabourin from the NSERC of Canada. Master's student Mario Godbout also received a scholarship from the École de technologie supérieure.

References

[1] R. Sabourin, Une Approche de Type Compréhension de Scène Appliquée au Problème de la Vérification Automatique de l'Identité par l'Image de la Signature Manuscrite, *Ph.D. Thesis*, École Polytechnique de Montréal (1990).

[2] W.R. Harrison, *Suspect Documents, Their Scientific Examination*, Nelson-Hall Publishers, Chicago (1981).

[3] R. Sabourin and J.-P. Drouhard, Off-line Signature Verification Using Directional PDF and Neural Networks, *Proceedings of the 11th ICPR*, The Hague, 321-325 (1992).

[4] R. Sabourin, J.-P. Drouhard, L. Gagné and N. Paquet, Automatic Handwritten Signature Verification Using Directional PDF and Neural Networks: A Feasibility Study, *Proceedings of the Conference and Exhibition on Industrial Automation*, Montréal, 20.17-20.20 (1992).

[5] D.J. Burr, Experiments on Neural Net Recognition of Spoken and Written Text, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **36**, 1162-1168 (1988).

[6] K. Fukunaga, *Introduction to Statistical Pattern Recognition (Second Edition)*, Academic Press Inc., New York (1990).

[7] M. Godbout, Évaluation de l'applicabilité des réseaux de neurones à la classification des images de signatures manuscrites, *Mémoire de Maîtrise*, École de technologie supérieure (1993).

[8] J. Maren, D. Jones and F. Franklin, Configuring and Optimizing the Back-Propagation Network, in *Handbook of Neural Computing Applications*, Academic Press, Santiago (1990).

[9] L. Xu, A. Krzyzak and C.Y. Suen, Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition, *IEEE Trans. on Systems, Man, and Cybernetics* **22**, 418-435 (1992).

[10] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley, New York (1990).

[11] M. Nadler and E.P. Smith, *Pattern recognition engineering*, John Wiley & Son, New York (1993).

[12] B. Dubuisson, *Diagnostic et reconnaissance des formes*, Hermès, Paris (1990).

[13] A.V. Ooyen and B. Neinhuis, Improving the convergence of the backpropagation algorithm, *Neural Networks* **5**, 465-471 (1992).

[14] E.B. Baum and D. Haussier, What size net gives valid generalization?, *Neural Computation* **1**, 151-160 (1989).

[15] T. Ash, Dynamic node creation in backpropagation networks, *Proceedings of the International Joint Conference on Neural Networks* **2**, 623-628 (1989).