Cursive Character Detection using Incremental Learning[*]

Jean-François Hébert, Marc Parizeau
Computer Vision and Systems Laboratory
Dept. of Electrical and Computer Engineering
Laval Univ., Ste-Foy (Qc), Canada, G1K 7P4
{jfhebert,parizeau}@gel.ulaval.ca

Nadia Ghazzali
Dept. of Mathematics and Statistic
Laval Univ., Ste-Foy (Qc), Canada, G1K 7P4
ghazzali@mat.ulaval.ca

Abstract

This paper describes a new hybrid architecture for an artificial neural network classifier that enables incremental learning. The learning algorithm of the proposed architecture detects the occurrence of unknown data and automatically adapts the structure of the network to learn these new data, without degrading previous knowledge. The architecture combines an unsupervised self-organizing map with a supervised Perceptron network to form the hybrid Self-Organizing Perceptron (SOP) network. Recognition experiments conducted on isolated characters taken in the context of cursive words show the promising incremental capabilities of this SOP network.

1. Introduction

One of the great challenge in on-line handwriting recognition, as in many other pattern recognition problems, is to design systems that are not only capable of automatic learning but also capable of learning in an incremental fashion. This paper presents a new hybrid neural network architecture that is capable of tackling this problematic, and evaluates its performances in the context of learning to recognize cursive characters in cursive words.

Neural network training may be passive, active or incremental. Passive learning is a one shot process where training is conducted on all available data. In contrast, an active strategy allows some form of interaction between learner and teacher. For example, an active process can concentrate on specific regions of the input domain. Incremental learning differs from passive and active learning in that it does not assume that all input data are available a priori. The main reasons for wanting active and incremental learning in neural networks are: 1) to allow training by parts on huge data sets, 2) to enable learning of new knowledge, 3) to adapt to knowledge evolution, and 4) to reinforce current knowledge. In order to achieve incremental learning, however, it appears necessary that the number of free parameters in the network be allowed to change over time, which translates to adding both new neurons and new connections in order to build constructive neural networks [1, 2, 3].

The architecture proposed in this paper combines a self-organizing neural network linked with a multi-layered feed-forward network. The idea is to take advantage of the modeling abilities of the self-organizing network for clustering the feed-forward network into specialized neurons.

The rest of this paper is organized as follows. First, Section 2 gives an overview of the cursive handwriting system in which the proposed neural architecture simply acts as a character classifier. The object of the paper is not to describe this system in full details, but rather to present the context in which the character classifier is used, and to define the specific problem that we wish to solve using incremental learning. Then, the details of our hybrid neural network are given in Section 3. Finally, Section 4 conducts experiments that demonstrate on cursive characters segmented in words the incremental learning abilities of this network.

2. Overview of the recognition system

One of the main problem in on-line cursive script recognition is the so-called segmentation/recognition dilemma where characters need both to be segmented before they can be recognized, and recognized before they can be segmented. Many approaches were developed for working around this dilemma [4]. But most of them are based on over-segmentation of the scripts in smaller units than characters (usually graphemes) and on analysis of all possible segmentation paths, using lexical constraints in order to limit combinatorial explosion. In contrast, our system uses a somewhat different strategy where no explicit a priori segmentation is made, and no lexical constraints are a priori imposed [5].

This system is based on a segmentation process where the objective is to find all instances of character classes in a cursive word. This search, conducted on a class by class basis, stems from an iterative positioning process that controls the

[*]This research was supported in part by NSERC grants to M. Parizeau and N. Ghazzali, and in part by an FCAR grant to M. Parizeau.

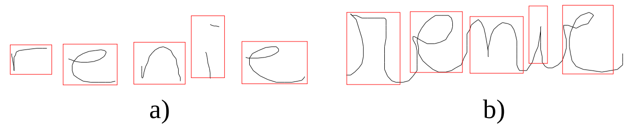

Figure 1. *Detection results on word "renie". The following enumerates the detected character in each window with its confidence value. a) 'r':1.0, 'e':1.0, 'n':1.0, 'i':1.0, 'e':1.0., b) 'n':0.4, 'e':1.0, 'n':0.3, 'i':1.0, 'e':1.0.*

panning and zooming of a spatial window of attention. The feasibility of this approach has already been demonstrated experimentally in the context of synthetic words constructed using randomly selected isolated characters, both concatenated and slightly overlapped to produce a cursive effect [5]. These experiments have shown that when a character instance is detectable by the character classifier, then our segmentation process is able to adjust a window of attention that partially encompasses the character, and make it converge to the bounding box of this character.

The output of the segmentation process is a set of hypotheses that begets a valuated segmentation graph whose best paths represent coherent interpretations of the unknown cursive word [6, 7]. Ideally, this graph should contain all true segmentation hypotheses and as few as possible false ones. Indeed, if a true hypothesis is missing in the graph, then the corresponding character cannot be part of the final word interpretation (remember that we do not wish at this point to use any linguistic information that could enable guessing the missing character). Thus our objective is to train our classifier network in order to maximize the detection of characters.

Up until now, our classifier network has been trained passively using Section 3 of data set Train-R01/V07 of the UNIPEN database [8], and tested using Section 3 of DevTest-R01/V02. These data set contain respectively $40\,092$ and $26\,560$ isolated lower case alphabetic characters ('a' through 'z') taken in the context of words or texts. Before training and testing, each isolated character of both data sets is mapped into a fuzzy vector of fixed diemsnion [9]. Recognition rates on the test data set are respectively 83.8%, 90.5%, 92.9%, 95.2% depending on whether one considers the top-1, top-2, top-3 or top-5 best classifier outputs. Even though the size of the training data set is large and the recognition results are relatively good on this testing data set, we have found that our classifier network is sometimes unable to detect some cursive letters within cursive words, especially for certain specific character classes. For example, Figure 1 shows two samples of the same word, but written by two different writers using different styles. The rectangles in this figure represent windows of attention. The first writer uses script handwriting and the classifier network is able to detect correctly, with a very high confidence, each of the letters. This probably implies that the handwriting style of this writer is well represented in the training data set (and indeed it is). For the second writer, however, the classifier did not succeed as well for letters 'r' and 'n', and the 'r' was even confused with an 'n'. There are two possible explanations for this phenomenon: 1) either the classifier has not learned these character styles correctly, or 2) these styles are not well represented in the training data set. In this paper, we assume the later alternative and wish to retrain the classifier using some new character instances[1] that are currently not well detected by the classifier network.

To do this, one could simply add the new samples to the training set and retrain the network. But this will be time consuming and will probably not work when the training data set is large. Our objective is to continue the training of the network in an incremental fashion, that is, by using only these new samples but without forgetting what was previously learned. In Section 4, experiments will be conducted to show that such incremental learning is possible.

3. The Self-Organizing Perceptron (SOP)

The Multi-Layer Perceptron (MLP) is certainly the most commonly used neural networks in the area of pattern recognition, and it has shown impressive results for many simple applications in recent years. However, it also has many defects that are well known and documented [1]. In particular, Gori has recently shown that the MLP tends to draw open separation surfaces in the input data space [10], and thus cannot reliably reject patterns. Another drawback of the MLP is the so-called moving target problem: since neurons on a layer do not communicate with one another, each neuron decides independently which part of the classification problem it will tackle [1]. To overcome this moving target problem, a conceivable solution is to specialize each hidden neuron by restricting them to act only within a localized region (a cluster) of the input space. The main idea behind the new neural network that is presented in the next sub-sections is to use an unsupervised self-organized network to clusterize a MLP.

In order to take advantage of the modeling abilities of an unsupervised self-organizing map, we design an hybrid Self-Organizing Perceptron (SOP) network in which the role of the self-organizing map is to cluster a Perceptron network into specialized groups of neurons. The idea is to link an already trained unsupervised map formed of q neurons $c_1, c_2, \ldots, c_q$ to a $n \times q \times m$ multi-layer Perceptron, where n is the dimensionality of the input data, q is the number of neurons located on a hidden layer, and m is the number of output neurons. Clusterization of the SOP is achieved through the computation for each neuron c_i, $i = 1, 2, \ldots, q$, of the map of a selection factor p_i which balances in regard

[1]These new samples could come from a human teacher that would pinpoint the correct window for the correct character.

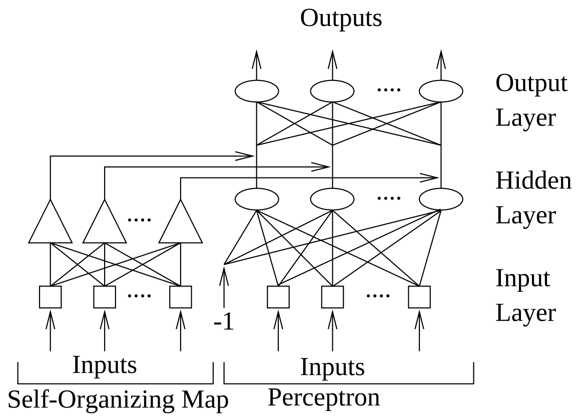

Figure 2. *SOP network architecture.*

to an input exemplar $\xi \in I\!R^n$ the output of a specific neuron located on the last hidden layer of the SOP (see Figure 2). For an input datum ξ, the selection factor p_i of a neuron c_i, $i = 1, 2, \ldots, q$, is computed as:

$$p_i = \exp\left(-\frac{||\omega_i - \xi||^2}{\sigma_i^2}\right), \qquad (1)$$

where $\omega_i \in I\!R^n$ is the position of neuron c_i in the input space, $||\cdot||$ denotes the Euclidean vectorial norm, and $\sigma_i \in I\!R^+$ is the mean length of all edges adjacent to neuron c_i. That last parameter σ_i controls the sphere of influence for neuron c_i. The resulting selection factors are used to balance the activation signal a_j, $j = 1, 2, \ldots, m$, of each output neuron in the SOP:

$$a_j = \sum_{i=1}^{q} p_i \, w_{ji} \, s_i, \qquad (2)$$

where s_i is the output signal provided by the i^{th} hidden neuron of the SOP and w_{ji} is the weight between that hidden neuron and the j^{th} output neuron.

The SOP network can be passively trained with an hybrid learning process. First, an unsupervised learning algorithm is used to construct the self-organizing map. Basically, the role of the unsupervised map for the SOP network is to generate a mapping from an original high-dimensional input space to a lower-dimensional topological structure which preserves the neighborhood relations contained in the input data. In practice, we use Fritzke’s Growing Neural Gas (GNG) [11] network because we have found that, most of the time, it achives a better mapping than other networks like the Kohonen’s Self-Organizing Map (SOM) [12] or the Fritzke’s Growing Cell Structures (GCS) [2]. Once this network is trained in an unsupervised fashion, a supervised learning algorithm such as the classic backpropagation algorithm then serves to adapt the synaptic weights of the MLP network. Note also that the backpropagation equations are slightly modified to take into account the influence of the selection factors.

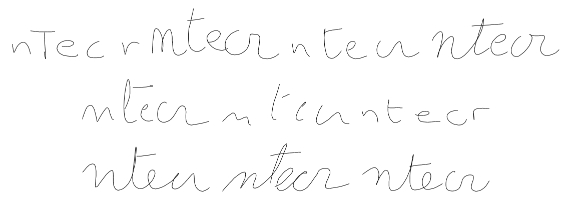

Figure 3. *Word “ntecr” written by the 10 writers.*

Both equations (1) and (2) ensure that input data located near each other in the input vector space will tend to activate the same group of neurons in the MLP during the supervised training of the SOP network. In contrast with standard backpropagation training, the use of selection factors to control network outputs allows to freeze the weight adjustments for inactive parts of the MLP. It is that feature that makes it possible to train a SOP network in an incremental fashion by adding new neurons into the structure. For full details, the reader is referred to Hébert et al. [13]

4. Experiments and results

In this section, experiments will show that a SOP network first trained passively on an isolated character data set can be retrained using only new samples of badly detected characters while preserving most of its previous knowledge. Furthermore, this retraining is conducted on only 2 out of 26 character classes in order to show that knowledge preservation is stable both within and between character classes. As stated in Section 2, passive training of a SOP network on the lower case characters (‘a’ through ‘z’) of Section 3 of data set Train-R01/V07 of the UNIPEN database [8] produces somewhat good recognition rates on the DevTest-R01/V02 test set. Indeed, these rates vary from 83.8% to 95.2% for the top-1 to top-5 hypotheses. The structure of the SOP network consists of 200 hidden neurons and 26 output neurons.

This trained SOP was thus integrated into our segmentation process [5] as the detector network, and was used to detect cursive letters in a custom cursive word data set that was manually segmented by a human operator. This data set contains 25 words written by 10 different writers (total of 250 words and 1 540 characters). Figure 3 illustrates the various styles of the ten writers. Next, two character classes were chosen, namely the ‘n’ and ‘r’ classes, because they contained many badly detected characters in this word data set. From a total of respectively 140 ‘n’ and 120 ‘r’ instances, 67 and 65 badly detected characters were extracted in order to retrain the SOP using the incremental learning mode as described in [13]. These new characters were randomly presented one by one to the SOP, and at the end, the final SOP network contained a total of 216 neurons, that is, 16 new neurons were added to the original SOP.

Now, in order to compare the performances of the detector network before and after incremental learning, a second

Table 1. *Detection results on the testing word data set, before and after incremental learning.*

Writer	Before			After		
	all[a]	‘n’	‘r’	all	‘n’	‘r’
#1	0.92	0.99	0.99	0.87	0.99	1.00
#2	0.78	0.43	0.35	0.78	0.70	0.59
#3	0.81	0.73	0.65	0.79	0.73	0.71
#4	0.72	0.74	0.52	0.72	0.85	0.77
#5	0.76	0.90	0.44	0.73	1.00	0.53
#6	0.56	0.31	0.54	0.53	0.53	0.72
#7	0.79	0.98	0.88	0.77	1.00	0.98
#8	0.73	0.89	0.17	0.73	0.99	0.64
#9	0.70	0.71	0.63	0.72	0.82	0.90
#10	0.71	0.91	0.42	0.74	0.93	0.82
Avg.	0.75	0.76	0.56	0.74	0.85	0.77

[a] “all” means all character classes except classes ‘n’ and ’r’.

word data set consisting of a different set of 25 words written by the same 10 writers was again segmented manually by a human operator. This time, the data set contains a total of $1\,520$ characters including 150 ‘n’ and 110 ‘r’ letters. Table 1 gives the average detection results on this new testing word data set on a per writer basis.

These detection results correspond to the network outputs for the window of attention selected by the human operator. An output can vary in the interval $[0, 1]$, 0 meaning no detection, and 1 meaning perfect detection. The columns labelled “all” in this table correspond to averages over all character classes, except classes ‘n’ and ‘r’. The other columns labelled ‘n’ and ‘r’ represent averages only for the corresponding character class. Relative to what could be gained to achieve perfect detection, these results correspond on average to a 45.8% increase for the ‘n’ character class and 47.2% for the ‘r’ character class. In contrast, the relative loss of detection over all other character classes is only 1.3%. Thus Table 1 shows that incremental learning is possible in order to increase detection of badly detected characters in the context of cursive words.

5. Conclusion

This paper has presented incremental learning experiments using a Self-Organizing Perceptron (SOP) network. The SOP network is an hybrid architecture combining a self-organizing map with a multi-layer Perceptron to form the Self-Organizing Perceptron (SOP) network. The unsupervised map is used to model the input data space and to partition the supervised Perceptron into clusters of neurons. This hybrid neural network was initially trained only on isolated characters taken from the UNIPEN database. Then, it was used in the context of cursive words in order to learn new instances of ‘n’ and ‘r’ characters that were not well represented in the initial training set. Results have shown that the proposed neural architecture could learn in an incremental fashion, that is by retraining the network using only the new pattern samples. This approach is currently being integrated into an interactive cursive handwriting recognition system where basic knowledge stems from large isolated character data sets that enable initial multi-writer capabilities. In this system, interaction with the user, who becomes a teacher, enables both knowledge evolution and reinforcement.

References

[1] S.E. Fahlman, C. Lebiere, “The Cascade-Correlation Learning Architecture”, in D.S. Touretzky (ed.), *Advances in neural information processing systems*, 2 (pp. 524-532). San Mateo, CA: Morgan Kaufmann Publishers, 1990.

[2] B. Fritzke, “Growing Cell Structures - A Self-Organizing Network for Unsupervised and Supervised Learning”, *Neural Networks*, 7(9):1441-1460, 1994.

[3] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, D. Rosen, “Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps”, *IEEE Trans. on Neural Networks*, 3(5):698-713, 1992.

[4] C.C. Tappert, C.Y. Suen, T. Wakahara, “The State of the Art in On-Line Handwriting Recognition”, *IEEE Trans. on Pattern and Machine Intelligence*, 12(8):787-808, 1990.

[5] J.-F. Hébert, M. Parizeau, N. Ghazzali, “Learning to Segment Cursive Words using Isolated Characters”, *Proc. of the Vision Interface Conference*, pp. 33–40, 1999.

[6] Parizeau M., Plamondon R., ”Allograph Adjacency Constraints for Cursive Script Recognition”, *Int. Workshop on Frontiers in Handwriting Recognition*, pp. 252-261, 1993.

[7] M. Parizeau, R. Plamondon, “A Fuzzy-Syntactic Approach to Allograph Modeling for Cursive Script Recognition”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 17(7):702-712, 1995.

[8] I. Guyon,L. Schomaker,R. Plamondon,M. Liberman,S. Janet, “UNIPEN Project of On-Line Data Exchange and Recognizer Benchmarks”, *Int. Conf. on Pattern Recognition*, 2:29-33, 1994.

[9] J.-F. Hébert,M. Parizeau,N. Ghazzali, “A New Fuzzy Geometric Representation for On-Line Isolated Character Recognition”, *Int. Conf. on Pattern Recognition*, 2:1121-1123, 1998.

[10] M. Gori,F. Scarselli,“Are Multilayer Perceptrons Adequate for Pattern Recognition and Verification?”, *IEEE Trans. on Pattern Analysis and Machine Intelligence*,20(11):1121-1132, 1998.

[11] B. Fritzke, “A Growing Neural Gas Network Learns Topologies”, In D.S. Touretzky, and T.K. Leen (eds), *Advances in Neural Information Processing Systems 7*, MIT Press, pp. 625–632, 1995.

[12] T. Kohonen, “The Self-Organizing Map”, *Proc. of the IEEE*, 78(9):1464-1480, 1990.

[13] J.-F. Hébert, M. Parizeau, N. Ghazzali, “A New Hybrid ANN Architecture for Active and Incremental Learning: the Self-Organizing Perceptron (SOP) Network”, accepted in *Int. Joint Conf. on Artificial Neural Networks*, 1999.