ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 561-566

SPATIAL-TEMPORAL ARTIFICIAL NEURONS APPLIED TO ONLINE CURSIVE HANDWRITTEN CHARACTER RECOGNITION

A. Rauf Baig

FAST-National Univ. of Computer & Emerging Sciences, Islamabad, Pakistan
Email: rauf.baig@nu.edu.pk

ABSTRACT

In this paper we present our latest experiments on the utilization of the recently developed Spatio-Temporal Artificial Neuron (STAN). This neuron has the capability to process asynchronous (continuous) spatio-temporal data sequences and compare them with the help of Hermitian distance. The problem addressed is that of online cursive (non-isolated) handwritten character recognition. We develop a system based on three modules: pre-processing, feature detection and character classification. The results obtained are encouraging and we also suggest further avenues of improvement in the system.
Keywords: STAN, cursive, handwritten, online, character recognition.

1. INTRODUCTION

Recently a neuron model called STAN has been developed [1,2,7], which emulates some of the aspects of the biological neuron. Its main feature is its capability to simultaneously handle the spatial as well as the temporal position of an event in a given sequence of events. This feature makes it suitable for applications which process data in which temporal positioning is also important (e.g. lip-reading [1,2]).

Online handwritten character recognition (HCR) is a spatial-temporal problem since the temporal ordering of the data generated during the writing of characters is available. Active research is going on in this field and it has been summarized in a few comprehensive survey papers, e.g. [5]. It has potential utilizations in many areas, one of which is the development of man-machine interfaces. Handwriting interface is not only more natural but also it is more feasible for many non-Latin languages (e.g. Chinese, Arabic, Persian) where the number of characters or the number of ways in which characters can be joined together is much higher than the keys of a normal key-board. However, HCR is a difficult problem because everyone writes in a slightly different way and the number of shapes obtained by combining different characters (cursive writing) is very high. Many techniques, including different types of artificial neural networks, have been applied to solve the problem with various degrees of success. In this paper, we extend the earlier work [3,4,7], on online isolated

ESANN'2004 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 28-30 April 2004, d-side publi., ISBN 2-930307-04-8, pp. 561-566

handwritten character recognition utilizing STAN and attempt cursive character recognition.

2. SPATIAL-TEMPORAL CODING & STAN

Consider a sequence of asynchronous events. An event is represented as an impulse *x* whose spatial and temporal aspects are simultaneously taken into account by coding it in the complex domain. In polar coordinates (η,φ) the magnitude η gives the amplitude and the angle φ gives the temporal position (or age) of the impulse from a reference point.

$$x = \eta e^{i\varphi} \text{ where } \tan(\phi) = \mu_T \tau \text{ and hence } x = \eta e^{i \arctan(\mu_T \tau)}$$

When a new impulse is emitted on a given component, it is accumulated with the previous impulses. The amplitude is made to decrease with time. Hence any given event is forgotten in due course of time

$$x(t) = \eta e^{-\mu_S \tau} e^{-i \arctan(\mu_T \tau)} \text{ where } \mu_S = \mu_T = 1/TW$$

The temporal window (TW) depends on the application and represents the size of the temporal window inside which one wishes to identify impulse sequences. This ST coding thus dynamically maps the incoming asynchronous events into a continuously evolving vector X. The time corresponding to the arrival of the latest impulse is taken as the reference point. All previous impulses have their temporal position updated (represented by phase angle) with reference to the current time. Each component of the vector X is thus updated as soon as a new impulse is presented to the input.

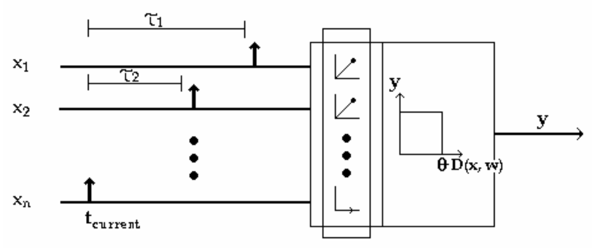

Fig. 1: *Update of input vector X at the arrival of each impulse and comparison with stored vector W. Output of an input if X and W are close enough.*

The STANs are spiking neurons, which work with the ST coding (Fig. 1). The weight W of a STAN is a complex vector and it represents the sequence to be detected by it. The comparison between X and W can be done by means of a Hermitien product V in one type of STAN and in another type it can be done by Hermitien distance D:

$$V(X,W) = \sum_{j=1}^{n} \overline{w_j} x_j \text{ and } D(X,W) = \sqrt{\sum_{j=1}^{n} (x_j - w_j)(\overline{x_j - w_j})}$$

where the bar denotes the complex conjugate.

3. CONTINOUS HCR SYSTEM

3.1. Data Acquisition & Pre-processing

3.1.1. Displacement and its quantization in 8 directions

From the digitizer tablet we acquire the basic signals of position coordinates of the contact of the pen on the surface of the tablet. A character consists of a sequence of position coordinates. As our first and basic preprocessing tool, we calculate displacements from these position coordinates. This process overcomes the spatial translation problem associated with absolute position coordinates

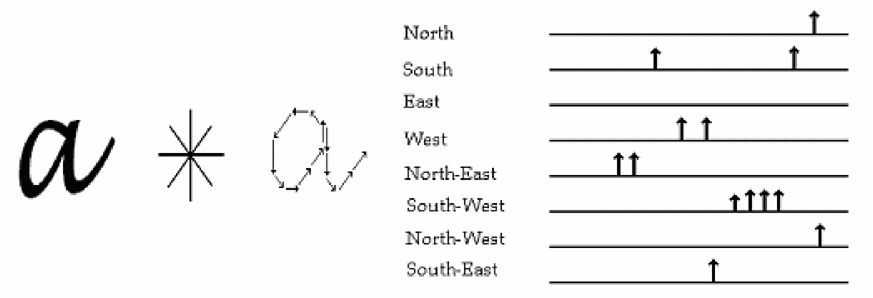

Fig. 2: *Each character is composed of displacements, which are converted into impulses. Thus a character is mapped into an 8 dimensional, time dependent, pattern of impulses.*

The calculated displacement between two points has an attached direction. The direction can be quantized, by making it equivalent to its nearest basic direction. By basic directions we mean the north, south, east, west, north-east, north-west, south-east and south-west. Previously we have used 4 basic directions, and projected (decomposed) the line to be quantized on its nearest vertical and horizontal axis [3,4], thus splitting each line into two lines. Here we increase the number of directions from 4 to 8 and avoid splitting of each line into its two basic component lines. We have experimentally observed that increasing the directions from 8 to 16 does not improve the results of the overall system.

3.1.2. Accumulation of displacement

Since our STANs are designed to detect sequences of asynchronous impulses, therefore we have to first convert quantized displacement into sequence of impulses. Since there are eight directions for quantization, therefore the sequence of impulses has eight components (Fig. 2).

We can consider each quantized displacement (between two successive position coordinates) as an impulse. However, this is not a feasible idea, as the displacement between two points of pen movement is very small, and hence there will be a lot of impulse for any given stroke. There are comparisons and decisions to be made by the next module at the arrival of each impulse, therefore the processing time of such a system will be high.

Usually the pen keeps on moving in a certain direction for several data acquisition points, therefore another method for conversion to impulses can be to add all the displacements in a certain quantized direction and emit an impulse when there is no further displacement in that direction and the direction of the pen movement changes. The amplitude of the impulse may be equal to the length of the accumulated displacement and its temporal position is the time when the pen changes direction. However, in cursive handwriting there is a risk that one character when joined with another character may produce a long line in a certain direction, and if the impulse is only at the changing of direction then there will be difficulties in identifying individual characters.

So as a compromise between two extremes, to keep the processing time short but without sacrificing efficiency, we have put a threshold on the length of the accumulated impulse. If the accumulated length of the line crosses the threshold an impulse of a magnitude equal to the size of the threshold is emitted and the accumulation starts again. If the accumulated length is below the threshold and the pen changes direction, then an impulse is emitted which has a magnitude equal to the accumulated length, and the accumulation starts again for the new direction. This threshold is writer dependent and is calculated from the training database by taking the average length of lines in all directions (there is only one threshold for all directions).

3.2. Detection of Primitives

Primitives are the basic components or parts of a character. The decomposition of a character into primitives is a vital step in our system because the boundaries of the individual characters are not known in cursive script and we cannot thus attempt to recognize the characters directly.

For detection of primitives, we first create their representative prototypes. For this purpose, we convert the continuous, but asynchronous, flow of impulses from the first module into ST vectors (formed by spatio-temporal coding of impulses). Each complex domain vector thus created contains in itself the information about the current impulse as well as a (exponentially weighted) history of previous impulses. These vectors can be considered as points in the Hermitian space. These points are then grouped into clusters with the help of Kmeans algorithm adapted for complex domain data (called ST-Kmeans) [1,2]. Each cluster center is the representative of a primitive of the characters present in the database. The optimum number of clusters for any given training database is determined experimentally by setting up the complete system and observing the final results for different number of clusters. The quality of clusters obtained by Kmeans can also be improved by the technique of hard-partitioning of data.

If there are k primitives then a layer of k STANs is used for their detection (one reserved for each primitive). Each STAN has the prototype (weight) vector of its primitive.

During the utilization phase, each STAN updates its ST vector at the arrival of each impulse from the first module. This ST vector is compared with the prototype vector. If the two are close enough, an impulse of unit length is produced at the output of the respective STAN, thus indicating the detection of a primitive.

3.3. Detection of Characters

The weights of STAN and the architecture of the classification module is according to the RCE algorithm [6] adapted to complex domain [1,2]. According to this algorithm we can have several clusters (and thus several prototypes) for a given class. For our problem, a class is a character. The procedure of classification is the same as that for the second module. The impulses (each impulse indicating the presence of a primitive) arrive from the second module. Each STAN continuously converts these incoming impulses to an ST vector and compares the ST vector with its prototype vector (weight vector). If the two vectors are sufficiently close to each other, an output impulse is produced.

The prototype vector stored in each STAN is the representative of the character to be recognized by that STAN. It is formed by ST coding of the correct sequence of impulses obtained from the second module (representing the correct succession of primitives) for a character.

3.4. Reset of System

For the time being the only higher level correctional technique utilized to improve efficiency is the detection of silence between two words. This is determined as a function of the time of pen lift and the position of next pen down. If the next pen down is a sufficient distance away from the pen lift in the horizontal direction, then it is assumed that the word has ended. This threshold is calculated on the basis of the average space found between words of the writer on whose writing the system has been trained. If the threshold is surpassed a trigger pulse is emitted, which resets all the system, thus enabling it to detect the next series of cursive characters in a more efficient way (the residual influence of the old characters due to ST coding is wiped off).

4. EXPERIMENTATION

We restrict ourselves to a single writer. The writing is also restricted to be in a natural handwriting form (no deliberate large or small characters are allowed: a conscious effort is made to keep the handwriting naturally consistent in height and width of characters). Furthermore, no delayed strokes or placing of dots are allowed. All such strokes (e.g. for t) and dots, which are normally placed after finishing a word, are to be placed immediately after writing the character. For establishing our training database, we store several paragraphs of cursive writing of a single writer and then separate (manually) ten occurrences of each character of the English alphabet (small letters only).

4.1. Determination of Weights

After the projection of each character onto the eight directions, we calculate the threshold for the emission of an impulse by the first module (Section 3.1.2).

The weights of second module (prototypes for primitives) are determined with the help of ST-Kmeans and for our training database we found the optimum number of primitives as 122 (see Section 3.2 and [1,2]).

The impulses obtained from the second module (representing the detection of primitives) have a unique pattern for each character. These sequences of impulses are grouped together by the supervised algorithm of ST-RCE (Section 3.3 and [1,2]). The cluster centers and their regions of influence obtained by the RCE algorithm are placed as the weights and thresholds of STANs of the second module.

4.2. Results

After setting up the system and its weights, we tested it with words written online by the same writer. The same constraints were imposed as for the training database (normal writing, no delayed strokes or dots). On a sample size of 1000 characters, the average correct recognition rate for individual characters is 72%. This is calculated as characters detected correctly minus inclusion errors divided by the total number of characters in the words.

5. CONCLUSION

We have attempted to tackle a difficult problem from a new point of view. Our results are encouraging when compared with other results found in the literature. In future, we intend to experiment with incorporation of a fourth module, which would have higher level information about the language. Also we would like to experiment with recognition of Urdu language, which is widely understood in South Asia.

6. REFERENCES

[1] A. R. Baig, "Une Approache Methodologique de l'utilisation des STAN Appliqué a la Reconnaissance Visuelle de la Parole", *PhD Thesis*, Univ. of Rennes-1, France, April 2000.

[2] A. R. Baig, R. Seguier, and G. Vaucher, "A Spatio-Temporal Neural Network Applied to Visual Speech Recognition", *ICANN Proceedings*, UK, pp. 797-802, Sept. 1999.

[3] N. Mozayyani, and G. Vaucher, "A Spatio-Temporal Perceptron for On-line Handwritten Character Recognition*", ICANN Proceedings*, 1997.

[4] N. Mozayyani, A. R. Baig, and G. Vaucher, "A Fully-Neural Solution for Online Handwritten Character Recognition*", IJCNN Proceedings*, USA, 1998.

[5] R. Plamondon, and S. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey", *IEEE Trans. on PAMI*, Vol. 22(1), pp. 63-84, Jan. 2000.

[6] D. Reilly, L. Cooper, and C. Elbaum. "A neural model for category learning", *Biological Cybernetics*, Vol. 45, 1982.

[7] G. Vaucher, A. R. Baig, and R. Seguier, "A Set of Neural Tools for Human-Computer Interactions", *Neural Computing & Applications*, Springer-Verlag, Vol. 9, Issue 4, Dec 2000.