A New Connectionist Model for Associative Retrieval of Harmonic Tunes

Seong-Won Yeo (1)
Min-Uk Han (2)
Chong-Ho Lee(3)
Department of Electrical Engineering, Inha University, Inchon Korea 402-751
(1)g9731492@inhavision.inha.ac.kr, http://anna.inha.ac.kr
(2)g9621018@inhavision.inha.ac.kr, http://anna.inha.ac.kr
(3)chlee@dragon.inha.ac.kr, http://anna.inha.ac.kr

Abstract

Many of the existing connectionist models based on neural associative memory have two separate stages for learning and retrieval, and are not structurally suitable for temporal pattern storage and classification in that the user must properly specify the duration of unit sector of continuously incoming patterns. In this paper, a new connectionist model of on-line associative memory for temporal pattern store and retrieval is suggested. The input signal that the suggested model deals with is a set of multi-channel discrete sequences of binary patterns which can be an interpretation of musical tune played on a keyboard. Simulation shows the suggested model can associatively retrieves a few stored patterns and adaptively tunes its stored pattern upon the change of input patterns.

1 Introduction

Most of stimuli which human receive with their sensory systems are diverse spatio-temporal patterns which continuously change along with space and time. In order to deal with such sequence of patterns, there is a request for intelligent processing method for temporal patterns.

A new connectionist model of on-line associative memory for temporal pattern processing is presented. The suggested memory model is based on on-line unsupervised learning[1]. It stores multi-channel time-varying binary signals[2] and retrieves one of the stored patterns in response to its similar pattern input.

2 Memory Structure

2.1 Neuron and Activation Function

The suggested memory model consists of series of layers. As shown in Figure 1, each layer can be visualized as a two-dimensional array of neurons. Each neuron is based on outstar[3] type neuron model. In this model, synapse connection does not take place at the back stage of neuron. Neuron owns weights in its forward direction and output of activation function is multiplied by these weights. Weight connections are stretched to neurons on the next layer.

A neuron on a layer has limited number of weights, as many as 10, towards next layer. It is much smaller than the number of all neurons on each layer. Which neurons on the next layer to be connected by the weights are not fixed and change during network processing. Initially all weights in a neuron are disconnected. As suggested memory model stores more temporal patterns, neurons gradually activate and connect weight by their necessity. If there is no more available weight for storing new patterns, the weakest weight connection is disconnected and reallocated for new pattern. This strategy is natural for human brain to forget trivial memories and pay more attentions on other important things.

Activation function is defined in Equation 1 and shown in Figure 2. Neuron fires binary output by this activation function. Net value of the activation function is accumulated by the previous layer. It has two threshold values and output is 1 if net value is near 1.

\[ f(\text{net}) = \begin{cases} 1, & 1-\delta < \text{net} < 1+\delta \\ 0, & \text{otherwise} \end{cases} \] (1)

2.2 Encoding and Decoding

The suggested memory model has encoding and decoding processes at the input and output stages.

Encoding process converts time domain temporal patterns to event domain[2] coded patterns while decoding process generates time domain output patterns from coded output patterns.
Time domain temporal pattern consists of several channels. It flows in each channel as a form of binary signals. Events are generated at every starting point of 1 signals of each channel. One event indicates one coded pattern vector is generated in the part of temporal pattern. Coded pattern vector has first element as time interval between previous and current events and its remaining elements are referred to a duration time of 1 signal of each channel.

Figure 3 shows an example of encoding process. Input signal consists of nine channels, and horizontal bars appeared in time domain indicate 1 signal being given to each of channels. Here, we define given pattern matrix, \( G \), as encoded temporal pattern that is given from input channels and to be stored by the memory model. This matrix consists of row vectors called given pattern vectors \( g_i, i=1...n \) which are in the form of coded pattern vector. The number of given pattern vectors is correspond to the number of events when the memory model encodes one temporal pattern. The layer has a two-dimensional array of neurons. Column represents channels and row represents length (duration time of 1 signal). Given pattern vector is applied to one layer as a form of activating neurons. Figure 3 also shows an example of applying given pattern vector, \( g_8 \), on a layer and activated neurons are marked

### 2.3 Overall Memory Structure

The suggested associative memory model consists of series of layers. There is an index that manifests which layer is currently receiving given pattern vector. Given pattern vectors, which are in the form of coded pattern vector and construct a part of a given pattern matrix, are sequentially applied to layers by this index.

The first layer associates first coded pattern vector with second coded pattern vector which lies on the second layer. The number of layers, \( L \), represents the number of coded pattern vectors that suggested memory model can store.

However, if the number of coded pattern vectors to be stored by the memory model is larger than the number of layers, the remaining coded pattern vectors are applied again from the first layer; that is, the last layer also associates \( L \)th coded pattern vector with \( L+1 \)th coded pattern vector by applying this vector to the first layer.

Once encoding process generates given pattern vector, it is applied to indexed layer. Then pattern process spans output pattern vector \( \{ o_i, i=1...n \} \) on this layer. With this output pattern vector, decoding process generate time domain output pattern. The overall memory structure is displayed on Figure 4.
3. Pattern Store and Retrieval

3.1 Weight Increase and Decrease

Weights are tuned by weight update algorithm which will be discussed in section 3.4. If a weight is connected to a correct neuron on the next layer, then this weight becomes increased. The weight increasing shows saturation property; that is, as learning of one pattern continues, weights are converged on some predetermined values based on equation 2.

\[ W_{i \rightarrow i} = W_{i \rightarrow i} + \frac{c}{N} (1 - W_{i \rightarrow i}) \]  

(2)

Here, \( c \) is a learning factor and \( N \) is the number of activated neurons on current layer minus one, and \( j, i \) denote activated neurons on previous and current layers respectively. By equation 2, weights are converged to the value of \( 1/N \).

If weight is connected to the close but not exactly correct neuron, then a new weight is created to connect this neuron by equation 3 and misleading weight is decreased by equation 4. Here \( d \) is a forgetting factor.

\[ W_{i \rightarrow i} = \frac{c}{N} \]  

(3)

\[ W_{i \rightarrow i} = W_{i \rightarrow i} - d \times W_{i \rightarrow i} \]  

(4)

3.2 Temporal Pattern Stored on Layers

When first given pattern vector is applied to the first layer and second given pattern vector applied to the second layer, activated neurons on both layers are connected by weight update algorithm. In this way, with successively applied temporal patterns, neurons which are activated by first given pattern vector can transfer outputs via sufficiently increased weights to the next layer. Then activated neurons on the second layer can represent second given pattern vector. This process can be identically applied to the \( i \)th and \( i+1 \)th pattern vectors. In this memory model, one layer can superpose several coded patterns vectors which are in different set of temporal patterns.

An example of pattern store is illustrated in Figure 5. Activated neurons are marked on the layers. Converged values of weights are also indicated. In this figure, three neurons on layer 1 are connected to all activated neurons on the layer 2 by the weights of 1/3. So net values of three neurons on layer 2 are accumulated to 1 and activation function fires 1.

3.3 Pattern process

Suggest associative memory model needs pattern process to store and to retrieve patterns which is summarized in Figure 6.

When zero inputs are received at the input stage in the middle of temporal pattern is being given, self-recall mode is activated. Weight update algorithm is not executed in the self-recall mode. In this mode, neurons on the current layer that are activated by the weight connections of the previous layer form a predicted pattern vector. Taking this vector as output pattern vector, associations are consecutively occurred to the next layers and remaining part of temporal patterns are retrieved without input.

Among neurons that are activated by the previous layer, which form a predicted pattern vector, output pattern vector is determined as neurons that are closely located with neurons of given pattern vector. If there is no weight that connects to the neuron of the given pattern vector from the previous layer, then new weight connection is created by weight update algorithm (see section 3.4).
activated neurons are considered during processing. The suggested Storing and retrieval can be done in real time tunes its stored pattern.

3.4 Weight Update

Weight update algorithm is summarized in Figure 7. If given pattern vector is slightly different from predicted pattern vector, this may mean given pattern vector is noisy. So generated output pattern vector is same as predicted pattern vector. However, If the different given pattern vector is continually applied, this may means a user wants to change the contents of stored pattern. By weight update algorithm, weight which is connected to predicted pattern vector becomes decreased and weight connecting new given pattern vector is created and increased. Eventually, predicted pattern vector become same as changed given pattern vector. In this way, the memory model adaptively tunes stored patterns.

4. Simulation

Figure 8 shows the computer simulation output in time domain of the memory model. It stores and retrieves two nine-channel temporal patterns. It can completely retrieve temporal patterns with only part of patterns given. It also shows how the memory model adaptively tunes its stored pattern.

5. Conclusion

In this paper a new on-line associative memory model suitable for storing and retrieval of temporal patterns is suggested. Storing and retrieval can be done in real time since neurons have limited number of weights and only activated neurons are considered during processing. The suggested memory model can adaptively tunes its stored patterns. Self-recall mode makes the memory model possible to retrieve whole temporal pattern providing the initial part of the pattern.

References


Figure 6. Pattern Process

Figure 7. Weight Update Algorithm

Figure 8. Simulation Output of the Memory Model and Progress of Tuning Stored Pattern (c=0.4, d=0.11)