SELF-ORGANIZING BOOLEAN NETWORKS FOR SPEECH RECOGNITION

Stefano Patarnello\textsuperscript{1}, Stefano Scarci\textsuperscript{2}

\textsuperscript{1}IBM ECSEC, via Giorgione 159, 00147 ROMA (Italy)
\textsuperscript{2}IBM Rome Scientific Center, via Giorgione 159, 00147 ROMA (Italy)

ABSTRACT

We show the application of a self-organizing Boolean network to speech recognition. The model consists of a set of two-input Boolean gates which has to implement a n-to-1 Boolean mapping through a learning-by-example procedure. The training scheme is based on an optimization process (Simulated Annealing). This approach is applied to a simple phoneme recognition task, achieving high accuracy.

INTRODUCTION

Speech processing seems a natural framework for experimenting the potentials of neural networks. Several models have been proposed [1] to address different classical problems in this context: grapheme-to-phoneme transcription, phoneme recognition, auditory models, etc.

In the speech recognition area the most popular approach is probably the one based on feed-forward networks trained through the back-propagation algorithm [2]. Although very promising, this technique has some drawbacks. The back-propagation algorithm is not guaranteed to converge to a proper network configuration. The number of layers of the structure and the number of units in each layer has to be fixed in advance.

One of the distinguishing features of the speech recognition problem is the dynamic nature of the input patterns, which are especially difficult to align and require complex network structures. A frame-by-frame memoryless pattern classifier would probably not achieve high recognition accuracy, being unable to exploit the temporal relationship between acoustic events. To circumvent this problem Time Delay Neural Networks have been proposed [3]. Another approach (often used also in conventional, HMM-based systems) consists in incorporating some dynamical information (e.g., spectral slope) into the feature vector.

An alternative approach to neural computing is provided by networks with looser analogy to biological systems. Many networks which use Boolean operators as "neurons" have been proposed [4]. The use of nodes consisting of general logical operators, rather than of a conventional nonlinear response, provides high computational effectiveness, when a suitable training procedure is performed. In particular, a successful approach consists of regarding the training process as an optimization search, where the function to minimize is the average error performed by the network on a set of examples. A very accurate optimization strategy is the \textit{simulated annealing} method [6], which guarantees convergence (at the expense of higher computer demands). Recently, a Boolean network has been proposed for the simulation of very simplified "conditioning" experiments, where the architecture is generalized to include memory units [7].

In order to study the possible application of Boolean networks to speech recognition, we focused on a simple problem: recognition of pre-segmented vowel utterances.

The paper is organized as follows. The next section contains an overview of the learning process of self-organizing Boolean networks. The following section describes in detail the recognition task under study and discusses issues of data representation and network structure; results of initial experiments are also given. The last section discusses plans of future work.

LEARNING IN BOOLEAN NETWORKS

Our architecture consists of a network of two-input, one-output Boolean gates, which we use to realize a binary mapping of a set of input bits into a set of output bits:

\[ o_j = F(l_i) \]

where the index \( i \) runs from 1 to \( N_i \), the total number of inputs, and the index \( j \) runs from 1 to \( N_o \), the total number of outputs. Each gate implements a Boolean operation among the 16 possible functions of two inputs into one output. This means that, besides nontrivial operators as the AND, OR, XOR, we consider also functions which simply transfer incoming information (e.g., the one which gives at its output the value of one of the two input bits) or even functions which destroy such information (e.g., the one which gives always zero, regardless of the input bits). The function that a given node performs is modified during the training of the network, as explained in the following.
There are some simple architecture constraints which tell how the nodes of the network can be connected. Each gate has a number which labels it (from 1 to \( N_0 \), where \( N_0 \) is the total number of gates, fixed at the beginning of the training) and the rule is that each of the two inputs of gate \( i \) can be connected either to the output of one of the \( i - 1 \) previous gates or to one of the \( N_i \) inputs of the network. To reduce the generality of the architecture, which would make training prohibitively complex, we fix that the \( N_0 \) output bits are taken at the output of the last \( N_0 \) gates of the network.

To define a configuration of this network we have to specify the pattern of the connections among gates and inputs, and the Boolean function that each of the \( N_0 \) gates perform. The training phase consists in selecting a configuration which implements the mapping \( F \) "best". It is often the case that the function \( F \) is given by a set of input-output instances (e.g., the training set which describes a pattern classification task). In this spirit, we propose a learning-by-example scheme. We take at random \( N_E \) examples of the mapping \( F \):

\[
\{ \bar{o}^y_j \} = F(\{ \bar{I}_i \})
\]

Here the superscript \( y \) labels the \( N_E \) examples of \( F \). The purpose of training is to select a configuration of the network which minimizes the error on the examples. For a given network configuration, the error is very well defined:

\[
E = \sum_{a=1}^{N_E} \sum_{j=1}^{N_0} | \bar{o}^y_j - \bar{o}^a_j |
\]

where \( \bar{o}^y_j \) is the result that the network produces at the output \( j \) for the example \( y \). This is obviously a function of the network configuration. So one has to set up a convenient search procedure in the space of the networks to minimize \( E \). In this respect, the choice of a proper strategy is crucial. Not surprisingly, if the mapping \( F \) is complex enough, the search for the optimal network is a hard optimization problem (it may even be NP-complete). We use an optimization technique powerful enough to formally guarantee convergence to the best solution (given the total number of gates of the network \( N_0 \)). The method is known as simulated annealing[6]: it consists in a stochastic search which allows to relax toward the absolute minimum of a "cost function" (the function \( E \) in this case). The approach is more general than a steepest descent algorithm, as the stochastic component allows to escape from solutions which are local minima for the function \( E \) (the exponentially long times related to NP-completeness mostly derive from the existence of such local minima).

In practice, a step of the procedure consists in producing a random "mutation" of the network configuration (i.e., changing one of the connections or the function performed by one of the nodes) and evaluating \( E \) for this new network. The new configuration is accepted if \( E \) decreases or stays the same (as in a greedy approach). If \( E \) increases, the new configuration is accepted with a probability which varies during training: in the first stages of the training essentially all mutations are accepted, whereas at the end only transitions which improve \( E \) are taken. Therefore the algorithm becomes gradually selective, preventing the system to get stuck in local minima.

Our approach has been fruitfully tested on different kinds of tasks, ranging from the realization of a binary adder via a learning-by-example approach [4] to applications in classical pattern recognition problems, such as printed character recognition. Some more general issues, such as the meaning of generalization in neural systems, have been successfully investigated using these Boolean networks [5].

As already mentioned, in most speech recognition problems the input-output relation strongly depends on the temporal context. To include this dynamic feature, we slightly modified the model as follows. Together with the \( N_I \) inputs that describe the new incoming data at time step \( t \), we include \( N_0 \) additional inputs, where we copy the output of the network at time \( t-1 \). In other terms, we include some kind of memory units which feed back into the network. A similar system was used to simulate a very simplified "conditioning" experiment [7], where the varying input described an external environment in which a sort of artificial organism had to survive (the neural network was optimized to plan the movements of this organism in the environment).

As it will be explained in detail in the next section, the specific speech recognition problem we study here consists essentially of a classifier which has to give a YES/NO (i.e. binary) answer. Thus the output used to evaluate the network (i.e. to compute \( E \)) consists of a single bit. The additional \( N_0-1 \) output bits have the only function of feeding back information concerning previous time steps. In the following we will refer to the \( N_0 \) fed back bits as state bits.

APPLICATION TO A PHONEME RECOGNITION PROBLEM

We tried the Boolean network approach on a very simple phoneme recognition problem. The task consists in deciding whether an utterance of a stressed vowel, known to belong to a set of vowels \( \nu_1 \nu_2 \ldots \nu_n \), is or is not produced by vowel \( \nu_i \). Competitive training, a strategy which has demonstrated very helpful in HMM recognizers (Maximum Mutual Information parameter estimation), and essential in neural systems, is employed.

The network is trained on a set of labeled vowel utterances by a single speaker. The vowels are not uttered in isolation, but as part of whole words, so they are subject to coarticulation. They are extracted from an acoustic database consisting of isolated utterances of Italian words aligned to their phonetic transcriptions. The alignments were produced by describing each phoneme with a Markov model and performing the Viterbi procedure, as reported in [8]. The acoustic representation of an utterance consists of a sequence of 20-dimensional vectors, corresponding to frames spaced by 10 msec. The vectors are computed according to an auditory model [9]; they consist, essentially, of a nonlinear transformation of the log-energy of the signal in 20 critical bands. It is the same signal processing performed by the first stage of the real-time large-vocabulary probabilistic IBM prototype speech recognizers for English [10] and Italian [11].

Representation of input data is crucial to the performance of the network. In the case of Boolean networks, it is desirable a coding scheme such that each bit of the input bears a simple relationship to the acoustic features of the signal. In our experiment the 20 bands are coded separately. The auditory model outputs a nonnegative integer \( x \) for each band \( 0 \leq x \leq M \). In the current implementation, \( x \) is a 16-bit integer, so \( M = 32767 \). We choose a set of \( x \) thresholds,
0 ≤ θi < θe < θf < M , and represent a value z by a string of z bits, where the i-th bit is 1 if x > θi, and 0 otherwise. This corresponds, in fact, to a quantization into s + 1 codewords α0 ... c_s, where c_s is represented by 1M−1 and is chosen if θ_s < x ≤ θ_e (this expression holds at the boundaries if we assume θ_0 = −1 and θ_s = M).

The thresholds θ_i can be chosen according to several criteria. In our experiments, we fixed a constant value for θ_i for all bands, and determined the thresholds so to maximize the entropy of the codewords. This simple criterion is met by examining a sample of input values and ensuring that they are distributed evenly in the s + 1 ranges [θ_i, θ_{i+1}], i.e. by an equalization procedure. A more complex but possibly better criterion, which we plan to study in the future, would be to maximize the mutual information between the codewords and the output of the network (the phoneme corresponding to the acoustic vector); this could bring to a variable number of thresholds per band, so that bands carrying more information would be quantized with higher resolution.

In the experiments described here, s = 8 for all bands, so that a vector is represented by 160 bits. The network consists of 10 gates, of which 5 carry state information; so N = 10 , N_0 = 5 , N_s = 160 . The state bits are set to 0 at the beginning of each utterance. The new value of the state bits is computed, for each speech frame, based on the input bits representing the acoustic vector for that frame and on the value of the state bits as computed at the previous frame. One of the state bits is also the output bit, which acts as the phoneme detector: the network is trained so that the output bit gives 1 when the frame is part of the utterance of a specific phoneme, and 0 when it is produced by any of the others.

In the first experiment the network was presented with 100 utterances of the Italian vowel /i/ and 100 utterances of the vowel /a/, and was trained to give 1 on the frames corresponding to /a/. The network converged to a simple configuration, consisting of an OR operation involving the output state bit and one of the input bits. The following table shows the behavior of the final network on the training sample. For the first line, the column marked θ_i reports the number of frames for which the correct output is 1 and the actual output j; ER is the error rate, i.e. the fraction of frames classified incorrectly, while CR is the fraction of frames classified correctly. The second line reports the same data for whole utterances. The output of an utterance is decided by majority filtering: if its frames produce n_i zeroes and n_j ones, the output of the utterance is zero if n_i > n_j, one if n_j > n_i, undecided if n_i = n_j. Therefore the values of ER and CR for utterances might add to less than 100%.

<table>
<thead>
<tr>
<th>Frames</th>
<th>0-0</th>
<th>1-1</th>
<th>0-1</th>
<th>1-0</th>
<th>ER</th>
<th>CR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1269</td>
<td>1212</td>
<td>5</td>
<td>132</td>
<td>5.2%</td>
<td>94.8%</td>
<td></td>
</tr>
<tr>
<td>Utt.</td>
<td>100</td>
<td>96</td>
<td>0</td>
<td>4</td>
<td>2.0%</td>
<td>98.0%</td>
</tr>
</tbody>
</table>

In the other experiments, the network was presented with utterances of five different vowels: /a/, /e/, /i/, /o/, /u/. Each vowel was represented in the training sample by 100 utterances.

The network was trained again to detect the /a/ sounds. The final configuration consisted of a single AND-NOT operation involving two input bits (i.e., no state information was used). The following table reports the performance on the training sample.

<table>
<thead>
<tr>
<th>Frames</th>
<th>Utt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5887</td>
<td>1008</td>
</tr>
<tr>
<td>334</td>
<td>336</td>
</tr>
</tbody>
</table>

Using the same input data, the network was trained to detect the /a/ sounds. The final configuration consisted of a single OR operation involving the output state bit and an input bit. The performance is reported in the following table.

<table>
<thead>
<tr>
<th>Frames</th>
<th>Utt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>5097</td>
<td>1987</td>
</tr>
<tr>
<td>397</td>
<td>84</td>
</tr>
</tbody>
</table>

Conclusions and Perspectives

We described an approach to the application of self-organizing Boolean networks to speech recognition. The approach was tested on a simple speaker-dependent vowel recognition task, achieving high accuracy on the training set. Learning is computationally expensive but the resulting networks are extremely simple and cheap.

More experiments are in our plans. We will focus on the representation of the input data and on the structure and learning strategy of the network, in order to apply the technique to more complex problems, without at the same time prohibitively increasing training requirements.

The recognition task we described was chosen essentially because of its simplicity, and is probably of little practical interest. Still, it shows some potential for future roles of Boolean networks in speech recognition systems. They could be incorporated into systems based on more traditional approaches (such as HMMs or multi-layer perceptrons), to perform tasks like phoneme labeling, vector quantization, or fast vocabulary preselection.
REFERENCES


