Neuroevolution Mechanism for Hidden Markov Model

Nabil M. Hewahi
Computer Science Department, Bahrain University, Bahrain
nhewahi@gmail.com

Abstract

Hidden Markov Model (HMM) is a statistical model based on probabilities. HMM is becoming one of the major models involved in many applications such as natural language processing, handwritten recognition, image processing, prediction systems and many more. In this research we are concerned with finding out the best HMM for a certain application domain. We propose a neuroevolution process that is based first on converting the HMM to a neural network, then generating many neural networks at random where each represents a HMM. We proceed by applying genetic operators to obtain new set of neural networks where each represents HMMs, and updating the population. Finally select the best neural network based on a fitness function.

Keywords: Neural Networks, Genetic Algorithms, Neuroevolution, Hidden Markov Model

1. Introduction

Hidden Markov Model (HMM) is a statistical model based on probabilities used in various applications such as cryptanalysis, machine translation, speech and hand recognition, natural language processing, gene prediction and bioinformatics [1][2][4][5][9][12-15]. A Markov model is a probabilistic process over a finite set, {S1, ..., Sk}, usually called its states. Each state-transition generates a character from the alphabet of the process. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by a HMM gives some information about the sequence of states. There are three problems associated with HMM, evaluation, decoding, and learning problems. The evaluation problem, given the parameters of the model, compute the probability of a particular output sequence, and the probabilities of the hidden state values given that output sequence. The evaluation problem can be solved by forward-backward algorithm. The decoding problem, given the parameters of the model, find the most likely sequence of hidden states that could have generated a given output sequence. This is solved by Viterbi algorithm. The learning problem, given an output sequence or a set of such sequences, find the most likely set of state transition and output probabilities. This problem is solved by the Baum-Welch algorithm [16].

Some of the applications that exploit the strengths of HMM is handwritten recognition, Gunter and Bunke [4] combined various HMMs at a more elementary level and shown that this sort of combination improves the system's performance. Another application is using HMM in stock market forecasting. This is performed by developing a fusion model of HMM, Artificial Neural Networks (ANN, NN) and Genetic Algorithms (GA) [5]. Hassan et.al [5], used a historical data for stock prices which become inputs for HMM using ANN, GA then used to optimize those inputs. Furui and Itoh [2] proposed a method for adapting phone HMMS to noisy speech using neural networks. They used noise HMM and signal-to-noise ratios as inputs. The trained networks were confirmed to be effective in recognizing new speakers under new noise and various signal-to-noise ratios conditions.

Hewahi [8] presented a modified version of Censored Production Rule (CPR) called Modified Censored Production Rules (MCPR). CPR is proposed by Michalski and Winston [10] to capture real time situations. MCPR can fit with hidden Markov model and present a scheme to compute the certainty values of the obtained conclusions out of the induced rules. To compute the certainty values for the rule actions (conclusions), the approach exploited only the probability values associated with the hidden Markov model without using any of the other well known certainty computation approaches. Hewahi [7] also proposed an intelligent networking management system based on the induced MCPRs extracted from a networking structure based on HMM. The advantage of using this technique is that MCPRs are very useful in real time applications and can be adapted over time based on the obtained experience of the networking working process. Again Hewahi[6] proposed a mechanism (algorithm) to evolve and select the best suitable HMM for a given problem using GA, this mechanism lacks to the training process that can be of great usefulness in finding the best HMM.

Based on the above mentioned research, the importance of using HMM is increasing rapidly.

Let us consider the HMM presented in Figure 1.

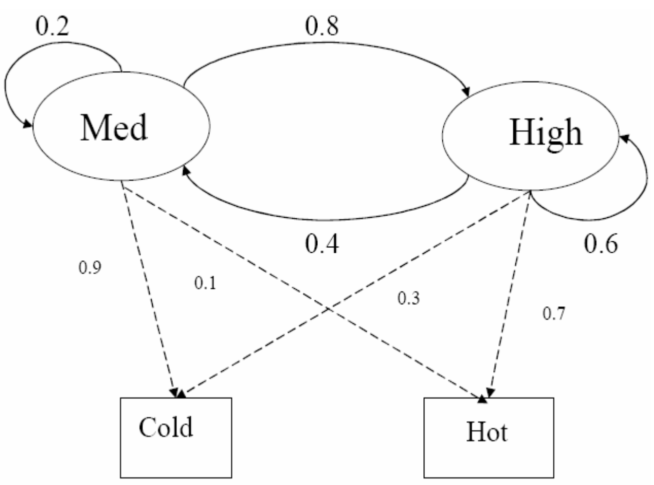

Figure 1. HMM to describe a relation between the states Med. and High with the observations (invisible states) cold and hot.

The summary of Figure 1 says:
P(cold | med) = 0.9, P(hot | med) = 0.1, P(cold | high) = 0.3, P(hot | high) = 0.7, P(med |med) = 0.2, P(high | med) = 0.8, P(high | high) = 0.6, P(med | high) = 0.4

The HMM in Figure 1can be represented in a rule structure form as shown in [8].

If Med Then Cold	0.9
If Med Then Hot	0.1
If High then cold	0.3
If High then Hot	0.7
If Med then Med	0.2
If Med then High	0.8
If High then High	0.6
If High then Med	0.4

In most of the application of HMM and neural networks, the neural networks are used to get inputs for HMM or to train historical group of HMM to predict certain result concerning with the application. Our main concern in this research paper is to be able to choose the best HMM model among various cases in terms of probability values. This will help the builders of HMM to use the most close HMM to their application. To perform this, we use neuroevolution approach by generating or evolving NNs through GAs, where each NN represents one chromosome.

2. The proposed approach

Due to the importance of HMM and its role in many AI related applications, we are concentrating on exploring how to get the best HMM for a certain application. Our proposed approach is based on Neural networks and genetic algorithms. The main problem is how to mimic the representation of HMM and apply it on a neural networks. Then how to evolve new HMM using GAs. The main steps that can be followed are:

1. Form a population of size n on HMMs for a certain problem. This population can be generated at random and based on the conditions that will be stated later.
2. Translate each HMM to a neural network.
3. Train each neural network with its domain through several examples using backpropagation algorithm.
4. Compute the fitness value for each neural network which is the number of problems solved correctly.
5. Convert the neural networks to its corresponding HMM chromosome form.
6. Apply genetic operators on the population and get new generation.
7. Compute the fitness value for the new generated chromosomes by converting them to neural networks and compute their validity with the test data. If the fitness value of the new generated chromosome is less than a threshold h, remove the corresponding chromosome. We do this to eliminate the bad chromosomes from the beginning. Actually we can do the same thing with the initial population.
8. Train the new generated neural networks and compute the fitness function for each one of them.
9. Replace the weak chromosomes with the more strong ones obtained from the generations.
10. Repeat steps from 2 to 9 until m number of generations is reached.
11. Select the best HMM chromosome (and the corresponding neural network) to make it your final HMM which you are going to deal with in your system.

To make some of the vague points more clear, we explain the following three points:

1. Representation of HMM in a neural network architecture.
2. Evolving new HMMs using GAs
3. Selecting the best HMM.

2.1 Representation of HMM in Neural Network Architecture

Under this step we need to represent the states and the observations and the connection between them. Based on Hewahi [6], the HMM model shown in Figure 2 can be represented as below:

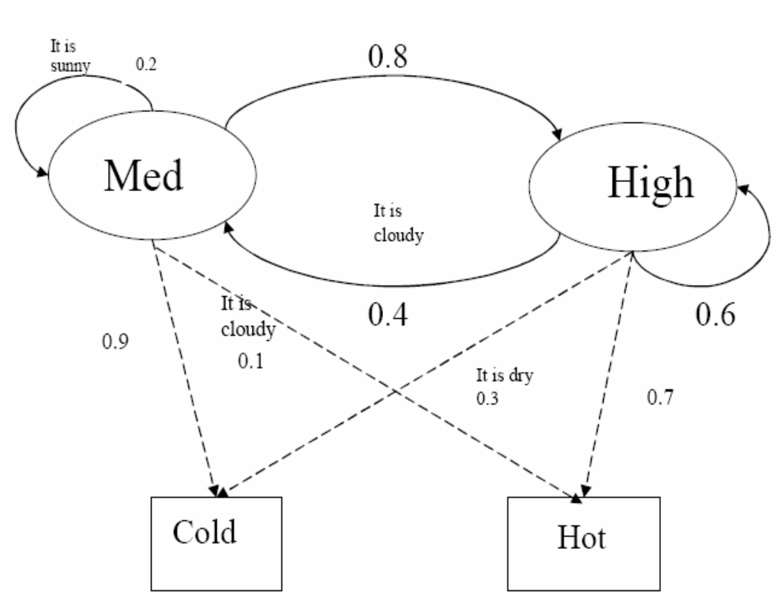

Figure 2. HMM with weights and necessary conditions on top of edges

If med then cold (0.9) unless It is cloudy then hot (0.1)
If high then hot (0.7) unless It is dry then cold (0.3)
If med then high (0.8) unless It is sunny then med (0.2)
If high then high (0.6) unless It is cloudy then med (0.4)

Based on the HMM structure in Figure 2, we can perform the following steps:

1. Make the number of nodes of inputs in the input layer of the NN as the number of states (visible states not the observations). Each input node represents one state.
2. Number of nodes in the output layer in the NN is equal to the number of states and observations (visible and invisible states), where each node corresponds to one state (visible or invisible).
3. We construct a hidden layer in NN with n number of nodes, where n is the same number of nodes in the input layer.
4. We make a connection from every input to every hidden layer node with a very negligible weight.
5. Connect every hidden node in the hidden layer to every node in the output layer.
6. Assign weights from the hidden layer to output layer in a way that as every node in the hidden layer corresponding to input state. The weight on top of the link between the hidden node to the output node is the probability value between the states in the HMM.

In our proposed structure, we injected a hidden layer to have a multilayer perceptron which is more efficient than single layer perceptron.

To make this process clear, Figure 3 shows the neural networks for the HMM presented in Figure 2.

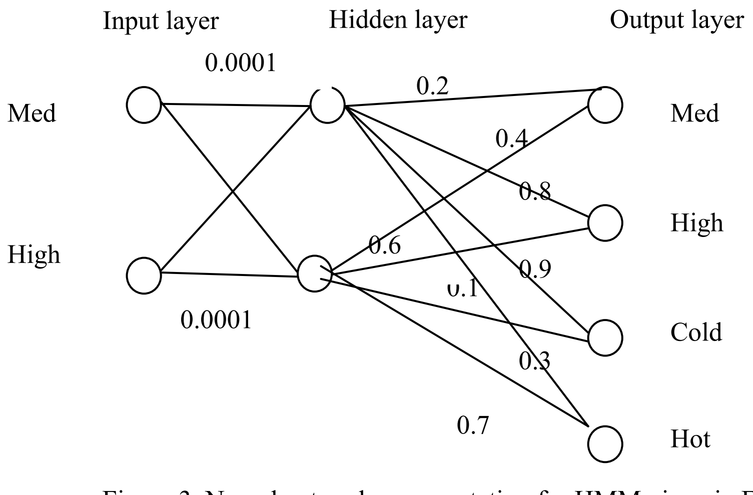

Figure 3. Neural network representation for HMM given in Figure 2.

2.2. Evolving HMM using GA

In this stage we use GA operators to generate various HMMs, and then obtain the best one among them. This process can be done as below:

a. Converting from NN to a chromosome structure

The chromosome which represents the HMM can be extracted from its corresponding neural network. The general structure of the chromosome is divided into two sections, input layer and hidden layer. Each section contains many slots, and each slot represents a weight from one node in that layer to a node in the next layer (from input to hidden and from hidden to output). The number of slots in the input layer is the same number of input nodes in the neural network. In the hidden layer, number of slots is equal to nodes in the output layer multiplied by the nodes in the hidden layer.

To make the idea clear, the chromosome representing the neural network in Figure 3 is shown in Figure 4.

Input layer		Hidden layer							
0.0001	0.0001	0.2	0.8	0.9	0.1	0.4	0.6	0.3	0.7

Figure 4. A chromosome structure for HMM shown in Figure 2.

b. Generating population

Generating a population of size *n* of HMMs at random can be performed with some restrictions:

- The weights representing the input layer in the chromosome should be always negligible as initial values.

- The weights which are involved in summation of 1.0 in the hidden layer part of the chromosome should be exactly 1.0.

Let us assume the following case

Visible states are 2 and invisible states (observations) are 3, , then we shall have a chromosome as shown in Figure 5.

Input		Hidden									
W1	W2	W11	W12	W13	W14	W15	W21	W22	W23	W24	W25

Figure 5. A chromosome structure in case we have 2 visible states and 3 invisible states

It is to be noted that the summation of W11 and W12 is equal to 1.0 (visible states) and the summation of W13,W14,W15 should be 1.0 (invisible states). Similarly the sum for W21 and W22 should be 1.0 (visible states) and the sum of W23, W24, and W25 should be 1.0 (invisible states). The values of W1 and W2 are initially assigned trivial values.
Based on the above situation, when genetic operators are applied, we must consider all the factors.

c. Genetic Operators

In this section, we discuss the potential genetic operators that can be applied on the HMM chromosome.

1. Crossover:

To apply crossover, we have two types of crossover, two point crossover and single point crossover.

Two point Crossover

For the two point crossover we get two parent HMMs and choose at random two cutting points for the weights that have a sum of 1.0 and swap the contents between the crossing points. This is illustrated in the example shown in Figure 6.

The two parents are

0.0001	0.0001	0.4	0.6	0.5	0.3	0.2	0.3	0.7	0.4	0.4	0.2

0.0001	0.0001	0.2	0.8	0.3	0.5	0.2	0.5	0.5	0.3	0.2	0.5

The offspring are

0.0001	0.0001	0.4	0.6	0.3	0.5	0.2	0.3	0.7	0.4	0.4	0.2

0.0001	0.0001	0.2	0.8	0.5	0.3	0.2	0.5	0.5	0.3	0.2	0.5

Figure 6. Two point crossover of HMM chromosomes.

One Point Crossover

This crossover is performed in the input layer part only. We choose a crossing cut point in the input layer part of the chromosome, and exchange everything before it. This is illustrated in Figure 7.

Parents

0.005	0.0001	0.4	0.6	0.5	0.3	0.2	0.3	0.7	0.4	0.4	0.2

0.0001	0.007	0.2	0.8	0.3	0.5	0.2	0.5	0.5	0.3	0.2	0.5

We get the following chromosomes

0.0001	0.007	0.4	0.6	0.5	0.3	0.2	0.3	0.7	0.4	0.4	0.2

0.005	0.0001	0.2	0.8	0.3	0.5	0.2	0.5	0.5	0.3	0.2	0.5

Figure 7. One point crossover of HMM chromosomes.

2. Mutation

For mutation, we propose three types.

a. Exchange mutation

Exchange two neighbor weights involved in a summation of 1.0. This is illustrated in Figure 8.

parent

0.005	0.0001	0.4	0.6	0.5	0.3	0.2	0.3	0.7	0.4	0.4	0.2

We get after exchange mutation

0.005	0.0001	0.4	0.6	0.3	0.5	0.2	0.3	0.7	0.4	0.4	0.2

Figure 8. Exchange mutation.

b. Adding-Reducing mutation

Adding a small value from one weight and decrement that value to another weight. The chosen weights should be involved in summation of 1.0. Figure 9 shows an example, we add 0.001 from one weight and decrement the same value from another weight.

Parent

0.005	0.0001	0.4	0.6	0.5	0.3	0.2	0.3	0.7	0.4	0.4	0.2

After adding and decrementing we get

0.005	0.0001	0.4	0.6	0.501	0.3	0.199	0.3	0.7	0.4	0.4	0.2

Figure 9. Adding-Reducing mutation.

c. Group mutation

This happens by swapping two complete groups with summation of 1.0 with the same criteria. Figure 10 shows a case of this.

Parent

0.005	0.0001	0.4	0.6	0.5	0.3	0.2	0.3	0.7	0.4	0.4	0.2

We get after group mutation

0.005	0.0001	0.4	0.6	0.4	0.4	0.2	0.3	0.7	0.5	0.3	0.2

Figure 11. Group mutation.

2.3 Selecting the best HMM

After performing n generations, we choose the best HMM chromosome (the corresponding NN) based on the fitness value. The fitness value of the neural network is related to how many problems does the HMM solve correctly.

4. Conclusion

In this paper we present a general mechanism to evolve HMMs using neuroevolution mechanism. This is done by converting HMMs to NNs then train the NNs using backpropagation algorithm and calculate the fitness value for each NN (chromosome). We then apply genetic operators , that is crossover and mutation to obtain new set of HMMs. Again the new offspring are being trained and are going to replace weak HMMs. Future directions could be applying the mechanism in various applications to scale its performance.

References

[1] L. Allison, L. Stern, T. Edgoose & T. I. Dix, ”Sequence complexity for biological sequence analysis”, Computers and Chemistry 24(1), pp.43-55, Jan. 2000.

[2] S. Furui and D. Itoh, "Neural-Network-based HMM Adaptation For Noisy Speech", Acoustics, Speech, and Signal Processing, 2001. Proceedings. (ICASSP '01,) IEEE International Conference on 02/2001; 1:365-368 vol.1., 2001.

[3] D. Goldenberg, “ Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley Longman Publishing, 1989.

[4] S. Gunter and H. Bunke, "A New Combination Scheme for HMM-based Classifiers and its Applications to HandWritten Recognition", 16th Inter. Conference on Pattern Recognition, pp. 332-337, 2002.

[5] M. Hassan, B. Nath and M. Kirley, "A Fusion Model of HMM, ANN and GA for stock market forecasting", Expert Systems with Applications, 33, pp. 171-180, 2007.

[6] N. Hewahi, "Genetic Algorithms Principles Towards Hidden Markov Model", BRAIN: Broad Research in Artificial Intelligence and Neuroscience, vol.2, issue 3, pp. 5-11, 2011.

[7] N. Hewahi, “An Intelligent Approach for Network Management Based on Hidden Markov Model”, Proceedings of the International Arab Conference for IT, ACIT’10, Libya, Dec.14-16, 2010.

[8] N. Hewahi, “Hidden Markov Model for Censored Production Rules”, Proceedings of the International Conference of Information Technology, ICIT’09, Jordan, May 3-5, 2009.

[9] J. Li, A. Najmi, R. M. Gray, Image classification by a two dimensional hidden Markov model, IEEE Transactions on Signal Processing, 48(2),pp. 517-33, February 2000.

[10] R. Michalski, P. Winston, “Variable precision logic”, Artificial Intelligence, 29, pp. 121-145, 1986.

[11] Lior Pachter and Bernd Sturmfels. "Algebraic Statistics for Computational Biology". Cambridge University Press, ISBN 0-521-85700-7, 2005.

[12] B. Pardo and W. Birmingham “Modeling form for on-line following of musical performances”, AAAI-05 Proc., July 2005.

[13] L. Rabiner, “A tutorial on hidden Markov model and selected applications on speech recognition”, Proceedings of IEEE, vol.77, no.2, 1989.

[14] K. Seymore, A. McCallum, and R. Rosenfeld, “Learning hidden Markov model structure for information extraction”, AAAI 99 Workshop on Machine Learning for Information Extraction, 1999.

[15] L. Stern, L. Allison, R. Coppel, and T. Dix, “Discovering patterns in Plasmodium falciparum genomic DNA”, Molecular and Biochemical Parasitology, 118(2) pp.175-186, 2001.

[16] http://en.wikipedia.org/wiki/Markov-model