# Hidden Markov Model Representation Using Probabilistic Neural Network

*Nabil M. Hewahi*
Department of Computer Science, University of Bahrain
Bahrain International Circuit, Zallaq, Bahrain, P.O. Box 32038
Phone: 17438888
nhewahi@gmail.com

**Abstract**

Hidden Markov Model (HMM) is a statistical network used in knowledge representation in many applications. In the same system/application, we could have several HMMs that might have similar or different visible and invisible attributes. In this paper, a theoretical framework proposal based on Probabilistic Neural Network (PNN) concept to represent all HMMs in a given system in one structure is presented. This representation framework will help in making the system provides results for cases that are not well formulated or provided in any of the given set of HMMs. The general idea of PNN has been adopted in this research to represent HMMs as patterns but the computation and representation are different. Our PNN will have at least two layers. The first layer is input layer, the second is the pattern and output layer, but we might have more than one pattern and output layer and the third layer is the sum and output layer. The proposed approach has been applied on three cases, one HMM, hierarchical HMM and independent related HMMs.

**Keywords:** Hidden Markov Model, Probabilistic Neural Networks, Prediction

## 1. Introduction

In this work, we are going to present a theoretical framework for representing Hidden Markov Model (HMM) based on Probabilistic Neural Networks (PNN). Many systems that use HMM might have several HMMs, or one or more Hierarchical Hidden Markov Models (HHMM), It would be very useful if we could represent all the used HMM in one structure that can help the user to give results for any combinations of inputs within the HMMs. Because the number of HMMs in such systems is usually limited and not big, we propose PNN to be as a tool to represent the HMM so that PNN can produce results for non-given input cases through the provided HMMs.

### 1.1. Hidden Markov Model

HMM is a well known statistical network that can be used in many applications such as natural language processing, speech and hand recognition, cryptanalysis, finance, virus detection, landmine detection, genomics, hand gesture, facial expressions bioinformatics and gene prediction (Bhusari & Pati, 2011), (Gales & Young, 2008), (Hamdi & Frigui, 2015), (Hewahi, 2010**),** (Naghizadeh, Rezaeitabar, Pezeshk, & Matthews, 2012), (Sherlock, Xifara, Telfer, & Begon, 2013). HMM has set of finite number of states, the state can be either visible or non-visible. Visible states are usually connected to each other with solid arrows whereas the visible states are connected to invisible states with dashed arrows. In top of arrows is the probability value of a certain state given the other state as evidence is presented. The summation of probabilities emerging from one visible state to all other visible states should be 1 and the summation of probabilities emerging from a visible state to all other invisible states should be 1. The invisible states do not have any connecting arrows to other invisible states. An example for HMM is given in Figure 1.

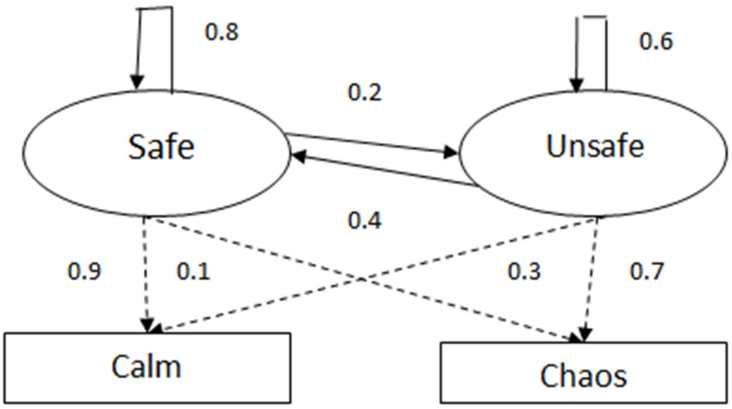



*Figure 1. HMM to describe a relation between the states "Safe" and "Unsafe" with the observations (invisible states) "Calm" and "Chaos".*

From Figure 1 we present HMM as

**a. The relation of visible states with visible states**

P(Safe|Safe)= 0.8, P(Unsafe| Safe) = 0.2, P (Unsafe| Unsafe) = 0.6, P(Safe| Unsafe) = 0.4

**b. The relation of visible states with non-visible states**

P (Calm | Safe) = 0.9, P(Choas | Safe)=0.1, P(Calm| Unsafe)=0.3, P(Chaos| Unsafe)=0.7

Some of the applications where the HMM is used are stated here. In (Bhusari & Pati, 2011) authors used HMM for detection of credit card fraud. In (Gales & Young, 2008) authors used HMM as speech recognition tool. Landmine detection using HMM was presented in (Hamdi **&** Frigui**, 2015)**. Hewahi (2010) proposed a method to use HMM in network management. In **(Naghizadeh et.al, 2012)** researchers proposed a modified HMM to be used in Protein Secondary Structure Prediction. In (Sherlock, et al., 2013) coupled HMM for disease interactions were developed. Application of HMM for classifying metamorphic virus was presented in (ShivaPrasad & RaghuKisore, 2015). A PhD thesis that is concerned with estimation of HMM and their applications in finance is prepared by Tenyakov (Tenyakov, 2014).

Further, some researchers worked to improve systems that use HMM for their representation. In (Hewahi, 2015) an approach to generate new HMMs using particle swarm optimization was presented. A neuroevolution approach was presented in (Hewahi, 2011a) to generate new HMMs where in (Hewahi, 2011b) a new approach using only genetic algorithm to generate new HMMs was proposed. In (Hewahi, 2011c) new methods to convert HMM to censored production rules which are used in real time system have been presented.

### 1.2. Hierarchical Hidden Markov Model

Fine and others (Fine, Singer & Tishby, 1998) proposed a Hierarchical Hidden Markov Model (HHMM) that combines more than HMM to represent related states in more than one HMM. Figure 2 presents HHMM.

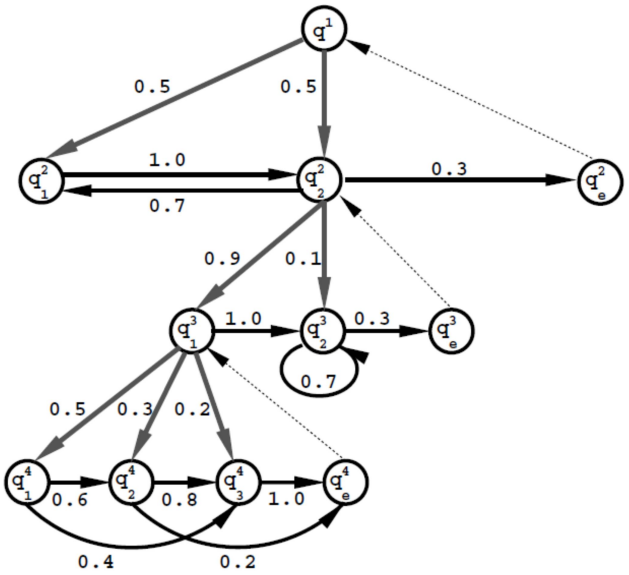



*Figure 2. HHMM with four levels. q is a state name, the upper value is the level of HMM and the lower value is the index of the state. The dashed line is the forced return to the root (Fine, et al., 1998).*

In (Fine, et al., 1998) authors show how HHMM can be used in building multilevel for English text, and unsupervised identification of repeated strokes in cursive handwriting.

## 4. Probability Neural Network

Probability Neural Network (PNN) is feed forward network based on supervised learning. The network has four layers, the first layer is the input layer, the second layer is the pattern layer, the third layer is summation layer and the fourth layer is the output layer. The first layer contains nodes for all the input attributes, the second layer contains a node for every instance in the data set, nodes produce the same output are arranged to be neighbors. The third layer contains neurons based on the number of different values for the class label. The fourth layer has one node that will produce the output of the neural networks. This method is also defined as an implementation of statistical algorithm called kernel discriminate analysis (Specht, 1992), (Specht, 1998). PNN has many advantages such that it does not need training, no local minimum issues, no further training is needed if new instances to be added to the dataset and finally the more examples are provided, the more opportunity to get the optimum solution. The main disadvantage of PNN is the need for large memory and it is specific for certain cases and domains. PNN is presented in Figure 3.

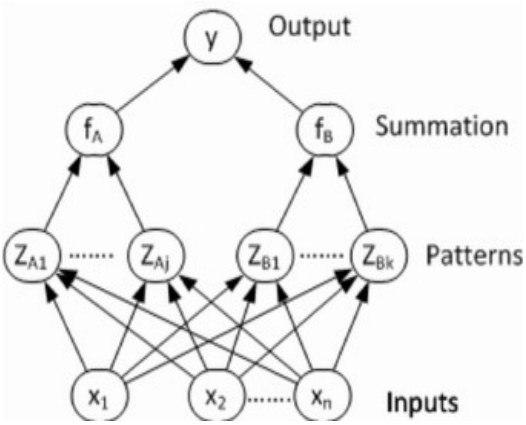



*Figure 3. PNN structure*

Figure 3 shows n X input attributes with two class labels $Z_A$ and $Z_B$, number of instances in the dataset that produces $Z_A$ has j examples where number of instances that produces $Z_B$ is k. The value for each node in the pattern layer is computed using a function called probability distribution function. The third layer has two nodes $f_A$ and $f_B$, each represents one of the possible outputs A or B. This is the obtained by using a formula called Gaussian formula which is the average of the corresponding inputs from the pattern layer. The output layer has one node which represents the winner among the two nodes in the summation layer, usually the output Y is the maximum value of $f_A$ and $f_B$. The PNN algorithm can be summarized as below (Specht, 1992) (Specht, 1998).:

a. Input neurons are represented using n attributes related to the given dataset. This represents the input layer.
b. For every instance in the dataset, create a node and arrange nodes with similar output (class label) to be neighbors. This represents the patterns layer. For every neuron in the patterns layer, probability distribution function formula is used to compute its value as shown below:

$$Y12 = e^{\frac{(X-X12)^2}{\sigma 2}} \quad (1)$$

Where $Y_{ij}$ is the probability distribution value for the $i^{th}$ output for the jth example, $\sigma$ is a smoothing parameter and X is the unknown input which we need to know its output

c. For every node in the summation layer use the Gaussian formula which is the average of all the probability distributions for the patterns having the same output as represented below:

$$g_i(X) = \frac{1}{ni} \sum_{k=1}^{ni} e^{\frac{(X-Xik)^2}{\sigma 2}} \quad (2)$$

$n_i$ is the number of instances having the same output (the $i^{th}$ output). $X_{ik}$ is the $k^{th}$ example having the $i^{th}$ output.

d. Obtain the output for the given input X by getting the maximum of all the Gaussian values obtained in the previous step. The used formula is

$$MAX(g_1,g_2,\ldots,g_i) \quad (3)$$

Several applications of PNN in many areas are existing. In (Araghi, Khaloozade, & Arvan, 2009) authors used PNN to identify ships. El Emary and Ramakrishnan (El Emary & Ramakrishnan, 2008) presented various applications of pattern classifications using PNN. In (Georgiou, Pavlidis, Parsopoulos, Alevizos, & Vrahatis, 2004) the performance of PNN in bioinformatics task has been optimized. In (Grim, Somol, & Pudil, 2005) PNN is used in Tic-Tac-Toe game.

In (Georgiou, et al., 2004) authors tried to improve the performance of PNN by incorporating a fuzzy class membership function for the weighting of its pattern layer neurons. In (Hewahi, 2011c), a proposal for representation of rule based system using PNN has been presented. The main concept is that in many systems based on rule based system, there are many rules that can be related and represented using PNN structure. This is being done to enable the system answer questions related to a set of inputs. Similarly, our proposed approach in this research is to present the existing set of HMMs in a system to be able to give answers for unknown situations.

## 2. The Proposed Approach

Our main goal in this paper is to represent HMMs in a form of PNNs so that the system will be able to give answers for unrepresented cases. PNNs is selected as a representation tool for HMM

because in most of the systems that use HMMs, they use few HMMs that can be represented easily in PNNs. As discussed before PNNs does not have weights on the links like the feed forward network based on backpropgation algorithm. In our approach, we are going to use the probability value between the states as weights for the links. This can be done among the visible states, and between the visible states and invisible states. It is to be noticed that there will be no links among invisible states. The similarity between our approach and PNN approach is only the general idea but not the computation, number of layers and the representation style. The adopted idea is to represent all used HMMs in the system in one neural network as examples. The assumption is that the given HMMs are accurate, complete and do not have inconsistency. In the proposed approach, there will be two layers or more. The used layers are one input layer, one or more pattern and output layers and one sum and output layer. We might have more than one layer related to pattern and output in case we have HHMMs. Below is the proposed approach. The points presented here are not a sequence but a general approach to be followed.

a. The input layer. This layer contains only visible states as input neurons. The intermediated or final visible states in HMMs will not be included in this layer. If we have more than one HMM, all input visible states in all HMMs will be as input nodes. In case where there are common states among the HMMs, only one state from each will be used as input.
b. The pattern and output layer. This layer contains neurons for visible and invisible states. The node might represent an output for a certain state or it could be only a representation for a pattern that needs to be connected to the sum and output layer. Only existing edges between the nodes in HMM will have links between the nodes in PNN. This means the network is not fully connected. The probability values between the states will act as weights of the links connecting the states in the PNN. In this layer similar nodes of the same state related to various HMMs are arranged as neighbors and given different index such as $Safe_1$, $Safe_2$ and $Safe_3$. Actually we might have more than one pattern and output layer specially in case of having hierarchical HMM.
c. The sum and output layer. This layer contains neurons that represent states that need to have summations of some of the neurons/states in the pattern and output layer (those nodes which are not output nodes in the pattern and output layer). For example there is a need in this layer to compute one value for Safe state obtained from $Safe_1$, $Safe_2$ and $Safe_3$ represented in the pattern and output layer.
d. In this approach, no need to have a separate output layer as PNN because in PNN we usually need one output which is the maximum of all obtained possible outputs. In our approach, all the obtained results are needed as output because each one of them represents the output of a certain state.
e. At each node in pattern and output layer, the average of the summation of multiplication of inputs with their corresponding link weighs to the node is computed.

$$A_i = Sum_i/n \qquad (4)$$

Where n is the number of links coming to the node that having input values other than 0.

$$Sum_i = \sum_{K=1}^{n} W_{ki} \cdot I_{ki} \qquad (5)$$

The idea of ignoring the input with value 0 from the computation is to maintain the original values of other states. This will be clarified in the examples explained later.

f. Sometimes a state X might have a value obtained/computed from a previous layer and this state has a link to another state Y in the next layer with a probability value as weight of the link between this node/state and other state in the next layer. The state Y might have inputs

from one or more states as X from the previous layer, to compute the value for state Y we use the following formula

$$\text{Temp-Sum}_{i} = \sum_{K=1}^{m} W_{ki} \cdot X_{ki} \qquad (6)$$

$$Y_i = \text{Temp-Sum}_i / m \qquad (7)$$

where m is the number of inputs going to node Y and $X_{ki}$ is the obtained value for the $k^{th}$ node X connected to $i^{th}$ Y node.

The assumption is that the visible state in any HMM can't be as invisible state in any other HMM and also any invisible state in any HMM can't be a visible state in any other HMM. This assumption is commonly true with most of the domains.

Figure 4 represented the proposed structure with one pattern and output layer.

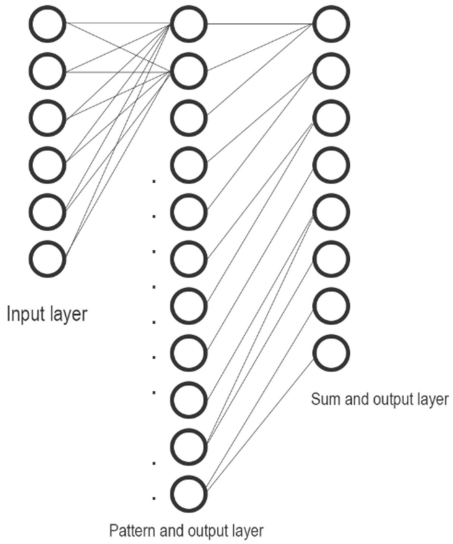



*Figure 4. PNN representation of HMM with 6 input attributes, 11 neurons in the pattern and output layer, 8 neurons in the sum and output layer*

In Figure 4 dots mean similar procedure of connection between the input layer and pattern and output layer is followed. It is to be noticed that only states having relation with other states in the pattern and output layer are having links. We have 6 input attributes; all these attributes are related to visible states in all HMMs existing in the system to be represented. In the pattern and output layer, visible and invisible states in all HMMs are represented as neurons but according to their hierarchy level. The more levels we have in HHMM, the more pattern and output layers we have. In some cases the same state say "Cloudy" can be computed through various HMMs, let us say "Cloudy-1", "Cloudy-2" and "Cloudy-3" are obtained from HMM1, HMM2 and HMM3 respectively, each one of these "Cloudy" will be represented in a different neuron and should be arranged in the layer as neighbors. In the sum and output layer, there are neurons to represent sum and outputs. This sum or output can be related to visible or non-visible state in all the existing HMMs in the system. The difference between the sum and output layer and pattern layer is that for example for "Cloudy" state we have one neuron to represent all "Cloudy" in all HMMs. As example the "Cloudy" neuron will have "Cloudy-1", "Cloudy-2" and "Cloudy-3" as its inputs to get one value for "Cloudy".

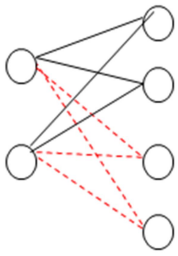
To make our approach clear, we apply it on three cases, the first case is on one HMM, the second case is using two HMMs and the third example is on a hierarchical HMMs.

### 2.1 Case of One HMM

In this section, we present two examples for representing one HMM as PNN, the first example is simple where the second is more complex. In the first example only two visible states and two invisible states are considered whereas in the second example there are three visible states and three invisible states.

#### *2.1.1 Example 1*

Let us consider HMM presented in Figure 1, as can be seen we have two visible states Safe and Unsafe and two invisible states Claim and Chaos. Figure 5 shows the representation of this HMM.



Input Layer: Safe, Unsafe

Pattern and output Layer: Safe, Unsafe, Calm, Chaos

Weights between the input layer and pattern layer are: 0.8, 0.2, 0.9, 0.1, 0.4,0.6,0.3,0.7 for the links Safe Safe, Safe Unsafe, Safe Calm, Safe Chaos, UnSafe Safe, Unsafe Safe, Unsafe Calm, Unsafe Chaos respectively.

*Figure 5. PNN representation of HMM presented in Figure 1*

In this example we have only two layers, input layer and the pattern and out layer. In this case the pattern and output layer is going to work as pattern layer and output layer because the values computed do not need any further computation and will be the as outputs for the states.

Let us assume that the values for Safe and Unsafe as 1 and 0 respectively. It is to be noted that in this case of input the relation is exclusive or relation which means one of inputs should be 1 and the other is 0.

According to formula (4), the outputs will be as below:

Safe = 0.8/1 =0.8, Unsafe = 0.2/1 = 0.2, Calm = 0.9/1=0.9, Chaos = 0.1/1 = 0.1

As noticed from the obtained results if 0 input is considered in the computation, the obtained values will not really represent the states. For example, if 0 input is considered, the results will be

Safe=(0.8+0)/2=0.4, Unsafe=(0.2+0)/2=0.1, Calm=(0.9+0)/2=0.45, Chaos=(0.1+0)/2= 0.05

As noticed, all the above obtained values are not correct concerning with the original HMM given in Figure 1, and that is why we ignore any input with value 0 and exclude it from the computation.

*2.1.2 Example 2*

Let us consider Figure 6, the figure has three visible states Humidity, Cloudy and Windy; and three invisible states Cold, Hot and Nice.

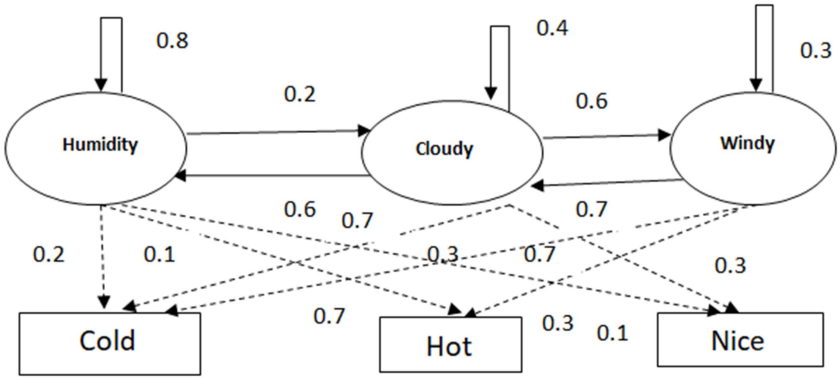
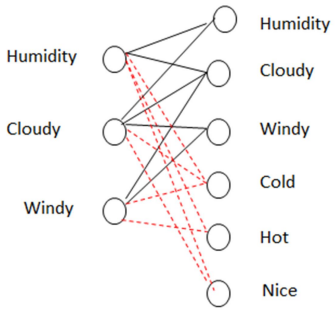



*Figure 6. HMM with three visible states and three invisible states*

The probability values given in Figure 6 are as below:
P(Humidity | Humidity)=0.8, P(Humidity | Cloudy)= 0.6, P(Cloudy | Cloudy)= 0.4, P(Cloudy | Humidity)= 0.2, P(Cloudy | Windy)= 0.7, P(Windy | Windy) = 0.3, P(Windy | Cloudy)= 0.6, P(Cold | Humidity)=0.2, P(Cold | Cloudy) = 0.7, P(Cold | Windy)= 0.7, P(Hot | Humidity)= 0.7, P(Hot | Windy)= 0.3, P(Nice | Humidity)= 0.1, P(Nice | Cloudy)=0.3




Input Layer: Humidity, Cloudy, Windy

Pattern and output Layer: Humidity, Cloudy, Windy, Cold, Hot, Nice

Weights between the input layer and pattern and output layer are: 0.8, 0.2, 0.2, 0.7, 0.1, 0.6,0.4,0.7,0.3,0.7,0.3,0.7,0.3 for the links Humidity Humidity, Humidity Cloudy, Humidity Cold, Humidity Hot, Humidity Nice, Cloudy Humidity, Cloudy Cloudy, Cloudy Cold, Cloudy Nice, Windy Cloudy, Windy Windy, Windy Cold, Windy Nice respectively.

*Figure 7. PNN representation of HMM presented in Figure 5*

Figure 7 shows the representation of HMM given in Figure 6. Assuming that the input values for the states are Humidity: 1, Cloudy: 1 and Windy: 0, the following would be the output:

Humidity = (1 x 0.8 + 1x0.6)/2 = 0.7 using formula (4)

Cloudy = (1 x 0.2 + 1 x0.4)/2 = 0.3 using formula (4)

Windy = 0

Cold = (1x0.2 + 1x0.7) /2 = 0.45 using formula (4)

Hot = (0.7 x 1)/1 = 0.7 using formula (4)

Nice = (1 x 0.1 + 1 x 0.3)/2 = 0.2 using formula (4)

### 2.2. Case of HHMM

In this example we are going to have a HHMM with three levels, level 1 is having two visible states, Safe and Unsafe, level 2 is having four Sates, two visible states and two invisible states, the two visible states are Crowded and Uncrowded and the two invisible states are Calm and Chaos. Level 3 is also having four states, two visible and two invisible. The two visible states are Frustration and Optimistic and the two invisible state are Late and On time. Figure 8 depicts the HHMM that shows the relation between the states.

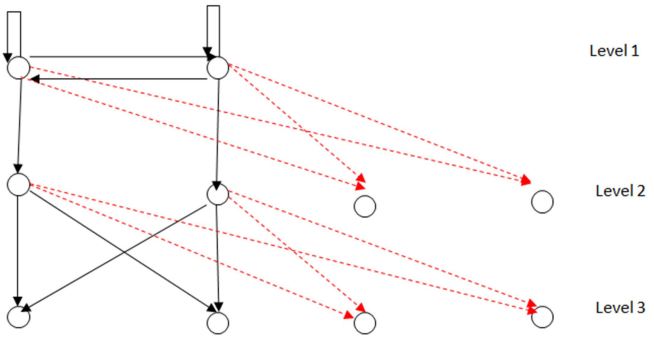



*Figure 8. Level 1: Safe and Unsafe, level 2: Crowded and Uncrowded, level 3 Frustration, Optimistic, Late and On time. The dashed lines are connecting the visible states with invisible states. The probability values between states are Safe Safe: 0.8, Safe Unsafe: 0.2, Safe Crowded: 0.8, Safe Uncrowded: 0.2, Safe Calm: 0.9, Safe Chaos: 0.1, Unsafe Safe 0.3, Unsafe Unsafe: 0.7, Unsafe Crowded: 0.2, Unsafe Uncrowded: 0.8, Unsafe Calm: 0.2, Unsafe Chaos: 0.8, Crowded Frustration:0.8, Crowded Optimistic: 0.2, Crowded Late: 0.7, Crowded On time:0.3, Uncrowded Frustration:0.6, Uncrowded Optimistic 0.4, Uncrowded Late: 0.3, Uncrowded On time: 0.7*

Figure 8 can be represented in the form of PNN as shown in Figure 9.

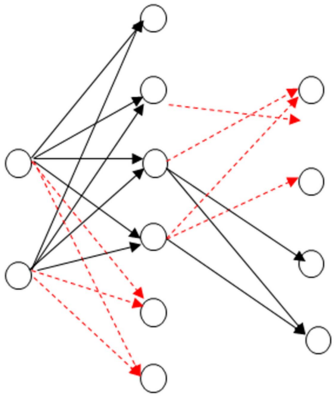


*Figure 9. PNN representation for the Hierarchal HMM shown in Figure 7. The first layer is the input layer, the second layer is the pattern and output layer and the third layer is the sum and output layer. Based on the top down nodes, the input layer is Safe and Unsafe, the pattern and output layer is Safe, Unsafe, Crowded, uncrowded, Calm and Chaos, and the sum and output layer is Late, On time, Frustration and Optimistic. The dashed links are used to connect visible states with invisible states*

In this example we have three layers, the first layer is the input layer, the second layer is the pattern and output layer and the third layer is the sum and output layer. In the pattern and output

layer there are patterns which represent outputs such as Safe, Unsafe, Calm and Chaos. The other two states in this layer Crowded and Uncrowded are nodes used as a bridge to get the rest of the outputs in the sum and output layer, these outputs are Late, On time, Frustration and Optimistic.

The values for the states will be as follows if the input values for the given attributes are Safe: 1 and UnSafe: 0

Safe= 0.8/1 = 0.8 using formula (4)

Unsafe=0.2/1=0.2 using formula (4)

Crowded = 0.8/1 = 0.8 using formula (4)

Uncrowded = 0.2/1 = 0.2 using formula (4)

Calm = 0.9/1=0.9 using formula (4)

Chaos= 0.1/1 =0.1 using formula (4)

Frustration= (0.8 x 0.8 + 0.2*0.6)/2= 0.38 using formula (7)

Optimistic = (0.8 x 0.2 + 0.2 x 0.4)/2 = 0.12 using formula (7)

Late = (0.8 x 0.7 + 0.2 x 0.3)/2 = 0.31 using formula (7)

On time= (0.8 x 0.3 + 0.2 x 0.7)/2= 0.19 using formula (7)

It is to be noted that the Crowded and Uncrowded might not be produced as output but used as intermediate states.

### 2.3. Case of Separate Related HMMs

In many systems that use HMMs, we might have independent related HMMs that. In this example, a representation of two separate and related HMMs is given. In this example we shall have two HMMs one of them is the one shown in Figure 1 and the second one is presented in Figure 10.

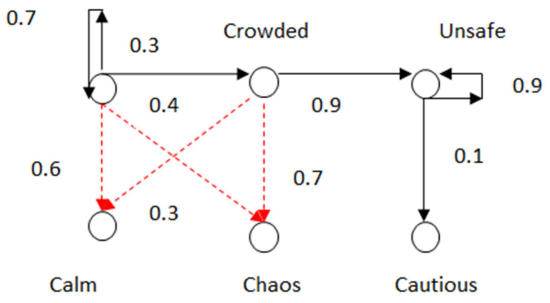



*Figure 10. The second HMM to be used in multiple separated related HMM example*

To represent HMM presented in Figure 1 and HMM presented in Figure 10, input layer contains all input states in the two HMMs, these states are Safe, Unsafe and Crowded. States such as Safe, Unsafe, Calm and Chaos need to have two nodes each in the pattern and output layer because each one of these states can be achieved through a different evidence of other states, for example Calm in HMM in Figure 1 can be reached through Safe and Unsafe states whereas Calm in Figure 10 can be reached through Safe and Crowded. This makes the pattern and output layer contains Calm 1 and Calm2. Similarly for other states. It is also to be noted that we have in the two HMMs Safe states but their values are different. This means we should have also two nodes to

represent Safe (call them Safe1 and Safe2) in the pattern and output layer. The representation of these two HMMs in one PNN would be as the one presented in Figure 11.

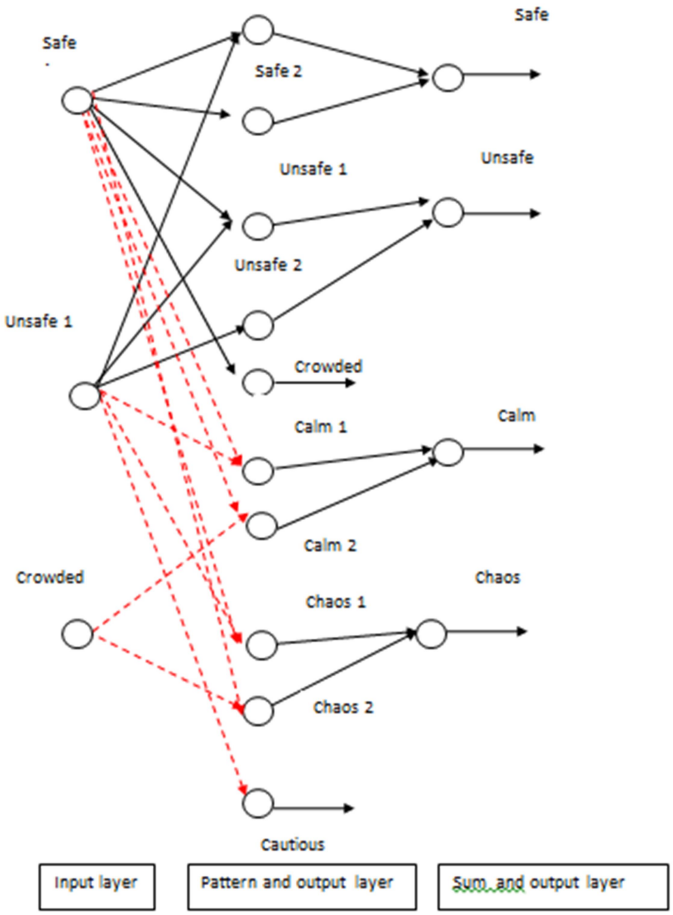



*Figure 11. Representation of the two HMMS given in figures Figure 1 and Figure 10 using the proposed approach. The dashed lines connect visible states with invisible states whereas the solid lines connect visible states with visible states*

The probability values between the states are the weights of the links as explained before assuming the index 1 is to represent the state in HMM in Figure 1 and index 2 is to represent the state in HMM in Figure 10. In this example, in the sum and output layer, we need to compute one value for each of Safe, Calm and Chaos. Let us assume the input values are Safe: 1, Unsafe: 0 and Crowded: 1. In the pattern and output layer, the state values are computed as below:

Safe1 = 0.8/1 = 0.8 using formula (4)

Safe2 = 0.7/1 = 0.7 using formula (4)

Unsafe1= 0.2 /1 =0.2 using formula (4)

Unsafe2 = 0 using formula (4)

Crowded = 0.1/1=0.1 using formula (4)

Calm1 = 0.9/1=0.9 using formula (4)

Calm2 = (0.6 + 0.3)/2= 0.45 using formula (4)

Chaos1= 0.1/1=0.1 using formula (4)

Chaos2 = (0.4+0.7)/2=0.55 using formula (5)

Cautious = 0

Based on the values obtained in the pattern and output layer, the state values for sum and output layer are computed as below:

Safe = (0.8+0.7)/2 = 0.75 using formula (7)

Unsafe = (0.2+ 0) / 2 = 0.1 using formula (7)

Calm = (0.9+0.45)/2 = 0.675 using formula (7)

Chaos = (0.1+0.55)/2= 0.325 using formula (7)

The values for Crowded and Cautious will be the same.

**3. Conclusions**

In this paper we presented a new approach to represent HMM using PNN. The work in this paper makes an attempt to utilize the advantages of HMM and PNN. The main advantages of HMM is its flexibility where it can be used in several applications, on the other hand, the main advantages of PNN is that it does not need any training and it is really good when we have small systems with accurate and complete data. The idea behind this representation is to be able to answer questions with given inputs when multiple HMMs are used and the question could not be related directly to the HMMs in the system. The proposed approach can be applied on single HMM, More the one related independent HMM or HHMM. The proposed approach uses ideas adopted from PNN where every HMM is represented in the neural network as every instance is represented in PNN. The proposed approach has at least two layers which are input layer, and pattern and output layer. The proposed representation can have multiple pattern and output layers, and the last layer is called sum and output layer in case this layer is existing. To demonstrate how the representation is implemented, four examples within three cases were presented, the discussed cases are one HMM, multiple related independent HMMs and HHMM. Some of the future work could be using and implementing this representation structure in various domains.

**References**

Araghi, L., Khaloozade, H., & Arvan, M. (2009). Ship Identification Using Probabilistic Neural Networks (PNN), Proceedings of the International MultiConference of Engineers and Computer Scientists 2009, Vol III, MECS 2009, Hong Kong, March 18 - 20.

Bhusari, V., & Pati, S. (2011). Application of Hidden Markov Model in Credit Card Fraud Detection, International Journal of Distributed and Parallel Systems, Vol.2, No.6, 203-2011.

El Emary, I., & Ramakrishnan, S. (2008). On the Application of Various Probabilistic Neural Networks in Solving Different Pattern Classification Problems, World Applied Sciences Journal, 4 (6), 772-780.

Fine, S., Singer, Y., & Tishby, N. (1998). The Hierarchical Hidden Markov Model: Analysis and Applications, in D. Haussler (Eds.), Machine Learning, 32, 41–62.

Gales, M., & Young, S. (2008). The Application of Hidden Markov Models in Speech Recognition, Foundations and Trends in Signal Processing, Vol. 1, No. 3,195-304.

Georgiou, V., Pavlidis, N., Parsopoulos, K., Alevizos, P., & Vrahatis, M. (2004). Optimizing the Performance of Probabilistic Neural Networks in a Bionformatics Task, Proceedings of the EUNITE 2004, 34-40.

Georgiou, V., Alevizos, P., & Vrahatis, M. (2008). Fuzzy Evolutionary Probabilistic Neural Networks, Artificial Neural Networks in Pattern Recognition, Lecture Notes in Computer Science, 2008, Vol. 5064, 113-124.

Grim, J., Somol, P., & Pudil, P. (2005). Probabilistic Neural Network Playing and Learning Tic-Tac-Toe, Pattern Recognition Letters-Special issue: Artificial Neural Networks in Pattern Recognition, Vol. 26, issue 12,1866-1873.

Hamdi, A., & Frigui, H. (2015). Ensemble Hidden Markov Models with Application to Landmine Detection, EURASIP Journal on Advances in Signal Processing.

Hewahi, N. (2015). Particle Swarm Optimization For Hidden Markov Model, International Journal of Knowledge and Systems Science, Vol. 6, No. 2, 1-12.

Hewahi, N. (2011a). Neuroevolution Mechanism for Hidden Markov Model, Broad Research in Artificial Intelligence and Neuroscience, Vol. 2, No. 4, 41-47.

Hewahi, N. (2011b). Genetic Algorithms Approach Towards Hidden Markov Model, Broad Research in Artificial Intelligence and Neuroscience**,** Vol. 2, issue 3,5-11.

Hewahi, N. (2011c). Probabilistic Neural networks For Rule Based Systems, International Journal of Advanced Research in Computer Science, Vol. 2, No. 2, 21-26.

Hewahi, N. (2009). Hidden Markov Model for Censored Production Rules, Proceedings of the International Conference of Information Technology, ICIT'09, Jordan, May 3-5.

Hewahi, N. (2010). 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.

Naghizadeh, S., Rezaeitabar, V., Pezeshk, H., & Matthews, D. (2012). A Modified Hidden Markov Model and Its Application in Protein Secondary Structure Prediction, Journal of Proteomics & Bioinformatics, 24-30.

Sherlock, A., Xifara, T., Telfer, S., & Begon, M. (2013). A Coupled Hidden Markov Model for Disease Interactions, Applied Statistics, Vol. 62, issue 4, 609–627, 2013.

ShivaPrasad, T., & RaghuKisore, N. (2015). Application of Hidden Markov Model for classifying metamorphic virus, Advance Computing Conference (IACC), IEEE, Banglore, India..

Specht, D. (1992). Enhancements to Probabilistic Neural Networks, International Joint Conference on Neural Networks, vol. I, 761-768.

Specht, D. (1998). Probabilistic Neural Networks for Classification, Mapping, or Associative Memory, IEEE International Conference on Neural Networks, vol. I, 525-532.

Tenyakov, A. (2014), Estimation of Hidden Markov Models and Their Applications in Finance, PhD Thesis, The University of Western Ontario.