RESEARCH ARTICLE

Channel status prediction for cognitive radio networks

Vamsi Krishna Tumuluru, Ping Wang* and Dusit Niyato

School of Computer Engineering, Nanyang Technological University, Singapore

ABSTRACT

The cognitive radio (CR) technology appears as an attractive solution to effectively allocate the radio spectrum among the licensed and unlicensed users. With the CR technology the unlicensed users take the responsibility of dynamically sensing and accessing any unused channels (frequency bands) in the spectrum allocated to the licensed users. As spectrum sensing consumes considerable energy, predictive methods for inferring the availability of spectrum holes can reduce energy consumption of the unlicensed users to only those channels which are predicted to be idle. Prediction-based channel sensing also helps to improve the spectrum utilization (SU) for the unlicensed users. In this paper, we demonstrate the advantages of channel status prediction to the spectrum sensing operation in terms of improving the SU and saving the sensing energy. We design the channel status predictor using two different adaptive schemes, i.e., a neural network based on multilayer perceptron (MLP) and the hidden Markov model (HMM). The advantage of the proposed channel status prediction schemes is that these schemes do not require a priori knowledge of the statistics of channel usage. Performance analysis of the two channel status prediction schemes is performed and the accuracy of the two prediction schemes is investigated. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS
channel status prediction; neural networks; hidden markov model; cognitive radio

*Correspondence
Ping Wang, School of Computer Engineering, Nanyang Technological University, Singapore.
E-mail: WANGPING@ntu.edu.sg

1. INTRODUCTION

1.1. Cognitive radio network

With the ever growing demand for spectrum in the unlicensed user systems (e.g., mobile Internet), dynamic spectrum allocation strategies have to be adopted. Recently, the U.S. Federal Communications Commission (FCC) has approved the opportunistic use of TV spectrum (UHF and VHF bands) by unlicensed users [1]. This concept can be extended to other licensed user systems through the cognitive radio (CR) technology [2]. Under the CR technology, a licensed user is referred to as the primary user while an unlicensed user is referred to as the secondary user because of the priority in accessing the licensed user spectrum. In most cases, the secondary users in a cognitive radio network (CRN) logically divide the channels allocated to the primary user spectrum into slots [3]. The slots left unused by the primary user are called spectrum holes or white spaces [2]. Within each slot the secondary user has to sense the primary user activity for a short duration and accordingly accesses the slot when it is sensed idle. The spectrum access by the secondary user should not cause any harmful interference to the primary user.

To minimize the interference to the primary users, the secondary users need a reliable spectrum sensing mechanism. Several spectrum sensing mechanisms were proposed in literature [4–6]; in some of which the secondary users are assumed to be able to sense the full spectrum. However, the secondary users are usually low-cost battery powered nodes. Due to the hardware constraint, they can sense only part of the spectrum [7]. On the other hand, due to the energy constraint, the secondary users may not have the willingness to waste energy to sense the spectrum part which is very likely to be busy. Hence, the key issue is to let the secondary users efficiently and effectively sense the channels in the licensed spectrum without wasting much energy. One way to alleviate this problem is by using spectrum sensing policies such as given in Reference [8,9]. The spectrum sensing policies distribute the spectrum sensing operation among different groups of nodes. These groups sense different portions in the licensed spectrum and share the sensing results...
with one another. Thus, full knowledge of the primary user activity in the licensed spectrum is obtained. Alternately, the spectrum sensing module can be made energy efficient by combining the sensing operation with a channel status prediction mechanism. The secondary user may predict the status of a channel based on the past sensing results and sense only if channel is predicted to be idle in next time slot. Thereby, the secondary user may use its sensing mechanism resourcefully. Besides, using channel status prediction, the effective bandwidth in the next slot may be estimated which allows the secondary users to adjust the data rates in advance.

The simplest way to design a channel status predictor is to use linear adaptive filters [10], which require the statistics of the licensed channel’s usage (e.g., second order statistics like autocorrelation or even higher order statistics). However, in most of the primary user systems, the channel usage statistics are difficult to obtain a priori (e.g., cellular band, public safety band, and microwave). Therefore, we explore the predictor design based on two adaptive schemes which do not require a priori knowledge of the statistics of channel usage.

### 1.2. Contribution of the paper

In this paper, we demonstrate the advantages of channel status prediction to the spectrum sensing operation in terms of improving the spectrum utilization (SU) and saving the sensing energy. Specifically, we present two adaptive channel status prediction schemes. For the first scheme, we propose a neural network approach using the multilayer perceptron (MLP) network [11], whereas for the second scheme, we introduce a statistical approach using the hidden Markov model (HMM) [12]. A qualitative analysis of the two channel status prediction schemes is performed. We evaluate the accuracy of the two prediction schemes in terms of the wrong prediction probability, denoted by \( P_e(\text{Overall}) \). Of particular interest is the wrong prediction probability (i.e., misdetection probability) given the real channel status is busy, denoted by \( P_e(Busy) \). \( P_e(Busy) \) is an important measure from the primary user’s standpoint because it indicates the level of interference to the primary user. \( P_e(\text{Overall}) \) is an important measure from a secondary user’s perspective because the goal of the secondary user is to minimize the interference to the primary users while maximizing its own transmission opportunities.

The rest of the paper is organized as follows. In Section 2, we present the related work. In Section 3, we propose the channel status predictor using the MLP neural network. In Section 4, we present the HMM-based prediction scheme. In Section 5, we present the simulation results for the two prediction schemes and demonstrate the effect of the two channel status prediction schemes in improving the SU and saving spectrum sensing energy. In Section 6, we provide a discussion on the implementation issues of the two prediction schemes. In Section 7, we present a qualitative comparison of the two prediction schemes. Finally, Section 8 concludes this paper.

### 2. RELATED WORK

The channel status prediction problem is considered as a binary series prediction problem [13]. The channel occupancy in a slot can be represented as busy or idle depending on the presence or absence of a primary user activity. A binary series \( x_1, x_2, \ldots, x_T \) is generated for the channel by sensing (or observing) the channel occupancy for a duration \( R \). The binary symbols 1 and -1 denote the busy and idle channel status, respectively. Using the binary series, the predictor is trained to predict the primary user activity in the next slot based on past observations. In a multiple channel system, a predictor is assigned to each channel.

In Reference [14], an autoregressive model using Kalman filter was used to predict the status of the licensed channel. However, this model requires knowledge of the primary user’s traffic characteristics, i.e., arrival rate and service rate, which may not be known a priori.

In Reference [13], a linear filter model followed by a sigmoid transform was used to predict the channel busy probability based on the past observations. The performance of the predictor suffered due to the non-deterministic nature of the binary series.

In Reference [15], a HMM-based channel status predictor was proposed. The primary user traffic follows Poisson process with 50% traffic intensity (i.e., 50% channel time is occupied by the primary users). The secondary user will use the whole time slot if the slot is predicted idle. However, in Reference [15], the accuracy of prediction is not provided. Another HMM-based predictor is also proposed in Reference [16], but it only deals with deterministic traffic scenarios, making it non-applicable in practice.

### 3. CHANNEL STATUS PREDICTION USING NEURAL NETWORK

The spectrum occupancy in most licensed user systems encountered in reality is non-deterministic in nature. Hence, it is appropriate to model such traffic characteristics using nonlinear adaptive schemes. Neural networks are nonlinear parametric models which create a mapping function between the input and output data. The advantage of neural networks over statistical models is that it does not require a priori knowledge of the underlying distributions of the observed process. In CRNs, it is difficult to obtain the statistics of channel usage by the primary users. Therefore, the neural networks offer an attractive choice for modeling the channel status predictor. Once the neural networks are trained, the computational complexity is significantly reduced. The neural network model, MLP, has been used in various applications, e.g., system identification and time series prediction [17,18].
3.1. MLP predictor design

The MLP network is a multilayered structure consisting of an input layer, an output layer, and a few hidden layers. Excluding the input layer, every layer contains some computing units (referred to as neurons) which calculate a weighted sum of the inputs and perform a nonlinear transform on the sum. The nonlinear transform is implemented using a hyperbolic tangent function. Neurons belonging to different layers are connected through adaptive weights. The output of a neuron $j$ in the $l$th layer, denoted by $y^l_j$, can be represented as

$$y^l_j = \frac{1 - \exp(-\nu^l_j)}{1 + \exp(-\nu^l_j)}$$  \hspace{1cm} (1)

where

$$\nu^l_j = \sum_i y^{l-1}_i w^l_{ji}$$  \hspace{1cm} (2)

Equation (2) represents weighted sum of the inputs coming from the output of the neurons in the $(l-1)$th layer using the adaptive weights (or parameters) $w^l_{ji}$, connecting the respective neurons. Equation (1) represents the nonlinear transform on $\nu^l_j$. This nonlinear transform gives an output in the range of $[-1, +1]$. If the inputs are coming from the input layer, $y^{l-1}_i$ in Equation (2) is replaced with the corresponding input. The total number of inputs in the input layer is referred to as the order of the MLP network and is denoted by $r$.

The number of hidden layers and the number of neurons in each layer depend on the application. For channel status prediction problem, we found the MLP network with two hidden layers to be sufficient. The first hidden layer has 15 neurons, the second hidden layer has 20 neurons, and the output layer has only one neuron. The order of the MLP predictor ($r$), which represents the length of the observation sequence (or slot’s status history) in our problem, is set to 4.

3.2. MLP predictor training

The MLP predictor training process is illustrated in Figure 1. The parameters of the MLP predictor are updated using the batch backpropagation (BP) algorithm [11]. The training patterns are obtained by ordering the entire binary series $x^n_i$ into input vectors $x_{r+1}^n = \{x_r, x_{r+1}, \ldots, x_{r+1}, x_{r+1}\}$ of length $t$ slots and the corresponding desired value $x_{r+1}$. For each input vector presented to the input layer of the MLP, the outputs of the neurons in each layer are calculated proceeding from the first hidden layer to the output layer using Equations (1) and (2). This computation is called forward pass. The output of the neuron in the output layer $y^l_i$ is referred to as the MLP output and is denoted by $\hat{x}_{r+1}$. $\hat{x}_{r+1}$ is treated as an estimate of the corresponding desired value $x_{r+1}$. The difference between the desired value and its estimate is called as the error $e_r$, which can be expressed as follows:

$$e_r = x_{r+1} - \hat{x}_{r+1} = x_{r+1} - y^0_r$$  \hspace{1cm} (3)

The objective of the training algorithm is to minimize this error $e_r$ by adapting the parameters $w^l_{ji}$ such that the MLP output approximately represents the desired value. In other words, the MLP predictor tries to create a mapping function between the input vector and the desired value. According to the BP algorithm [11], it is easier to minimize the mean square error than to directly minimizing the error $e_r$. The mean square error criterion can be expressed as

$$E = \frac{1}{2} \sum e_r^2$$  \hspace{1cm} (4)

Based on the BP algorithm [11], the parameters are updated as follows:

$$w_r = w_{r-1} + \Delta w_r$$  \hspace{1cm} (5)

$$\Delta w_r = -\eta \frac{\partial E}{\partial w_r} + \beta \Delta w_{r-1}$$  \hspace{1cm} (6)

In Equations (5) and (6), $w_r$ represents the parameter $w^l_{ji}$ at time instant $t$ (also denoted by $w^l_{ji}$), while $\eta$ and $\beta$ represent the learning rate and the momentum term, respectively. $\eta$ can be chosen from the range (0, 1), while $\beta$ can be chosen from the range (0.5, 0.9) in Reference [10,11]. In the simulations, the values of $\eta$ and $\beta$ are set to 0.2 and 0.9, respectively. The partial derivative $\partial E/\partial w_r$ in Equation (6) is calculated successively for each neuron by proceeding backwards from the output layer to the input layer. This computation is called backward pass. The partial derivative $\partial E/\partial w_r$ can be expressed in terms of the variables $e_r, y^l_i$.\[100-111\]
\[ \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial w_{ji}'} = \frac{\partial E}{\partial w_{ji}''} = \frac{\partial E_{\text{wi}}}{\partial y_{ji}} \frac{\partial y_{ji}}{\partial w_{ji}} \] (7)

The partial derivatives \( \frac{\partial y_{ji}}{\partial v_{ji}} \) and \( \frac{\partial v_{ji}}{\partial w_{ji}} \) in Equation (7) are calculated by the following expressions, based on Equations (1) and (2):

\[ \frac{\partial y_{ji}}{\partial v_{ji}} = (1 - y_{ji})(1 + y_{ji}) \] (8)

\[ \frac{\partial v_{ji}}{\partial w_{ji}} = y_{ji}^{l-1} \] (9)

The partial derivative \( \frac{\partial E}{\partial w_{ji}} \) is calculated in two ways depending on whether the neuron \( j \) has a desired output value or not [11]. For the neuron in output layer \( o \), a desired output value \( y_{o,i} \) exists. Therefore, \( \frac{\partial E}{\partial w_{ji}} \) for the parameter \( w_{ji} = w_{ji}' \) connecting the output layer neuron to a neuron \( i \) in the preceding layer \( (o-1) \) is expressed as

\[ \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial w_{ji}'} = \frac{\partial E}{\partial w_{ji}''} = \frac{\partial E}{\partial e_{i}} \frac{\partial e_{i}}{\partial y_{ji}} \frac{\partial y_{ji}}{\partial v_{ji}} \frac{\partial v_{ji}}{\partial w_{ji}} \] (10)

From Equation (3), \( \frac{\partial e_{i}}{\partial y_{ji}} \) is given by

\[ \frac{\partial e_{i}}{\partial y_{ji}} = -1 \] (11)

From Equation (4), \( \frac{\partial E}{\partial e_{i}} \) is given by

\[ \frac{\partial E}{\partial e_{i}} = \frac{\partial E}{\partial \epsilon_{i}} = \epsilon_{i} \] (12)

Substituting \( v_{ji}' = v_{ji}', y_{ji}^{l-1} = y_{ji}^{l-1}, \) and \( w_{ji}' = w_{ji}' \) in Equations (8) and (9), we obtain

\[ \frac{\partial v_{ji}'}{\partial v_{ji}'} = (1 - y_{ji}')(1 + y_{ji}') \] (13)

\[ \frac{\partial v_{ji}'}{\partial w_{ji}'} = y_{ji}^{l-1} \] (14)

Substituting Equations (11)–(14) in Equation (10), \( \frac{\partial E}{\partial w_{ji}} \) for the parameter \( w_{ji} = w_{ji}' \) connecting the output layer neuron to a neuron \( i \) in the preceding layer \( (o-1) \) is given by

\[ \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial w_{ji}'} = \epsilon_{i}(-1)((1 - y_{ji}')(1 + y_{ji}')(y_{ji}^{l-1}) \] (15)

For a neuron \( j \) in a hidden layer \( l \), a desired output value does not exist. In this case, \( \frac{\partial E}{\partial w_{ji}} \) is calculated in terms of an error term called local gradient, denoted by \( \delta_{j}(t) \). The local gradient \( \delta_{j}(t) \) for a neuron \( j \) in the hidden layer \( l \) can be expressed as

\[ \delta_{j}(t) = \frac{\partial E}{\partial w_{ji}'} = \frac{\partial E}{\partial y_{ji}} \frac{\partial y_{ji}}{\partial v_{ji}} \] (16)

In order to calculate the local gradient \( \delta_{j}(t) \) at time instant \( t \) for every neuron \( j \) in a hidden layer \( l \), \( \delta_{j}(t) \) is initialized at the output layer neuron and calculated recursively for every neuron \( j \) in layer \( l \) by proceeding backwards from the output layer to the first hidden layer. The initialization can be given by

\[ \delta_{j}(t) = \frac{\partial E}{\partial y_{ji}} = \frac{\partial E}{\partial e_{i}} \frac{\partial e_{i}}{\partial y_{ji}} = \frac{\partial E}{\partial e_{i}} \frac{\partial e_{i}}{\partial y_{ji}} \frac{\partial y_{ji}}{\partial v_{ji}} \] (17)

The recursive equation for \( \delta_{j}(t) \) is given by

\[ \delta_{j}(t) = (1 - y_{ji}')(1 + y_{ji}') \sum_{k} \delta_{k}^{l+1}(t)w_{j,k}^{l+1} \] (18)

The partial derivative \( \frac{\partial E}{\partial w_{ji}} \) for a parameter \( w_{ji} = w_{ji}' \) connecting the neuron \( j \) in a hidden layer \( l \) to a neuron \( i \) in the layer \( (l-1) \) is expressed in terms of \( \delta_{j}(t) \) as

\[ \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial w_{ji}'} = \frac{\partial E}{\partial y_{ji}} \frac{\partial y_{ji}}{\partial v_{ji}} \frac{\partial v_{ji}}{\partial w_{ji}} = \delta_{j}(t)y_{ji}^{l-1} \] (19)

For batch BP algorithm [11], the parameter \( w \) (or \( w_{ji}' \)) is not updated at the instant \( t \), only \( \Delta w_{r} \) is calculated. After showing all patterns, the parameter \( w \) is updated by the average value of \( \Delta w_{r} \), as follows:

\[ w_{new} = w_{old} + \frac{1}{R - r} \sum_{t=r}^{R-1} \Delta w_{t} \] (20)

where \( R \) is the length of the entire binary series \( x_{i}^{k} \).

The BP algorithm [11] is repeated until the minimum of the mean square error or the maximum number of iterations is reached. Once training is complete, we test the MLP predictor by randomly observing \( r \) successive slots \( x_{r+1 \ldots r+1}^{k} \) and computing the MLP output \( \hat{x}_{r+1}^{k} \). By using a decision threshold at the MLP output, the predicted value can be expressed as binary symbol

\[ \text{if } \hat{x}_{r+1}^{k} \geq 0 \text{ then } \hat{x}_{r+1}^{k+1} = +1 \]

\[ \text{if } \hat{x}_{r+1}^{k} < 0 \text{ then } \hat{x}_{r+1}^{k+1} = -1 \] (21)

The accuracy of the MLP predictor is evaluated in Section 5.
4. CHANNEL STATUS PREDICTION USING HMM

4.1. Hidden Markov model

Consider a system having \( N \) states, the set of states can be denoted as \( S = \{S_1, S_2, \ldots, S_N\} \). At a time instant \( t \), the system enters state \( q_t \) depending on some probabilities associated with the state transitions. If the state transitions follow the Markov property, then the probability of a state transition can be expressed as

\[
P(q_t = S | q_{t-1} = S, q_{t-2} = S, \ldots) = P(q_t = S | q_{t-1} = S)
\]  

(22)

Suppose that the states are associated with \( M \) discrete symbols, and the set of symbols is denoted as \( V = \{v_1, v_2, \ldots, v_M\} \). After every state transition, a symbol \( O_t ( \in V ) \) is emitted by state \( q_t ( \in S ) \) depending on some probability distribution. Suppose that only the symbol sequence is observable while the state sequence is hidden, this gives rise to the HMM. Therefore, the HMM can be formally stated as a statistical model in which the observed process is assumed to be generated in response to another stochastic process which is hidden and follows the Markov property.

The HMM has found remarkable applications in speech recognition [12] and bioinformatics [19]. While HMM-based channel status prediction was proposed in the literature [15,16], these papers do not give a detailed analysis of the HMM predictor design. Therefore, we explain the HMM predictor design in more detail.

In order to model the HMM, it is necessary to specify the following:

- the number of symbols, \( M \)
- the number of states, \( N \)
- the observation sequence, \( O = \{O_1, O_2, \ldots, O_T\} \)
- the state transition probabilities, \( a_{ij} = P(q_t = S_j | q_{t-1} = S_i) \) subject to the conditions \( a_{ij} \geq 0 \) and \( \sum_{j=1}^{N} a_{ij} = 1 \)
- the symbol emission probabilities, \( b_j(v_m) = P(O_t = v_m | q_t = S_j) \) subject to the conditions \( b_j(v_m) \geq 0 \) and \( \sum_{m=1}^{M} b_j(v_m) = 1 \), \( 1 \leq j \leq N \)
- initial state distribution, \( \pi = \{\pi_1, \ldots, \pi_N\} \), where \( \pi = P(q_1 = S_i) \) and satisfies the conditions \( \pi_i \geq 0 \) and \( \sum_{i=1}^{N} \pi_i = 1 \).

The HMM can be denoted by the notation \( \lambda = [\pi, A, B] \), where \( A \) is an \( N \times N \) state transition matrix containing the probabilities \( a_{ij} \) where \( i \) denotes the rows and \( j \) denotes the columns, and \( B \) is an \( N \times M \) emission matrix containing the probabilities \( b_j(v_m) \) where \( j \) denotes the row and \( m \) denotes the column.

4.2. HMM predictor

The HMM prediction scheme is illustrated in Figure 2. Consider the following sequence of channel occupancies \( \{O_1, O_2, \ldots, O_T, O_{T+1}\} \) where the channel statuses busy and idle are denoted by 1 and -1, respectively. The objective of the HMM predictor is to predict the symbol \( O_{T+1} \) based on the past \( T \) observations. To predict the symbol \( O_{T+1} \), the HMM should be able to generate the observation sequence \( O = \{O_1, O_2, \ldots, O_T\} \) with maximum likelihood probability. Hence, the parameters \( \lambda = [\pi, A, B] \) are adapted to maximize the likelihood probability of generating the observation sequence, i.e., maximize the probability \( P(O|\lambda) \).

Once training is completed, the joint probability of observing the sequence \( O \) followed by a busy slot or an idle slot at instant \( T + 1 \) is calculated. In other words, the joint probabilities \( P(O, 1|\lambda) \) and \( P(O, -1|\lambda) \) are calculated. The slot occupancy at instant \( T + 1 \) is predicted according to decision rule given by

\[
\text{if } P(O, 1|\lambda) \geq P(O, -1|\lambda) \text{ then } \hat{O}_{T+1} = 1
\]

\[
\text{if } P(O, 1|\lambda) < P(O, -1|\lambda) \text{ then } \hat{O}_{T+1} = -1
\]  

(23)

where \( \hat{O}_{T+1} \) is the predicted value.

4.2.1. HMM training

The Baum-Welch Algorithm (BWA) [12] is an iterative method to estimate the HMM parameters \( \lambda = [\pi, A, B] \) such that the probability \( P(O|\lambda) \) is maximized. To estimate the parameters \( \lambda = [\pi, A, B] \), the BWA defines the following variables:

- Forward variable \( a_t(i) = P(O_1, O_2, \ldots, O_t, q_i = S_i|\lambda) \), for \( 1 \leq i \leq N \)
Cognitive radio technology\textsuperscript{9}

1. Backward variable $\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T, \xi_t = S_i|\lambda), \text{for } 1 \leq i \leq N$
2. $\xi_t(i, j) = P(q_t = S_i, q_{t+1} = S_j|O_t, \lambda)$ for $1 \leq i, j \leq N$, the probability of being in state $S_i$ at instant $t$ and in state $S_j$ at instant $t + 1$ given the observation sequence $O$ and the model $\lambda = [\pi, A, B]$
3. $\gamma_t(i) = P(q_t = S_i|O_t, \lambda)$ for $1 \leq i \leq N$, the probability of being in state $S_i$ at instant $t$ given the observation sequence $O$ and the model $\lambda = [\pi, A, B]$.

The estimation formulas for the parameters of the model $\lambda = [\pi, A, B]$ are expressed in terms of the variables $\xi_t(i, j)$ and $\gamma_t(i)$ as follows:

\begin{equation}
\alpha_t(i) = \frac{\prod_{t=1}^{T-1} \xi_t(i, j) b_j(O_{t+1}) \hat{\beta}_{t+1}(j)}{\sum_{i=1}^{N} \prod_{t=1}^{T-1} \xi_t(i, j) b_j(O_{t+1}) \hat{\beta}_{t+1}(j)} \tag{24}
\end{equation}

\begin{equation}
b_j(v_m) = \frac{\sum_{i=1}^{T} \alpha_t(i) \gamma_t(i)}{\sum_{i=1}^{N} \alpha_t(i) \gamma_t(i)} \tag{25}
\end{equation}

\begin{equation}
\pi_t = \gamma_t(i) \tag{26}
\end{equation}

In Equation (24), the numerator represents the expected number of transitions from state $i$ to state $j$ over duration $T - 1$, while the denominator represents the expected number of times a transition is made from state $i$. The numerator in Equation (25) represents the expected number of transitions from state $i$ and symbol $v_m$ is observed after the transitions. In the Equations (24)–(26), $\xi_t(i, j)$ and $\gamma_t(i)$ are calculated as follows:

\begin{equation}
\xi_t(i, j) = \frac{\alpha_t(i) \alpha_t(j) b_j(O_{t+1}) \beta_t(j)}{P(O_t|\lambda)} \tag{27}
\end{equation}

\begin{equation}
\gamma_t(i) = \sum_{i=1}^{N} \xi_t(i, j) \tag{28}
\end{equation}

The forward and backward variables in Equations (27) and (28) are calculated recursively. The forward variable $\alpha_t(i)$ is calculated as follows:

Initialization:

\begin{equation}
\alpha_t(i) = \pi_t \beta_t(1), \quad 1 \leq i \leq N \tag{29}
\end{equation}

Recursion:

\begin{equation}
\alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq i \leq N \tag{30}
\end{equation}

Termination:

\begin{equation}
P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i), \quad 1 \leq i \leq N \tag{31}
\end{equation}

The backward variable is calculated as follows:

Initialization:

\begin{equation}
\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{32}
\end{equation}

Recursion:

\begin{equation}
\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j), \quad 1 \leq i \leq N \tag{33}
\end{equation}

Equation (31) provides the formula for the probability of observing the sequence $O$ given the model $\lambda = [\pi, A, B]$. The parameters $\lambda = [\pi, A, B]$ are re-estimated using the equations for a maximum of $K$ iterations or till the maximum $P(O|\lambda)$ is reached.

To avoid the possibility of underflow, the forward and backward variables are scaled (or normalized) while calculating Equations (29)–(33). The scaling operation [12] can be expressed as follows:

\begin{equation}
\widetilde{\alpha}_t(i) = \left( \prod_{t=1}^{T} c_t \right) \alpha_t(i), \quad 1 \leq i \leq N \tag{34}
\end{equation}

\begin{equation}
\widetilde{\beta}_t(i) = \left( \prod_{t=1}^{T-1} c_t \right) \beta_t(i), \quad 1 \leq i \leq N \tag{35}
\end{equation}

where $\widetilde{\alpha}_t(i)$ and $\widetilde{\beta}_t(i)$ represent the scaled forward and backward variables, respectively, while $c_t$ represents the scaling coefficient, which can be calculated as

\begin{equation}
c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{36}
\end{equation}

Accordingly, the variables $\xi_t(i, j)$ and $\gamma_t(i)$ are calculated by replacing $\alpha_t(i)$ and $\beta_t(i)$ with their scaled versions $\widetilde{\alpha}_t(i)$ and $\widetilde{\beta}_t(i)$ in Equations (27) and (28). Because of using the same coefficients for the forward and the backward variables, the estimation formulas given in Equations (24)–(26) remain unchanged. However, the likelihood probability $P(O|\lambda)$ cannot be truly estimated due to the scaling operation. Instead, the log likelihood probability $\log(P(O|\lambda))$ can be calculated as follows:

\begin{equation}
\log(P(O|\lambda)) = \log \left[ \sum_{i=1}^{N} \alpha_T(i) \right]
\tag{37}
\end{equation}

\begin{equation}
= \log \left[ \sum_{i=1}^{N} \left( \widetilde{\alpha}_T(i) \left/ \left( \prod_{t=1}^{T} c_t \right) \right. \right) \right]
\tag{38}
\end{equation}

\begin{equation}
= \log \left[ \sum_{i=1}^{N} \widetilde{\alpha}_T(i) \right] - \log \left[ \prod_{t=1}^{T} c_t \right] \tag{39}
\end{equation}

\footnote{The computations (29)–(33) involve multiplication of probabilities which causes the variables $\alpha_t(i)$ and $\beta_t(i)$ to tend to zero as $t$ tends to a large value.}
Thus, when scaling is used, the modified BWA estimates the model \( \lambda = [\pi, A, B] \) such that the log likelihood probability of generating the observation sequence \( O \), \( \log(P(O|\lambda)) \), is maximized.

During the training, the observation sequence length \( T \) is set to 80, the number of states \( N \) in the HMM is set to 10, and the maximum number of iterations \( K \) is set to 10. Initial values of the probabilities for state transition \( (A) \), state emission \( (B) \), and state initial distribution \( \pi \) are chosen arbitrarily. However, the conditions for \( a_{ij}, b_{j}(v_{w}), \) and \( \pi_{i} \), given in Section 4.1, should be satisfied. The log likelihood of generating an observation sequence \( O \), \( \log(P(O|\lambda)) \) by the HMM at every iteration \( k \) of the training is shown in Figure 3.

4.2.2. HMM testing.

After training, the joint probabilities \( P(O, 1|\lambda) \) and \( P(O, -1|\lambda) \) are calculated as follows:

\[
P(O, 1|\lambda) = \sum_{i=1}^{N} \alpha_{T+1}^{-1}(i) \tag{39}
\]

\[
P(O, -1|\lambda) = \sum_{i=1}^{N} \alpha_{T+1}^{+1}(i) \tag{40}
\]

where \( \alpha_{T+1}^{-1}(i) \) and \( \alpha_{T+1}^{+1}(i) \) are given by

\[
\alpha_{T+1}^{-1}(i) = \left[ \sum_{j=1}^{N} \alpha_{T}(j)a_{ji} \right] \cdot b_{i}(O_{T+1} = -1), \quad 1 \leq i \leq N \tag{41}
\]

\[
\alpha_{T+1}^{+1}(i) = \left[ \sum_{j=1}^{N} \alpha_{T}(j)a_{ji} \right] \cdot b_{i}(O_{T+1} = 1), \quad 1 \leq i \leq N
\]

Based on \( P(O, 1|\lambda) \) and \( P(O, -1|\lambda) \), the predicted value is found using Equation (23). When scaling is used, the log likelihood probabilities \( \log(P(O, 1|\lambda)) \) and \( \log(P(O, -1|\lambda)) \) are calculated similar to Equation (38) and the channel status at the time instant \( T + 1 \) is predicted in favor of the maximum of the two values. The accuracy of the HMM predictor is evaluated in Section 5.

5. SIMULATION AND ANALYSIS

For the purpose of simulation, the primary user traffic on a channel is assumed to follow Poisson process.\(^1\) The ON and OFF times of the channel are drawn from geometric distributions. For different traffic scenarios, we vary the traffic intensity \( \rho \) and the mean inter-arrival time \( t_{\text{inter}} \) of the traffic bursts. The traffic intensity is related to the mean inter-arrival time as follows.

\[
\rho = \frac{\text{mean ON time}}{\text{mean ON+OFF time}} = \frac{t_{\text{ON}}}{t_{\text{inter}}} \tag{43}
\]

where \( t_{\text{ON}} \) is the mean time that the primary user is active on a channel for each traffic burst. A minimum of 50% traffic intensity is maintained for a channel.

The accuracy of the two prediction schemes is evaluated using two performance measures, \( P_{\rho}^{s}(\text{Overall}) \) and \( P_{\rho}^{s}(\text{Busy}) \), for various traffic scenarios. We also investigate the probability of wrongly predicting the idle channel status (i.e., the channel is predicted to be busy when it is actually idle or the so-called false-alarm probability), denoted by \( P_{\rho}^{0}(\text{Idlle}) \). For both prediction schemes, it is observed that for a given mean inter-arrival time \( t_{\text{inter}} \) and traffic intensity \( \rho \), there are only small changes in the probability \( P_{\rho}^{s}(\text{Idlle}) \). Hence, \( P_{\rho}^{s}(\text{Idlle}) \) is shown for different mean inter-arrival times \( t_{\text{inter}} \) but only for traffic intensity of \( \rho = 0.5 \).

5.1. Performance of the predictors under stationary traffic conditions

First, we evaluate the performance of the two predictors under stationary\(^2\) traffic conditions. We have chosen the same traffic settings as used in Reference [15]. The length

\(^{1}\) In the simulation, we use Poisson process to generate the primary user traffic. However, the channel status predictors are not restricted to Poisson process and are applicable to any traffic distribution.

\(^{2}\) Stationary means that for each simulation scenario, the primary user traffic statistics (i.e., the traffic intensity \( \rho \) and the mean inter-arrival time \( t_{\text{inter}} \)) remain unchanged over time.
of the testing data for both the predictor models under each traffic setting is chosen as 30,000 slots. Figure 4 shows the performance of the MLP predictor in predicting the busy channel status for various traffic scenarios. It can be seen that the MLP predictor performance improves when the traffic intensity $\rho$ increases. This is because, for a given mean inter-arrival time $t_{\text{inter}}$, as $\rho$ increases, the number of slots occurring with busy channel status also increases which leads to more correlation in the primary users’ channel occupancy data. However, when $\rho$ is equal to 50%, there is little correlation in the primary users’ channel occupancy data. Hence, the worst case scenario occurs when the traffic intensity $\rho$ is 0.5 (mean ON time = mean OFF time). The correlation also decreases as the mean inter-arrival time decreases.

Figure 5 shows the overall performance of the MLP predictor for various traffic scenarios. It can be seen that $P^5_{\text{Overall}}$ is slightly higher than $P^5_{\text{Busy}}$ under the same traffic scenario. This is because of the inclusion of the wrong predictions when channel status is idle. Table I shows the performance of the MLP predictor in predicting the idle channel status. The traffic intensity is set to 50%. It can be observed that the probability of wrongly predicting the idle channel status, $P^5_{\text{Idle}}$, ranges from 4.8 to 10.1% when the mean ON/OFF time varies from 11 slots to five slots, respectively.

The performance of the HMM predictor in predicting the busy channel status for different traffic scenarios is shown in Figure 6. Similar to the result shown in Figure 4, for a given mean inter-arrival time $t_{\text{inter}}$, the performance improves as the traffic intensity $\rho$ of the channel increases.

Figure 7 shows the overall performance of the HMM predictor for various traffic scenarios. Similar to the MLP

![Figure 4](image_url)

**Figure 4.** Performance of the MLP predictor in predicting the busy channel status for various traffic scenarios.

![Figure 5](image_url)

**Figure 5.** Overall performance of the MLP predictor in predicting the channel status for various scenarios.

![Figure 6](image_url)

**Figure 6.** Performance of the HMM predictor in predicting the busy channel status for various traffic scenarios.

![Figure 7](image_url)

**Figure 7.** Overall performance of the HMM predictor in predicting the channel status for various scenarios.

<table>
<thead>
<tr>
<th>Mean ON time (slots)</th>
<th>Mean OFF time (slots)</th>
<th>$P^5_{\text{Idle}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>0.101477</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>0.063528</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>0.058236</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>0.052713</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>0.047875</td>
</tr>
</tbody>
</table>
The HMM predictor also shows a slightly higher value for \( P^\text{r}(\text{Overall}) \) than the probability \( P^\text{r}(\text{Busy}) \) under the same traffic scenario. This is due to the same reason as observed for the MLP predictor. Table II shows the probability of wrongly predicting the idle channel status \( P^\text{r}(\text{Idle}) \) for the HMM predictor when we set the traffic intensity to 50% and vary the mean ON/OFF time. It is observed that \( P^\text{r}(\text{Idle}) \) ranges from 5.1 to 10.3% when the mean ON/OFF time varies from 11 slots to five slots, respectively.

### 5.2. Performance of the predictors under non-stationary traffic conditions

Consider a licensed channel with different primary user traffic distributions at different time intervals (e.g., each interval has 6000 slots) as shown in Table III. Consider the training data for the MLP predictor to be generated during the interval \([t_0, t_1]\). The testing data for both the predictors is obtained during the interval \([t_1, t_2]\). The performance of the MLP and the HMM predictors is analyzed by calculating the percentage of wrong predictions in the duration \([t_1, t_2]\). The percentages of wrong predictions for the MLP and HMM predictors are 6.31 and 7.08%, respectively. The performance of the MLP predictor can be improved by retraining the MLP for a short duration in each interval. For instance, when the MLP is retrained using 1000 observations at the beginning of each interval, the percentage of wrong predictions reduces to 4.07%.

### 5.3. Performance enhancement due to channel status prediction

We demonstrate the advantages of applying the two channel status prediction schemes in spectrum sensing using two performance measures (to be explained):

- Percentage improvement in SU, denoted by \( \text{SU}_{\text{imp}}(\%) \),
- Percentage reduction in sensing energy, denoted by \( \text{SE}_{\text{red}}(\%) \).

#### 5.3.1. Improvement in spectrum utilization

Consider a CRN containing \( N_\text{ch} \) licensed channels with different primary user traffic distributions. Every channel is logically divided into slots and the slot size is constant over all the channels. Each secondary user (CR) is able to sense only one channel during a slot due to the hardware constraint. We also assume that every secondary user stores a short history of the sensing results for every channel. This information can be collected from neighbors over a common control channel.

We consider two types of secondary users, \( \text{CR}_{\text{sense}} \) device and \( \text{CR}_{\text{predict}} \) device. We assume that both device types use the same sensing mechanism and have the same level of sensing accuracy. A \( \text{CR}_{\text{sense}} \) device randomly selects a channel at every slot and senses the status of that channel, while a \( \text{CR}_{\text{predict}} \) device individually predicts the status of all channels based on their respective slot history, before sensing. The channel to be sensed by the \( \text{CR}_{\text{predict}} \) device is randomly selected among those channels with idle predicted status.

SU can be defined as the ratio of the number of idle slots discovered by the secondary user to the total number of idle slots available in the system over a finite period of time (e.g., 30,000 slots).

\[
\text{SU} = \frac{\text{Number of idle slots sensed}}{\text{Total number of idle slots in } N_\text{ch} \text{ channels}}
\]

(44)

The percentage improvement in SU due to channel status prediction can be expressed as

\[
\text{SU}_{\text{imp}}(\%) = \frac{\text{SU}_{\text{predict}} - \text{SU}_{\text{sense}}}{\text{SU}_{\text{sense}}}
\]

(45)

where \( \text{SU}_{\text{sense}} \) and \( \text{SU}_{\text{predict}} \) represent the SU for the \( \text{CR}_{\text{sense}} \) and \( \text{CR}_{\text{predict}} \) devices, respectively. Substituting Equation (44) in Equation (45), \( \text{SU}_{\text{imp}}(\%) \) can be given by

\[
\text{SU}_{\text{imp}}(\%) = \frac{I_{\text{predict}} - I_{\text{sense}}}{I_{\text{sense}}}
\]

(46)

where \( I_{\text{sense}} \) and \( I_{\text{predict}} \) represent the number of idle slots sensed by the \( \text{CR}_{\text{sense}} \) and the \( \text{CR}_{\text{predict}} \) devices, respectively.

In the simulation, we consider the system with different number of channels \( N_\text{ch} \) (the channels are added sequentially according to Table IV). Table V shows that a \( \text{CR}_{\text{predict}} \) device using MLP and HMM predictors can discover more idle slots than a \( \text{CR}_{\text{sense}} \) device. The percentage of improvement in SU is more than 68% when the \( \text{CR}_{\text{predict}} \) device uses MLP (HMM) predictor.
Table IV. Different channel models for the primary user system.

<table>
<thead>
<tr>
<th>Channel index</th>
<th>$\rho$</th>
<th>$t_{total}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5625</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>0.6667</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>0.6818</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>0.7</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td>18</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>22</td>
</tr>
<tr>
<td>9</td>
<td>0.5</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>0.7</td>
<td>20</td>
</tr>
</tbody>
</table>

5.3.2. Reduction in sensing energy.

Considering a single licensed channel scenario, a $CR_{sense}$ device senses all the slots whereas a $CR_{predict}$ device only senses when the channel status of the slot is predicted to be idle. In other words, when the slot status is predicted to be busy, the sensing operation is not performed, thereby saving energy. We assume both device types use the same sensing mechanism and have the same level of sensing accuracy. If we assume one unit of sensing energy is required to sense one slot, then the total sensing energy required for a $CR_{sense}$ device in a finite duration of time (e.g., 30,000 slots) can be given by

$$SE_{sense} = \text{(Total number of slots in the duration)} \times \text{(unit sensing energy)} \quad (47)$$

while the total sensing energy required by the $CR_{predict}$ device can be given by

$$SE_{predict} = SE_{sense} - \left( \left( B_{predict} \right) \times \text{(unit sensing energy)} \right) \quad (48)$$

where $B_{predict}$ is the total number of busy slots predicted by the $CR_{predict}$ device.

Therefore, using Equations (47) and (48), the percentage reduction in the sensing energy can be given by

$$\text{SE}_{red}(\%) = \frac{SE_{sense} - SE_{predict}}{SE_{sense}} \times 100 \quad (49)$$

Table VI shows the percentage of reduction in the sensing energy for different traffic settings when $CR_{predict}$ devices with MLP predictor and HMM predictor are used, respectively. It can be seen that as $\rho$ increases for a given $t_{total}$, more busy slots are predicted and hence more sensing energy is saved.

6. IMPLEMENTATION ISSUES

In this section, the aspects related to model selection and computational complexity are discussed.

6.1. Model complexity of MLP predictor

The MLP predictor can predict the channel status based on a small number of inputs (channel statuses of the past slots) $\tau$. From the simulations, we found that for channel status prediction $\tau$ in the range of [4, 10] slots is sufficiently long. However, there is no straightforward method to identify the optimal number of hidden layers and the optimal number of neurons in each layer. In the simulations, the size of the MLP predictor is chosen arbitrarily. By selecting sufficient number of neurons and appropriate learning rate, the channel status predictor can be implemented using the MLP network. A detailed analysis of the parameter selection for the MLP network is explained in Reference [11].

6.2. Computational complexity of MLP predictor

The MLP uses two computation phases during training, namely the forward pass and the backward pass. However, only the forward pass is required after training. Unlike the HMM model, the MLP predictor is trained only once and the training is done offline. The computational complexity of the MLP predictor is directly related to the network size. Considering the forward pass, with the addition of a
In the measurements process of different traffic intensities ρ and different mean inter-arrival times $t_{\text{E}}$, neuron to the layer $j$, the number of multiplication and addition operations increases by $(L_i + L_k)$ and $(L_i - 1) + L_k$, respectively, where $L_i$ and $L_k$ are the number of neurons in the layers preceding and succeeding layer $j$, respectively. However, if a layer $j$ is added to the MLP network, the number of multiplication and addition operations increases by $L_i(L_i + L_k)$ and $L_i(L_i - 1) - L_k(L_i - 1)$, respectively, where $L_i$ denotes the number of neurons in layer $j$. The MLP predictor may suffer from the local minima problem which slows down the training. However, the faster versions of the BP algorithm can be used to overcome the local minima problem [20].

### 6.3. Model complexity of HMM predictor

In the simulations, the number of states $N$ in the HMM predictor is kept the same when modeling different observation sequences $O = \{O_1, O_2, \ldots, O_T\}$ and predicting the symbol $O_{T+1}$ following the sequences. However, it is not an optimal solution, since $N = 10$ might not yield the maximum likelihood probability $P(O|\lambda)$ for some observation sequences. To determine the optimal number of states for modeling an observation sequence is extremely challenging and even infeasible in practice because of the large number of trials needed [12]. Hence for practical reason, the number of states is fixed during the simulations. Table VII shows the performance of the HMM predictor for different states $N$. For this experiment, the length of the observation sequences $T = 80$, maximum number of iterations $K = 10$, and the traffic model has mean inter-arrival time $t_{\text{E}} = 20$ and $\rho = 0.5$. It can be seen that the predictor performance improves with the increase in the number of states. However, as the number of states $N$ increases, the number of model parameters to be estimated $\lambda = [\pi, A, B]$ also increases.

#### 6.4. Computational complexity of HMM predictor

To predict the channel status at the $(T + 1)$th time instant, we have to compute the estimates of the model $\lambda = [\pi, A, B]$ such that the probability $P(O|\lambda)$ is maximized. Unlike the MLP predictor, the HMM predictor training is done in real time (online learning). As shown in Figure 2, the estimation and prediction steps are repeated when a new observation $O$ is made. Hence, it is necessary that all computations are completed before the occurrence of the $(T + 1)$th slot for the HMM predictor to be applicable.

The length of the observation sequence is chosen as $T = 80$ slots so that the state transition probabilities and state emission probabilities are reliably estimated. By increasing the value of $T$, a better estimate of the model $\lambda = [\pi, A, B]$ can be obtained. However, if larger values of $T$ are used in real-time then the HMM has to wait for a longer duration to collect the observation symbols. Alternately, we may investigate approaches whereby $T$ can be iteratively reduced.

The computational complexity of the HMM predictor is related to the number of states $N$ and the size of the observation sequence $T$. Consider the computation of the forward variable $\alpha_t(i)$ which is computed after training, it takes approximately $N^2T$ calculations.

### 7. QUALITATIVE COMPARISON

The HMM-based channel status prediction schemes [15,16], mentioned in Section 2, do not provide details of the model like number of states $N$ and length of the observation sequence $T$, hence making it difficult for comparison with the two proposed channel status prediction schemes. In this section, the pros and cons of the two proposed channel status prediction schemes is provided through a qualitative comparison of their performance and implementation issues which arise during the design. Table VIII shows a comparison of the properties of the two proposed prediction schemes.
<table>
<thead>
<tr>
<th>Property</th>
<th>MLP predictor</th>
<th>HMM predictor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive parameters</td>
<td>$w_I^*$</td>
<td>$A, B$ and $\pi$</td>
</tr>
<tr>
<td>No. of slot observations</td>
<td>$t$</td>
<td>$T$</td>
</tr>
<tr>
<td>No. of adaptive parameters</td>
<td>$\sum L_i L_{i+1}$</td>
<td>$N$</td>
</tr>
<tr>
<td>Training criteria</td>
<td>Minimize mean square error</td>
<td>Maximize $\log(P(O</td>
</tr>
<tr>
<td>Mode of training</td>
<td>Offline, only once</td>
<td>Online, repeated</td>
</tr>
<tr>
<td>Problems during training</td>
<td>Local minima problem</td>
<td>No problems</td>
</tr>
<tr>
<td>No. of calculations in prediction after training</td>
<td>$\sum (L_i L_{i+1} + L_{i+1} (L_i - 1))$</td>
<td>$N^2 T$</td>
</tr>
</tbody>
</table>

$L_i$ denotes the number of neurons in the layer $i$ of the MLP predictor, $i$ ranges from input layer to the last hidden layer. For the input layer, $i$, simply denotes the number of inputs.

7.1. Performance comparison

It is evident from the simulations that the two prediction schemes perform similarly under the same traffic scenario while the performance of MLP predictor is slightly better than that of the HMM predictor. This is because, the MLP predictor is a fully trained model whereas the HMM predictor is not optimally designed to cater to all observation sequences, because the number of states in the HMM is fixed.

Further, the MLP predictor requires fewer past observations (or slot occupancy history) than the HMM predictor to predict the channel status in the next slot.

The MLP predictor is trained only once whereas the HMM predictor is trained repeatedly (see Figure 2). However, for time varying traffic scenario, the MLP predictor can be retrained periodically for better performance.

7.2. Design comparison

During the training, the MLP predictor can get trapped in local minima but the HMM predictor faces no such problem. However, the local minima problem can be remedied by using modified versions of the BP algorithm [20].

Both the prediction schemes require considerable computations and memory to store the parameters for the training process. However, after training, the memory requirement for the MLP predictor is significantly reduced. The number of parameters in the MLP predictor is higher than that of the HMM predictor. However, the number of computations after training is lower for the MLP predictor than that for the HMM predictor. In our simulations, the MLP predictor has 380 parameters while the HMM has 130 parameters, while the number of calculations after training is 724 for MLP predictor and 8000 for HMM predictor.

8. CONCLUSION

Channel status prediction is important to CRNs because it can greatly save the sensing energy and help the secondary users to exploit the spectrum holes more efficiently. A reliable channel status prediction mechanism should ensure a lower probability of wrong predictions of the channel status. As the statistics of channel usage in CRNs are difficult to be determined, we rely on adaptive schemes which do not require such a priori knowledge. We have investigated two such adaptive schemes for channel status predictor design, a novel MLP predictor and a HMM predictor. A qualitative analysis of the two prediction schemes has been presented using various simulations. We have also presented the issues regarding the design of the MLP network and the HMM as channel status predictors.

REFERENCES

Access Networks (DySPAN), Baltimore, Maryland, USA, November 2005; 224–232.

AUTHORS’ BIOGRAPHIES

V. K. Turnuluru, P. Wang and D. Niyato

Cognitive radio technology