Predictive Channel Access in Cognitive Radio Networks based on Variable order Markov models

by

Chamara Devanarayana

A Thesis
submitted to the Faculty of Graduate Studies of The University of Manitoba,
in Partial Fulfilment of the Requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba R3T 5V6 Canada

Predictive Channel Access in Cognitive Radio Networks based on Variable order Markov models

by

Chamara Devanarayana

A Thesis
submitted to the Faculty of Graduate Studies of The University of Manitoba,
in Partial Fulfilment of the Requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Abstract

The concept of Cognitive radio enables the unlicensed users to share the spectrum with licensed users, on the condition that the licensed users have preemptive priority. The use of the channel by unlicensed users should not result in more than acceptable interference level to the licensed users, if interference occurs. The sense and react strategy by unlicensed users sometimes does not lead to acceptable level of interference while maintaining an acceptable data transfer rate for the unlicensed users.

Proactive channel access has been proposed for the purpose of reducing the interference to primary users and to reduce the idle channel search delay for the secondary users. There are many methods used in the literature to model the channel state fluctuations. Based on these models the future channel states are predicted.

In this thesis we introduce a predictive channel usage scheme which is capable of reducing the interference caused by the unlicensed users. Furthermore our scheme is capable of increasing the data rates the unlicensed users experience through the reduction of the idle channel identification delay. In our scheme no assumptions are made about the distribution of licensed user channel usage. We learn the traffic characteristics of the channels using a learning scheme called Probabilistic Suffix Tree algorithm.

Acknowledgements

The valuable time and effort of many individuals has gone into coming up with this thesis although my name appear on the front page. First of all I would like to thank my advisor Dr. Attahiru S. Alfa for the time, guidance, inspiration and the encouragement provided from the start of the program to the instant of completion of the thesis. It would not have been possible for me to complete it without his valuable support.

Secondly, I would like to thank all the lecturers in the department of Electrical and Computer Engineering and outside the department for their valuable support and guidance. Next I would like to thank the administrative staff of the Faculty of Graduate Studies and the department of Electrical and Computer Engineering. Specially to our graduate student advisor for all the help provided.

A big thank you go to all my colleagues in the CNER lab whose support and friendship was invaluable. Another big thank you goes to my friends and relatives in Winnipeg whose affection suppressed the feeling that I am far away from home.

Last but not least I am forever in debt to my parents for the person I am today. I cannot thank you enough for the love and support you provided me all this time.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Electromagnetic Spectrum is a natural resource. The assignment of frequencies in this spectrum is managed by government organizations such as the FCC in the US and the Spectrum and Radio Policy directorate in Canada, which assigns the spectrum to organizations on fixed long term contracts to operate on a particular geographic region. Although this method worked in the past, vast growth in the Wireless services has pushed the FCC to find more efficient spectrum allocation mechanisms to avoid spectrum scarcity [1]. The measurement operation carried out by the FCC's Enforcement Bureau in 2002, in single locations in the US, for a limited time period gave some insight into the real cause of the widely accepted notion of spectrum scarcity [2]. According to the report of the Spectrum Efficiency working group, while some bands are heavily used there exists bands which have availability in either time, frequency or both. The recommendations made by the Spectrum efficiency working group to the FCC consists of promoting flexible use of spectrum, development and deployment of advanced technologies and promoting secondary markets for spectrum [2]. A novel technology called by the names Cognitive Radio (CR), neXt Generation (XG) or Software Defined Radio has the capability to implement those recommendations [1]. In what follows in this section we present the concepts, functionalities and the applications of Cognitive radio.

1.1 Overview of Cognitive Radio

As mentioned above there is an issue of"artificial spectrum scarcity" [3] created by the static allocation policy of spectrum management institutions. Due to this policy there exists what is called spectrum holes in the electromagnetic spectrum. The definition of a spectrum hole can be given as:"*A spectrum hole is a band of frequencies assigned to a primary user, but, at a particular time and specific geographic location, the band is not being utilized by that user*" [4]. In this definition the primary users refer to the entities that have entered into a contract with the spectrum management institutions to use a particular band of frequency in particular geographic location. These spectrum holes can be exploited by the use of Cognitive radios which opportunistically use the spectrum. These cognitive radios are called by the name Secondary Users or Unlicensed users. In order to exploit the spectrum holes efficiently such that the primary users are not affected significantly, the Cognitive radios should be able to identify which portion of the spectrum is busy and which portion is idle, select the best available channel, share the channels with other secondary users and to release the channel when the primary user appears [5].

As one can see cognitive radio being an opportunistic user of the spectrum should have a lot of capabilities to operate on diverse frequency bands (eg: GSM900, GSM1800, UHF, VHF), diverse standards (eg: GSM, WLAN, WiMAX) and diverse modulation schemes, and should be capable of switching from one standard to another in a very short time duration [6]. Since the cognitive radio is using the spectrum owned by some other entity it should sense the spectrum to find out whether the licensed user is appearing. This sensing can be complex since the licensed users will be using different standards on different frequency bands. The hardware and the software in the cognitive radio should have a high degree of flexibility in order to provide the above mentioned capabilities. This flexibility is referred to as reconfigurability. A cognitive radio is capable of reconfiguring itself instantaneously if necessary. Furthermore a cognitive radio should be capable of supporting multiple services such as telephony, data and video streaming, and should be capable

of supporting two or more independent transmitting and receiving channels at the same time [6].

A cognitive radio possesses two main characteristics, cognitive capability and reconfigurability. The ability to detect the availability of spectrum holes, analyzing the characteristics of the spectrum holes and deciding on what band to use depending on the data rate, bandwidth and transmission mode are called the cognitive capabilities [5]. The reconfigurability refers to the ability of the cognitive radio to change, the frequency of operation, the modulation scheme used, the transmission power used and the communication standard used [5]. The frequency of operation may have to be reconfigured due to the appearance of a primary user on a previously discovered spectrum hole or when a more suitable channel is discovered. Some applications have low error tolerance while others require more throughput. When the spectrum hole or channel band is finite the cognitive radio has to use a modulation scheme with high efficiency to get higher throughput and this increases the error rate. The amount of transmit power should be controlled within the regulatory maximum so that one gets the tolerable error rate at the data rate one requires. Then another secondary user who is sufficiently far away can use the same channel there by increasing the spectrum efficiency. The cognitive radio should be able to change from one communication standard to another almost instantly when required.

Another feature in Cognitive radio which helps to provide Quality of Service is the "*spectrum mobility*" [5]. This feature enables the cognitive radio to reconfigure itself seamlessly so that in a spectrum hand off the applications running on the cognitive radio suffers minimum performance degradation. As one can see this feature should work on all the layers on the network stack to provide a seamless spectrum hand off. Spectrum hand off means the change of operating frequency band due to the appearance of a primary user, change of spectrum holes due to movement in space or can be due to fading and shadowing.

We will discuss the most desirable feature in cognitive radio, spectrum sensing, in the next section.

1.2 Overview of Channel Sensing methods

Channel sensing is one of the key functionalities of a cognitive radio. The concept of cognitive radio is tightly wound over channel sensing. In the cognitive radio jargon channel sensing means much more than the measurement of signal power over some bandwidth. In cognitive radio channel sensing involves the gathering of information on the usage of a channel over a number of dimension such as frequency, time, space and code [7]. Channel sensing in frequency domain implies the detection of frequency bands which are not used. In time domain it implies the detection of temporal variations of a particular band. The space dimension includes both distance and azimuth [7]. The advancements in antenna design have enabled beam-forming thus the cognitive radio should not only measure the power level in a particular band but also the direction from where the signal arrives [7]. Channel sensing in code dimension refers to the identification of spread spectrum signals which can be Frequency Hopping Spread Spectrum (FHSS) or Direct Sequence Spread Spectrum (DSSS) and finding out whether there exists possibility to use an orthogonal code to transmit secondary payload [7].

As one can see channel sensing or spectrum sensing in cognitive radio requires antennas that have a wide operating range, analog to digital converters which sample in the range of GHz with very high resolution, low noise power amplifiers that have a wide dynamic range and digital signal processors which are capable of operating in extreme speeds [7]. The other challenge in spectrum sensing is the hidden terminal problem. This problem arises when the signals from the primary user operating on a particular channel is undetectable at the cognitive user due to shadowing, severe multipath fading or high penetration loss of a building [8]. As one can see the cognitive radio is not always in the propagation path between the primary transmitter and primary receiver, thus the propagation path between primary transmitter and cognitive radio receiver is independent of the path between primary transmitter and receiver. Therefore the cognitive radio not only has to have a receiver which is 30-40dB more sensitive than the primary receiver but also has to make decision of

whether the channel is in use looking at the measurements of the indirect path [9]. These are the main challenges on spectrum sensing faced by researchers these days.

In channel sensing what we basically do is, testing whether the primary user is present or not. Thus it can be reduced to a binary hypothesis testing problem, where the two hypotheses are given below [8]:

$$\begin{aligned} \mathcal{H}_0 &: \text{Primary user is not in operation} \\ \mathcal{H}_1 &: \text{Primary user is in operation} \end{aligned}$$

Then the performance of the sensing method can be measured by the probability of false alarm P_f, probability of missed detection P_m and probability of correct detection P_d. False alarm is mistakenly concluding the presence of primary user activity on the channel, when there is none. Missed detection is concluding that, the primary user is not operating on a channel, when primary user is operating. They can be mathematically expressed as [8]:

$$P_f = Prob\left\{Decision = \mathcal{H}_1 | \mathcal{H}_0\right\}, \tag{1.2.1}$$

$$P_m = Prob\left\{Decision = \mathcal{H}_0 | \mathcal{H}_1\right\}, \tag{1.2.2}$$

$$P_d = Prob\left\{Decision = \mathcal{H}_1 | \mathcal{H}_1\right\}. \tag{1.2.3}$$

In the next subsection we discuss some of the channel sensing methods that exist in the literature and their pros and cons.

1.2.1 Matched filter Detection

Matched filter detection is the most optimal detection technique where the signal characteristics of primary user such as bandwidth, operating frequency, modulation type and order,

pulse shaping and frame format is known by the secondary user [7]. This method is an optimal detection method of a signal in the presence of additive white Gaussian noise [8]. In the case of additive white Gaussian noise the received signals given the respective hypotheses can be presented by the following formula [10].

$$\begin{aligned} \mathcal{H}_0: &\quad Y[n] = W[n] && n = 0, 1, ..., N-1 \\ \mathcal{H}_1: &\quad Y[n] = X[n] + W[n] && n = 0, 1, ..., N-1 \end{aligned} \tag{1.2.4}$$

In the above equation $W[n] \sim \mathcal{N}(0, \sigma^2)$. The optimum matched filter can be obtained by the correlation of the unknown signal $Y[n]$ with the known signal or the template of the signal $X[n]$ and finding out whether it is larger than a threshold γ. Which can be formulated as [10]:

$$T(\mathbf{Y}) = \sum_{n=0}^{N-1} Y[n]X[n] \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma \tag{1.2.5}$$

The test statistic $T(\mathbf{Y})$ is derived from the Neyman-Pearson criteria which tries to maximize the probability of detection for a given probability of false alarm [11]. According the the Neyman-Pearson criteria hypothesis $\mathcal{H}_1$ is chosen if the condition given in the Inequality (1.2.6) is met, where $\mathbf{Y}$ is a vector of length n, $prob(\mathbf{Y}|\mathcal{H}_i)$ is an n dimensional normal distribution conditioned on hypotheses $\mathcal{H}_1$ and $\mathcal{H}_0$, and γ is a threshold. The quantity $L(\mathbf{Y})$ is known as the likelihood ratio.

$$L(\mathbf{Y}) = \frac{prob(\mathbf{Y}|\mathcal{H}_1)}{prob(\mathbf{Y}|\mathcal{H}_0)} \geq \gamma \tag{1.2.6}$$

For a given P_d and P_f the number of samples required can be given as a function of $\frac{1}{SNR}$ when no fading or shadowing is assumed [12]. Therefore as the SNR lowers the number of samples required grows as $O\left(\frac{1}{SNR}\right)$. The main advantages of matched filter detection are optimality in detection and short sensing time [13]. The main disadvantages are the requirement of perfect knowledge of transmitted signal, requirement of perfect synchronization with the transmitter and high power consumption since various receiver algorithms

have to be executed, having separate one for each communication standard [13].

1.2.2 Energy Detection

This method is also known by the name energy meter or periodogram. It is the most generally used method to sense because of its low complexity and its capability to sense without the knowledge of the transmitted signal [7]. In this case we compare the energy of the signal with a threshold γ which can be represented as [10]:

$$T(\mathbf{Y}) = \sum_{n=0}^{N-1} Y^2[n] \underset{\mathcal{H}_0}{\overset{\mathcal{H}_1}{\gtrless}} \gamma \tag{1.2.7}$$

In this case different formulas can be derived for probability of detection and false alarm considering presence or absence of multi-path fading. The Reader is referred to [7, 10, 12] for more information on that. The main advantages of this scheme are simplicity in implementation, low computational complexity and scheme being optimal in the case primary user signal is unknown [13]. The main disadvantages in this scheme include requiring accurate knowledge of noise power in finding out the threshold γ, significant degrading of performance in fading and shadowing, inability to differentiate noise, primary user signals and secondary user signals and inefficiency in detecting spread spectrum signals [7, 13].

1.2.3 Cyclostationary Detection

Naturally in communication signals there exists one or multiple periodicities within their random fluctuations due to sine wave carriers and repetitive pulsing or keying [14]. Although the secondary user has no knowledge about the carrier phase or clock timing these periodicities can be exploited in detection of primary user signals using cyclostationary detection. For more information see [14]. The advantages of this scheme include the ability to distinguish between primary user, noise and secondary user, not requiring complete information about the signal but have to know the cyclic frequencies, being more effective in sensing direct sequence spread spectrum signals and having high performance in low SNR

conditions [7, 13, 14]. The main disadvantage is the complexity of computations [13].

1.2.4 Wavelet based Detection

In this method the spectrum is sensed as a whole within a wide band without using tunable Bandpass filters to measure the signal power in that narrow band [15]. A wide band of frequencies, where the information on the starting frequency and ending frequency is known to the receiver, is analyzed using the Wavelet transform and the sub-bands are identified by the property that, within sub-bands the power spectral density has a smooth distribution but between adjacent sub-bands the power spectral density changes abruptly [15]. After identifying the sub-bands the signal power level in each sub band is estimated. This scheme requires extremely high sampling rates which may be impossible in some instances and have to have very fast signal processors to calculate the power spectral density of the wide band signal and to get the wavelet transform. Considering these facts the authors of [15] came up with another scheme using sub-Nyquist sampling in [16]. In this scheme the spectrum is assumed sparse in the frequency domain and compressed sensing is used to get a coarse power spectrum density function and the wavelet transform is used to identify the sub-bands.

1.3 Different spectrum sharing models for Cognitive Radio.

As given in [5] the spectrum sharing models can be differentiated according to the network architecture, spectrum allocation behavior and spectrum access technique. The cognitive radio networks can be categorized as Centralized networks and Decentralized networks according to the network architecture. Then they can be categorized according to the spectrum allocation behavior as Cooperative and Non-cooperative. Finally they can be categorized as Overlay and Underlay according to the spectrum access technique. These

categories are further explained in what is to follow in this subsection.

1.3.1 Centralized Network

In the centralized network a secondary base station or a central spectrum server makes the decisions on channel assignment to Cognitive radio users [17]. In this architecture the cognitive radio base station gathers the spectrum sensing information from the secondary users. Using this information it runs an assignment algorithm and informs the secondary users of the channels to be used on a predefined broadcast channel [17]. As one can see having the global picture the base station can provide a globally optimal channel assignment but according to [17] there are some problems with this approach. The first one would be the requirement of a common control channel which should be pre-assigned and all secondary users should have interference free access. The assumption of existence of such a channel is assumed in most spectrum sharing solutions [5]. The second drawback is the processing complexity of the base station being high. In [18] the authors assume that secondary users coordinate with each other using a low power ultra-wide-band channel as in the underlay networks.

1.3.2 Decentralized Networks

Decentralized networks are mainly used in the cases where building infrastructure is not preferred [5]. In this architecture the cognitive radios use distributed algorithms in performing spectrum assignment. Each user does its own channel sensing and coordinates with the nearby cognitive radios and determines who gets which channel [17]. These algorithms are iterative so the neighborhood to which the cognitive radio sends data and from which it receives data should be limited. Otherwise it can incur large delays and the network will become unscalable [17]. The issue of common control channel is there in distributed architecture too, since to communicate the transmitter and the receiver should perform a handshake.

1.3.3 Cooperative and Non-cooperative networks

In general all the centralized cognitive radio networks are cooperative networks and some decentralized networks can be cooperative as well [5]. In [19] they formulate Game theoretic models for both Non-cooperative and Cooperative networks. The main difference between these schemes lies in the choice of the utility function. In non-cooperative networks each user tries to maximize the gain it can achieve without considering about the utility of others. In contrast to that in cooperative games when a user tries to maximize its own utility it tries to satisfy the minimum utility required by other users in the network. An advantage of Non-cooperative networks is the advantage of being able to function with minimum communication with other cognitive radio users. The Cooperative networks can improve the overall network utilization but have higher communication overheads. So in situations where exchange of excess information is not feasible one should choose a non-cooperative solution.

1.3.4 Overlay and underlay networks

In overlay networks a cognitive radio senses narrow band channels and finds out a channel which is not used by a primary user and transmits on that channel. If multiple channel are available it uses the best available channel. Best channel can be in terms of signal to noise ratio, bandwidth, or the expected *idle* time left in the channel. Thus it tries to avoid interference to the primary users. On the other hand underlay networks spread the signal power over the entire bandwidth so that they appear as noise to the primary user. This spreading of the signal energy can be done by using techniques like Code division multiple access [20, 21] or Ultra wide band radio [22]. But this should be done with care since if the cognitive radio increases the noise floor of the primary user above a particular level it will cause interference to it. In [23], the authors investigate the impact caused on primary users measured by the outage probability using overlay, underlay and interference avoidance underlay. The interference avoidance underlay scheme also spreads its power

over the entire band but it notches out or nullifies the frequencies in which the cognitive radio senses primary user activity [23]. In the analysis the authors of [23] found out that, interference avoidance schemes introduce less interference to the primary users and in the event of imperfect sensing, interference avoidance underlay scheme performs better than the overlay scheme.

1.4 Applications of Cognitive Radios

In the 802.22 standard of the IEEE Cognitive radios are considered as a solution to provide rural areas with broadband Internet. These kinds of networks are called by the name Wireless regional area networks where the network is a centralized one with a radius of 33 km [24]. The VHF and the UHF spectrum bands are released for the purpose of making of rules for this standard. These bands are chosen because the broadcast television spectrum is very sparsely used and it has very good propagation characteristics. The primary users who should be protected in these frequency bands are the television broadcasters and viewers, Wireless microphone users in concerts, private land mobile radio services and commercial mobile radio services [24]. In this standard minimum planned downstream data rate is 1.5 Mbps and the upstream peak throughput is 384 kbps [24].

Another application of cognitive radio is a military application called joint tactical radio system which is considered as the backbone of the US Army's proposed future combat system. Originating in mid 1990s the main aim of the joint tactical radio system program was to replace the 25-30 families of radio systems with software defined radios which can operate in the entire frequency spectrum [25]. These software defined radios should also be able to communicate with the legacy radios used by the department of defense. The main purpose of the joint tactical radio system is enabling the seamless connectivity to all levels command and providing direct access airborne and battle field sensors [25]. Since the future combat system overwhelms the spectrum currently owned by the department of defense they will have to use dynamic spectrum allocation. One of the cognitive radio

applications in joint tactical radio system is given in [26]. This application tries to adapt the radio band usage from air to ground based on the GPS coordinates of the location of the aircraft so that it will not cause interference to communications of ground troops.

Applying cognitive radio concepts on public safety is proposed by the FCC [27] to provide the first responders with better connectivity in emergency situations. Usually in this kind of scenarios the radio spectrum usage becomes concentrated both in space and in time since all the emergency workers will have no alternative communication methods other than the wireless radios [28]. Using cognitive radios one can avoid using the congested radio bands in contrast to the legacy radios, where the band of operation is fixed. In addition the cognitive radios having the capability of operating in diverse standards and frequencies, can communicate with the legacy radios. Another issue in public safety bands is they being scattered in a large frequency band and being owned by different authorities. Thus that strict allocation policies create artificial spectrum scarcity in emergency situations and cognitive radio can be a solution to that [28].

1.5 Importance of Channel State Prediction

The channel sensing can be classified into two categories; one is the reactive channel sensing and the other one is the proactive channel sensing. Most of the work in cognitive radio channel sensing is done on reactive sensing and only a few researchers have worked on proactive sensing.

In reactive sensing the secondary users only sense the spectrum when they have some pay load to deliver to another secondary user or receive from another secondary user. So assuming secondary users use periodic narrow band sensing, they have to sense the licensed spectrum sequentially till they find an idle band to use [29]. It is hard to gain some statistical knowledge out of this channel sensing, since the sensing operation for a given channel is performed at random times [29]. The knowledge of regular channel sensing outcomes can be used to create a statistical model of the primary user channel usage for each licensed

channel. Using this model it is possible for the secondary user, to find out the most optimal channel sensing pattern and intelligently schedule channel access in advance [3,29]. Building this statistical channel usage model and using it to schedule channel sensing is called proactive channel sensing or channel state prediction.

The sense and react policy of the secondary users will unavoidably disrupt communications of both the secondary users and primary users since the appearance of a primary user within the time period between sensing slots is undetectable to the secondary user [3]. Because of the unexpected interruptions in this reactive sensing policy the secondary user cannot meet any quality of service requirements and they do not have any idea of how long the current idle slot is going to last on average. This knowledge on the channel behavior can be used to predict the effective bandwidth in the next time slot which allows the cognitive radio to adjust data rates accordingly [30].

In proactive channel sensing the secondary users have to periodically sense the channels even if they do not have any data to be sent. In the scheme discussed in this thesis we require the secondary users to sense the channels even though they do not have any data to send since the other secondary users are dependent on it. This periodic regular sensing policy in proactive channel sensing may seem to be draining power from the cognitive radio system but it can reduce the idle channel search delay when the secondary user wants to transmit or receive a payload and there will be less interruptions to both secondary and primary users. Thus there is an advantage in using the channel state prediction schemes.

1.6 Outline of the thesis

The objective of this thesis is to propose a channel state prediction algorithm which can be used to reduce the interference felt by the primary users and to increase the throughput of the secondary users. We use a predictor which depends on the Markov properties observed in the channel state fluctuation patterns. The algorithm we use for prediction is called the probabilistic suffix tree algorithm.

The contributions presented in this thesis are:

1. Use of Probabilistic Suffix Tree algorithm for channel state modeling and prediction. This method has been successfully used in biological sequence analysis, speech and language modeling, text analysis and extraction, music generation and classification, hand writing recognition etc.

2. Ordering of the available licensed channels in order to reduce the idle channel discovery delay based on the expected idle time in each channel.

3. Switching the channel used by the secondary user pro-actively based on channel state prediction in order to reduce interference to both primary and secondary users. We create a channel model using an algorithm called Probabilistic suffix tree algorithm and based on that do the predictions.

This thesis is organized as follows: Chapter 2 contains a literature review of the channel state prediction models found in the literature. In the same chapter we are discussing the pros and cons of those schemes with respect to our model. Chapter 3 presents an overview of the algorithm we use to build the channel model. We predict the channel states using this channel model. Chapter 4 presents our channel state prediction scheme, which we use to pro-actively switch the channels used by the secondary users when they are already transmitting. The same channel state prediction scheme is used in selecting the channels to look for spectrum opportunities when they are starting to transmit data. In Chapter 5 we present the simulation results of our prediction scheme and of another scheme found in the literature. Chapter 6 concludes the thesis. In addition, Appendix A presents an example of the probabilistic suffix tree algorithm and the respective variable order Markov model.

Chapter 2

Survey and comparison of the existing schemes with the proposed scheme

There are many methods in the literature used for channel state prediction. These methods can be divided mainly into two categories. The first category of prediction schemes assumes a channel usage model for the licensed channels (eg: [29] , [3]) and the predictions are carried out using the assumed model. The second category of predictors tries to model the channel usage pattern based on the past channel usage data. This second category of predictors uses one of three approaches to model the channel usage namely Hidden Markov Models, Neural networks or Regression techniques (eg: [30], [31] and [32]). The method we propose uses a technique called Probabilistic Suffix tree algorithm which belongs to the category of Variable order Markov Models.

2.1 Prediction of channel status using past channel usage data

In this subsection we discuss few of the methods used in channel usage modeling. After building these models the cognitive radio will be able to predict the future channel states given the past channel states. Channel state in this context means whether the channel is

occupied by a primary user or not. We call the channel state *busy* when the channel is occupied by a licensed user and call it *idle* when the channel is not occupied.

2.1.1 Modeling and prediction of licensed bands using Hidden Markov Models

The first channel sensing model we are going to discuss is the Hidden Markov model. It is called a hidden Markov model since the states of the Markov chain are not physically identifiable. The only visible part to the outside world is the symbol which is generated by a distribution conditioned on the state the Markov chain is currently in. A Hidden Markov process is a bivariate stochastic process [33]. The process consists of a finite state homogeneous Markov chain and another set of conditionally independent random variables given the above mentioned Markov chain [34]. The Markov chain cannot be observed externally and thus given the name Hidden Markov Process [33]. The only observation that can be made is of the random variable which only depends on the state of the Markov chain at a given instant.

A discrete time hidden Markov model consists of a *N*-state $\mathbf{Q} = \{q_1, q_2, ..., q_N\}$ Markov chain and at each transition time *t* the Markov chain enters into a new state based on a transition probability matrix $\mathbf{A}$ given the previous state. Then at that particular time instant a discrete observation output symbol $v_i \in \mathbf{V} = \{v_1, v_2, ...v_M\}$ is produced based on a probability distribution $\mathbf{B}$ which only depends on the current state of the Markov chain. For example $v_i \in \mathbf{V} = \{idle, busy\}$. Let us take a coin tossing example where a person behind a screen is tossing a coin. The person has two coins one is a fair coin and the other is biased towards the heads, and the probability of getting a head by tossing it, is 0.6 . The person who is tossing chooses the fair coin with probability 0.6 and the biased coin with probability 0.4, given he/she used the fair coin in the previous experiment. He/She uses the fair coin with probability 0.3 and the biased coin with probability 0.7, given he/she used the biased coin in the previous experiment. So as we can see for an outside person only

the result of the toss is visible. The coin with which the toss is carried out is not known to the observer. In this example the state space $\mathbf{Q} = \{fair, biased\}$. The distributions of the state transition probability vector whose states are *fair* and *biased* for this example is given by $\mathbf{A}_{toss}$, which is shown in Equation (2.1.1). The probability distribution of the occurrence of *heads* or *tail* given the type of coin for this example is given by $\mathbf{B}_{toss}$, which is shown in Equation (2.1.2). The general initial state distribution of the Markov chain is given by $\pi = (\pi_1, \pi_2, ..., \pi_N)$ which can be the steady state distribution or some other user defined distribution [33]. $\mathbf{A}$ and $\mathbf{B}$ in general, are defined in equations (2.1.4) and (2.1.5) respectively [33]. In compact notation a hidden Markov model is usually expressed mathematically as $\lambda = (\mathbf{A}, \mathbf{B}, \pi)$.

$$\mathbf{A}_{toss} = \begin{cases} prob(fair|fair) = a_{\mathcal{FF}} = 0.6 \\ prob(biased|fair) = a_{\mathcal{FB}} = 0.4 \\ prob(fair|biased) = a_{\mathcal{BF}} = 0.3 \\ prob(biased|biased) = a_{\mathcal{BB}} = 0.7 \end{cases}, \tag{2.1.1}$$

$$\mathbf{B}_{toss} = \begin{cases} prob(heads|fair) = prob(\mathcal{H}|\mathcal{F}) = 0.5 \\ prob(tail|fair) = prob(\mathcal{T}|\mathcal{F}) = 0.5 \\ prob(heads|biased) = prob(\mathcal{H}|\mathcal{B}) = 0.6 \\ prob(tail|biased) = prob(\mathcal{T}|\mathcal{B}) = 0.4 \end{cases}. \tag{2.1.2}$$

$$\begin{aligned} \mathbf{A} &= \{a_{ij}\}, \\ a_{ij} &= Pr(q_j \text{ at time } t+1 | q_i \text{ at time } t), \end{aligned} \tag{2.1.3}$$

$$\begin{aligned} \mathbf{B} &= \{b_j(k)\}, \\ b_j(k) &= Pr(v_k \text{ at time } t | q_j \text{ at time } t). \end{aligned} \tag{2.1.4}$$

The above mentioned parameters $\mathbf{A}$, $\mathbf{B}$ and π should be estimated based on a given string of past observation data. The algorithm used in this estimation is the Baum-Welch algorithm [31]. A separate model has to be built for each channel if the channel usage is different statistically. If the model for a given channel j is $\lambda_j = (\mathbf{A}_j, \mathbf{B}_j, \pi_j)$ one can calculate the probability of the next slot being *idle* or *busy* given the model and the channel usage data. Let the channel usage data string be $y_1^T = (y_1, y_2, ..., y_T)$ where $y_i = \{busy, idle\}, \forall i = 1, 2, ..., T$. The decision on whether the next slot is going to be *idle* in channel j is based on the following inequality where δ is a threshold.

$$Pr(y_1^T, idle|\lambda_j) - Pr(y_1^T, busy|\lambda_j) \geq \delta \tag{2.1.5}$$

$Pr(y_1^T, idle|\lambda_j)$ is the probability of having the channel state sequence $y_1\ y_2\ y_3\ ...\ y_T\ idle$ according to the hidden Markov model of the channel j. If Inequality (2.1.5) is satisfied the next slot is predicted to be *idle*. The method to calculate these probabilities $Pr(y_1^T|\lambda_j)$ can be found in [31] and [33]. If we recall the coin tossing example and let $y_1^2 = heads\ tail$ we can calculate the $Pr(y_1^2, heads|\lambda_{toss})$ as given in Equation (2.1.6). Let the initial state distribution be $\pi_{toss} = \{prob(fair) = \pi_{\mathcal{F}} = 0.5, prob(biased) = \pi_{\mathcal{F}} = 0.5\}$. $Pr(y_1^2, heads|\lambda_{toss})$ is the probability of getting a sequence of coin flipping out comes $heads\ tail\ heads$ using the model $\lambda_{toss} = \{\mathbf{A}_{toss}, \mathbf{B}_{toss}, \pi_{toss}\}$, where the outcome in the right hand side is the most recent. According to [33] this calculation has very high complexity if done directly so a method called Forward-backward algorithm is used to reduce the complexity. When using hidden Markov models one has to maintain a large number of past observations and the model parameter estimation techniques are highly complex [32].

$$\begin{aligned} Pr(y_1^2, heads|\lambda_{toss}) \ = \ & \sum_{i=\mathcal{F},\mathcal{B}} \sum_{j=\mathcal{F},\mathcal{B}} \sum_{k=\mathcal{F},\mathcal{B}} \pi_i \cdot prob(\mathcal{H}|i) \cdot a_{ij} \cdot prob(\mathcal{T}|j) \\ & \cdot a_{jk} \cdot prob(\mathcal{H}|k) \end{aligned} \tag{2.1.6}$$

2.1.2 Modeling and prediction of licensed bands using Binary time series

The data the cognitive radio gathers is the state of the channel which is either *idle* or *busy*. Therefore for each channel the data can be viewed as a time series. Using this time series of channel states the conditional success of the current trial can be estimated [32]. Let the current status of the channel be given by B_t, where $B_t = 0$ when the channel is *idle* and $B_t = 1$ when the channel is *busy*. Probability of the channel being *busy* in the next time slot conditioned on the information known upto time $t-1$ $\mathcal{F}_{t-1}$, can be given as $P_\beta(B_t = 1|\mathcal{F}_{t-1})$ [35]. The information upto time $t-1$ $\mathcal{F}_{t-1}$, can include the past data on the channel status and/or any other variable or a set of variables which is known to influence the channel status. Where $\beta = (\beta_1, \beta_2, ...\beta_d)$ denotes the model parameters and d is the order of the model. Using regression for some parameter vector β, the expectation of the channel state in the next time slot given the information available upto the time instant $t-1$ can be represented by Equation (2.1.7) [32].

$$\mu_t = E[B_t|\mathcal{F}_t - 1] = \sum_{k=1}^{d} \beta_k B_{t-k} \tag{2.1.7}$$

The expected value of B_t should lie between $[0, 1]$ but it cannot be assured in the above Equation [36]. Therefore a continuous function that maps any real number to the above range should be used. The family of such functions is known as link functions [32]. The authors of [32] use the Sigmoid function. To find the value of μ_t the parameter vector β should be determined first. This estimation is done using a vector of past observations. As one can see the above discussed model is for a single channel. In the work discussed in [32] they consider a regression model with multiple channels. If one has R channels the states of the R channels in a time epoch t can be written as a vector $\mathbf{S}_t = [S_{1t}, S_{2t}, ..., S_{Rt}]^T$. Using these vectors they device a $VAR(p)$ model, where p is the number of past time slots we use, in predicting the future channel states. In modeling the channel states as a $VAR(p)$ model, they find the covariance and the intercept vectors using genetic algorithms

and neural networks and find the probability of the next slot being idle of R channels, given the history of those channels for the past p time slots. For more information about $VAR(p)$ models refer to [37]. In this multivariate binary regression model since the function has to be transformed using a link function, which is non-linear, a closed form solution is not possible so a numerical approach is taken [32]. As one can see in this technique the process of finding out the coefficient matrices are complex and a globally optimum solution is not possible due to the higher dimension.

2.1.3 Modeling and prediction of licensed bands using Multiple Layer Perceptron

In this method the aim is to develop a model which takes in a vector of past channel observations and predicts the next channel observation. In this method the next channel outcome is predicted directly rather than giving a probability. Multiple layer perceptron is an Artificial Neural network which arranges neurons in layers and connect them using weighted lines. A multiple layer perceptron has an input layer, an output layer and between those layers it has some hidden layers. The models created have a non-linear relationship between inputs and output [30]. As in the previous methods in [30] the channel states consist of *idle* and *busy* and the values -1 and 1 represents the respective states. To train the neural network the input and the respective output data are required. Those values are obtained by the binary time series of -1 and 1, taken by sampling the channel for a time period. Let that vector be represented by $\mathbf{x}_1^T = (x_1, x_2, ..., x_t, ..., x_T)$. If the number of inputs of the input layer of the neural network is τ and we want to predict the status of the next time slot $t+1$ using the past τ time slots, the input vector would be $\mathbf{x}_{t-\tau+1}^t = (x_t, x_{t-1}, ..., x_{t-i}, ..., x_{t-\tau+1})$. The number of inputs in the multilayer perceptron is called the order of the neural network [30].

Two layers of a multi layer perceptron is given in Figure 2.1.1. Note that not all connections are shown to avoid making the graph unreadable. All the nodes of layer $l-1$ should be connected to each node in layer l. If the output of node j in layer $l-1$ is y_j^{l-1}

$\forall j = \{1, 2, ..., m\}$, the input to a node i in layer l is the weighted sum of outputs of each node in layer $l - 1$. Since the outputs should be restricted to $[-1, 1]$ a hyperbolic tangent function is used on the weighted sum which is a non-linear transform function [30].

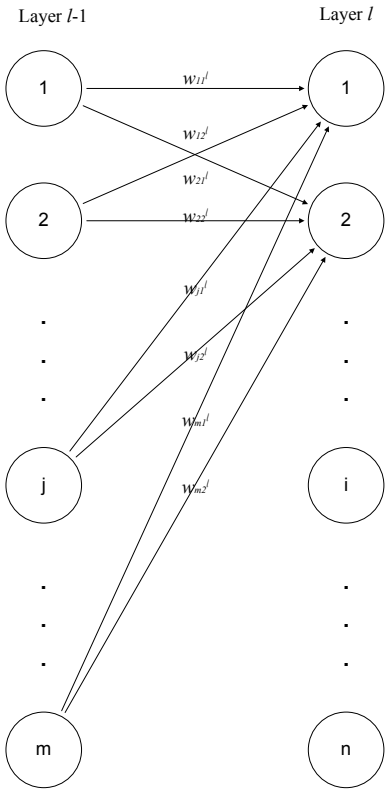


Figure 2.1.1: Two layers of a Multy Layer Perceptron

All the layers are similar to Figure 2.1.1, other than the input layer which does not have any neurons or computation units [30]. The output of the neuron i in the l^{th} layer can be represented by Equation (2.1.8) [30].

$$y_i^l = \frac{1 - \exp\left(\sum_{j=1}^{m} y_j^{l-1} w_{ji}^l\right)}{1 + \exp\left(\sum_{j=1}^{m} y_j^{l-1} w_{ji}^l\right)} \tag{2.1.8}$$

The weights on each edge are then optimized so that the output gives minimum error for the training sequence. This weight optimization is done using the Back propagation algorithm [30]. Further information with regard to the Batch propagation algorithm can be found in [30].

In the multilayer perceptron technique the number of layers and the neurons per layer depends on the application [30]. Thus for different traffic patterns different architectures are required. The complexity of training is high for neural networks which can be a problem if the traffic patterns change depending on the time of the day.

2.2 Prediction of channel status using an assumed channel usage model

In contrast to the earlier subsection, the models discussed here assume a distribution for the usage of a channel by a primary user and based on that channel model, predict the channel status of the next time slot, based on the history of the channel usage. In this section we will discuss mainly three types of assumed channel models such as independent Erlang distributed alternating *busy* and *idle* times [29], independent exponentially distributed alternating *busy* and *idle* times [29] and alternating-periodic exponential model [3].

In all of the models mentioned above an expression is derived for the probability of a particular channel being idle, given the sensing result of the past time slot. For the Erlang distributed case and exponentially distributed case the expression is derived using the theory of alternating renewal processes. For these two cases the channel alternates between *busy* and *idle* states and the duration for which the channel is in either of the states is statistically governed by Erlang distribution and the exponential distribution respectively. The *busy* and *idle* durations have different rate parameters and they are assumed to be independent of each other. In the alternating-periodic exponential model the period of *idle* (*busy*) is exponentially distributed with parameter λ while the *busy* (*idle*) duration is fixed at a constant T [3].

The equations of the formulas for the probability of a channel being in the state *idle* in the next time slot, given the information on the channel state on the current time slot for Erlang, Exponential and Alternating-periodic exponential channel usage patterns are given in equations (2.2.1) taken from [29],(2.2.2) taken from [3] and (2.2.3) taken from [3]

respectively. Where Δ_i is the time elapsed since the last sensing operation, λ_{Y_i} and λ_{X_i} are the rate parameters of the distribution of the *busy* and *idle* times respectively, $\hat{\Delta}_i = \Delta_i - nT$ and d_i is the most recent sensing result available.

$$P^i_{idle}(\Delta_i) = \begin{cases} \frac{1}{2} + \frac{1}{2}\exp(-\Delta_i)\cos(\Delta_i) & \text{if } d_i = idle \\ \frac{1}{2} - \frac{1}{2}\exp(-\Delta_i)\cos(\Delta_i) & \text{if } d_i = busy \end{cases} \tag{2.2.1}$$

$$P^i_{idle}(\Delta_i) = \begin{cases} \frac{\lambda_{Y_i} + \lambda_{X_i}\exp(-(\lambda_{X_i}+\lambda_{Y_i})\Delta_i)}{\lambda_{X_i}+\lambda_{Y_i}} & \text{if } d_i = idle \\ \frac{\lambda_{Y_i} - \lambda_{X_i}\exp(-(\lambda_{X_i}+\lambda_{Y_i})\Delta_i)}{\lambda_{X_i}+\lambda_{Y_i}} & \text{if } d_i = busy \end{cases} \tag{2.2.2}$$

$$P^i_{idle}(\Delta_i) = \begin{cases} \frac{1}{T}\int_0^T \sum_{n=0}^{\lfloor \frac{\Delta_i - x}{T} \rfloor} \frac{\lambda^n(\hat{\Delta}_i - x)^n}{n!\exp(\lambda(\hat{\Delta}_i - x))} dx & \text{if } \Delta_i > T \\ \frac{1}{T}\int_0^{\Delta_i} \exp(-\lambda(\Delta_i - x))dx & \text{if } \Delta_i < T \end{cases} \tag{2.2.3}$$

The methods discussed above assumed that the channel usage behaves statistically according to a particular distribution. Exponential distribution is a memory less distribution which means the residual time does not depend on how long the process has been going on. Field studies done on the sojourn time of *idle* and *busy* times have shown that this assumption of exponentially distributed sojourn times of channel states cannot be taken for granted [38]. Therefore to find the closest match for the sojourn times a distribution fitting must be done and due to the fact that most data networks have self similarity none of the above discussed models will be close enough approximations [39].

2.3 Proactive channel ordering scheme

In the prediction schemes which use Binary Time series regression model [32] and Erlang distributed channel usage model [29] described above, the main goal is to arrange the

available channels in the descending order of the probability of the channel being idle in the next time slot. When ordered in this pattern the cognitive radio senses the channels in this order which reduces the idle channel search time [29]. The ordering of the channels in [34] is done based on the difference between the probability of having an *idle* slot and probability of having a *busy* slot. The channels are ordered only if this difference is greater than a threshold.

In [3] after calculating the probability of the channel being *idle*, the channels are ordered based on two schemes. The two methods are Proactive planning I and Proactive planning II. In Proactive planning I scheme, the channels are ordered in the descending order of the length of the remaining *idle* period, which is shown in Equation (2.3.1) [3]. In Proactive planning II scheme the channels are ordered based on the probability that the remaining idle time of a channel i being greater than that of the current channel and that probability being greater than a threshold. The equation of the probability is given in Equation (2.3.2) [3], where λ_{x_c} is the rate parameter of the *idle* time distribution of the current channel, and P_c and P_i are the probability of the current channel c and some other given channel i being *idle* in the next time slot respectively.

$$E(T_i) = \frac{P_i}{\lambda_{x_i}} \tag{2.3.1}$$

$$P(T_i > T_c) = P_i - \frac{\lambda_{x_i}}{\lambda_{x_i} + \lambda_{x_c}} P_i P_c \tag{2.3.2}$$

When ordering channels in the Proactive planning II scheme, channels are put in order of probability $P(T_i > T_c)$ from highest to the lowest if that probability is greater than a threshold [3].

2.4 Proactive channel switching

In this scheme secondary users switch the channel they use, if the current channel in use is predicted to be occupied by a primary user or if there exists a channel which has better expected idle time than that of the current channel in use [3]. When switching the channels based on prediction information the secondary user looses the opportunity to transmit in the time taken to switch the channel. But if the channel, it switches to has a better remaining *idle* time, which even compensates for the lost transmit time, then that loss of switch time will not be a waste. If the secondary user acts reactively, which means switching only if it detects a primary user on it's current channel in use, then it causes interference to the primary user for some short time period and in that time period the secondary data sent may have a high error rate, making the data useless.

As one can see the benefits of proactive switching depend on the accuracy of the prediction model [3]. A prediction scheme which is perfect does not exist. Therefore prediction errors are inevitable, which gives rise to wrong switching decisions. In [3] they identify two kinds of wrong switching decisions. The first one is called *Dumb Switch I* which refers to the scenario of secondary user predicting a channel j to be better than channel i and finding out that the channel j is occupied after switching the channel to channel j. This scenario causes the secondary user to interfere with the primary user and loose the available time on channel i to transmit. The second type of wrong switching is called *Dumb Switch II*, which refers to a secondary user switching from channel i to channel j, predicting channel j has longer *idle* time than the current channel i, and finds out actually shorter *idle* time is available in channel j. If the prediction scheme avoids these two dumb switches it is called a *Smart Switch* [3]. Prediction schemes can be evaluated using the percentage of *Dumb Switch I*, *Dumb Switch II* and *Smart* switches made in given time period. Graphical representations of these three kinds of switchings are given in figures 2.4.1, 2.4.2 and 2.4.3 respectively. In the simulations done we evaluate our scheme based on these measures.

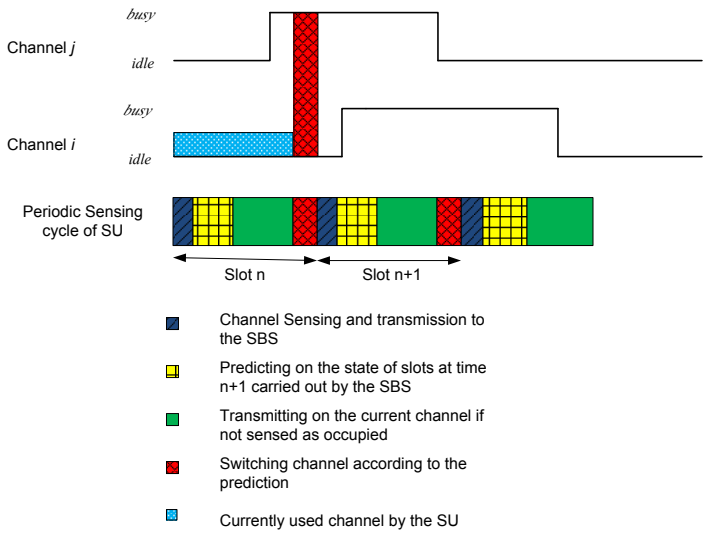


Figure 2.4.1: An illustration of Dumb Switch 1

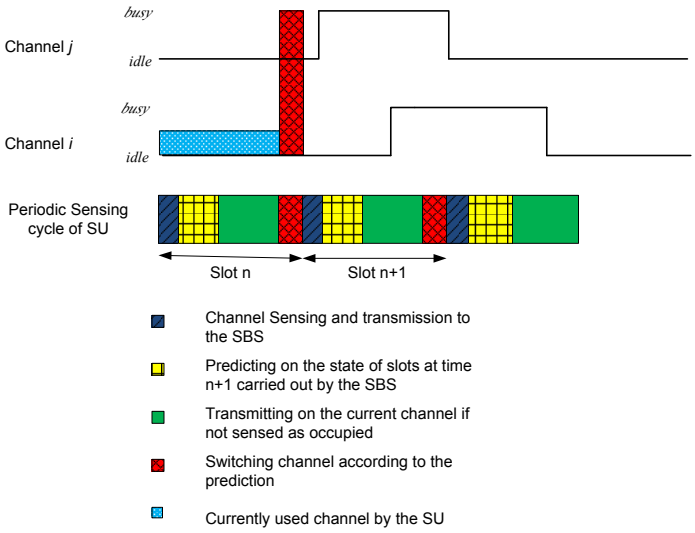


Figure 2.4.2: An illustration of Dumb Switch 2

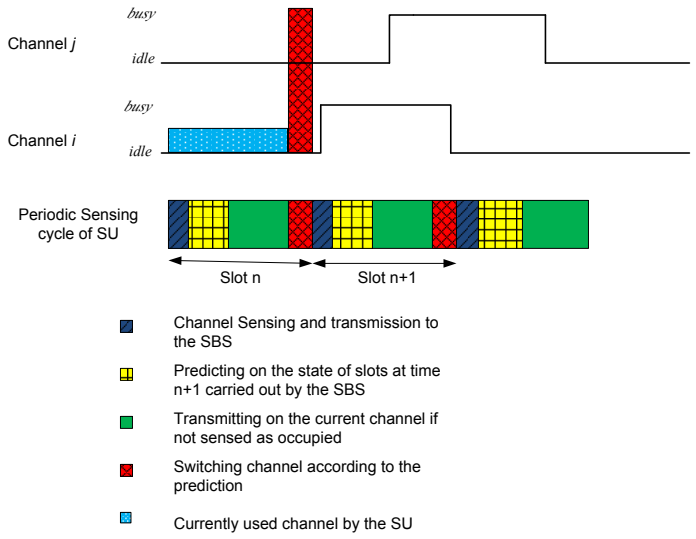


Figure 2.4.3: An illustration of Smart Switch

Chapter 3

Fundamentals on Probabilistic Suffix Tree (PST) Algorithm

The most commonly used Markov chain in the literature is the one step Markov chain, i.e. a chain which keeps memory of only the one last step. In this thesis we are using a multiple step Markov chain. Instead of keeping one step memory, we keep memory up to m steps. We call the amount of memory the Markov chain keeps the Order of the Markov Chain. So if the order of the Markov chain is m, the next transition probabilistically depends on the last m observations. For example let '1' and '0' represent the states *busy* and *idle* of a particular channel, a fixed order Markov model of order m has strings of channel observations of the past m time slots, as its state labels. It means that there are 2^m number of states in this particular case. So as one can see the the number of states grows exponentially with m and only Markov chains with low order can be practically used [40]. The higher the number of states the higher will be the complexity of calculating the probabilities. On top of that only a fraction of the states will be highly used. An example of a fixed order Markov model of order two is shown in Figure 3.0.1. The bold arrows indicate transitions with the occurrence of a *busy* sensing result and the dashed arrows represent the transitions with an *idle* sensing result. The values indicated on the arrows are the probabilities at which the given transition takes place, for example P_{32} is the probability of going from state '11' to

state '10'. In this figure, we add the new observation to the right hand side of the string labeling the current node at transitions and choose a number of symbols which is equal to the order of the model starting from the right hand side. Symbolically the the state at time n, $A_n = \{\sigma^{n-m+1}\ \sigma^{n-m+2}\ ...\sigma^n\}$. Where σ^{n-j} is the channel state observation j time slots before the current observation.

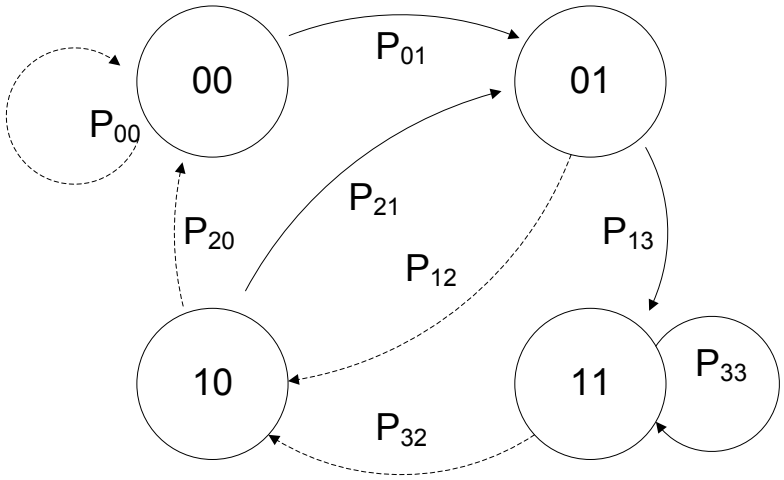


Figure 3.0.1: An example of a fixed order Markov model of order two

In variable order Markov models as opposed to fixed order Markov models, we only consider the states which are highly abundant on the training sequence. Therefore the number of states in variable order Markov models are less than that of fixed order Markov model of the same order [41]. Furthermore it contains a set of parameters which enables one to control the number of states. If we take a variable order Markov model of order $D = 2$, its states are labeled with binary strings of length $l \leq 2$. The set of states in variable order Markov model is a subset S of $\Lambda = \{0, 1, 00, 10, 01, 11\}$, chosen according to an algorithm based on the typical behavior of the transitions. But no sequence in this subset S is a suffix of another sequence in S, for example 1 and 01 both cannot be in set S. A suffix s' of a string s is defined as $s' \in \{s_i s_{i-1} ... s_l | 1 \leq i \leq l\} \cup \{e\}$ where $s = \{s_1 s_2 ... s_l\}$ [40]. The other requirement about the state space S is that for every string $s \in S$, if the probability of having $\sigma \in \{1, 0\}$ after s is greater than zero, there should be a string $\hat{s} \in S$ which is a suffix of $s\sigma$, where $s\sigma$ is the concatenation of σ to the end of the

string s [40]. The set S is known as a *Suffix Set* (a suffix free set). As one can see in this scheme different states may contain different memory lengths upper bounded by D. An example of a variable memory Markov model is given in Figure 3.0.2.

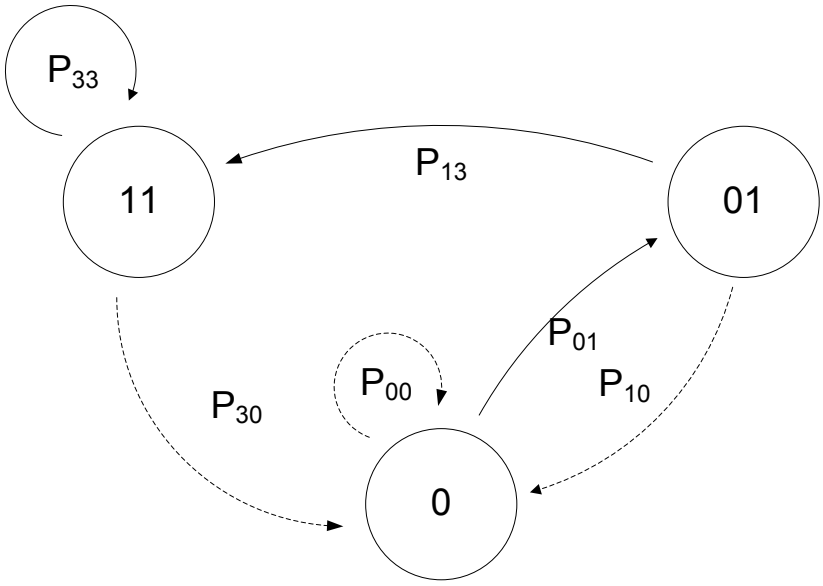


Figure 3.0.2: An example of a Variable order Markov Model of order two

The distribution generated by a variable order Markov model can indeed be generated by a fixed order Markov model of the same order. The only difference is that the variable order Markov model is able to do it in a much more succinct manner, which is computationally more efficient and consumes less memory [40, 42]. When creating a fixed order Markov model we only have to worry about the conditional probabilities i.e. the probability of occurrence of either a $'1'$ or a $'0'$ after a string labeling a particular state. Since the state labels are already fixed when the order is decided. In variable order Markov models, we have to find out the binary strings of $'1's$ and $'0's$ of length less than or equal to D, the exclusion of which, will affect the distribution and use them as states and the conditional probabilities also have to be found out.

Given a variable order Markov model or the fixed order model, if we want to find out the probability of occurrence of either 1 or 0 after a string of past observations s, first we try to find out the state having a label which matches with any suffix $\tilde{s}$ of s. Then we find

the probability distribution of the next symbol by the transition probability of that state, given that most recent channel observations are on the right hand side of the string.

As mentioned above to find out the states that matter in the variable order Markov model we use an algorithm called the Probabilistic suffix tree algorithm, which was introduced in [40]. This algorithm builds a tree, which we can use to build a variable order Markov model. To use this algorithm we need a training sequence of sufficient length which represents channel behavior under normal circumstances. The probabilistic suffix tree can be calculated using either a single string of channel states or multiple channel state strings of length greater than $D+1$. When we run this algorithm on the training sequence it identifies a set of strings S of 1s and 0s which are abundant in the training sequence and they should satisfy some conditions given in Algorithm 1. The algorithm for the discovery of a *Suffix Set* S given a training sequence σ_1^m of length m is explained in the next subsection.

3.1 Probabilistic suffix tree learning algorithm

This algorithm is based on [40, 41]. If we take the probability distribution of the channel usage pattern of a particular channel to be P, we can denote the *empirical probability distribution function* $\tilde{P}$, based on a sample string generated by P. For example P can be a Bernoulli distribution with parameter $\frac{1}{2}$ and it generates a sequence 001011001. Looking at this sequence assuming it is Bernoulli we can empirically calculate its parameter to be $\frac{4}{9}$. As one can see the empirical distribution can be somewhat deviated from the original distribution if the sample sequence is not long enough. In information theory the distance between two distributions is measured with a distance measure called *Kullback-Leibler* divergence and it can be represented as in Equation (3.1.1) where P_M^N and P_T^N are two steady state distributions generated by two Markov chains having N states [40,43]. Σ represents a finite alphabet which is $\{1, 0\}$ in our case and Σ^N represents all the possible permutations of alphabet Σ of length exactly equal to N. $P_M^N(r)$ and $P_T^N(r)$ are the probabilities of the two chains being in at an arbitrary state r in the respective Markov chains.

$$D_{KL}[P_M^N||P_T^N] = \sum_{r\in\Sigma^N} P_M^N(r) log\left(\frac{P_M^N(r)}{P_T^N(r)}\right) \tag{3.1.1}$$

The main purpose of the algorithm is to find out a hypothesis which resembles the distribution which generated the sequence, where the Kullback-Leiber distance between the actual and the empirical distribution is less than ϵ per state with probability $1-\delta$ where $0 < \epsilon < 1$ and $0 < \delta < 1$. For our algorithm to generate a good hypothesis a training sequence of sufficient length which represents the general channel usage is needed. But what length is sufficient is a difficult question to answer. In [40] the authors introduce two kinds of training methods. In one method the algorithm is fed with a number of independent sequences each of which are at least of length $D+1$ and in the second, the algorithm is fed with one long sequence of the change in channel states with time. Assuming the distribution is generated with a variable order Markov chain having n states and the order is D, they derived formulas for the number of sequences for the first training method and the length of the string of channel states for the second method. But to find the length of minimum sequence length required for one string of channel state data, according to the method introduced by Ron et al. we need the stochastic transition matrix of the channel state distribution process, which is the same thing we are trying to find out. In the training with multiple independent samples there is no guarantee that the sample strings are independent. What we can only do is taking sets of data so there is sufficient time gap in taking one set of data and the other set or we can get the data from several origins which are known to have less correlation. Therefore there is no analytic way of finding out the minimum length required for the sequence to be a representation of the distribution in general.

3.1.1 Calculation of probabilities with one sequence of training data

Assuming the sequence of channel state data represent the stochastic distribution, we can express the probability of occurrence of a string s in the training sequence, given by $\tilde{P}(s)$ to be roughly the number of occurrences of s in the training sequence divided by the maximum

number of occurrences possible of strings with length D, given the variable order Markov model is of order D. Equation (3.1.3) [40] gives the definition for $\tilde{P}(s)$ where the training sequence $\sigma_1^m = \sigma_1\sigma_2...\sigma_m$ is of length m, and $\sigma_i = \{1, 0\}$, $\forall i = \{1, 2, ..., m\}$. The indicator function $\chi_j(s) = 1$ if $\sigma_{j-|s|+1}...\sigma_j = s$ and $\chi_j(s) = 0$ otherwise. $|s|$ is the length of string s. As an example let $\sigma_1^{10} = 1110010000$ and $s = 00$. When $j = 2$, $\chi_2(00) = 0$ since $\sigma_1\sigma_2 = 11$ is not equal to $s = 00$. When $j = 5$, $\chi_5(00) = 1$, since $\sigma_4\sigma_5 = 00 = s$.

$$\chi_j(s) = \begin{cases} 1 & ; \sigma_{j-|s|+1}...\sigma_j = s \\ 0 & ; Otherwise \end{cases}, \tag{3.1.2}$$

$$\tilde{P}_{D,m}(s) = \frac{1}{m - D - 1} \sum_{j=|s|}^{m} \chi_j(s). \tag{3.1.3}$$

As one can see $\tilde{P}(s)$ is not a proper distribution but it gives the idea of relative abundance of a string s in the training sequence σ_1^m. If the training sequence represents the distribution the sequence was generated the empirical state probability $\tilde{P}(s)$ should be arbitrarily close to the stationary probability of the state s, $\pi(s)$ in the actual distribution with probability $1 - \delta$ where $0 < \delta < 1$ [40]. Now we can define the probability of occurrence of a symbol $\sigma = \{1, 0\}$, after a binary sequence s as given in Equation (3.1.4) [40]. All the symbols have the usual meaning. $s\sigma$ is the concatenation of σ to the end of s.

$$\tilde{P}(\sigma|s) = \frac{\sum_{j=|s|}^{m-1} \chi_{j+1}(s\sigma)}{\sum_{j=|s|}^{m} \chi_j(s)}. \tag{3.1.4}$$

3.1.2 Calculation of probabilities with multiple independent sequences of training data

Assuming we are given m' independent channel state sequences $\sigma^1, \sigma^2, ..., \sigma^{m'}$ of length $l \geq D + 1$ which represent the underlying distribution, where D is the order of the variable order Markov model. We can express the empirical probability of occurrence of a string s of channel states, of length at most D, as given in Equation (3.1.5) where $\chi_j^i(s)$ is defined

in (3.1.6) [40]. $\sigma^i = \sigma_1^i \sigma_2^i ... \sigma_l^i$ where $\sigma_j^i = \{1, 0\}$ for $1 \leq i \leq m'$ and $1 \leq j \leq l$.

$$\tilde{P}(s) = \frac{1}{m'(l-1)} \sum_{i=1}^{m'} \sum_{j=D}^{l-1} \chi_j^i(s), \tag{3.1.5}$$

$$\chi_j^i(s) = \begin{cases} 1 & ; \sigma_{j-|s|+1}^i ... \sigma_j^i = s \\ 0 & ; Otherwise. \end{cases} \tag{3.1.6}$$

The probability of occurrence of a symbol $\sigma = \{1, 0\}$ after a string s is defined in Equation (3.1.7) [40].

$$\tilde{P}(\sigma|s) = \frac{\sum_{i=1}^{m'} \sum_{j=D}^{l-1} \chi_{j+1}(s\sigma)}{\sum_{i=1}^{m'} \sum_{j=D}^{l-1} \chi_j(s)}. \tag{3.1.7}$$

In the algorithm explained in the next section, we will be using the probabilities $\tilde{P}(s)$ and $\tilde{P}(\sigma|s)$ without referring to which type of training sequence is provided to the algorithm since the algorithm only depends on the value of the probabilities, not on how it is calculated.

3.1.3 Algorithm to build the Probabilistic suffix tree

All Markov predictors depend on the property that, the probability distribution of the next symbol can be approximated by conditioning it on the previous D symbols, which is called by the name "*short memory principle*" [42]. If we choose the memory D to be too small than required, the distribution will be incapable of capturing all the dependencies between the symbols which degrades prediction efficiency [42]. On the other hand if we choose it to be too high it will over-fit the training sequence and give a high prediction error.

The probabilistic suffix tree algorithm has a favorable property of avoiding the over-fitting of the model to the training sequence if the maximum memory kept is too high. Therefore we can afford to keep D to be higher than required since the algorithm takes care of it [42]. But this advantage does not come free. For the algorithm to take care of this issue, more tunable parameters are introduced [42]. This algorithm is governed by

five parameters. These parameters control the upper bound of the number of states in the variable order Markov model. The first parameter is the maximum memory of the variable order Markov model, denoted by D. Value of D is the maximum number of ones and zeros in the strings in set S used to name the states of the Markov Chain or the maximum level to which the tree is grown. The second parameter is the minimum probability of abundance P_{min} in the training sequence, of any binary string s' of length $|s'|(\leq D)$, for s' to be a member of the tree. This is a necessary condition but it is not sufficient. The third parameter α denotes the minimum value of which the conditional probability $\tilde{P}(\sigma|s')$ can take for a binary string s' of length $|s'|(\leq D)$ and a symbol $\sigma \in \{1, 0\}$. This is also a necessary condition a string s' should meet in order for it to be a node on the tree. The fourth parameter r is a threshold which determines whether the string s contributes additional information in predicting the next symbol σ than its longest suffix $\hat{s}$ which is a node on the tree (if $s = \sigma_1\sigma_2...\sigma_{k-1}\sigma_k$ then $\hat{s} = \sigma_2...\sigma_{k-1}\sigma_k$). The final parameter γ is the probability assigned to $\tilde{P}(\sigma|s)$, if it is zero for any symbol σ and state s.

As one can see in the above algorithm we start with a tree with the single node representing the null string e and as the algorithm progresses we add nodes, the inclusion of which is mandatory for the distribution to be correct. We add a node v labeled by a string s to the tree $\bar{T}$ if the following criteria are satisfied by the string s. First the empirical probability of the occurrence of string s in the training sequence $\tilde{P}(s)$ should be larger than a threshold P_{min}. Because of this requirement we avoid the exponential growth in the number of strings to be tested in the algorithm. Then we check whether the probability of occurrence of a symbol $\sigma \in \{1, 0\}$ after the string s, $\tilde{P}(\sigma|s)$ is greater than a threshold α. Finally we check to see whether $\tilde{P}(\sigma|s)$ is greater than the probability of getting σ after the longest suffix $suffix(s)$ of s. We check this by taking the ratio $\frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|suffix(s))}$ and finding out whether it is greater than r where $r > 1$. We add the extended version of s which is σs where $\sigma = \{1, 0\}$ to the set of strings $\bar{S}$ to be tested if it should be inserted to the tree, if the probability of occurrence of σs in the training sequence is greater than P_{min}, immaterial of whether s is included in the tree or not. This is done because it is possible to have a string s'

Algorithm 1 PST Learning Algorithm
Phase 1

Initialize $\bar{T}$ and $\bar{S}$: $\bar{T}$ is a binary tree having a root node
e denoting empty string & $\bar{S}$ is an empty set

$\bar{S} \Leftarrow \{\sigma | \sigma \in \{1, 0\}$ and $\tilde{P}(\sigma) \geq P_{min}\}$

while $\bar{S} \neq \emptyset$, pick any $s \in \bar{S}$ and **do**

remove s from $\bar{S}$

if $\exists \sigma \in \{1, 0\}$ s.t $\tilde{P}(\sigma|s) \geq \alpha$ and $\frac{\tilde{P}(\sigma|s)}{\tilde{P}(\sigma|Suffixs)} > r$ **then**

add the node corresponding to s with all the nodes on the path starting from the deepest node in $\bar{T}$ which is a suffix of s to s, to $\bar{T}$

end if

if $|s| < D$ **then**

$\Sigma' = \{1, 0\}$
while $\Sigma' \neq \emptyset$ **do**

remove σ' from Σ'

if $\tilde{P}(\sigma' s) \geq P_{min}$ **then**

add $\sigma' s$ to $\bar{S}$
end if
end while
end if
Continue to next $s' \in \bar{S}$
end while

who has a conditional distribution substantially different from its parent. Parent of a node in our tree is labeled by the longest suffix of that particular node. A completely worked out example of how a string is used to build a model using this algorithm is given in Appendix A.

After performing Algorithm 1 on the training sequence there is a possibility that, for some of the nodes, the probability of occurrence of either 1 or 0 after the string s labeling that node ($\gamma_s(1)$ or $\gamma_s(0)$) is zero. Further more there is a possibility that the internal nodes of the tree $\bar{T}$ may only have one of the children with respect to the occurrence of either 1 or 0. We correct those problems in the phase 2 of the algorithm given in Algorithm 2 [40]. In the first case we assign a probability of γ in the nodes where the probability

of occurrence of either 1 or 0 after the string labeling the node is zero or less than γ i.e. if $\gamma_s(\sigma) \leq \gamma$ then make $\gamma_s(\sigma) = \gamma$ where $\sigma = \{1, 0\}$ this is shown in the upper right diagram of Figure 3.1.1. After continuing the procedure for every such node re-normalize $\{\gamma_s(0), \gamma_s(1)\}$ for all the nodes s on the tree $\bar{T}$, as shown in the bottom right diagram in Figure 3.1.1. In the second case we should enter the missing children of each internal node in the tree $\bar{T}$. Let the string corresponding to the internal node with only one of the children be s and let the child corresponding to symbol σ be missing. Then we enter that particular missing node $s' = \sigma\ s$ in $\bar{T}$ as a child of s. After entering the node we should calculate the probability of occurrence $\forall\sigma = \{0, 1\}$ after the binary string s' i.e. the distribution $\{\gamma_{s'}(0), \gamma_{s'}(1)\}$. These probabilities are calculated using Equation (3.1.8), where s is the longest suffix of s' in $\bar{T}$ [40]. Continue this procedure for every internal node with missing children. The insertion of missing nodes and calculating the probabilities is given in the bottom left diagram of Figure 3.1.1. The values in curly parenthesis $\{\gamma_s(0), \gamma_s(1)\}$ gives the probability of occurrence of a 0 or a 1 after the string labeling the node s respectively. The time complexity of probabilistic suffix tree learning algorithm is $O(Dm^2)$ and the space complexity is $O(Dm)$ where D is the order of variable order Markov model and m is the length of the training sequence [41]. The time complexity in calculating $\gamma_s(\sigma)$using the tree is $O(D)$ [41].

$$\gamma_{s'}(\sigma) = \tilde{P}(\sigma|Suffix(s'))(1 - 2\gamma) + \gamma \tag{3.1.8}$$

Algorithm 2 PST Learning Algorithm
Phase 2

Initialize $\hat{T}$ to be $\bar{T}$

Extend $\hat{T}$ by adding all missing sons of internal nodes.

For each node in the set $\hat{T} - \bar{T}$ labeled by string s' calculate the conditional probability $\gamma_{s'}(\sigma)$ for each $\sigma = \{1, 0\}$ using Equation (3.1.8).

if $\exists$ leaf nodes labeled with sting s which belong to both $\bar{T}$ and $\hat{T}$ s.t $\tilde{P}(\sigma|s) \leq \gamma$ for $\sigma \in \{1, 0\}$ **then**

Let $\tilde{P}(\sigma|s) = \gamma_s(\sigma) = \gamma$ for that particular s and σ

end if

Re-normalize the conditional distribution $\gamma_s(0), \gamma_s(1)$ for each leaf node labeled by s.

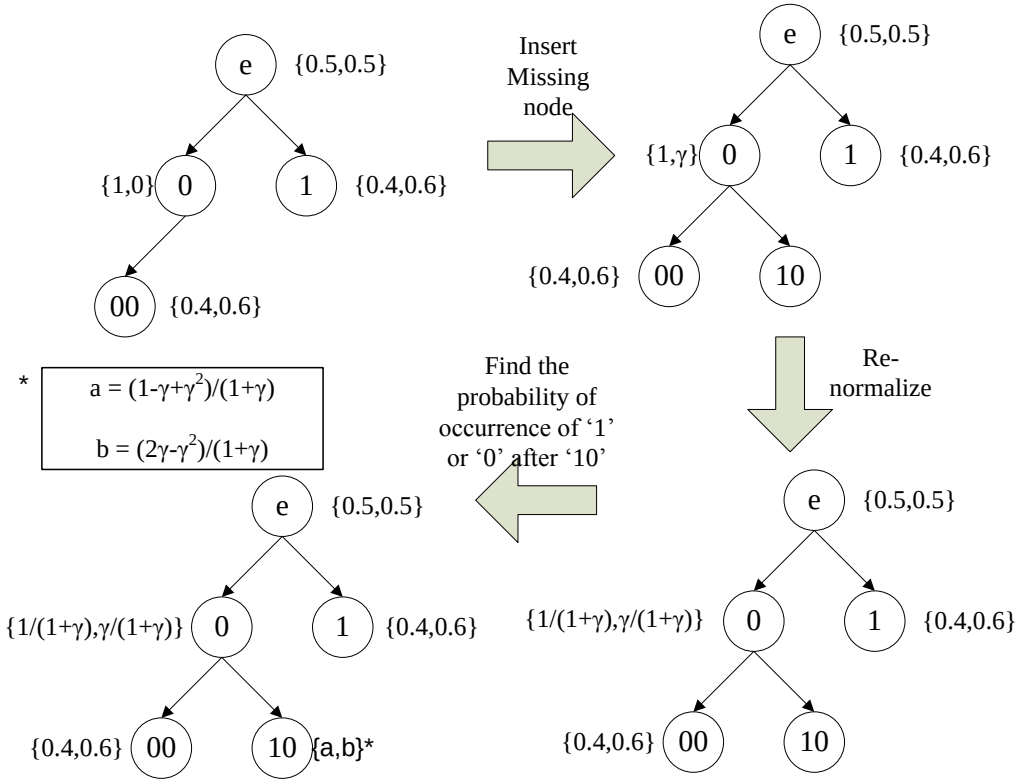


Figure 3.1.1: Inserting missing children

Although we are talking about a Markov chain, in the algorithm we are creating a tree $\hat{T}$. In this tree if we take the leaf nodes to be the respective state labels of the equivalent Markov chain, we might not be able to define the transition function for all the states s and

all symbols σ. This happens in the case where for a string s labeling a state and a symbol $\sigma = \{1, 0\}$ the state space does not contain a suffix of the string $s\sigma$. In this case we have to extend the tree $\hat{T}$ to a larger tree T' such that the transition function is well defined on the leafs of the extended tree T' [40]. This statement can be expressed in the implementation form as extending the tree until there exists a leaf node or an internal node having a label equal to the longest prefix of s, for every leaf node s in the tree T' [40], where the longest prefix of $s = \sigma_1\sigma_2...\sigma_k$ is $prefix(s) = \sigma_1\sigma_2...\sigma_{k-1}$. In [40] the authors prove that, there exists a Markov chain which is an equivalent predictor to the tree we derive, in terms of the next symbol probability distribution function. Since we are only interested in the probability of the next symbol being 1 or 0 given the past D symbols both representations give the same result. Phase 1 of the learning algorithm was given in Algorithm 1. When performing this algorithm the conditional probability distribution $\tilde{P}(\sigma|s)$ of the next symbol σ given the binary string labeling a given node s is saved within each of the node s in $\bar{T}$ as calculated in the algorithm. For the extended tree T' the probability of the next symbol σ for a given node having label s is the same as tree $\hat{T}$ for common nodes. For the added nodes in T' which are not present in $\hat{T}$, the probability of getting symbol σ after a string s', for each new node having label s' in T', is equal to the probability of having σ after string s in the tree $\hat{T}$ where s is the longest suffix of s' in the tree $\hat{T}$. This conditional probability mapping from the original tree $\hat{T}$ to the extended tree T' is given in Equation (3.1.9). Where $\gamma'_s(\sigma)$ is the probability of occurrence of a symbol σ after a string s labeling a node in tree T' and $\gamma_s(\sigma)$ is the probability of occurrence of a symbol σ after a string s labeling a node in tree $\hat{T}$.

$$\gamma'_s(\sigma) = \begin{cases} \gamma_s(\sigma) & ; \text{ if } s \in \hat{T} \cap T' \\ \gamma'_s(\sigma) & ; \text{ Where } s' \text{ is the longest suffix of } s \text{ in } \hat{T} \end{cases} \tag{3.1.9}$$

After the tree T' is complete we can create the corresponding Markov chain for that tree or we can do the prediction using the tree. An example of populating the tree using this algorithm is given in Appendix A. The resulting variable order Markov model is given

in Figure A.3.1. In deriving the resulting tree we assume the following values for the parameters: $D = 3, P_{min} = 0.006, \alpha = 0, r = 1.05$ and $\gamma = 0.0006$. Our training sequence is the same we assumed in Subsection 3.1, $\sigma_1^{10} = 1110010000$.

Chapter 4

Predictive Channel access using PST Algorithm

In this chapter we discuss the Predictive channel access scheme in cognitive radio networks using the Variable order Markov models, which is the main focus of the thesis. The assumed network topology for the cognitive radio network used in this thesis is discussed in Section 4.1. Then the process of creating a string of channel state data for multiple channels, which can be used as the training sequence to build the variable order Markov, is discussed in Section 4.2. In Section 4.3 we will present on how to obtain the probability of next slot being *idle* given the history of channel status, using the probabilistic suffix tree and the corresponding variable order Markov model. In Section 4.4, we will discuss how to intelligently order the channels so as to find an idle channel, at the least possible delay. Finally in Section 4.5 we explain how to intelligently switch channels, before the Primary user appears in the channel.

4.1 System overview

In the scheme proposed in this thesis we assume both the *Primary Network* and the *Secondary Network* to be centralized. The primary users posses the exclusive right to the

spectrum. The secondary users utilize the spectrum in an opportunistic fashion. As one can see this is a typical set up of a centralized cognitive radio overlay network. In the *secondary network* the *Secondary Base Station* takes decisions on the channels each secondary user should use, and channels each secondary user should switch to at the event of primary user arrival. Similarly the *Primary Base Station* controls the access of primary users. Thus we can make the safe assumption that both the primary base station and the secondary base station have perfect knowledge of the channels used by primary users and secondary users, respectively. We further assume that, primary base station having the information about the past channel usage of primary users has a record of general channel usage information, and this usage information is available for the secondary base station. It should be noted that we do not assume this data is current but we assume the data is statistically equivalent to the behavior of primary user channels in general.

The secondary users are assumed to be equipped with a widely tunable antenna, a sensing unit and a radio unit which is capable of sensing all the N primary channels and transmitting on the same. Further more these units are capable of operating on *Ultra Wide Band* mode for short periods of time for the exchange of control information. A similar assumption is taken in [18]. We adopt a listen before talk policy for the secondary users. The secondary users sense the channel they transmit on periodically and then send the result to the secondary base station, just after sensing, using an ultra wide band control channel. In our setup every secondary user who is transmitting senses the channel it transmits on and the secondary users not transmitting may be required to sense a particular channel periodically by the secondary base station. In this way the secondary base station is aware of the availability of a subset of channels, of the N channels. The secondary base station temporarily assigns these channels to the secondary users who needs to communicate. A model diagram of this setup is shown in Figure 4.1.1.

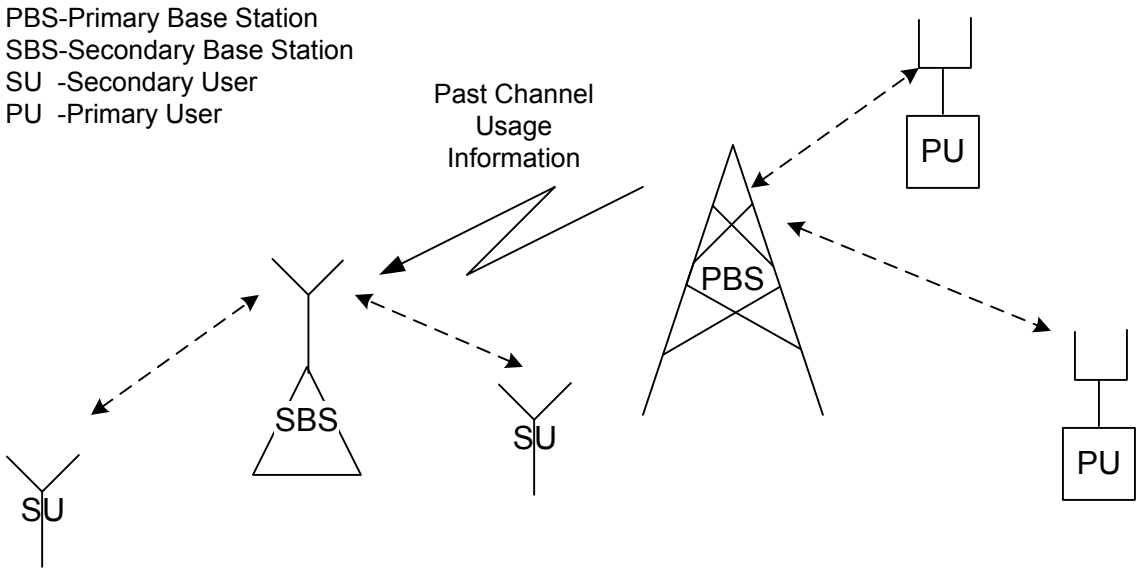


Figure 4.1.1: Diagram of the Network Model

4.2 Preprocessing of input from primary base station

As explained in Section 4.1 the primary base station makes the channel status information of the N channels available to the secondary base station. We assume that this information is in a form from which we can derive a time series of *busy* and *idle* data values. The time interval between two consecutive data points in the derived time series is equal to the time period between two sensing time slots T_{sense}. We assume that the secondary users perform channel sensing periodically. Therefore the time interval between two consecutive channel sensing operations is constant and can be denoted by T_{sense}. It consists of two parts. The first t_{sense} time interval is spent on channel sensing and transmission of sensing outcome to the secondary base station. The sensing time required depends on the modulation scheme used by the primary users, the sensing method adopted and the tolerable missed detection and false alarm probabilities [29]. The latter time interval $t_{transmit}$ is used to transmit data, given the channel is sensed to be *idle*. A diagrammatic representation of the sensing time is given in Figure 4.2.1.

The optimum time period between two consecutive sensing slots T_{sense} for a particular channel depends on the level of activity of that channel. A sensing rate adaption algorithm

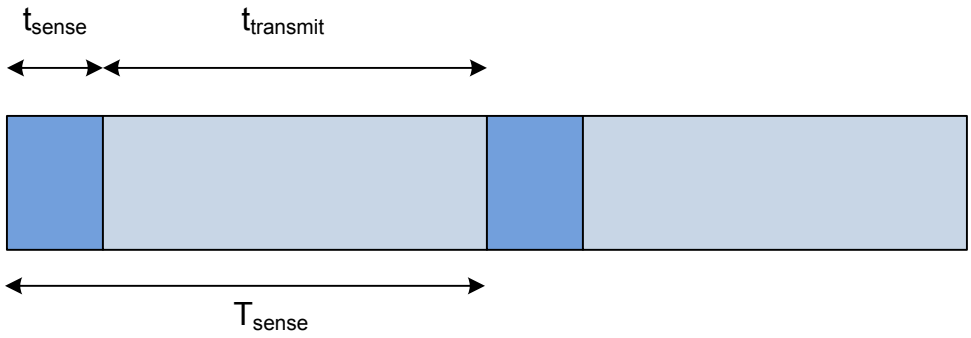


Figure 4.2.1: Periodic channel sensing operation

was derived for a channel model with exponentially distributed *busy* period followed by an exponentially distributed *idle* period in [29]. In that rate adaptation algorithm, they try to find the optimal sensing rate such that, a balance is struck between the probability of discovery of an *idle* period of a channel and the data transmission rate. Frequent channel sensing implies that the data transmission time will be less. The argument for doing sensing rate adaptation is, if the sensing is performed less frequently the secondary user can miss the entire *idle* period of the channel and the primary user will be exposed to higher interference. Further more it is a fact that, the secondary user can only utilize the portion of the *idle* period after discovery. In this paper, we adopt the convention of denoting the *busy* period with 1 and the *idle* period with 0. An illustration of this is given in Figure 4.2.2. Our ultimate goal is to build a model using this data and do predictions on the future channel status. The method we use to build the model was discussed in Chapter 3. In the next subsection, the method of prediction we use is discussed.

4.3 Prediction

We discussed the process of creating the probabilistic suffix tree and the variable order Markov chain in Section 3. The training sequence for the probabilistic suffix tree algorithm would be the preprocessed binary sequence, which was created by the preprocessing of the information sent by the primary base station. First we create the probabilistic suffix tree for

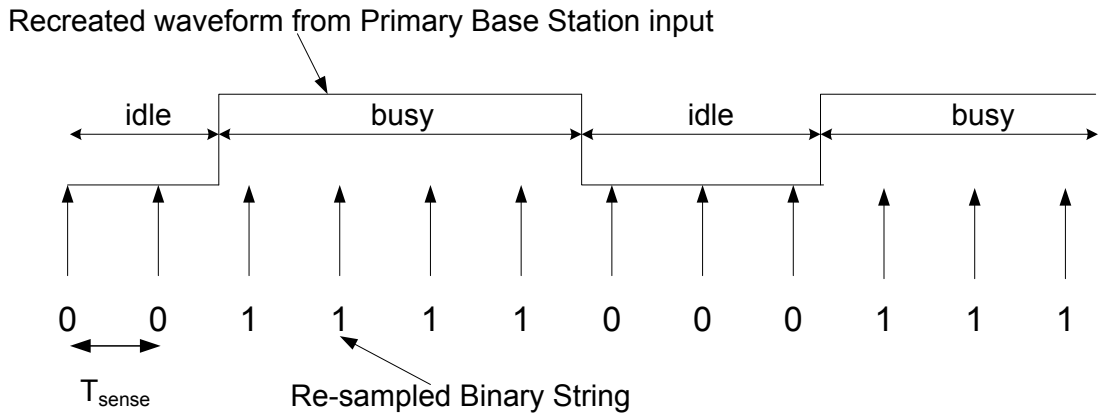


Figure 4.2.2: Producing the Binary Sequence for Model formulation

each channel where we have information on the past channel status. Then if the Markov chain is preferred over the tree we can create the corresponding Markov model. Having the channel model created as a tree or as a Markov Chain, prediction of the next symbol can be done using either the tree or the Markov chain. First let us discuss the prediction procedure using the tree. The prediction procedure using the Markov chain is explained after explaining the tree model.

4.3.1 Prediction using the probabilistic suffix tree

For a given probabilistic suffix tree $\bar{T}$ and the channel history for the last D slots we traverse the tree starting from the node e which denotes the empty string. This traversing is done according to the past channel states starting with the most recent channel observation and going backwards in the string of the past D channel states. Traversing the tree, we will reach a leaf node either exhausting the string of channel history or not exhausting the channel history. When this leaf node is reached, we use the probability distribution of the occurrence of an *idle* slot or a *busy* slot for that particular node in predicting the next channel status. The decision of whether it is an *idle* slot or a *busy* slot is taken based on a threshold. Let us take the final example tree calculated in the Appendix A which is shown in Figure 4.3.1 and find the probability of occurrence of an *idle* slot given the past states of

the channel 001. In traversing the tree first we take the most recent sample of the history string, which is 1 and goto the right child of e since it is 1 and if it was 0 we do otherwise. Then take the next most recent sample, which is 0 and then go to the left child of the current node. We reach the node 01 which is a leaf node before exhausting the channel history. Based on the probability distribution of node 01 and a threshold we can decide whether it will be an *idle* slot or a *busy* slot. As one can see the probabilities of occurrence of 0 and 1 after the sequence 01 are 0.9994 and 0.0006 respectively. According to the construction of the tree we conclude that the left most 0 of the history string 001 does not give us more information about the next slot and we conclude that, the probabilities of occurrence of 0 and 1 after the sequence 001 is equal to that of 01. If the threshold is 0.5 we can predict the channel to be *idle* in the next time slot.

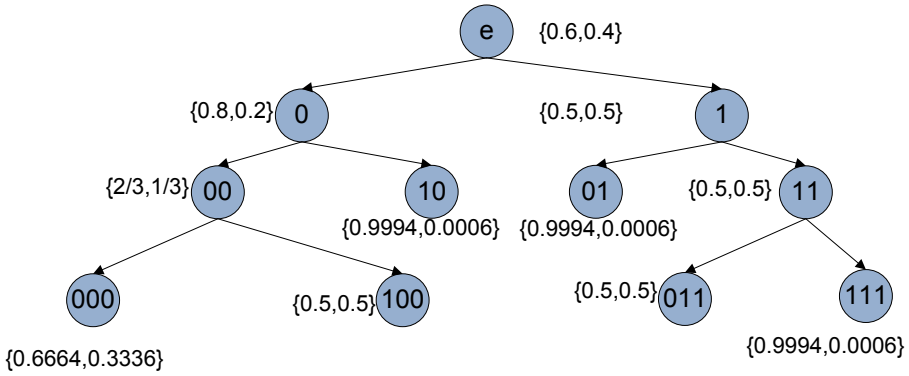


Figure 4.3.1: Probabilistic Suffix Tree

4.3.2 Prediction using the Markov Chain

In prediction using the Markov chain, first we derive the corresponding Markov Chain of the tree $\bar{T}$ using the labels of the leaf nodes of the tree, as explained in Section 3.1.3. Given the string of past D observations of the channel and the Markov Chain, we search for the state label that matches with the longest suffix of the channel history sequence. The longest suffix of the channel history sequence have the most recent observations of the channel for the last $k(k \leq D)$ time slots. Assume the state that matches with the longest suffix is $\hat{s}$. Using the transition probabilities of the state $\hat{s}$ we can find the probability of occurrence of

an *idle* slot or a *busy* slot. Using a threshold we can predict the channel status of the next slot. Using the example Markov chain derived in Appendix A let us find the probability of channel being *idle* given the channel states of the past three time slots, 001. The Markov chain is given in Figure 4.3.2. First search for a state having label 001 in Figure 4.3.2. Since no matching state is found, search for a state having label equal to 01, which is the longest suffix of 001. There exists a state having label 01. Then the probabilities of occurrence of 0 and 1 would be the same as in Subsection 4.3.1.

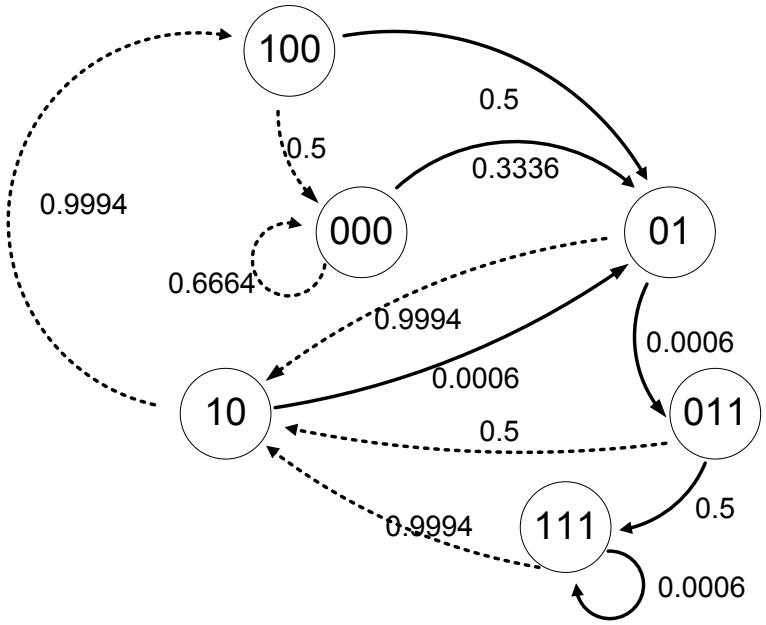


Figure 4.3.2: Variable order Markov model

As mentioned earlier the channel state prediction would be based on a threshold. We can set the threshold of the next slot being *idle* to be close to 0.5 or close to 1.0. When it's close to 1.0 more opportunities will be wasted and on the other hand, if it is close to 0.5 we will cause more disruption to the primary users. So we should decide on the threshold based on the prediction error and the interference tolerance level of the primary user. We can diagrammatically represent the process of predicting as in Figure 4.3.3.

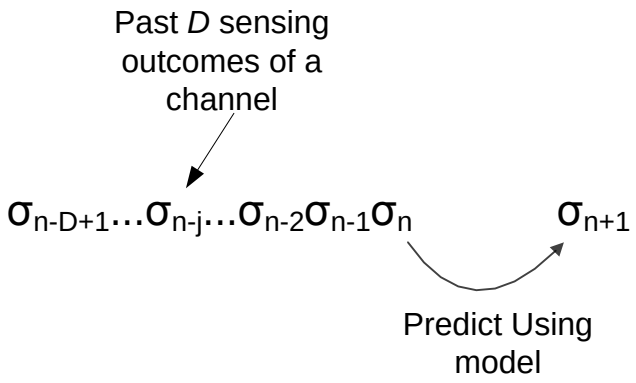


Figure 4.3.3: Prediction

4.4 Proactive Spectrum access

In reactive spectrum access, SUs search for idle channels, only when they detect a primary user on the channel they were using. So they randomly select a channel and listen to it, to identify the state of the channel. If the channel is *idle* the secondary user uses it and if not the search continues till it finds one. In searching, secondary users do not take into account the expected available idle time on the particular channel, because of this the secondary users can be interrupted very frequently. On the other hand the primary users can experience high interference. In order to reduce the interruptions to the Primary and secondary users we have to order the channels in the ascending order of activity so that the secondary users go for the channel with highest expected availability first, saving it from interfering with the primary user and high variability in the idle channel search delay. Two such ordering schemes are given below.

4.4.1 Proactive Channel Ordering Scheme 1

In Proactive Channel Ordering, secondary base station first calculates the probability of channel i being idle in time slot $n+1$, using our channel usage model, given the history of that particular channel for the past D time slots including slot n. If the probability is greater than a threshold $Idle^1_{min}$ we consider it as an idle channel in the next time slot $n+1$. Then

secondary base station finds the probability that the channel is idle in the time slot $n+2$, given it was predicted to be idle in slot $n+1$. If the probability is greater than $Idle_{min}^{(n+2)-n}$ we consider it as an idle slot. SBS continues this for a window of l slots till a slot is predicted not to be idle. Note that all of these predictions are carried out at time slot n. The ordering of channels is done in the descending order of the number of consecutive idle slots in the window of l slots. If two or more channels have the same number of expected idle slots τ, the secondary base station orders them based on the value of the product of the probabilities β of a particular channel being idle in the $\tau \leq l$ slots, see Equation (4.4.1).

$$\beta = Idle_i^{n+1} Idle_i^{n+2} ... Idle_i^{n+\tau}, \tag{4.4.1}$$

where $Idle_i^m$ is the probability of channel i being idle in the time slot m.

4.4.2 Proactive Channel Ordering Scheme 2

In this scheme we order the channels in the descending order of the product of β and τ which is a measure of expected number of idle slots δ. δ is defined in Equation (4.4.2). As one might think this scheme can order a channel with more idle slots below a channel with lower number of idle slots in the case where the channel with low number of idle slots have a higher β value. For example let us take a scenario having two channels. The channel-1 has one idle time slot with probability 0.9 and the channel-2 has two idle time slots, where the probability of the slot one being idle is 0.6 and the slot two being idle is 0.5. In this case this scheme would prefer channel-1 over channel-2. This selection might look bad but if one wants to minimize the interference caused to the primary user this might be the attractive alternative since the probability of the prediction being wrong is less in this scheme.

$$\delta = \tau \cdot \beta \tag{4.4.2}$$

4.5 Proactive Channel Switching

In the concept of proactive channel switching, the secondary user tries to get an idea of the channel usage pattern of the primary user and based on that idea, it switches the channel in use even before a primary user appears on it. First the Secondary base station finds out the probability that, each channel will be idle in the next time slot based on the channel usage model. Then if finds out the expected number of time slots each channel will be idle contiguously in the window of next l time slots. Based on these measures it determines the free channel to switch to. The switching scheme is explained in the next subsection.

4.5.1 Proactive Channel Switching Scheme

In Proactive Channel Switching the secondary users switch the channel in use before the primary user actually arrives in that channel if it predicts one of two scenarios. The first scenario is, the probability of the currently used channel being occupied by the primary user in the next time slot being high. It can be represented using the probability of the current channel being idle in the next time slot, given the history of it for the past D slots, and a threshold as $P(0|s_c) < Idle^1_{min}$. The second is, the existence of a channel j, which has a higher expected idle time T_j^{idle} than the current channels' idle time T_c^{idle} plus the switching delay θ. So if the models' predictions are accurate we can reduce the interference experienced by the primary user and increase the QoS experienced by the secondary user. But this should be done with utmost care otherwise this scheme can actually increase the level of interference. Channel selection in time slot $n+1$, $Channel^{n+1}$ can be represented by Equation (4.5.3) where s_c is the channel history of the current channel for the past D slots. Channel j's expected idle time, T_j^{idle} can be calculated using Channel Ordering Scheme 1 or Scheme 2. In Scheme 1, τ represents the number of expected contiguous idle slots. Then you can assume $T_j^{idle} = \tau_j$, where τ_j is the number of idle slots expected in channel j. In Scheme 2, δ gives a measure which can be approximated to T_j^{idle} in the unit of slots. The value Ch^{max} defined in Equation (4.5.1) represents the index of the channel

which has the highest expected *idle* time depending on whether the current channel is going to be *idle* or *busy* in the next time slot.

$$Ch^{\max} = \begin{cases} argmax_j T_j^{idle}, j \in \{1, ..., N\} \setminus \{c\} & ; P(0|s_c) < Idle_{min}^1 \\ argmax_j T_j^{idle} - \theta_j, j \in \{1, ..., N\} & ; P(0|s_c) \geq Idle_{min}^1 \end{cases}, \tag{4.5.1}$$

where θ_j is expressed by Equation (4.5.2) and c is the current channel number.

$$\theta_j = \begin{cases} \theta & ; j \neq c \\ 0 & ; j = c \end{cases} \tag{4.5.2}$$

$$Channel^{n+1} = \begin{cases} Ch^{max} & ; T_{Ch^{max}}^{idle} \geq \varepsilon \\ \text{Stop transmission} & ; T_{Ch^{max}}^{idle} < \varepsilon \end{cases}, \tag{4.5.3}$$

where ε is a threshold.

Chapter 5

Simulation

In order to create the channel model according to the probabilistic suffix tree scheme we need a training sequence. Because of the unavailability of real data we had to create a set of training data as strings of '1's and '0's. For this we used a renewal process with independent Erlang-5 distributed *busy* time and an *idle* time. The process of creating the training sequence is given in the next subsection.

5.1 Creating the training sequence using the Erlang-5 distribution

As mentioned above in the simulation conducted we assumed a channel with Erlang-5 distributed *busy* and *idle* periods with the rate parameters 2.5 and 1.5 respectively. Rate parameters where chosen such that the primary channel utilization is some where in between $0\% - 50\%$ We simulated the channel switching of one secondary user using 3 channels. Three channels were chosen since we were only considering the channel access of one secondary user. The secondary user was assumed to carry out periodic sensing with periods of 0.2 units. Erlang-5 distribution is a continuous time distribution so we have to re-sample it to generate the binary time series. A sum of n exponentially distributed independent random variables having parameter λ are distributed according to Erlang-n distribution

with parameter λ [44]. We can generate exponential random variables using the Inverse transform method. If the exponential variables are $X_1, X_2, X_3, ..., X_n$, we can present the Erlang-n distributed random variable Y as in Equation 2.2.1.

$$Y = X_1 + X_2 + X_3 + ... + X_n. \tag{5.1.1}$$

If $\forall i = \{1, 2, 3..., n\}, U_i$ are independently distributed according to the standard uniform distribution we can write the equation for the Erlang (n, λ) distributed random variable as in Equation 5.1.2.

$$Y = -\frac{1}{\lambda} \log \left(\prod_{i=1}^{n} U_i \right). \tag{5.1.2}$$

In our case we should generate Erlang-5 distributed *idle* and *busy* periods alternatively with parameters 1.5 and 2.5 respectively. We make use of the uniform random number generator in Matlab ® to generate the standard uniform random numbers. We initialize ten random streams to make the generated uniform random variables independent and make use of five of them to generate the exponentially distributed *idle* time and the other five to generate the exponentially distributed *busy* time according to Equation (5.1.2). After generating the alternating *idle* and *busy* time periods we re-sample the times into slots of 0.2 time units and generate a string of 1s and 0s representing *busy* and *idle* channel states. This string will serve as our training sequence. We generate 3 such strings representing the sampled channel states of 3 channels.

5.2 Creating the training sequence using the Markovian arrival process

In the distribution generation with the Erlang-5 distribution we considered the *idle* and the *busy* periods to be independently distributed from each other. But this might not be the case. Therefore we further test our algorithm with *idle* and *busy* periods which are

correlated. To generate a distribution having correlation between the different channel states we use the Markovian arrival process. Using the Markovian arrival process one can generate a wide variety of distributions changing the values in the governing matrix [45]. The governing matrix of a Markovian arrival process resembles a transition matrix of a discrete time Markov chain with state space $\Gamma = \{1, 2, ..., n_b, n_b + 1, ..., n_b + n_i\}$ and transition matrix D presented in Equation (5.2.1) taken from [45].

$$D = \begin{bmatrix} D_b & d_{bi} \\ d_{ib} & D_i \end{bmatrix} \tag{5.2.1}$$

In Equation (5.2.1) the matrices D_b and D_i are substochastic with order n_b and n_i respectively. Substochastic means that the sum of the entries of each row in matrices D_b and D_i is less than or equal to one. They represent the transition probabilities within the *busy* state and the *idle* state without changing the state. The matrices d_{bi} and d_{ib} represent the transition probabilities from a *busy* state to an *idle* state and vice versa respectively. The dimensions of matrix d_{bi} are $n_b \times n_i$ and that of matrix d_{ib} are $n_i \times n_b$. Since the matrix D is strictly stochastic the matrices D_b, D_i, d_{bi} and d_{ib} should satisfy the conditions mentioned in equations (5.2.2) and (5.2.3), where $\mathbf{1}$ is column vector of ones [45].

$$D_b\mathbf{1} + d_{bi}\mathbf{1} = \mathbf{1} \tag{5.2.2}$$

$$D_i\mathbf{1} + d_{ib}\mathbf{1} = \mathbf{1} \tag{5.2.3}$$

It is evident that, to generate a string of channel status observations according to a Markovian arrival process one has to create a matrix D which governs it. For the simulation purpose we developed the Markovian arrival process as given in [39, 45]. That particular Markovian process exhibits what is known as "Pseudo long range dependent self-similar characteristics" [39] . Self similarity means that the burstiness of traffic is the same over different time scales, i.e. if we measure the traffic going on a particular link and plot the

traffic as a graph of amount of packets per 1s, 10s, 100s, 1000s etc. the graph will only be scaled in amplitude, the shape will remain the same [39]. In a Brownian motion process $X(t)$, if the statistics of $X(t)$ and $\frac{X(rt)}{r^H}$ are indistinguishable the process is said to be self-similar with parameter H, where r is the scaling variable. The structure of the Markovian arrival process given in [39] has burstiness similar to Ethernet traffic [39]. This Markovian arrival process depends on mainly three parameters a, b and n. Parameter n governs the number of states in the discrete time Markov chain on which the Markovian arrival process is based on. Given n we calculate the parameter a such that the process generates a distribution which has a given average burst length $E[B]$, where $E[B]$ is the average number of time slots the channel will be *busy* consecutively. $E[B]$ is defined in Equation (5.2.4) [45]. We then calculate b so that the distribution generated has the given utilization factor of the primary channel ρ_{PU} using Equation (5.2.5). The utilization factor gives the percentage of time the channel is busy on average.

$$E[B] = \left(\sum_{k=1}^{n-1} a^{-k} \right)^{-1}, \tag{5.2.4}$$

$$\rho_{PU} = \frac{1 - \frac{1}{b}}{1 - \frac{1}{b^n}}. \tag{5.2.5}$$

The structure of matrix D which represents the Markovian arrival process is given in Equation (5.2.6). The lines drawn in the matrix divides matrix D into block matrices as in the Equation (5.2.1).

$$D = \left[\begin{array}{c|ccccc} 1 - \sum_{k=1}^{n-1} \frac{1}{a^k} & \frac{1}{a} & \frac{1}{a^2} & \cdots & \frac{1}{a^{n-1}} \\ \hline \frac{b}{a} & 1 - \frac{b}{a} & & & \\ \left(\frac{b}{a}\right)^2 & & 1 - \left(\frac{b}{a}\right)^2 & & \\ \vdots & & & \ddots & \\ \left(\frac{b}{a}\right)^2 & & & & 1 - \left(\frac{b}{a}\right)^2 \end{array}\right]. \tag{5.2.6}$$

We set the size of the matrix to be 6×6 and let the *idle* and *busy* states to have 1 phase

and 5 phases each. In accordance with the earlier explanation $n_b = 1, n_i = 5$.

To generate the *idle* and *busy* states according to this matrix first we have to find out the stationary distribution π which gives the information on the probabilities of choosing a state $\gamma = \{1, 2, 3, ..., 6\}$ at start. Let $\pi = \{\pi_1, \pi_2, ..., \pi_6\}$. Then the first state is chosen by generating a uniformly distributed random variable u and testing if it satisfies the Inequality (5.2.7) for some value i, where $\pi_0 = 0$. The value i for which the inequality is satisfied, is the state the system is in, at that time epoch.

$$\gamma_0 = \left\{ i \;\middle|\; \sum_{k=0}^{i-1} \pi_k < u \leq \sum_{k=1}^{i} \pi_k \right\}. \tag{5.2.7}$$

After finding γ_0 which is the initial state, if $1 \leq \gamma_0 \leq n_b$, we set the channel state to *busy* and if $n_b + 1 \leq \gamma_0 \leq n_b + n_i$, we set the channel state to *idle*. Thereafter the channel state at time t is generated using the same method but the distribution used is not π, instead the row of D corresponding to the state of the Markov chain at time epoch $t - 1$ is used as the distribution. The D matrix used for simulations is given in Equation 5.2.8. This Matrix was generated such that the sequence generated have *busy* slots 40% of the time i.e. $\rho_{PU} = 0.4$ and it has average bursts of length $E[B] = 10$.

$$D = \left[\begin{array}{c|cccccc} 0.9000 & 0.0909 & 0.0083 & 0.0008 & 0.0001 & 0.0000 \\ \hline 0.1458 & 0.8542 & 0 & 0 & 0 & 0 \\ 0.0213 & 0 & 0.9787 & 0 & 0 & 0 \\ 0.0031 & 0 & 0 & 0.9969 & 0 & 0 \\ 0.0005 & 0 & 0 & 0 & 0.9995 & 0 \\ 0.0001 & 0 & 0 & 0 & 0 & 0.9999 \end{array}\right] \tag{5.2.8}$$

5.3 Setting thresholds

In the scheme proposed there is no analytic way of finding the optimum thresholds which aids us in detecting idle time slots. Therefore one can only find the optimum values using trial and error. There are several performance measures we take into consideration when

determining these thresholds. The first one is the Dumb Switch I. Lower number of Dumb Switch I erroneous switches means that the scheme is predicting well. The second performance measure is the number of Dumb Switch II performed by the predictor. In this measure too lower number of Dumb Switch II erroneous switches means the predictions are good. But these measures alone will not give us much detail about the quality of prediction, since the cognitive radio has the option of not transmitting in a given slot. If we set the thresholds too high, it will rarely transmit, there by wasting many opportunities. Therefore we have the performance measure Unused slots, to make sure not much opportunities are wasted. So one should set the thresholds such that a balance is struck between the above mentioned measures. Although Dumb Switch II performance measure can be easily used in simulations, using it can be quite cumbersome in real time, since at a given time we do not know how long the given idle period is going to last. The performance measure which is linked directly to the level of interference to primary users is the Dumb Switch I. So the lower the measure then the lower will be the level of interference caused on the primary users. The performance measure Unused slots is linked to the level of spectrum utilization of the secondary users. Lower unused slots mean high spectrum utilization.

In Section 4.4.1 we used several thresholds to detect the number of continuous *idle* slots in a window of l slots given by $Idle^j_{min}$, where $j = \{1, 2, 3, ..., l\}$. The superscript j gives the distance from the current time slot to the time slot where predictions are made. For $j = 1$ the threshold should be predetermined. The other thresholds are calculated using the probabilities of the channel being idle from $n + 1^{th}$ slot up to the $n + j - 1^{th}$ slot for $j \geq 2$, given the history as given in Equation (5.3.1). n is the current slot for which we have sensing information and $Idle^p_i$ is the probability of channel i being idle in time slot p.

$$Idle^j_{min} = \frac{Idle^j_{min}}{\prod_{k=1}^{j-1} Idle_i^{n+k}} \quad j \geq 2 \tag{5.3.1}$$

5.4 Numerical results for Erlang-5 distributed channel states

Table 5.1: Simulation Parameters Erlang-5

Parameter	Value
D	10
P_{min}	0.006
α	0
r	1.05
γ	0.0006
l	5
$Idle^1_{min}$	[0.5:0.1:0.9]
ε Scheme 1	3
ε Scheme 2	2.5
θ	1

In the numerical results presented we use the performance measures Dumb Switch I, Dumb Switch II, Smart Switch and Unused slots explained in Section 2.4.

We are comparing our schemes denoted by PST Scheme 1 and PST Scheme 2 with the scheme presented in [3] which is denoted by California in the plotted graphs. We simulated for 30000 slots and replicated the simulation 20 times to do a mean and error plot. In plotting graphs for the scheme in [3] we used the Proactive Planning II scheme explained in Section 2.3 with the threshold $T = 0.45$. Note that the plots are shifted with respect to each other in places where they overlapped each other.

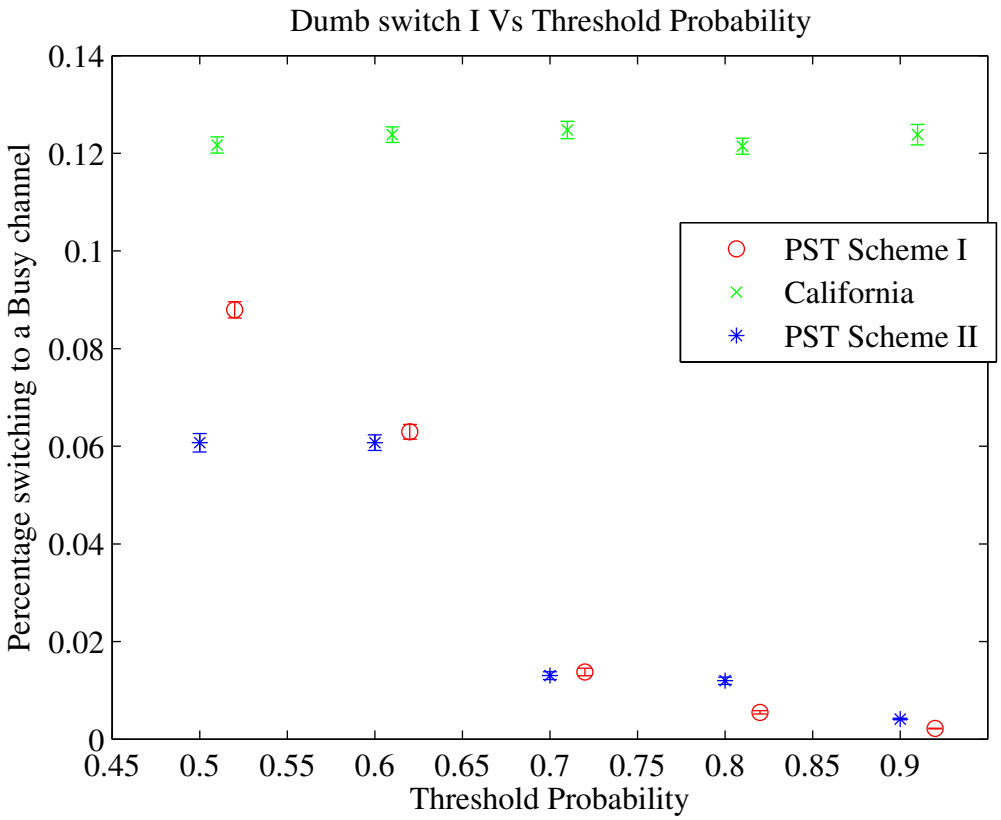


Figure 5.4.1: Dumb Switch I for Erlang-5 distributed channel states

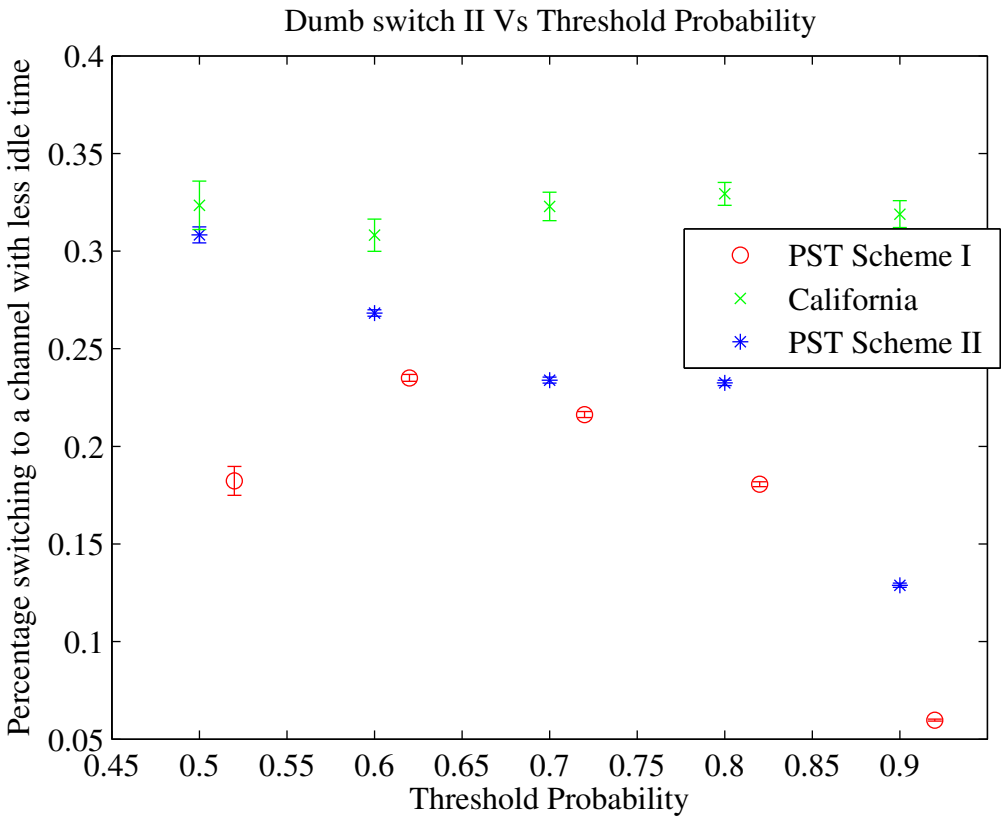


Figure 5.4.2: Dumb Switch II for Erlang-5 distributed channel states

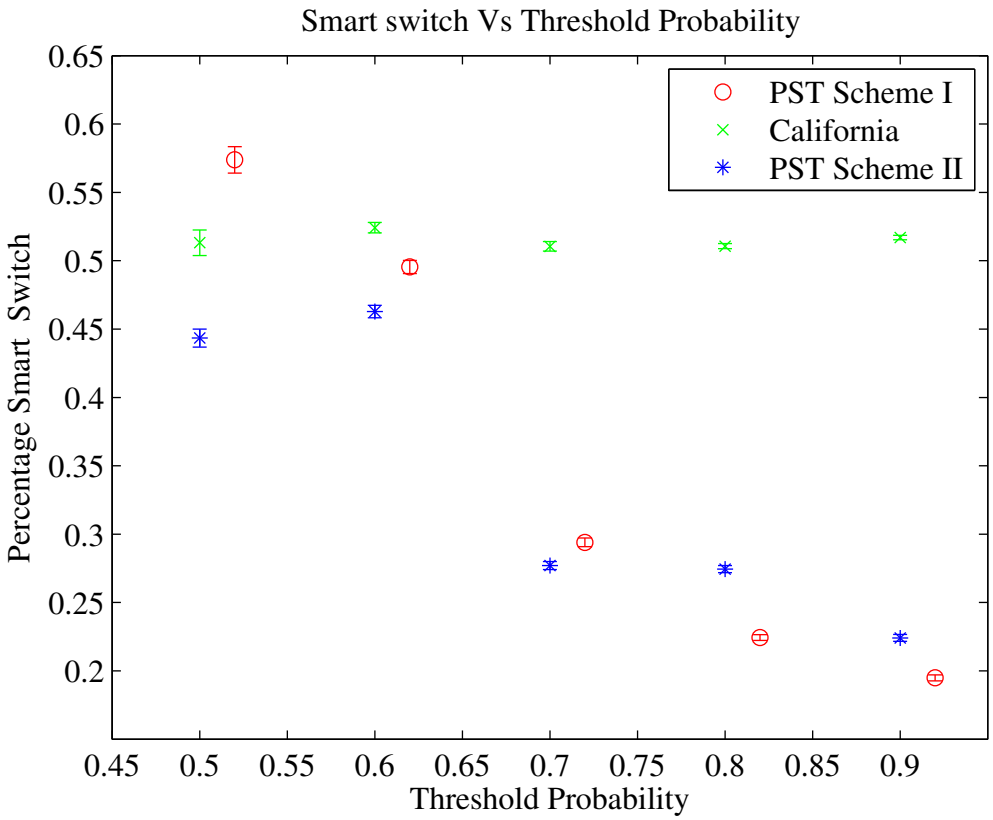


Figure 5.4.3: Smart switch for Erlang-5 distributed channel states

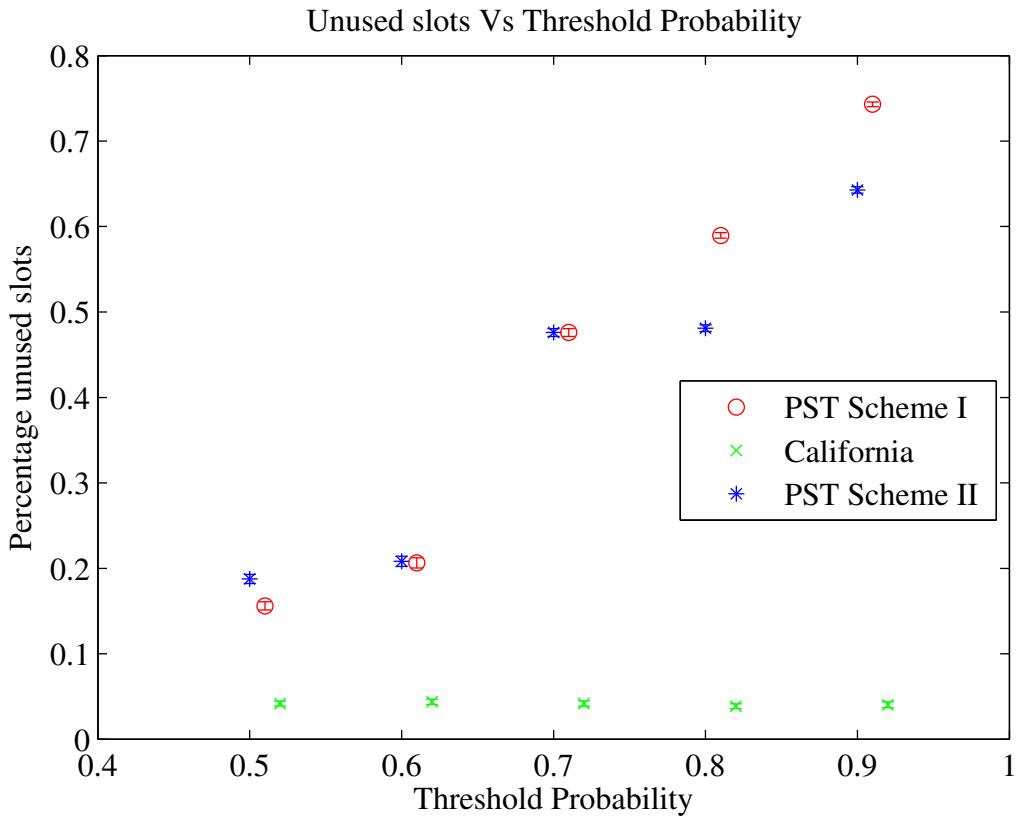


Figure 5.4.4: Unused Slots for Erlang-5 distributed channel states

5.5 Numerical results for channel states distributed according to the Markovian arrival process

Table 5.2: Simulation Parameters Markovian arrival process

Parameter	Value
D	10
P_{min}	0.006
α	0
r	1.05
γ	0.0006
l	5
$Idle^{i}_{min}\ \forall i = \{1, 2..., l\}$	[0.4:0.1:0.8]
ε Scheme 1	2
ε Scheme 2	1.75
θ	0.5
ρ_{PU}	0.4
$E[B]$	10
Slot duration	0.05

The graphs below show the performance measures for the proposed schemes. Since we used a discrete time channel state distribution model which was a discrete time Markovian arrival process, we could not directly compare it with the scheme presented in [3]. Therefore we assumed a slot duration and matched it to an exponential distribution having the same mean. As one can presume the performance of the scheme in [3] will depend on the time slot length we used. In this scheme the function to calculate the probability of occurrence of an *idle* time epoch given the channel state of last sensing instant is also a function of this slot duration. We tested this scheme for a series of slot time durations

such as 0.2, 0.1, and 0.05 and the performance was almost the same. Based on our simulations the scheme given in [3] has inferior performance. This could be due to the fact that the assumption of independently distributed *idle* and *busy* time slots does not hold in the Markovian arrival process. The other schemes in the literature which we explained in Section 2.1 were too complex to implement and in their work they do not present results of a similar analysis. In the multilayer perceptron technique given in [30], the best number of inputs, the number of hidden layers and the neurons in each layer are decided first, and then the network is trained adjusting the weights. The authors mentioned what these parameters should be for the case of geometrically distributed *idle* and *busy* times but there is no evidence that those are the best values for the case of *idle* and *busy* times distributed according to the Markovian arrival process. On top of that the learning rate parameter, the stopping condition and whether they used a momentum term to reduce the oscillations were not mentioned. In the case of hidden Markov models the model will depend on the initial model assumed since the Baum-Welch re-estimation converges to a local maximum always. In [31] the authors do not mention the number of states in the hidden Markov Model they used, the initial distributions for the occurrence of *busy* or *idle* time slots and the state transition probabilities they used. In [32] the calculation of model parameters were done based on genetic algorithms these algorithms too will converge to a local minimum with high probability. The order of the model they used was also not mentioned in the reference. Thus making it difficult to compare the performance of our scheme with theirs.

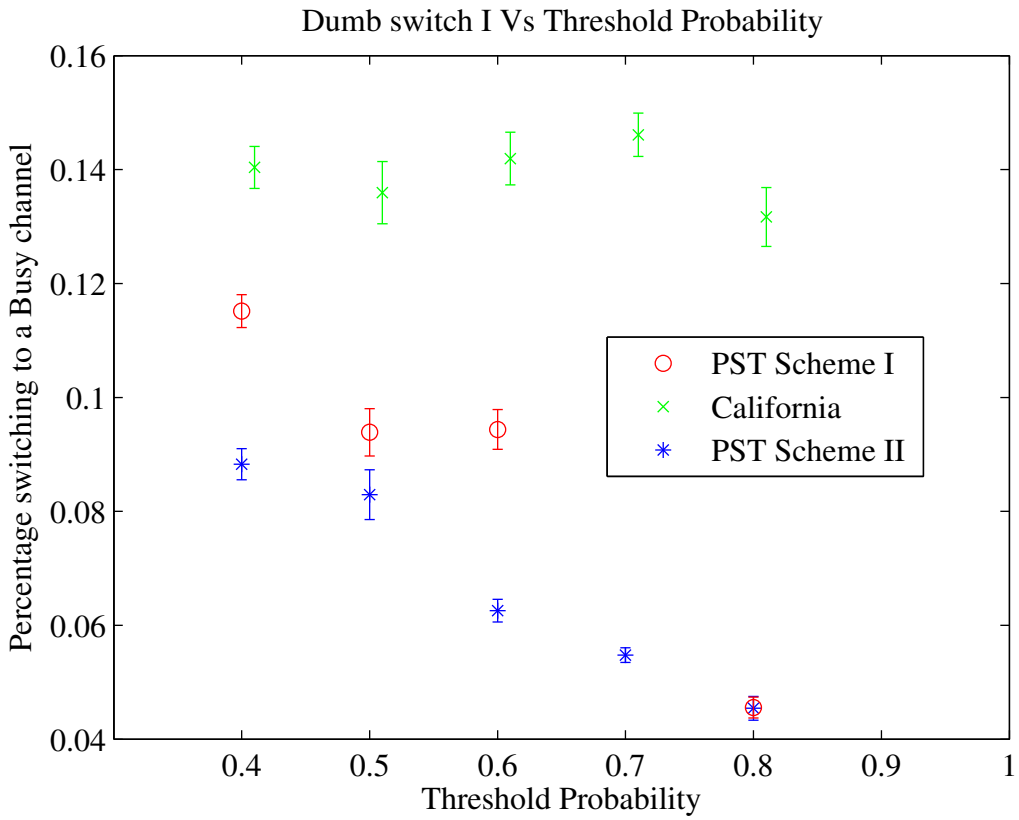


Figure 5.5.1: Dumb Switch I for channel states distributed according to the Markovian arrival process

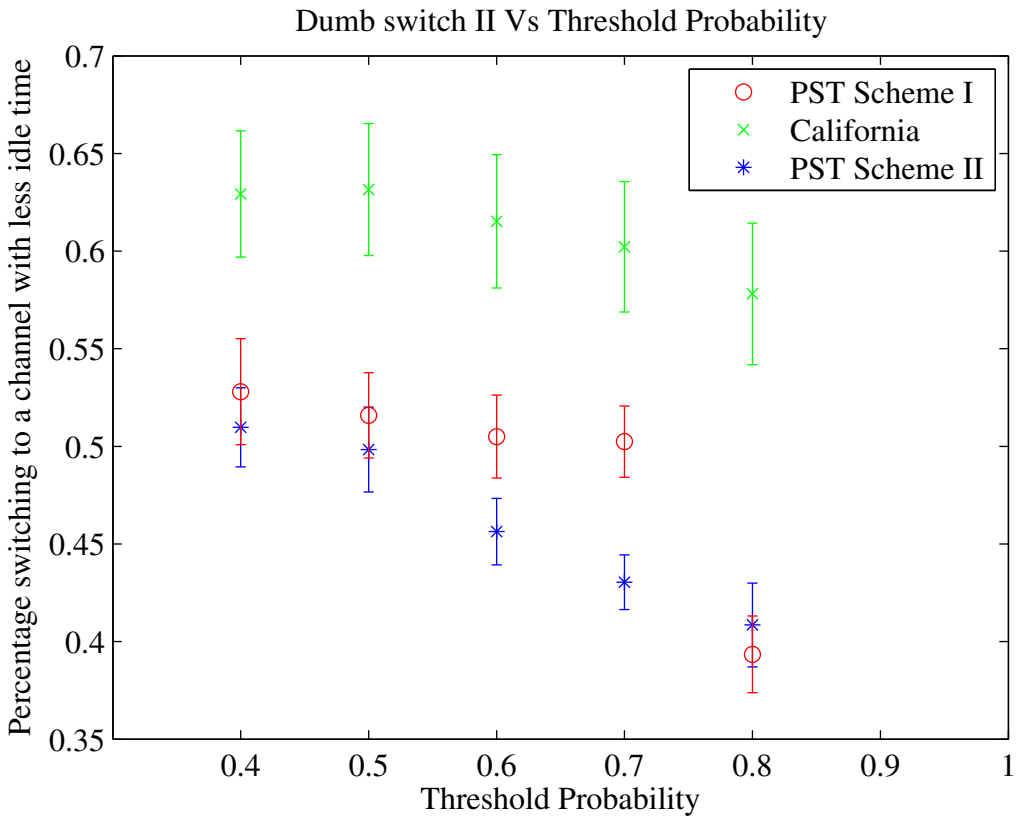


Figure 5.5.2: Dumb Switch II for channel states distributed according to the Markovian arrival process

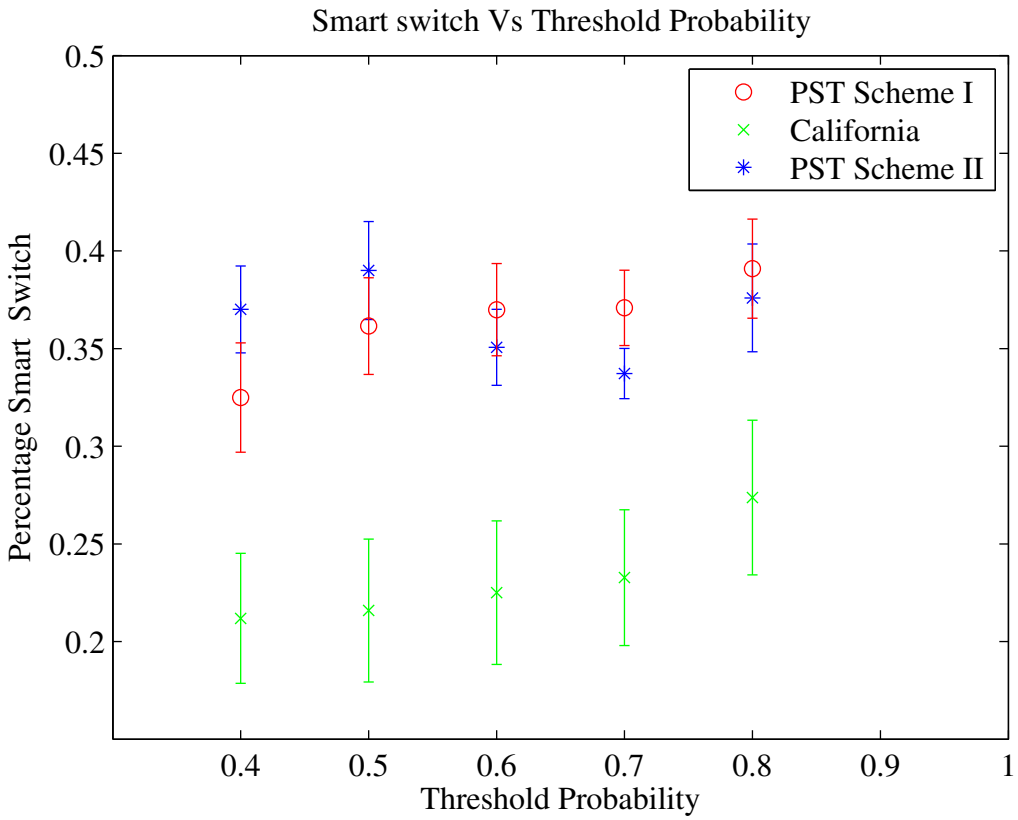


Figure 5.5.3: Smart switch for channel states distributed according to the Markovian arrival process

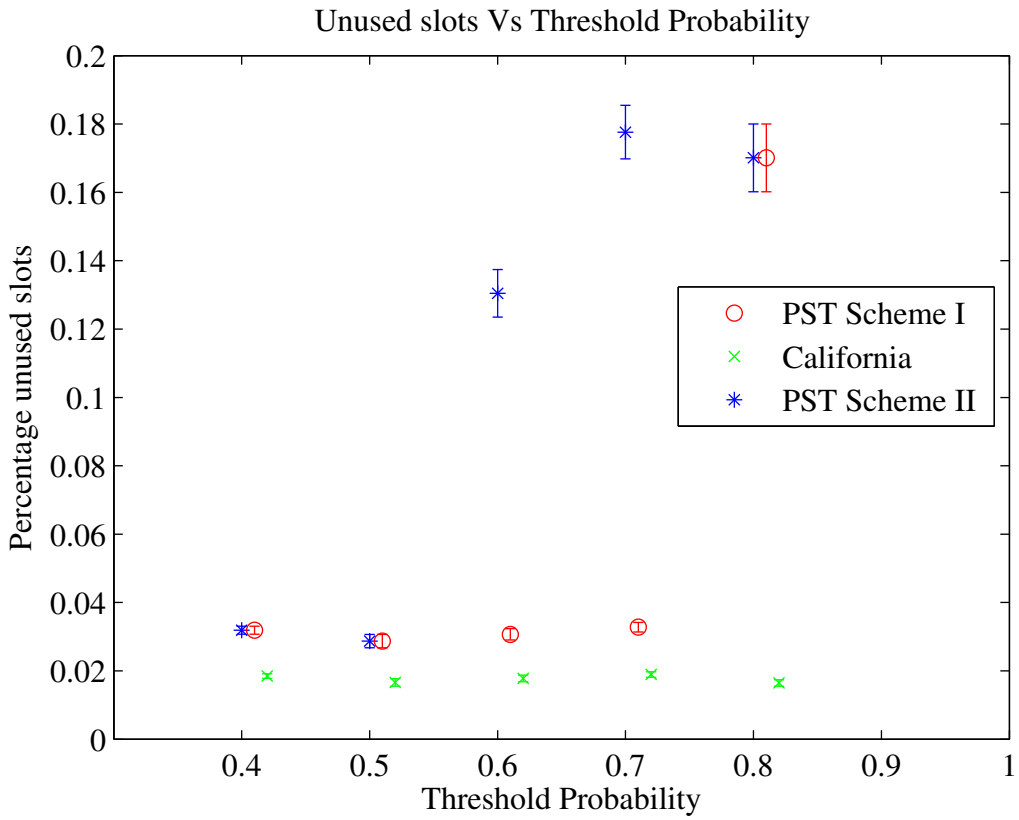


Figure 5.5.4: Unused Slots

5.6 Numerical results for Erlang-5 distributed channel states with imperfect sensing

In this simulation we consider the imperfect sensing scenario of the cognitive radios and its impact on the prediction schemes. We assume that the spectrum sensing scheme used by the cognitive radios is energy detection which was explained in Section 1.2.2. We also assume the channel only distorts the signals by additive white Gaussian noise. Then we get the matching probabilities of missed detection and false alarms for a particular noise power and number of samples tested assuming the energy detector uses the Neumann-Pearson optimality criteria. We change the probability of misdetection and using the matching false alarm probability we plot the graphs for our performance measures. In the graphs

we assume a mean signal to noise ratio of $20dB$. The number of samples used to get the result will be 5. In the graphs shown the percentage of Dumb Switch 1s increases with the misdetection probability which should be the case. But the Dumb Switch 2s get reduced which is surprising and the Smart switching also increases marginally. The unused slots decrease because of misdetection. In Figure 5.6.1 we show an example scenario of imperfect sensing. One observation we made about the prediction models in general is, if the sensing result in the previous slot concludes it to be *idle*, they predict the next slot to be *idle* with high probability and vice versa. Therefore whenever the channel status is misdetected within a time period where the channel is busy (time epochs t_1 to t_4 in Figure 5.6.1) other than the edge (time epoch t_5) where the *busy* status changes to *idle*, we can possibly end up in a Dumb Switch 1. If the edge sensing result is misdetected one can predict the *idle* slot earlier with high probability. If it is predicted early there is a high possibility that it has the highest possible number of slots. In that case if that channel is chosen there is a high possibility that the switching to be smart thus reducing the Dumb Switch 2s. When the misdetection probability is high the chance of above scenario occurring increases which explains the surprising result in Figure 5.6.3.

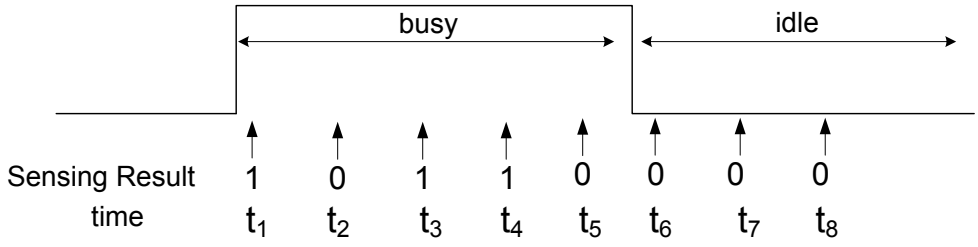


Figure 5.6.1: An example scenario of imperfect channel sensing

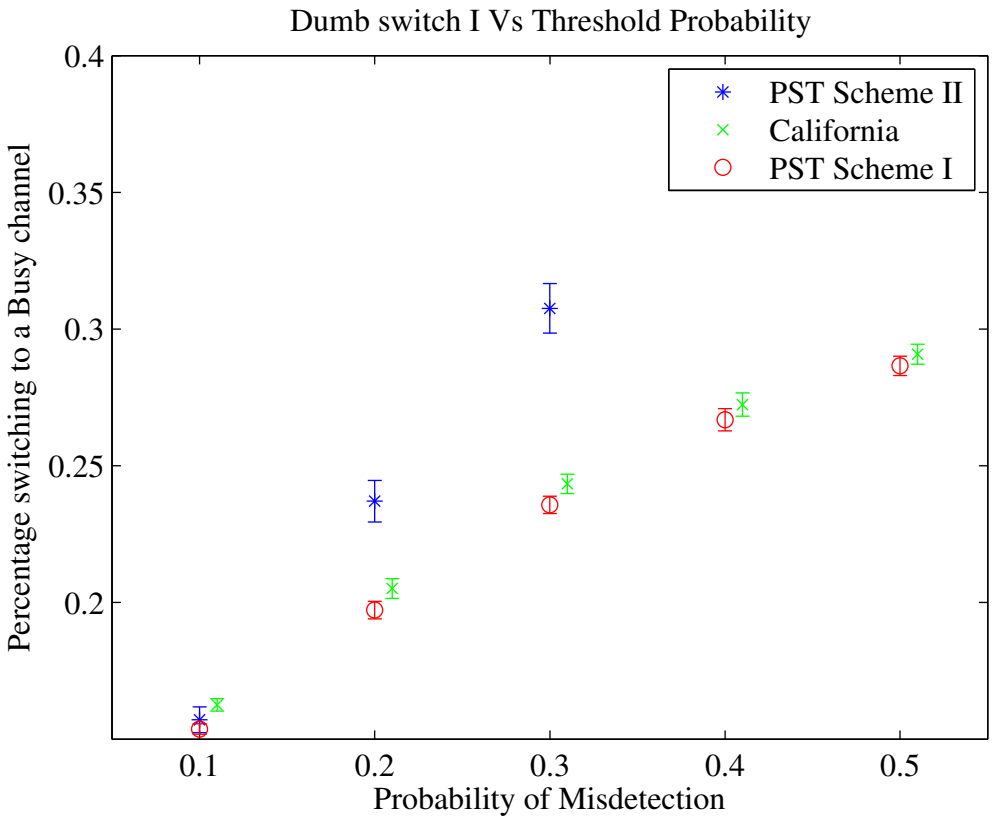


Figure 5.6.2: Dumb Switch I for Erlang-5 distributed channel states with imperfect sensing

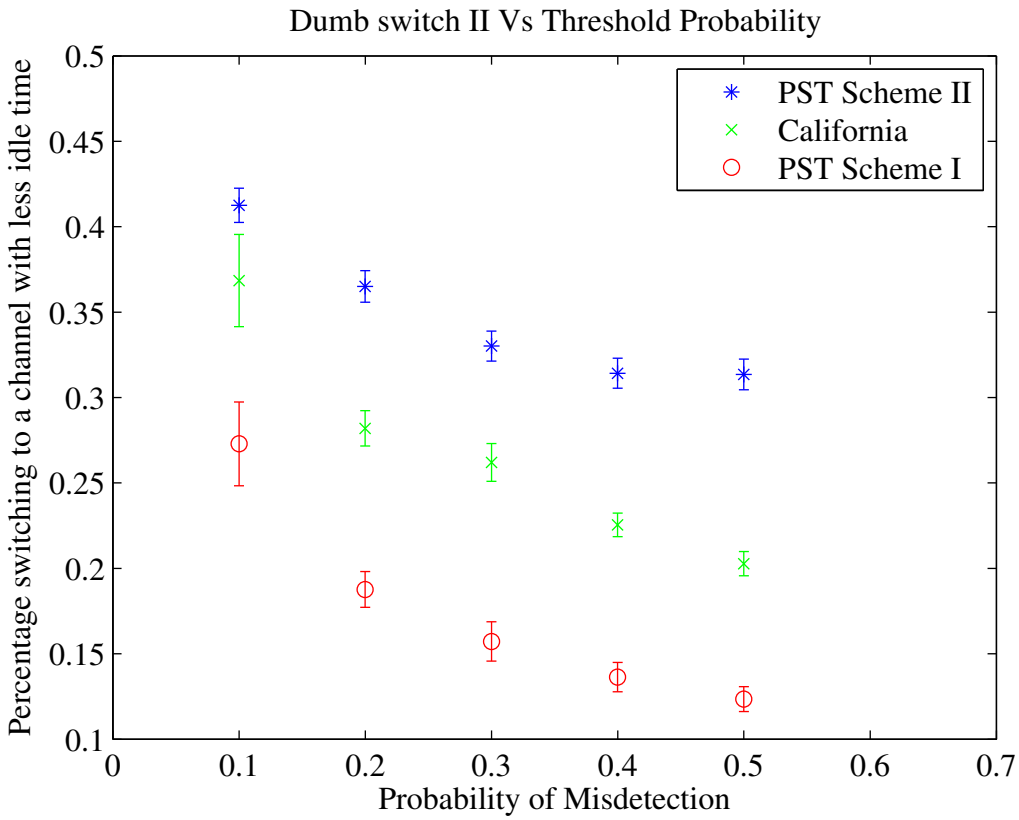


Figure 5.6.3: Dumb Switch II for Erlang-5 distributed channel states with imperfect sensing

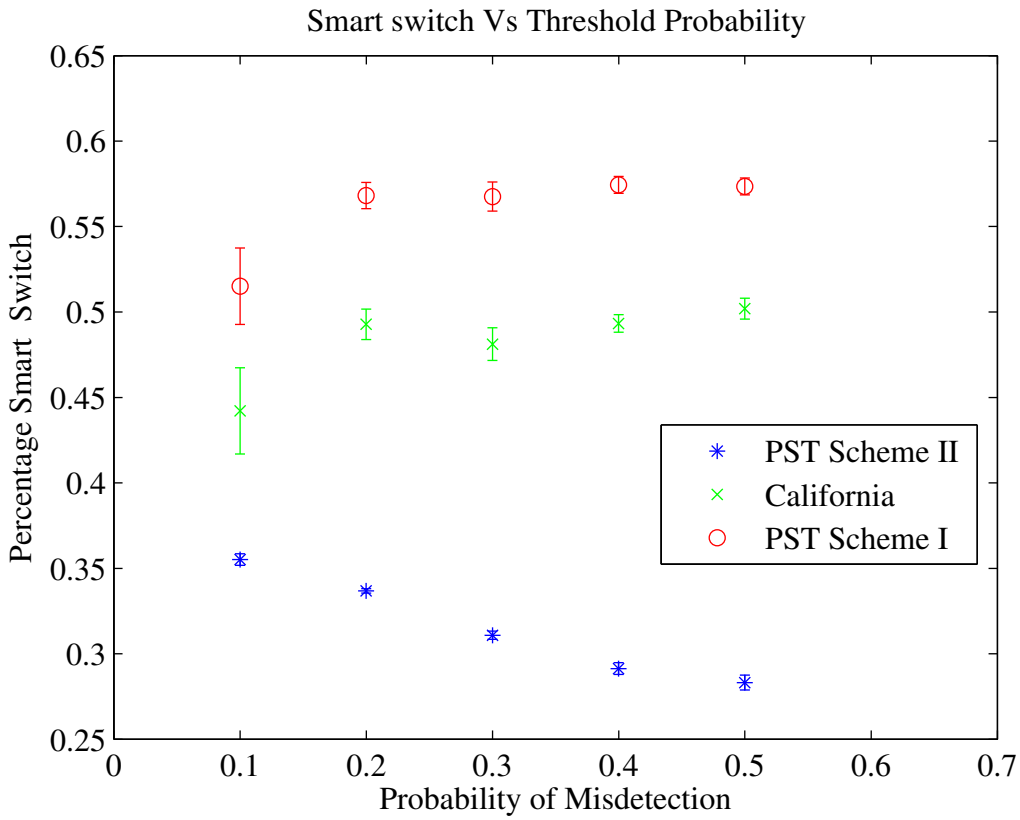


Figure 5.6.4: Smart switch for Erlang-5 distributed channel states with imperfect sensing

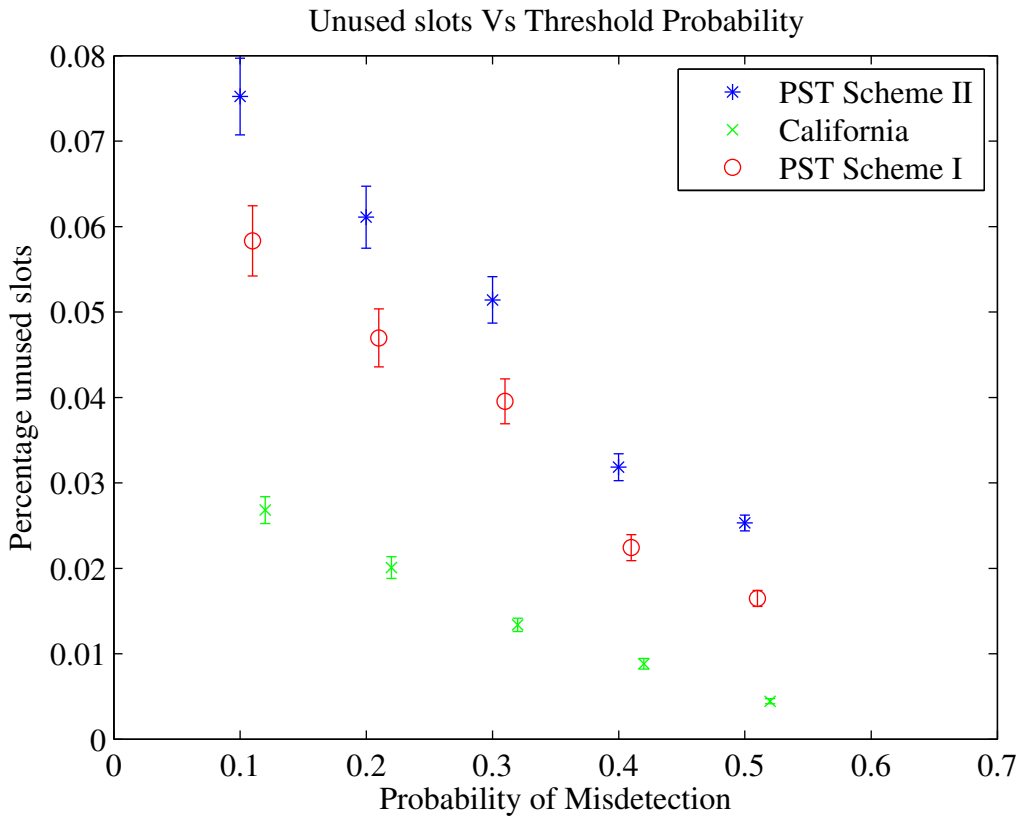


Figure 5.6.5: Unused Slots for Erlang-5 distributed channel states with imperfect sensing

5.7 An empirical probability distribution of the channel search delay for Erlang-5 distributed channel states

In this simulation we tried to find out the number of time slots it took to search for an *idle* channel using the two schemes we developed, the scheme found in [3] and a reactive channel sensing scheme. In the reactive channel sensing scheme, the cognitive radio just randomly selects a channel to search for *idle* slots. We present the results as a probability distribution in Figure 5.7.1. It confirms that using proactive switching the cognitive radio will be able to find *idle* channels with out much delay. This can lead to the enhancement of the quality of service experienced by the secondary users in terms of transmission delay. Furthermore it increases the channel utilization.

Table 5.3: Simulation Parameters Idle channel search delay

Parameter	Value
D	10
P_{min}	0.006
α	0
r	1.05
γ	0.0006
l	5
$Idle^{1}_{min}$	0.5
ε Scheme 1	2
ε Scheme 2	2
θ	1

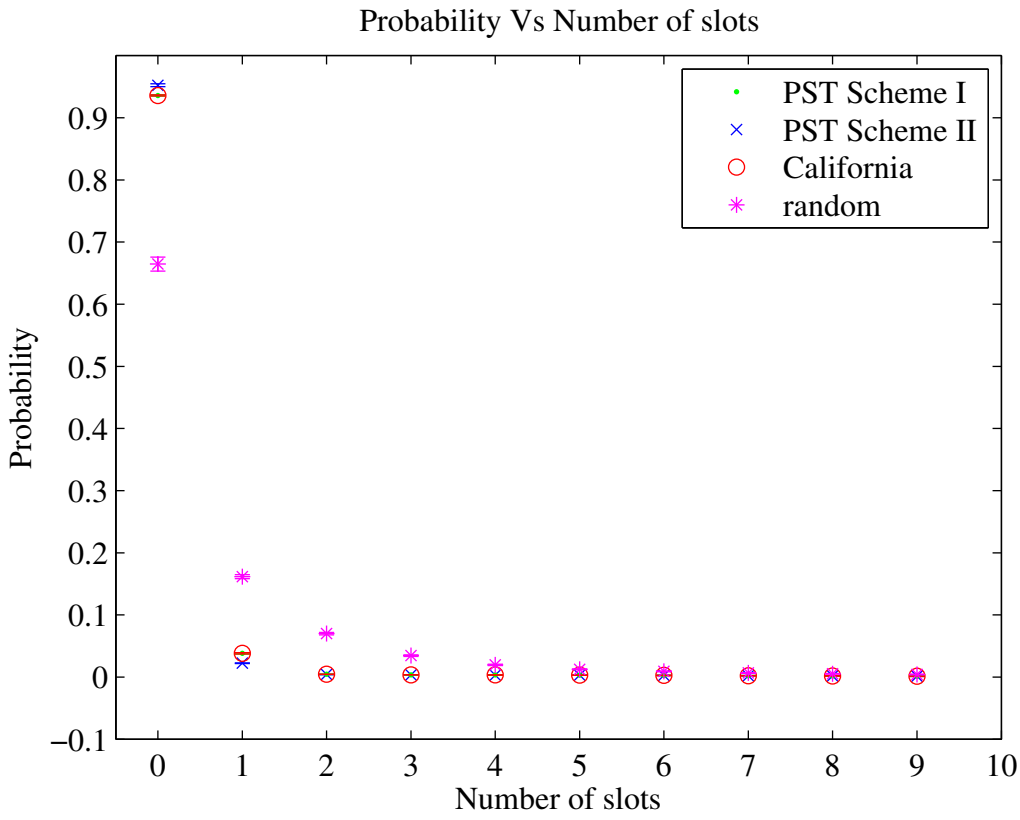


Figure 5.7.1: Empirical distribution function for idle channel search delay for Erlang-5 distributed channel states

5.8 Comparison with the scheme given in [30]

As we were unable to implement the Multilayer perceptron method given in [30] we tested our method of prediction by replicating their method of testing. In their test they assume the channel states to have sojourn times distributed according to the Geometric distribution. They presented one of the results in table form therefore we could compare it with the results we got using our scheme. According to the results we got our scheme performed better. They tested the scheme assuming a channel utilization of 0.5 and they tested for different inter-arrival times. The metric they presented in the table (Table I) was the prediction error given the channel is *idle*, $P^e(idle)$. They also plotted the graph of prediction error given the channel is *busy* but it was hard to get the values by looking at it. The comparison

results are given in Table 5.4.

Table 5.4: Comparison with the scheme given in [30]

Mean *idle* slots	Mean *busy* slots	$P^e(idle)$ in [30]	$P^e(idle)$ in our scheme
5	5	0.101477	0.0885
8	8	0.063528	0.0581
9	9	0.058236	0.0522
10	10	0.052713	0.0474
11	11	0.047875	0.0436

Chapter 6

Concluding remarks

6.1 Conclusions and Comments

The focus of this research is on prediction techniques of primary user channel states and using that information in efficient channel access of cognitive radios. In this thesis we presented some prediction schemes found in the literature. We mainly categorized them as the schemes which assume a channel usage pattern and schemes which come up with usage patterns based on the past channel usage data of the primary users. We also discussed the pros and cons of the schemes with respect to the scheme we used in prediction. We focused our attention on centralized spectral access schemes where both the primary network and the secondary network are centralized. We generated training and test data according to both an alternating Erlang-5 renewal process and a Markovian arrival process and used the probabilistic suffix tree algorithm to learn the distribution and predict the future channel states given the history of channel states in that channel. We compared our scheme with the scheme in reference [3] for the Erlang-5 distributed channel state changing scheme. Due to the complexity and since they have not done similar tests as in our case we did not compare our scheme with the other methods introduced in Chapter 2. We further test the performance of our scheme by generating a channel state sequence using a discrete time Markovian arrival process. In this case we could not compare it with the scheme in [3]

directly since that scheme is based on a continuous distribution. Therefore we did an approximate comparison where we assumed one discrete time slot to be of certain length and based on the mean of the Markovian arrival process we generated a exponential distribution. In this approximation the smaller the time slot length better was the comparison therefore we used a slot length of 0.05 time units in arriving with the results. Then for the channel states which are distributed according to the Erlang-5 renewal process we tested the prediction schemes for the case where the secondary user channel sensing is imperfect. We plot the graphs of performance measures for different probabilities of misdetection and matching false alarm probabilities. As mentioned above we could not implement other prediction schemes mentioned in Chapter 2 therefore we used our method of prediction on a test conducted in [30] and compared their results given in Table I in the respective paper with our scheme. Unfortunately the same could not be done to the other two schemes. In [32] the authors only gave a sample run graph of predicted values and observed values and in [31] the authors plot the cumulative distribution function of the Signal to interference ratio. The Signal to interference ratio cannot be compared with out knowing the physical distribution of the secondary user and the four primary users they used.

The channel modeling scheme we used has some advantages over the other schemes explained in Section 2. We test our scheme in different settings in sections 5.4, 5.5 and 5.6. In the first two cases we assume perfect sensing scenario. In Section 5.4 we compare our scheme with the scheme explained in Section 2.2. In the results shown our scheme gave better results. In Section 5.5 we tested our scheme when the primary user channel access is busty and have characteristics explained in Section 5.2. Since the scheme explained in Section 2.2 could only be used in continuous time scenario we were not able to compare it with our scheme. Finally we tested our scheme in the scenario of imperfect spectrum sensing and again our scheme gave better results than the Scheme explained in Section 2.2.

In summary, we can conclude that proactive spectrum access is a smart way to reduce the interference caused to the primary users. Since it orders the channels in the reducing order of the odds of finding an *idle* channel these schemes reduces the *idle* channel search

delay as shown in Section 5.7 which can improve the throughput experienced by the secondary users. The algorithm used by us has many advantages over the channel modeling algorithms found in the literature. These advantages make our scheme less complex and more computationally efficient.

6.2 Proposals for future work

In the current work we assumed that the channel sensing period is fixed but as one may note the channel sensing period should depend on the distribution of the primary user channel access. If the sensing period is too high we may incur too much interference to the primary users and if it is too low the time available for the secondary user to transmit data becomes less. Therefore based on the interference and the data rate one should be able to find the optimal sensing period for each channel.

Another area of improvement would be to automate the selection of the thresholds we used in our Proactive channel ordering scheme 1 and scheme 2 based on the the unused slots and the amount of dumb switch 1's to a particular channel. As one may assume if ϵ Scheme 1 and ϵ Scheme 2 are too low we will be accommodating more channels as good channels to transmit on but in the process we will increase the percentage of Dumb switch 1s. If those thresholds are too high we will only offer a few channels for the cognitive radios increasing the percentage of unused slots.

In the current work we considered only a single secondary user scenario. It would be interesting to see how the channel allocations should be done in a multiple user scenario using some medium access scheme. The applicability of the current scheme in a decentralized cognitive radio network will also be an interesting area to explore.

Appendix A

Example calculation of the Probabilistic suffix tree

This section shows the calculations conducted in populating the Probabilistic suffix tree shown in Figure A.2.1. The tree population and the necessary calculations conducted in and after each iteration is shown below. The values in the parenthesis give the relative abundance of a zero occurring and a one occurring respectively given the string naming the node is observed in the previous slot or slots. In the final tree derived after the phase 2 the values in the parenthesis are the probabilities of zero occurring and a one occurring respectively given the string naming the node is observed in the previous slot or slots. The training sequence and the threshold values used in the algorithm are given in Table A.1.

Description	Value
P_{min}	0.006
α	0
r	1.05
D	3
γ	0.0006
σ_1^{10}	1110010000

Table A.1: Threshold values and Training sequence used in Algorithm

A.1 Phase I of the Algorithm

In the tree graphs shown below, the blue color nodes denote the strings added to the tree $\bar{T}$. The white color nodes denote the strings to be tested in the set $\bar{S}$. The nodes in red denote the strings that have failed the test.

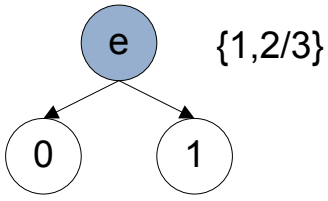
Iteration 1

Initialize : $\bar{T} = \{e\}$ and $\bar{S} = \{\ \}$

$\tilde{P}(0) = \frac{6}{10-3-1} \geq P_{min}$ {Using Equation 3.1.3}

$\tilde{P}(1) = \frac{4}{10-3-1} \geq P_{min}$ {Using Equation 3.1.3}

$\bar{S} = \{0, 1\}$ and $\bar{T} = \{e\}$



Remove 0 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|0) = \frac{4}{6} \geq \alpha$, $\frac{\tilde{P}(0|0)}{\tilde{P}(0|e)} = \frac{\frac{4}{6}}{1} < r$

Check for 1: $\tilde{P}(1|0) = \frac{1}{6} \geq \alpha$, $\frac{\tilde{P}(1|0)}{\tilde{P}(1|e)} = \frac{\frac{1}{6}}{\frac{2}{3}} < r$

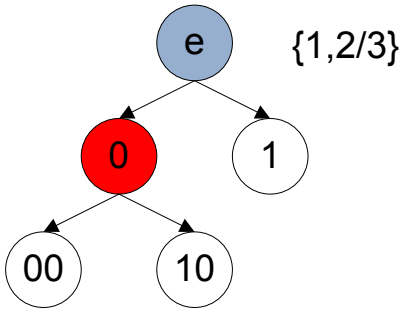
$\bar{S} = \{1\}$ and $\bar{T} = \{e\}$

$|'0'| = 1 < D$

$\tilde{P}(00) = \frac{4}{6} \geq P_{min}$

$\tilde{P}(10) = \frac{2}{6} \geq P_{min}$

$\bar{S} = \{1, 00, 10\}$ and $\bar{T} = \{e\}$



Iteration 2

Remove 1 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|1) = \frac{2}{4} \geq \alpha$, $\frac{\tilde{P}(0|1)}{\tilde{P}(0|e)} = \frac{\frac{2}{4}}{1} < r$

Check for 1: $\tilde{P}(1|1) = \frac{2}{4} \geq \alpha$, $\frac{\tilde{P}(1|1)}{\tilde{P}(1|e)} = \frac{\frac{2}{4}}{\frac{2}{3}} < r$

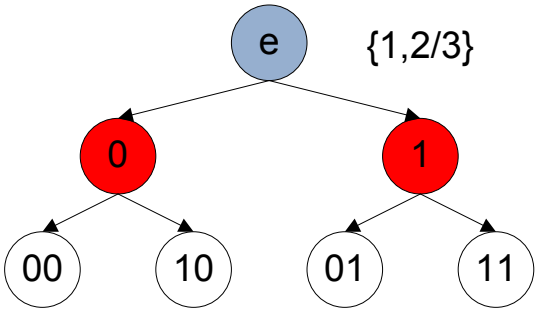
$\bar{S} = \{1\}$ and $\bar{T} = \{e\}$

$|'1'| = 1 < D$

$\tilde{P}(01) = \frac{1}{6} \geq P_{min}$

$\tilde{P}(11) = \frac{2}{6} \geq P_{min}$

$\bar{S} = \{00, 10, 01, 11\}$ and $\bar{T} = \{e\}$



Iteration 3

Remove 00 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|00) = \frac{2}{4} \geq \alpha$, $\frac{\tilde{P}(0|00)}{\tilde{P}(0|0)} = \frac{\frac{2}{4}}{\frac{4}{6}} < r$

Check for 1: $\tilde{P}(1|00) = \frac{1}{4} \geq \alpha$, $\frac{\tilde{P}(1|00)}{\tilde{P}(1|0)} = \frac{\frac{1}{4}}{\frac{1}{6}} > r$

Add node 00 to the tree with its predecessors

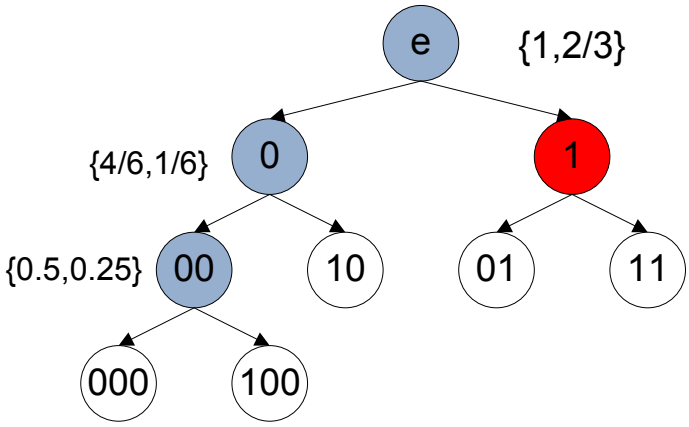
$\bar{S} = \{10, 01, 11\}$ and $\bar{T} = \{e, 0, 00\}$

$|'00'| = 2 < D$

$\tilde{P}(000) = \frac{2}{6} \geq P_{min}$

$\tilde{P}(100) = \frac{2}{6} \geq P_{min}$

$\bar{S} = \{10, 01, 11, 000, 100\}$ and $\bar{T} = \{e, 0, 00\}$



Iteration 4

Remove 10 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|10) = \frac{2}{2} \geq \alpha$, $\frac{\tilde{P}(0|10)}{\tilde{P}(0|0)} = \frac{\frac{2}{2}}{\frac{4}{6}} \geq r$

Check for 1: $\tilde{P}(1|10) = \frac{0}{4} \geq \alpha$, $\frac{\tilde{P}(1|10)}{\tilde{P}(1|0)} = \frac{\frac{0}{4}}{\frac{1}{6}} < r$

Add node 10 to the tree with its predecessors

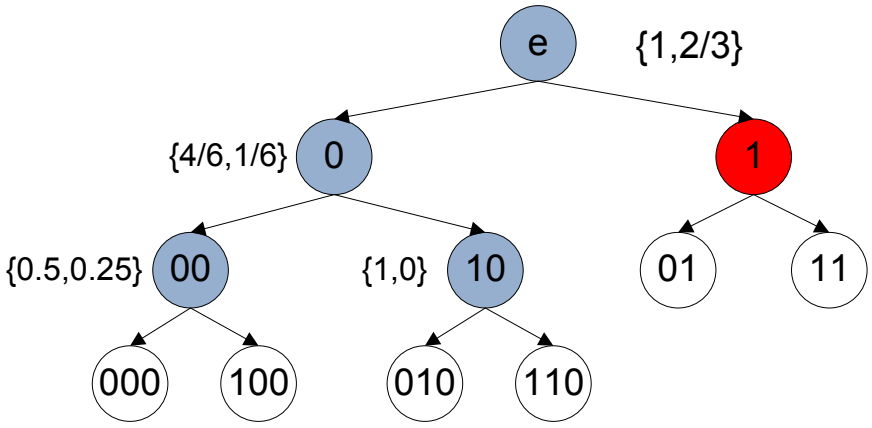
$\bar{S} = \{01, 11\}$ and $\bar{T} = \{e, 0, 00, 10\}$

$|'10'| = 2 < D$

$\tilde{P}(010) = \frac{1}{6} \geq P_{min}$

$\tilde{P}(110) = \frac{1}{6} \geq P_{min}$

$\bar{S} = \{01, 11, 000, 100, 010, 110\}$ and $\bar{T} = \{e, 0, 00, 10\}$



Iteration 5

Remove 01 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|01) = \frac{1}{1} \geq \alpha$, $\frac{\tilde{P}(0|01)}{\tilde{P}(0|1)} = \frac{\frac{1}{1}}{\frac{2}{4}} \geq r$

Check for 1: $\tilde{P}(1|01) = \frac{0}{1} \geq \alpha$, $\frac{\tilde{P}(1|01)}{\tilde{P}(1|1)} = \frac{\frac{0}{1}}{\frac{2}{4}} < r$

Add node 01 to the tree with its predecessors

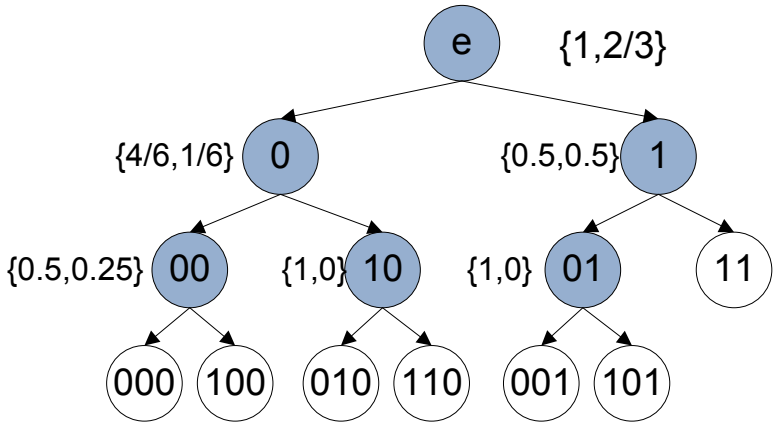
$\bar{S} = \{11, 000, 100, 010, 110\}$ and $\bar{T} = \{e, 0, 00, 10, 01\}$

$|'01'| = 2 < D$

$\tilde{P}(001) = \frac{1}{6} \geq P_{min}$

$\tilde{P}(101) = \frac{0}{6} < P_{min}$

$\bar{S} = \{11, 000, 100, 010, 110, 001\}$ and $\bar{T} = \{e, 0, 00, 10, 01\}$



Iteration 6

Remove 11 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|11) = \frac{1}{2} \geq \alpha$, $\frac{\tilde{P}(0|11)}{\tilde{P}(0|1)} = \frac{\frac{1}{2}}{\frac{2}{4}} < r$

Check for 1: $\tilde{P}(1|11) = \frac{1}{2} \geq \alpha$, $\frac{\tilde{P}(1|11)}{\tilde{P}(1|1)} = \frac{\frac{1}{2}}{\frac{2}{4}} < r$

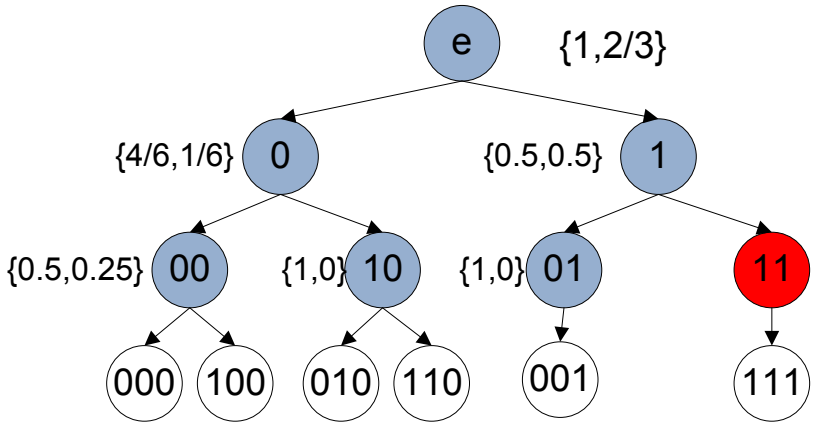
$\bar{S} = \{000, 100, 010, 110, 001\}$ and $\bar{T} = \{e, 0, 00, 10, 01\}$

$|'11'| = 2 < D$

$\tilde{P}(011) = \frac{0}{6} < P_{min}$

$\tilde{P}(111) = \frac{1}{6} \geq P_{min}$

$\bar{S} = \{000, 100, 010, 110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01\}$



Iteration 7

Remove 000 from $\bar{S}$ and check if it satisfies conditions to be on the tree

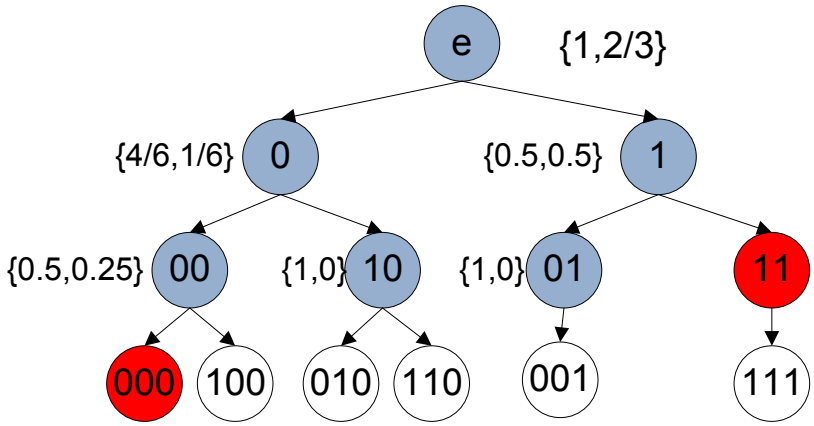
Check for 0: $\tilde{P}(0|000) = \frac{1}{2} \geq \alpha$, $\frac{\tilde{P}(0|000)}{\tilde{P}(0|00)} = \frac{\frac{1}{2}}{\frac{2}{4}} < r$

Check for 1: $\tilde{P}(1|000) = \frac{0}{2} \geq \alpha$, $\frac{\tilde{P}(1|000)}{\tilde{P}(1|00)} = \frac{\frac{0}{2}}{\frac{1}{4}} < r$

$\bar{S} = \{100, 010, 110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01\}$

$|'000'| = 3 \geq D$

$\bar{S} = \{100, 010, 110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01\}$



Iteration 8

Remove 100 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|100) = \frac{1}{2} \geq \alpha$, $\frac{\tilde{P}(0|100)}{\tilde{P}(0|00)} = \frac{\frac{1}{2}}{\frac{2}{4}} < r$

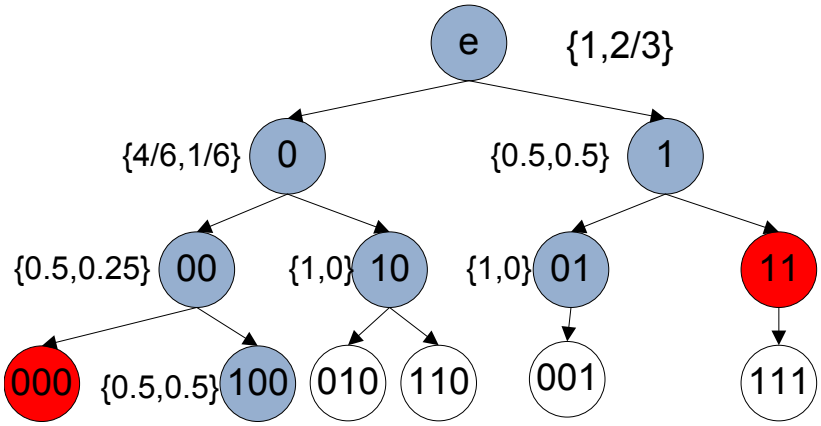
Check for 1: $\tilde{P}(1|100) = \frac{1}{2} \geq \alpha$, $\frac{\tilde{P}(1|100)}{\tilde{P}(1|00)} = \frac{\frac{1}{2}}{\frac{1}{4}} \geq r$

Add node 100 to the tree with its predecessors

$\bar{S} = \{010, 110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$

$|'100'| = 3 \geq D$

$\bar{S} = \{010, 110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$



Iteration 9

Remove 010 from $\bar{S}$ and check if it satisfies conditions to be on the tree

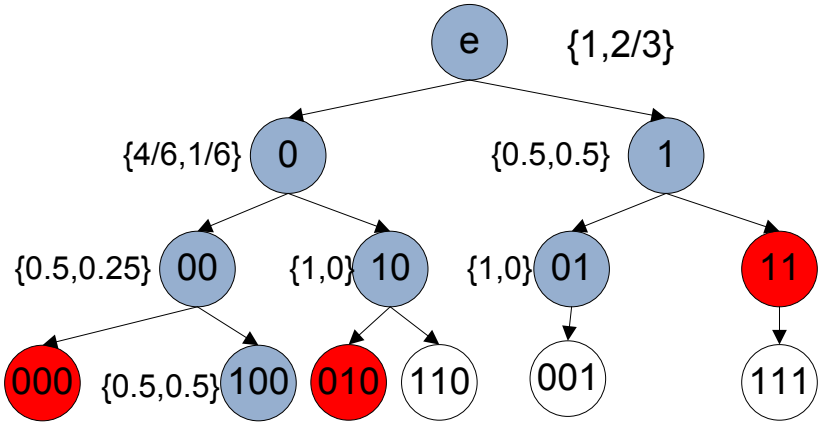
Check for 0: $\tilde{P}(0|010) = \frac{1}{1} \geq \alpha$, $\frac{\tilde{P}(0|010)}{\tilde{P}(0|10)} = \frac{\frac{1}{1}}{\frac{2}{2}} < r$

Check for 1: $\tilde{P}(1|010) = \frac{0}{1} \geq \alpha$, $\frac{\tilde{P}(1|010)}{\tilde{P}(1|10)} = \frac{0}{\frac{1}{2}}$ undefined

$\bar{S} = \{110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$

$|'010'| = 3 \geq D$

$\bar{S} = \{110, 001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$



Iteration 10

Remove 110 from $\bar{S}$ and check if it satisfies conditions to be on the tree

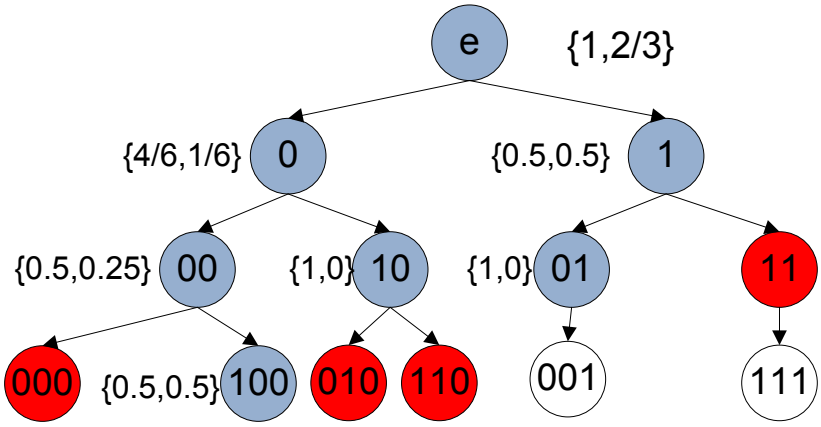
Check for 0: $\tilde{P}(0|110) = \frac{1}{1} \geq \alpha$, $\frac{\tilde{P}(0|110)}{\tilde{P}(0|10)} = \frac{\frac{1}{1}}{\frac{2}{2}} < r$

Check for 1: $\tilde{P}(1|110) = \frac{0}{1} \geq \alpha$, $\frac{\tilde{P}(1|110)}{\tilde{P}(1|10)} = \frac{\frac{0}{1}}{\frac{0}{2}}$ undefined

$\bar{S} = \{001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$

$|'110'| = 3 \geq D$

$\bar{S} = \{001, 111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$



Iteration 11

Remove 001 from $\bar{S}$ and check if it satisfies conditions to be on the tree

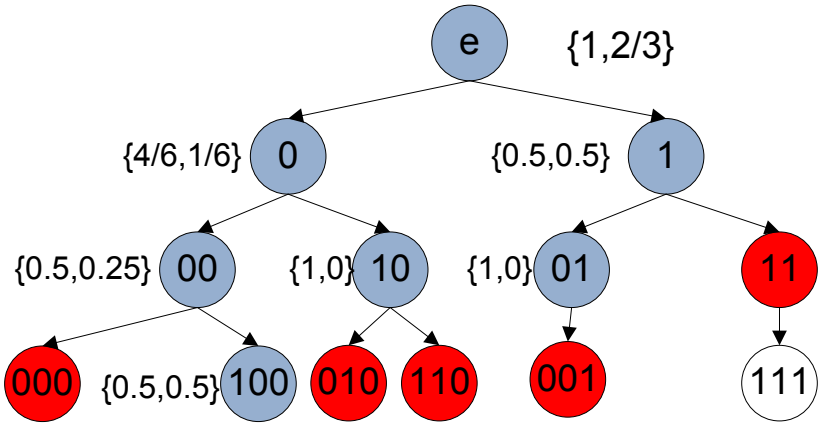
Check for 0: $\tilde{P}(0|001) = \frac{1}{1} \geq \alpha$, $\frac{\tilde{P}(0|001)}{\tilde{P}(0|01)} = \frac{\frac{1}{1}}{\frac{1}{1}} < r$

Check for 1: $\tilde{P}(1|001) = \frac{0}{1} \geq \alpha$, $\frac{\tilde{P}(1|001)}{\tilde{P}(1|01)} = \frac{\frac{0}{1}}{\frac{0}{1}}$ undefined

$\bar{S} = \{111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$

$|'001'| = 3 \geq D$

$\bar{S} = \{111\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100\}$



Iteration 12

Remove 111 from $\bar{S}$ and check if it satisfies conditions to be on the tree

Check for 0: $\tilde{P}(0|111) = \frac{1}{1} \geq \alpha$, $\frac{\tilde{P}(0|111)}{\tilde{P}(0|11)} = \frac{\frac{1}{1}}{\frac{1}{2}} \geq r$

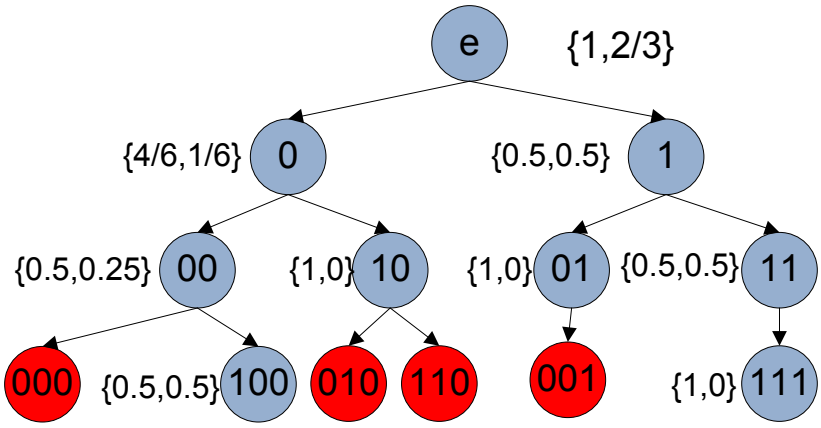
Check for 1: $\tilde{P}(1|111) = \frac{0}{1} \geq \alpha$, $\frac{\tilde{P}(1|111)}{\tilde{P}(1|11)} = \frac{\frac{0}{1}}{\frac{1}{2}} \geq r$

Add node 111 to the tree with its predecessors

$\bar{S} = \{\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 100, 111\}$

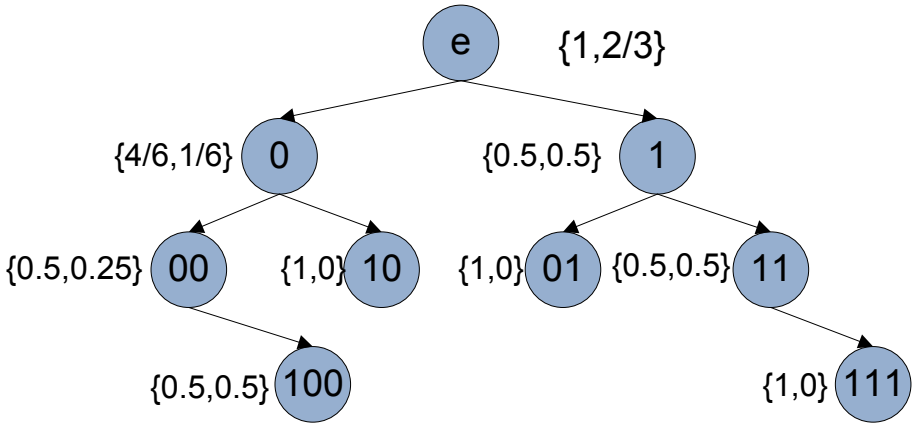
$|'111'| = 3 \geq D$

$\bar{S} = \{\}$ and $\bar{T} = \{e, 0, 00, 10, 01, 1, 100, 11, 111\}$



A.2 Phase II of the Algorithm

Start with the tree $\bar{T}$ obtained in the phase one of the algorithm. The tree $\bar{T}$ is given below.



Add the missing children of internal nodes. Calculated the conditional probabilities of the added nodes using the Equation 3.1.8.

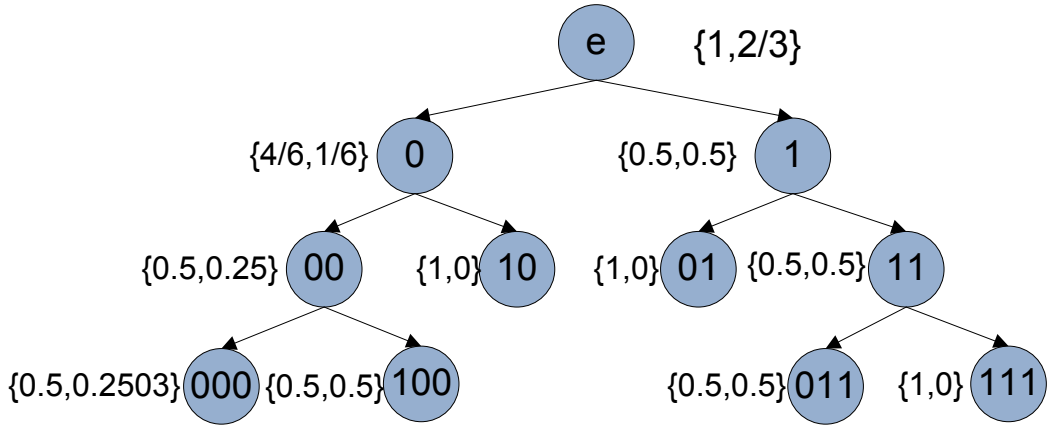
Calculating conditional probabilities of node 000 using Equation 3.1.8.

$$\begin{aligned}\gamma_{000}(0) &= 0.5(1 - 2 * 0.0006) + 0.0006 \\ &= 0.5\end{aligned}$$

$$\begin{aligned}\gamma_{000}(1) &= 0.25(1 - 2 * 0.0006) + 0.0006 \\ &= 0.2503\end{aligned}$$

Calculating conditional probabilities of node 011 using Equation 3.1.8.

$$\begin{aligned}\gamma_{011}(0) &= 0.5(1 - 2 * 0.0006) + 0.0006 \\ &= 0.5 \\ \gamma_{011}(1) &= 0.5(1 - 2 * 0.0006) + 0.0006 \\ &= 0.5\end{aligned}$$



In instances where the conditional probabilities are less than γ, we equate them to γ. Re-normalize the conditional distributions.

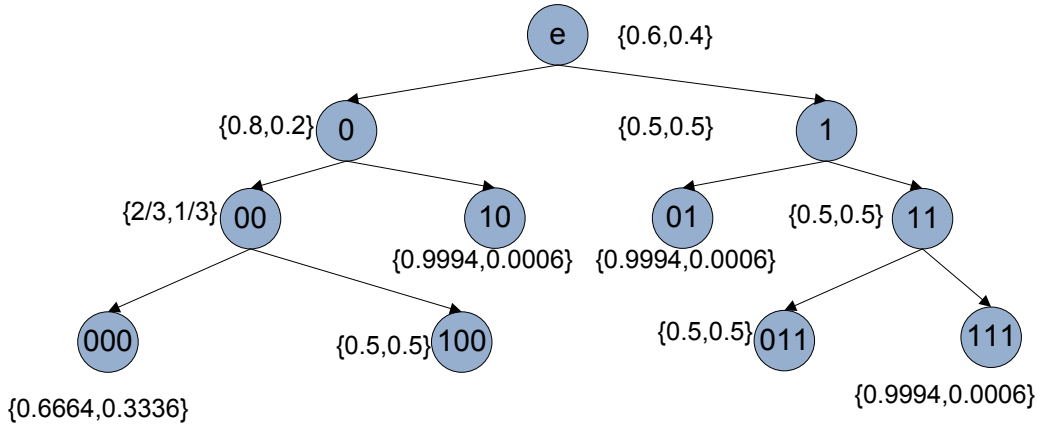


Figure A.2.1: Probabilistic Suffix Tree

A.3 Deriving the variable order Markov model

The variable order Markov model representation of the probabilistic suffix tree given in Figure A.2.1 is given in Figure A.3.1. No change to the tree is necessary since it already

has the features given in Section 3.1.3.

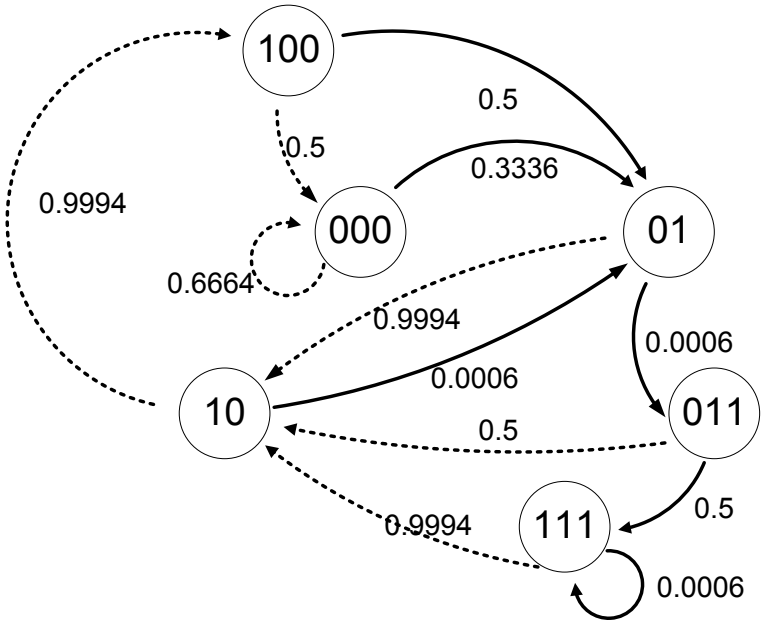


Figure A.3.1: Variable order Markov model

Bibliography

[1] "FCC, ET Docket No 03-222 Notice of proposed rule making and order, December 2003." 2003.

[2] "Federal Communications Commission Spectrum Policy Task Force Report of the Spectrum Efficiency Working Group," 2002.

[3] L. Yang, L. Cao, and H. Zheng, "Proactive channel access in dynamic spectrum networks," *Physical Communication*, vol. 1, no. 2, pp. 103–111, Jun. 2008. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1874490708000268

[4] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, Feb. 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1391031

[5] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, no. 13, pp. 2127–2159, Sep. 2006. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128606001009

[6] F. K. Jondral, "Software-Defined RadioBasics and Evolution to Cognitive Radio," *EURASIP Journal on Wireless Communications and Networking*, vol. 2005, pp. 275–283, 2005. [Online]. Available: http://www.hindawi.com/journals/wcn/2005/652784/abs/

[7] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Communications Surveys & Tutorials*, vol. 11, no. 1, pp. 116–130, 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4796930

[8] W. Zhang and K. Letaief, "Cooperative Spectrum Sensing," in *Cognitive Wireless Communication Networks*, E. Hossain and V. Bhargava, Eds. Springer US, Dec. 2007, vol. 7, no. 12, ch. 4, pp. 115–138. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-68832-9_4

[9] D. Cabric, S. Mishra, and R. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004.*, pp. 772–776, 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1399240

[10] R. Tandra, "Fundamental limits on detection in low snr," Masters Thesis, UNIVERSITY OF CALIFORNIA, BERKELEY. [Online]. Available: www.eecs.berkeley.edu/~sahai/Theses/Rahul_MSThesis.pdf

[11] S. M. Kay, *Fundamentals of Statistical Signal Processing, Volume 2: Detection Theory*. Prentice Hall, 1998. [Online]. Available: http://www.amazon.com/Fundamentals-Statistical-Signal-Processing-Detection/dp/013504135X

[12] H. Hu, "Cyclostationary approaches to signal detection and classification in cognitive radio Systems," in *Cognitive Radio Systems*, W. Wang, Ed. InTech, 2009, ch. 5, pp. 51–76. [Online]. Available: http://www.intechopen.com/articles/show/title/cyclostationary-approach-to-signal-detection-and-classification-in-cognitiveradio-systems

[13] L. Gavrilovska and V. Atanasovski, "Spectrum Sensing Framework for Cognitive Radio Networks," *Wireless Personal Communications*, Feb. 2011. [Online]. Available: http://www.springerlink.com/index/10.1007/s11277-011-0239-1

[14] W. Gardner and C. Spooner, "Signal interception: performance advantages of cyclic-feature detectors," *IEEE Transactions on Communications*, vol. 40, no. 1, pp. 149–159, 1992. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=126716

[15] Z. Tian and G. Giannakis, "A wavelet approach to wideband spectrum sensing for cognitive radios," in *Cognitive Radio Oriented Wireless Networks and Communications, 2006. 1st International Conference on*. Ieee, 2006, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4211139

[16] Z. Tian and G. B. Giannakis, "Compressed Sensing for Wideband Cognitive Radios," *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, pp. IV–1357–IV–1360, Apr. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4218361

[17] C. Peng, H. Zheng, and B. Y. Zhao, "Utilization and fairness in spectrum assignment for opportunistic spectrum access," *Mobile Networks and Applications*, vol. 11, no. 4, pp. 555–576, May 2006. [Online]. Available: http://www.springerlink.com/index/10.1007/s11036-006-7322-y

[18] P. A. Kumar Acharya, S. Singh, and H. Zheng, "Reliable open spectrum communications through proactive spectrum access," *Proceedings of the first international workshop on Technology and policy for accessing spectrum - TAPAS '06*, pp. 5–es, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1234388.1234393

[19] C. Comaniciu and N. Nie, "Adaptive channel allocation spectrum etiquette for cognitive radio networks," *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pp. 269–278, 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1542643

[20] K. Gilhousen, I. Jacobs, R. Padovani, a.J. Viterbi, L. Weaver, and C. Wheatley, "On the capacity of a cellular CDMA system," *IEEE Transactions on Vehicular Technology*, vol. 40, no. 2, pp. 303–312, May 1991. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=289411

[21] T. S. Rappaport, *Wireless Communications*. Prentice Hall, 1996. [Online]. Available: http://www.amazon.com/Wireless-Communications-Theodore-S-Rappaport/dp/0133755363

[22] Liuqing Yang and G. Giannakis, "Ultra-wideband communications: an idea whose time has come," *Signal Processing Magazine, IEEE*, vol. 21, no. no.6, pp. 26– 54. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1359140&isnumber=29810

[23] R. Menon, R. Buehrer, and J. Reed, "Outage probability based comparison of underlay and overlay spectrum sharing techniques," *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pp. 101–109. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1542623

[24] C. Cordeiro, K. Challapali, D. Birru, and N. Sai Shankar, "IEEE 802 . 22 : The First Worldwide Wireless Standard based on Cognitive Radios," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN*, 2005, pp. 328–337. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1542649&isnumber=32916

[25] A. Feickert, "The Joint Tactical Radio System (JTRS) and the Army's Future Combat System (FCS): Issues for Congress." Washington D.C., USA . UNT Digital Library., Tech. Rep., 2005. [Online]. Available: http://digital.library.unt.edu/ark:/67531/metacrs7941/.

[26] R. D. Hinman, "APPLICATION OF COGNITIVE RADIO TECHNOLOGY TO LEGACY MILITARY WAVEFORMS," in *Military Communications Conference*, Washington, DC, 2006, pp. 1–5.

[27] "Cognitive Radio for Public Safety." [Online]. Available: http://transition.fcc.gov/pshs/techtopics/techtopic8.html

[28] W. Lehr and N. Jesuale, "Public Safety Radios Must Pool Spectrum," *IEEE Communications Magazine*, no. March, pp. 103–109, 2009.

[29] H. Kim and K. G. Shin, "Adaptive MAC-layer Sensing of Spectrum Availability in Cognitive Radio Networks," University of Michigan, Tech. Rep., 2006. [Online]. Available: http://www.eecs.umich.edu/techreports/cse/2006/CSE-TR-518-06.pdf

[30] V. K. Tumuluru, P. Wang, and D. Niyato, "A neural network based spectrum prediction scheme for cognitive radio," in *IEEE International Conference on Communications*, vol. 294, no. 3, Mar. 2010, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5502348&isnumber=5501741

[31] I. Akbar and W. Tranter, "Dynamic spectrum allocation in cognitive radio using hidden Markov models: Poisson distributed case," *Proceedings 2007 IEEE SoutheastCon*, pp. 196–201, 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4147414

[32] S. Yarkan and H. Arslan, "Binary Time Series Approach to Spectrum Prediction for Cognitive Radio," *2007 IEEE 66th Vehicular Technology Conference*, pp. 1563–1567, Sep. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4349981

[33] L. Rabiner and B. Juang, "An introduction to hidden Markov models." *ASSP Magazine, IEEE*, vol. Appendix 3, no. January, pp. 4 – 16, Jan. 1986. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1165342&isnumber=26219

[34] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Transactions on Information Theory*, vol. 48, no. 6, pp. 1518–1569, Jun. 2002. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1003838

[35] B. Kedem and K. Fokianos, *Regression Models for Time Series analysis*, 1st ed. Wiley Series in Probability and Statistics. John Wiley And Sons, Inc, 2002.

[36] D. R. Cox, "The Regression Analysis of Binary Sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 20, no. 2, pp. 215–242, Jan. 1958. [Online]. Available: http://www.jstor.org/stable/2983890

[37] Helmut Lütkepohl, *New Introduction to Multiple Time Series Analysis*, 1st ed. Springer US, 2005.

[38] J. Cai and a. S. Alfa, "Optimal Channel Sensing in Wireless Communication Networks with Cognitive Radio," *2009 IEEE International Conference on Communications*, pp. 1–5, Jun. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5199277

[39] S. Robert and J.-Y. L. Boudec, "New models for pseudo self-similar traffic," *Performance Evaluation*, vol. 30, no. 1-2, pp. 57–68, Jul. 1997. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0166531696000557

[40] D. Ron, Y. Singer, and N. Tishby, "The power of amnesia: Learning probabilistic automata with variable memory length," *Machine Learning*, vol. 25, no. 2-3, pp. 117–149, 1997. [Online]. Available: http://www.springerlink.com/index/10.1007/BF00114008

[41] R. Begleiter, R. El-yaniv, and G. Yona, "On Prediction Using Variable Order Markov Models," *Journal of Artificial Intelligence Research*, vol. 22, pp. 385–421, 2004. [Online]. Available: http://dx.doi.org/10.1613/jair.1491

[42] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by Markov chains," *Ieee Wireless Communications*, vol. 16, no. April, pp. 56–64, 2009. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4907561&isnumber=4907549

[43] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. New York, New York, USA: Wiley-Interscience, 2006.

[44] M. Haugh, "Generating Random Variables and Stochastic Processes," 2004. [Online]. Available: www.columbia.edu/~mh2078/MCS04/MCS_generate_rv.pdf

[45] A. S. Alfa, V. Pla, J. Martinez-bauset, and V. Casares-Giner, "Discrete time analysis of cognitive radio networks with saturated source of secondary users," in *Performance Evaluation of Cognitive Radio Networks. Workshop*, Valencia, Spain, 2011. [Online]. Available: http://personales.upv.es/~jmartine/Publications/PE_CRN2011.pdf