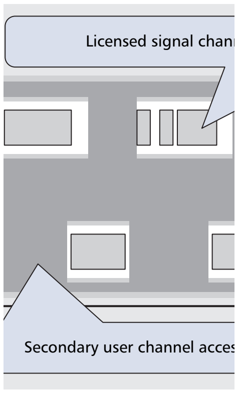
APPLICATIONS OF MACHINE LEARNING TO COGNITIVE RADIO NETWORKS

CHARLES CLANCY, DEPARTMENT OF DEFENSE
JOE HECKER, SAIC
ERICH STUNTEBECK, GEORGIA TECH
TIM O'SHEA, NC STATE



The authors describe a concrete model for a generic cognitive radio to utilize a learning engine. The goal is to incorporate the results of the learning engine into a predicate calculus-based reasoning engine.

ABSTRACT

Cognitive radio offers the promise of intelligent radios that can learn from and adapt to their environment. To date, most cognitive radio research has focused on policy-based radios that are hard-coded with a list of rules on how the radio should behave in certain scenarios. Some work has been done on radios with learning engines tailored for very specific applications. This article describes a concrete model for a generic cognitive radio to utilize a learning engine. The goal is to incorporate the results of the learning engine into a predicate calculus-based reasoning engine so that radios can remember lessons learned in the past and act quickly in the future. We also investigate the differences between reasoning and learning, and the fundamentals of when a particular application requires learning, and when simple reasoning is sufficient. The basic architecture is consistent with cognitive engines seen in AI research. The focus of this article is not to propose new machine learning algorithms, but rather to formalize their application to cognitive radio and develop a framework from within which they can be useful. We describe how our generic cognitive engine can tackle problems such as capacity maximization and dynamic spectrum access.

INTRODUCTION

In today's literature, *cognitive radio* is often treated as a buzz word rather than a scientific term since it has been used by so many different people to mean so many different things. The most generally accepted definition is a radio that can sense and adapt to its environment. The term *cognitive* implies awareness, perception, reasoning, and judgment. However, nowhere is *learning* required.

So far this has fueled much research into policy-based cognitive radios. These are radios whose operation is governed by a reasoning engine that examines the current state of the environment and makes decisions on how the radio should operate. An example of this might be an IEEE 802.11 modulation controller that switches from 16-quadrature amplitude modulation (QAM) to quaternary phase shift keying (QPSK) to binary PSK (BPSK) as the signal-to-noise ratio (SNR) decreases [1].

Generic learning-based cognitive radio is a relatively unchartered research area. Various projects have used techniques such as genetic algorithms to evolve radio parameters with the goal of optimizing performance [2]. In contrast, this article examines the fundamentals of learning and reasoning, and proposes an architecture to use them together. We then apply the framework to two common problems in cognitive radio: capacity maximization and dynamic spectrum access.

In this article we describe the cognitive radio architecture, and discuss reasoning and learning engines. We then describe applications and how they work with the described model, and outline our cognitive radio implementation. We then conclude the article.

COGNITIVE RADIO ARCHITECTURE

A software radio (SR) can be defined as a radio implemented with generic hardware that can be programmed to transmit and receive a variety of waveforms. Cognitive radio is often thought of as an extension to software radio, and here we treat it as such. A cognitive radio extends a software radio by adding an independent cognitive engine, composed of a knowledge base, reasoning engine, and learning engine, to drive software modifications. A well defined application programming interface (API) dictates communi-

This work was completed while J. Hecker, E. Stuntebeck, and T. O'Shea were with the Laboratory for Telecommunication Sciences, U.S. Department of Defense, which funded this research. The opinions expressed in this document represent those of the authors, and should not be considered an official opinion or endorsement by the Department of Defense or U.S. federal government.

The SR exports variables that are either read-only or read-write. The read-only parameters represent statistics maintained by the SR, such as signal to noise ratio or bit error rate. The read-write variables represent configurable parameters such as transmit power, coding rate, or symbol constellation.

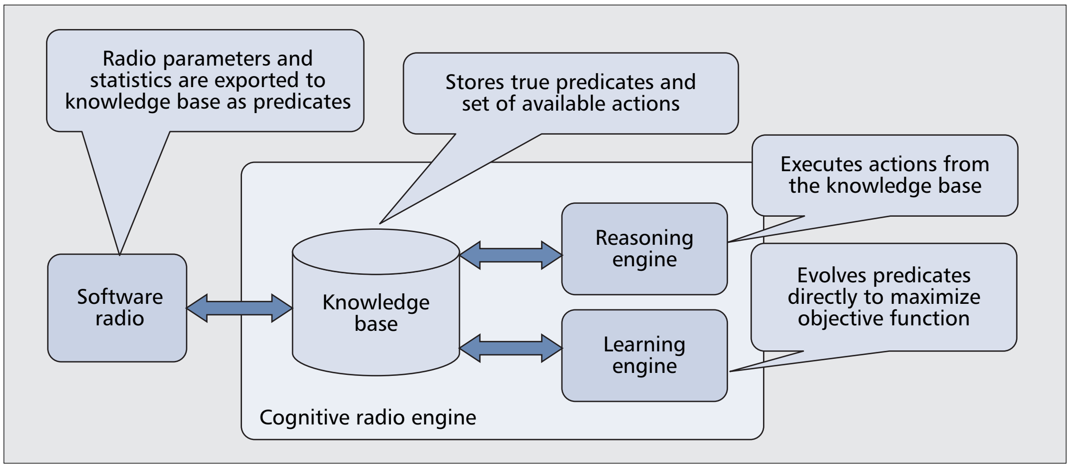

Radio parameters and statistics are exported to knowledge base as predicates
Stores true predicates and set of available actions
Executes actions from the knowledge base
Reasoning engine
Software radio
Knowledge base
Evolves predicates directly to maximize objective function
Learning engine
Cognitive radio engine

Figure 1. *Cognitive radio architecture showing the interactions between the software radio, knowledge base, and policy and learning engines.*

cation between the cognitive engine and the SR. Figure 1 illustrates this architecture and the interaction between various components.

At any given time, the cognitive engine generates conclusions based on information defined in the knowledge base, the radio's long-term memory. These conclusions are the result of extrapolations of this information based on reasoning or learning. The reasoning engine is what is often referred to in artificial intelligence (AI) literature as an expert system. The learning engine is responsible for manipulating the knowledge base from experience. As lessons are learned, the learning engine stores them in the knowledge base for future reference by the reasoning engine. Depending on the application, the learning engine may only be run to train a newly initialized radio, or it could be run periodically as the radio operates.

The following sections describe each of the two engines in more detail.

Reasoning and Planning

The SR exports variables that are either read-only or read-write. The read-only parameters represent statistics maintained by the SR, such as SNR and bit error rate. The read-write variables represent configurable parameters such as transmit power, coding rate, and symbol constellation.

These radio parameters are bound to predicates in the knowledge base. Knowledge bases are very common in AI planning. The one we describe here contains two basic data structures. The first is a logic expression made up of predicates that represents the state of the environment. Predicates are expressions in first-order logic that evaluate to either true or false.

The second set of data contained within the knowledge base is actions. Actions define operations the reasoning engine could perform to change the state of its environment. Actions consist of a set of preconditions and postconditions. Preconditions must be inferable from the knowledge base and evaluate true for the action to be selected. An action's postconditions describe the modified state of the knowledge base.

To better illustrate this discussion, consider the following example, the objective of which is to decrease the modulation rate with a decrease in SNR.

The knowledge base contains the following predicates:[1]

$$modRate(QPSK) \wedge snr(5\text{ dB}) \tag{1}$$

and the following action

action: decreaseModulationRate
precond: $modRate(QPSK) \wedge snr(\leq 8\text{ dB})$ (2)
postcond: $\neg modRate(QPSK) \wedge modRate(BPSK)$

The reasoning engine uses planning, which is a field of AI that works with logic.[2] At any given time, it looks at the current state and determines which actions are executable in that state. All the possible resulting states are then evaluated to see which is optimal, where optimality is determined by an objective function $f_R(\cdot)$.

In our current example, we can successfully infer the preconditions from our knowledge-base. As a result, the **decreaseModulationRate** action is executed and the postconditions are applied to the knowledge-base, resulting in

$$\begin{aligned} KB' &= KB \wedge \text{postcond} \\ &= (modRate(QPSK) \wedge snr(5\text{ dB})) \wedge \\ &\quad (\neg\, modRate(QPSK) \wedge modRate(BPSK)) \\ &= modRate(BPSK) \wedge snr(5\text{ dB}) \end{aligned} \tag{3}$$

Observe how modulation is changed from QPSK to BPSK when the SNR drops below 8 dB. While this example may seem elementary, it provides the fundamentals for reasoning in our cognitive radio.

This example also illustrates the necessity for a learning engine. In complex radios with many inputs and many outputs, a countless number of actions would be necessary to account for all possible radio states. Rather than preprogramming this list of actions, the learning engine should auto-generate these actions based on experience.

Learning

The learning engine is responsible for augmenting the list of actions available to the radio that allow it to adapt to a changing environment.

[1] *Note the commonly used notation in predicate calculus is as follows: logical AND is* ∧, *logical OR is* ∨, *and logical NOT is* ¬.

[2] *We mostly consider scenarios that use forward chaining rather than backward chaining for inferencing, since often we do not have a particular goal state.*

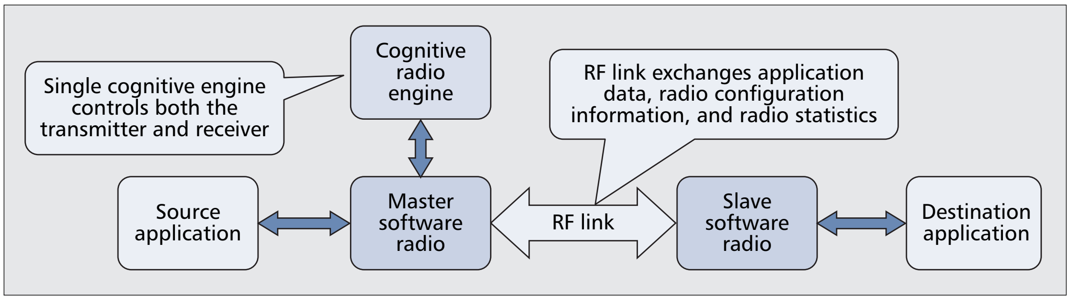

Cognitive radio engine
Single cognitive engine controls both the transmitter and receiver
RF link exchanges application data, radio configuration information, and radio statistics
Source application
Master software radio
RF link
Slave software radio
Destination application

Figure 2. *Configuration of a basic simplex cognitive radio communications link.*

Evolutionary techniques such as genetic algorithms could evolve the radio's state in order to maximize the objective function. After each state change, the resulting state will be evaluated. This process continues until a globally optimal state is found.

Many different learning algorithms are available to a cognitive radio, including hidden Markov models [3], neural networks [4], and genetic algorithms [5].

Nearly every learning technique involves the use of an objective function to determine the value of the learned data. In a cognitive radio, these objective functions reflect the overall goal of the application, such as maximizing channel capacity. The goal of the learning engine is to determine which input state will optimize the objective function. However, unlike the objective function used by the reasoning engine, there is no simple mathematical relationship between the system inputs and the objective function.

In order for learning to be necessary, the precise effects of the changing inputs on the outputs must not be known. This is often the case in non-ideal radio channels. For example, we typically know that decreasing coding rate will also decrease bit error rate, but we do not necessarily know by how much for an arbitrary communications channel. Using a learning engine, we can estimate our channel statistics; then, using that data, we can make decisions that are optimal for our current radio frequency (RF) environment.

More formally, we define the radio's state-space $\mathcal{S}$. A particular state $s \in \mathcal{S}$ is made up of both the radio's input predicates i and output predicates o. That is, $s = i \wedge o$.

We have an objective function $f_L: \mathcal{S} \to \mathbb{R}$ that can be evaluated on a particular state $s \in \mathcal{S}$ and returns a real number. State s_1 is preferable to state s_2 if $f_L(s_1) > f_L(s_2)$.

The learning algorithm will try to precisely characterize $f_L(\cdot)$ in an effort to find an optimal state. For example, a neural network using unsupervised learning could try to find statistical correlations between inputs and outputs to come up with a mathematical representation of $f_L(\cdot)$.

Similarly, evolutionary techniques such as genetic algorithms could evolve the radio's state in order to maximize the objective function. After each state change, the resulting state is evaluated. This process continues until a globally optimal state is found.

Let us assume that the learning algorithm evaluated N different inputs $i_1, i_2, \ldots, i_N$ with resulting outputs $o_1, o_2, \ldots, o_N$ before finding the optimal one. This implies that

$$f_L(i_N \wedge o_N) \geq f_L(i_n \wedge o_n)\ \forall 1 \leq n \leq N-1. \quad (4)$$

Thus, we now know that if our radio is ever in state $i_1 \wedge o_1, \ldots, i_{N-1} \wedge o_{N-1}$, the optimal behavior is to set the radio inputs to i_N. This equates to our learning engine generating the following actions for the policy engine:

action: learnedActionX
precond: $\forall n \leq N-1\ i_n \wedge o_n$ (5)
postcond: $\neg\ i_n \wedge i_N$.

If the radio environment changes, however, o_N may not result from i_N, and i_N may no longer be optimal. If this occurs, the learning engine should be aware of the changes and remove the suboptimal learned action from the knowledge-base. This can be expressed as

action: unlearnActionX
precond: $i_N \wedge \neg o_N$ (6)
postcond: $\neg$**learnedActionX**.

The cognitive process, both reasoning and learning, is then reinstatiated in search of a new optimal state. The next section describes concrete instantiations of the policy and learning engines for particular applications.

Applications

In this section we describe several applications of learning to real radio problems. For each application, the inputs and outputs of the SR are defined, along with an objective function. We then describe strategies for how a cognitive radio can solve the problem.

All the applications assume the basic communications architecture shown in Fig. 2. This architecture defines a simplex link between a source and a destination SR. The source radio has a cognitive engine that controls the source radio's parameters, and uses the existing link to exchange configuration information (e.g., modulation scheme, frequency, or bandwidth) and statistics (e.g., noise power or bit error rate).

In order for this system to work, we must assume that both the master and slave radio start off with the same initial configuration c_0. When the cognitive radio decides a change to configuration c_{i+1} is necessary, the master SR sends a message containing c_{i+1} to the slave using configuration c_i. A simple example of this would be a frequency hopping radio system where a hop to frequency f_{i+1} is signaled by sending a control message at frequency f_i.

To extend this basic architecture to a duplex link between two SRs, a cognitive radio engine can be added to the second radio. However, here we assume that each cognitive engine acts as an independent agent controlling only outbound data and not communicating with other cognitive radio engines.

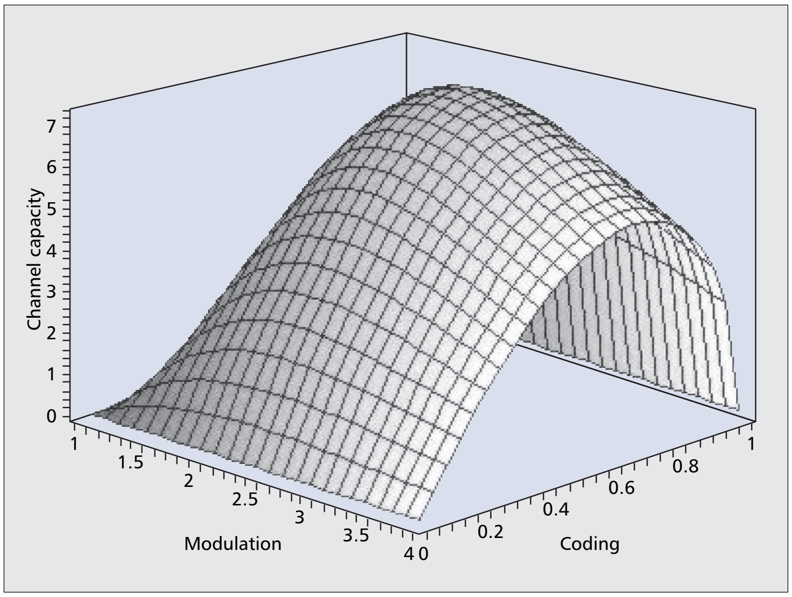


Figure 3. *Sample objective function for a particular RF environment, showing optimality over modulation type 2^{2m}-QAM and coding rates from 0 to 1, interpolated onto a continuous surface.*

To extend this model to a multiparty or ad hoc network, each radio would maintain independent state in its cognitive radio engine for each destination node, as each pairwise directed communications channel could have different properties. Certainly these channels are not completely independent, so some shared state can be used to more quickly find optimal solutions.

Capacity Maximization in an AWGN Channel

One very typical application of cognitive radio is to create a radio that can adapt to a channel to maximize its overall performance. Here we assume a fixed power, frequency, and bandwidth, since these parameters are more commonly thought of as being within the realm of dynamic spectrum access, detailed later.

Thus, here we try to select a modulation type and coding rate to maximize capacity. For an additive white Gaussian noise (AWGN) channel, we can compute channel capacity using the Shannon-Hartley law [6], and our goal is to find a modulation type and coding rate that can get us as close to capacity as possible.

Let us assume we have M modulation types $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_M$ and N coding types $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N$. Modulation type $\mathcal{M}_i$ can transmit d_i data bits per symbol and has a probability of bit error $e_i(S)$ for SNR S. Coding type $\mathcal{C}_j$ has rate r_j and can correct c_j bit errors per block of size b_j bits. For this application we assume an AWGN channel; therefore, $e_i(S)$ can be computed directly [6].

Our goal is to maximize capacity, so our objective function should reflect the capacity of our channel. For an arbitrary input configuration $\mathcal{M}_i \wedge \mathcal{C}_j$, we can compute channel capacity $C_{i,j}$ by computing the raw data rate multiplied by the probability of not receiving c_{j+1} or more bit errors. This results in a suitable objective function $f_R(\cdot)$.

Thus, our goal is to find values $\mathcal{M}_i$ and $\mathcal{C}_j$ that maximize the objective function subject to our environment's SNR S.

A key thing to notice here is that our radio output predicate $snr(S)$ is independent of our inputs, and as a result, a pure policy-based radio can be used. The learning engine is not necessary. Given the above-defined objective function, the cognitive radio's reasoning engine can determine the optimal radio operating parameters.

A knowledge base for maximizing channel capacity can be composed with several basic actions. The first can be used to set the modulation type.

action: switchModType(A,B)
precond: *modType*(A) (7)
postcond: $\neg$*modType*(A) $\wedge$ *modType*(B).

The second can be used to set the coding type.

action: switchCodeType(A,B)
precond: *codeType*(A) (8)
postcond: $\neg$*codeType*(A) $\wedge$ *codeType*(B).

The reasoning engine will start with the initial state and derive all possible resulting states achievable with the above two actions. For all resulting states, the objective function will be evaluated, and the radio will then execute the actions that lead to that state. In this section our key assumption is an AWGN channel where $e_i(S)$ can be derived for every possible modulation scheme. In the next section we look at a channel where $e_i(S)$ is not known, and learning algorithms must be used.

Capacity Maximization in a Non-AWGN Channel

In the previous section we examined the problem of capacity maximization for an AWGN channel. We demonstrated that a cognitive radio can perform this without using a learning engine since the radio outputs were independent of the radio inputs. In this section we consider a non-AWGN channel with fading non-Gaussian noise and interference. In this channel we cannot know how any given modulation and coding scheme will perform without first trying it.

In order to avoid a brute force search, we must assume a continuous convex solution space. To achieve this, there is a logical ordering of modulation types and coding types that results in a continuous convex objective function, such that if we evaluate the objective function for modulation type $\mathcal{M}_i$ and find it to be less than $\mathcal{M}_{i+1}$, the objective function evaluated for $\mathcal{M}_{i-1}$ must be less than $\mathcal{M}_i$.

This will allow the use of an algorithm that performs a gradient search to find the objective function maxima. If the solution space is not continuous and convex, it does not mean we cannot find a solution, only that we cannot find it as quickly.

In this application our objective function will depend heavily on our outputs. Assuming our slave SR can measure the corrected bit success rate coming out of the decoder, the resulting capacity is the product of our bit success rate, coding rate, and raw modulation data rate. As before, we can use this as our objective function $f_L(\cdot)$.

As an example, a sample objective function has been plotted in Fig. 3 for various modulation types and coding rates. Since it is convex, hill climbing will efficiently yield an optimal configuration.

Dynamic Spectrum Access

Dynamic spectrum access (DSA) involves locating frequency bands and times when a cognitive radio can transmit without causing harmful interference to other transceivers [7]. For example, consider a cognitive radio network operating in the UHF television bands, where transmission is permissible provided devices can guarantee they will not interfere with licensed broadcasts.

More concretely, the goal is to locate center frequencies, bandwidths, and times when a cognitive radio can transmit, while maximizing capacity and minimizing interference. This type of problem is very different from capacity maximization where we wanted to learn how our radio inputs affected our radio outputs. DSA involves learning where and when *other* radios will be transmitting.

If we assume all signals with which we are trying to coexist are continuous in time, the problem simplifies to one solvable by a policy-based radio. Imagine our SR exports predicates to the knowledge base regarding detected signals $s_1, s_2, \ldots, s_N$ that are of the form

$$signalFreq(s_i, f_i) \wedge signalBW(s_i, W_i). \quad (9)$$

Our goal is to find some f_c and W that do not overlap any detected signal, while maximizing W and consequently the radio's capacity.

First, let us assume we have a helper function notOverlap(f_c, W, s_i) that evaluates to true if the band occupied by a signal centered at f_c with bandwidth W does not overlap signal s_i. Then we can define our predicate,

action: moveBand($\mathbf{f_{old}}$, $\mathbf{W_{old}}$, $\mathbf{f_{new}}$,$\mathbf{W_{new}}$)
precond: $\forall i \leq N \; notOverlap(f_{new}, W_{new}, s_i)$
$(centerFreq(\text{f\{old\}}) \wedge bandwidth(W_{old}))$
postcond: $\neg(centerFreq(f_{old}) \wedge bandwidth(W_{old}))$
$\wedge (centerFreq(f_{new}) \wedge bandwidth(W_{new})).$
(10)

Then we define our objective function,

$$f_R(centerFreq(f_c) \wedge bandwidth(W)) = \text{W}. \quad (11)$$

We now have a policy-based cognitive radio that will search out the largest continuous piece of bandwidth for communication.

In more complex noise and interference environments, this approach is too simplistic, as different frequency bands may have different noise floors. Extension to this environment will require the use of a learning engine. Imagine our SR exports predicate $snr(S)$ for the currently tuned center frequency and bandwidth. This gives us more information, and we can define a better objective function based on the Shannon-Hartley law:

$$f_L(centerFreq(f_c) \wedge bandwidth(W) \wedge snr(S)) \quad (12)$$
$$= W \log_2(1 + S).$$

This learning objective function can only be evaluated when S is known, and this can only be computed after the radio has been tuned to center frequency f_c.

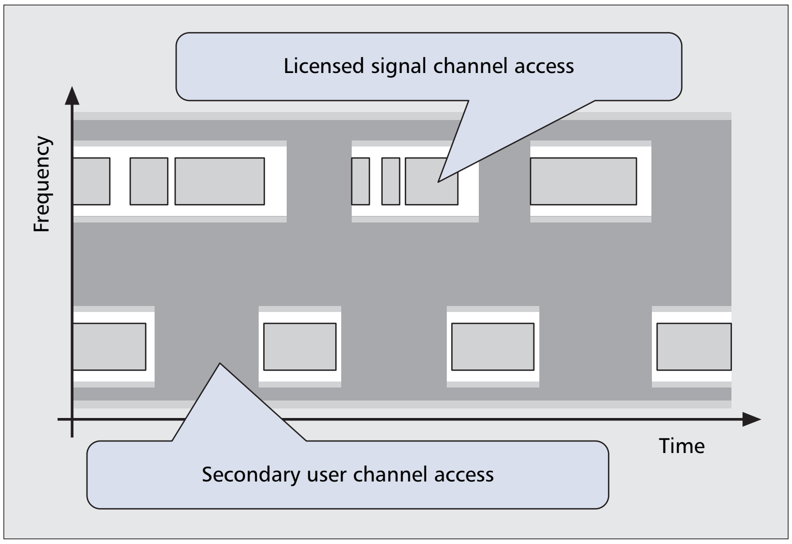


Figure 4. *Example showing cognitive radio coexisting with bursty signals in both frequency and time.*

Another interesting extension to this idea is to consider non-continuous signals, such as ones using time-division multiple access. For these situations, we can coexist not only in frequency but also in time. Using various cyclostationary statistical learning algorithms [8], we can compute the expected length of channel vacancy and incorporate that into our decision making.

Imagine a neural network has computed for us the probability $P_i(t)$ that signal s_i is transmitting at time t, where the probability is periodic over time τ_i. Our learning engine can then export to the knowledge base predicates describing this signal.

$$signalFreq(s_i, f_i) \wedge signalBW(s_i, W_i)$$
$$\wedge (\forall t < \tau, P_i(t) > \alpha \; signalON(s_i, t) \quad (13)$$
$$\wedge (\forall t < \tau, P_i(t) < \alpha \; signalOFF(s_i, t).$$

A threshold α is used to indicate that the signal is present with very low probability. We can now extend our notOverlap function to take into consideration whether s_i is transmitting at time t, and our radio can compute its optimal course of action.

action: moveBand($\mathbf{f_{old}}$, $\mathbf{W_{old}}$, $\mathbf{f_{new}}$, $\mathbf{W_{new}}$)
precond: $\forall i \leq N : notOverlap(f_{new}, W_{new}, s_i, t)$
postcond: $\neg(centerFreq(f_{old}) \wedge bandwidth(W_{old}))$
$\wedge (centerFreq(f_{new}) \wedge bandwidth(W_{new}))$
(14)

We can use the same objective function as before, or incorperate SNR if necessary.

Implementation

To build on the ideas developed in this article, we have implemented a cognitive radio based on a generic cognitive engine [9]. It combines OSSIE, which is an open source software communications architecture (SCA) core framework implementation from Virginia Tech, with the Soar cognitive engine from the University of Michigan.

Within it we have implemented both the policy-based capacity maximization and the learning-

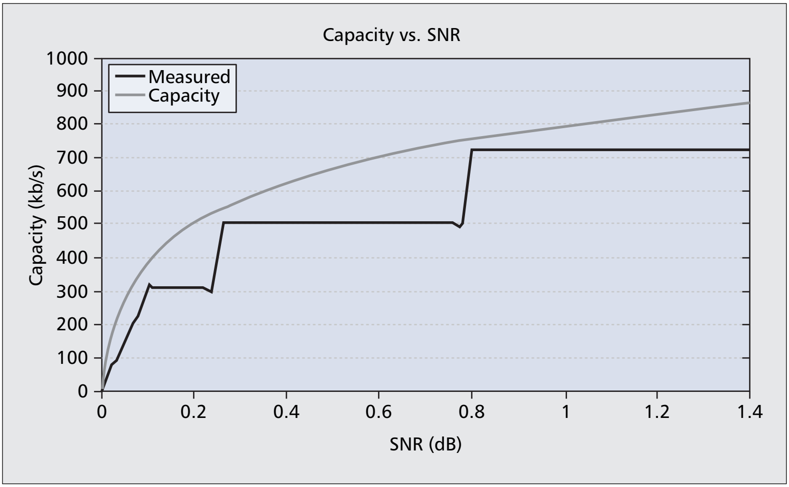


Figure 5. *Theoretical and achieved capacity as a function of signal to noise ratio.*

based capacity maximization. The radio achieves the optimal configuration of modulation and coding in an environment with time-varying noise power (Fig. 5). One of the biggest factors to consider in an implementation is adaptation time. The more *intelligence* you put behind the decision making process, generally the slower it is. If your cognitive engine is constantly running, searching for better radio configurations, it can utilize a significant portion of your processor time. Our policy radio adapts on the order of tens of milliseconds, while our learning radio adapts on the order of hundreds of milliseconds. We are currently working to implement dynamic spectrum access and adaptation to interference and fading channels within OSCR.

Conclusion

Since the introduction of cognitive radio in 1999 [10], there have been many high-level discussions on proposed capabilities of cognitive radios. In this article we have tried to formalize some of the architecture behind these ideas and the applications for which they are most suited, and give some insight into the differences between reasoning and learning.

Certainly there is a great deal of future work in the field of cognitive radio, and in particular applications of machine learning to cognitive radio. The architecture described here is flexible enough to address many different applications provided they can be expressed in predicates, actions, and objective functions. The real work will be mapping potential applications to predicate calculus.

References

[1] IEEE 802.11, "Working Group on Wireless Local Area Networks," http: //www.ieee802.org/11/
[2] T. Rondeau *et al.*, "Cognitive Radios with Genetic Algorithms Intelligent Control of Software Defined Radios," *SDR Forum Conf. 2004*.
[3] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, 1989.
[4] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1998.
[5] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989.
[6] J. Wozencraft and I. Jacobs, *Principles of Communication Engineering*, Waveland Press, 1990.
[7] P. Leaves *et al.*, "Dynamic Spectrum Allocation in a Multiradio Environment Concept and Algorithm," *Conf. 3G Mobile Commun. Technologies 2001*.
[8] T. Clancy and B. Walker, "Predictive Dynamic Spectrum Access," *SDR Forum Conf. 2006*.
[9] E. Stuntebeck *et al.*, "Architecture for an Open-Source Cognitive Radio," *SDR Forum Conf. 2006*.
[10] J. Mitola, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*, Ph.D. dissertation, KTH, 2000.

Biographies

CHARLES CLANCY (clancy@ltsnet.net) is a senior researcher with LTS and adjunct professor at the University of Maryland. He recieved his M.S. in electrical engineering from the University of Illinois and Ph.D. in computer science from the University of Maryland. His research interests include next-generation wireless networks and wireless security.

JOE HECKER (heckerj@clemson.edu) is an engineer with SAIC. He received his M.S. in electrical engineering from Clemson University. His research interests include artificial intelligence and pattern recognition.

ERICH STUNTEBECK (eps@gatech.edu) is a Ph.D. candidate in the Electrical and Computer Engineering Department at Georgia Tech. His research interests include wireless sensor networks.

TIM O'SHEA (tjoshea@ncsu.edu) is a graduate student at North Carolina State, pursuing an M.S. in electrical engineering. His research interests include software-defined radio implementation and spectrum sensing.