

Security Threats to Signal Classifiers Using Self-Organizing Maps

T. Charles Clancy
tcc@umd.edu

Awais Khawar
awais@umd.edu

Department of Electrical and Computer Engineering
University of Maryland, College Park

Abstract—Spectrum sensing is required for many cognitive radio applications, including spectral awareness, interoperability, and dynamic spectrum access. Previous work has demonstrated the ill effects of primary user emulation attacks, and pointed out specific vulnerabilities in spectrum sensing that uses feature-based classifiers. This paper looks specifically at the use of unsupervised learning in signal classifiers, and attacks against self-organizing maps. By temporarily manipulating their signals, attackers can cause other secondary users to permanently misclassify them as primary users, giving them complete access to the spectrum.

In the paper we develop the theory behind manipulating the decision regions in a neural network using self-organizing maps. We then demonstrate through simulation the ability for an attacker to formulate the necessary input signals to execute the attack. Lastly we provide recommendations to mitigate the efficacy of this type of attack.

I. INTRODUCTION

A number of new cognitive radio technologies rely on signal classification. Its use is required when a radio system needs to differentiate different types of transmitters in a particular frequency band. Radios may need to do this for spectral situational awareness, interoperability, or dynamic spectrum access (DSA). For DSA, signal classification techniques typically combine signal processing and pattern matching to allow the secondary user to authenticate a primary user signal. This can be done by extracting specific features of the signal through a variety of possible signal processing techniques and then matching these characteristics to a pattern of a known primary user. These features can range from the spectral shape of the primary users signal to high order cyclostationary features of the signal [1].

A major area of study has been the use of neural networks for classifying features extracted from signals [2], [3], [4], [5]. Recently research has begun on pointing out key security issues related to the use of machine learning in cognitive radio networks [6], [7], specifically related to attacks to spectrum sensing [8], [9], [10], and attacks that can fool signal classifiers [11].

This paper focuses on the use of unsupervised learning in feature-based signal classification. Unsupervised learning is a very powerful tool in building cognitive radios that require minimal preconfiguration. These radios can learn from the ground up the properties of other devices in their environment. In the context of neural networks, self-organizing maps are often used for unsupervised learning [12]. In this paper we delve

into the use of self-organizing maps for signal classification, and then present scenarios whereby the output classes of the neural network can be manipulated by an attacker. For DSA systems, the result is that attacker signals are misclassified as primary users by other devices in the environment, without the attackers needing to mimic the spectral properties of primary users. The attackers would then have unrivaled access to the spectrum as other secondary users moved to other bands.

The remainder of this paper is organized as follows. Section 2 introduces unsupervised signal classification using self-organizing maps. Section 3 develops specific attacks against these maps. Section 4 simulates attacks for specific waveforms. Section 5 describes mitigation techniques and concludes.

II. CLASSIFICATION USING SELF-ORGANIZING MAPS

In this section we present a basic model for signal classification useful for dynamic spectrum access. We describe the fundamentals of feature-based signal classification, and detail an unsupervised-learning approach that uses self-organizing maps.

A. Feature-Based Signal Classification

Consider a system of machine learning where a series of signal values $x_n(t)$ are presented to a signal classifier. The goal of the signal classifier is to determine whether $x_n(t)$ is a primary user P or a secondary users S . The correct class \hat{C}_n can be selected by maximizing the following conditional probability:

$$\hat{C}_n = \operatorname{argmax}_{C \in \{P, S\}} P(C|x_n(t)) \quad (1)$$

Using Bayes Rule, this can be rewritten as

$$\hat{C}_n = \operatorname{argmax}_{C \in \{P, S\}} \frac{P(x_n(t)|C)P(C)}{P(x_n(t))} \quad (2)$$

The value of $P(x_n(t))$ is not part of the maximization, and therefore can be removed, resulting in:

$$\hat{C}_n = \operatorname{argmax}_{C \in \{P, S\}} P(x_n(t)|C)P(C) \quad (3)$$

The *a priori* probability $P(C)$ can be computed using prior knowledge of the breakdown between primary and secondary users in a particular frequency band.

The *a posteriori* probability $P(x_n(t)|C)$ can be computed by first projecting $x_n(t)$ into a features space using transform

$F : \mathbb{C}^\infty \rightarrow \mathbb{R}^N$. This reduces the dimensionality of the problem by examining specific features of $x_n(t)$ rather than $x_n(t)$ itself. We then use a classification engine to compute likelihoods $L_C(\cdot)$ of the various classes. Specifically,

$$\begin{aligned} L_C(F(x_n(t))) &\propto P(F(x_n(t))|C) \\ &\approx P(x_n(t)|C) \end{aligned} \quad (4)$$

In unsupervised learning, we feed a series of feature vectors $F_n = F(x_n(t))$ into our classification engine, and it then outputs class P or S based on its acquired knowledge. At no point do we provide any annotated training data to the classifier that helps it make decisions about which points belong to which class. Two key modes of unsupervised learning are K -means clustering [13] and self-organizing maps [12].

B. Self-Organizing Maps

Self-organizing maps are a type of neural network where individual weights are evolved to fit the input data. Imagine neurons n_i located on a lattice in a 1- or 2-dimensional space, called *map space*. Let n_i represent the location in that space of the neuron. Each neuron has an associated weight vector $w_i \in \mathbb{R}^N$ which is a point in *weight space*. These neurons serve to map points from the N -dimensional weight space to the low-dimensional map space.

For each input feature vector F_n , the first step in classification is to select the neuron with closest weight vector. In particular,

$$\hat{j} = \underset{j}{\operatorname{argmin}} \|F_n - w_j\|_2 \quad (5)$$

We then update all weight vectors of all neurons, where the magnitude of the update is a function of the distance between the neuron and $n_{\hat{j}}$:

$$w_i \leftarrow w_i + \eta_n a_{i,\hat{j}} (F_n - w_i) \quad (6)$$

where $a_{i,\hat{j}}$ is an activation metric based either Euclidean or link distance (e.g. $a_{ii} = 1$, $a_{ij} = 0.5$ if i and j are neighbors, and otherwise $a_{ij} = 0$), and η_n is an exponentially-increasing time metric for some time-constant τ :

$$\eta_n = \eta_0 e^{n/\tau} \quad (7)$$

The definition of $a_{i,\hat{j}}$ means that weight vector updates will most influence the selected neuron and its neighbors. The definition of η_n means that the more samples fed into the system, the less the system will update its weight vectors. This damping effect allows convergence. The values η_0 and τ affect this convergence rate.

Initial values for weight vectors w_i can be selected randomly using a uniform distribution, a rough approximation of the weight space probability distribution, or they can be selected based on initial feature vectors. In particular, for this last approach assume feature vectors F_1, F_2, \dots, F_n are known, where $n \geq N$. We can place these into a matrix \mathcal{F} as follows:

$$\mathcal{F} = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{bmatrix} \quad (8)$$

From \mathcal{F} we can compute its eigenvalues $\lambda_1, \dots, \lambda_N$ and eigenvectors e_1, \dots, e_N . Assuming neurons are in a 2-dimensional space, and given λ_j and λ_k are the two largest eigenvalues, e_j and e_k are the principle eigenvectors that span a 2-dimensional space best fit for the original N -dimensional data. For lattice point n_i we can compute its initial weight vector w_i by taking n_i 's 2-dimensional coordinate on the plane spanned by e_j and e_k and projecting it into the higher-dimensional \mathbb{R}^N space.

If the two signal classes P and S have sufficiently separable signatures in the weight space \mathbb{R}^N , the weight vector of map neurons in cluster together after . By looking at weight vector densities across the lattice, the clusters can be identified, and decision boundaries can be placed between them. For each new signal received, the map can be used to classify it, while simultaneously updating itself with the new information.

Figure 1 shows examples of self-organizing maps applied to a feature set in \mathbb{R}^3 with three statistically distinct classes. Figure 1(a) shows the convergence of the neuron weights to the location of the input data points. The lines between points indicate that the neurons associated with those weight vectors are neighbors in map space. Figure 1(b) plots in map space how close a neuron is to its neighbors in weight space. We immediately notice three areas of higher density, separated by areas of lower density. This representation can be used to form boundary regions between classes.

III. THREATS TO SELF-ORGANIZING MAPS

In this section we consider a generic self-organizing map whose goal is to distinguish between two classes. First we derive analytically the feasible set of input signals that would inductively manipulate the decision regions, and then we show a more complex example through simulation.

A. Analytical Derivation

Here we consider, for simplicity, a 1-dimensional map. After $n - 1$ iterations from previous input data, the weight vectors for neurons n_i are w_i , respectively.

Our goal is to create an input vector $x_n(t)$ that will cause a neuron currently on the border between two classes to switch classes. Applying this technique inductively, an attacker can arbitrarily shift the decision region between primary and secondary users, causing more signals to be classified as primary, decreasing competition for spectral resources.

Let's assume neurons n_1, \dots, n_ℓ are linearly arranged and evenly spaced, and without loss of generality, $n_i = i$. Assume nodes n_1, \dots, n_{i-1} are classified as primary users, and nodes n_i, \dots, n_ℓ are classified as secondary users. Our goal is to cause n_i to be classified as a primary user.

To accomplish this, the input signal must have a feature vector closer to w_i than any other weight vector. This introduces our first constraint:

$$\|F(x_n(t)) - w_i\|_2 < \|F(x_n(t)) - w_j\|_2 \quad \forall j \neq i \quad (9)$$

In other words, $x_n(t)$ must have a feature vector that causes n_i to be the winning neuron. Let \mathcal{X}_1 be the space of feasible vectors $x_n(t)$ that satisfy this space.

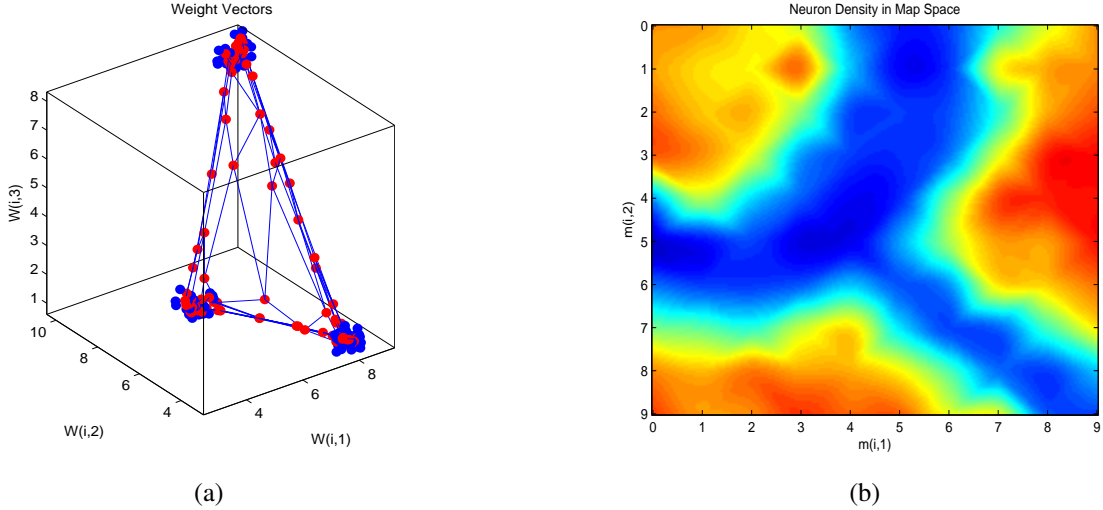


Fig. 1. (a) Weight vectors for a self-organizing map, demonstrating convergence to unannotated input data made up of three distinct classes; (b) Densities of neurons in weight space, plotted as a function of their location in map space

Next, the map will update the weight w_i , and the goal is for this update δ_i to cause neuron n_i to shift from being classified as secondary to primary. However this update also affects n_i 's neighbors. For n_i and its neighbors, the update is defined as

$$\delta_j = \eta_n a_{j,i} (F(x_n(t)) - w_j) \quad (10)$$

Assuming activation values of 1 for the winning neuron, 0.5 for its neighbors, and 0 for all other neurons, we have the following:

$$\begin{aligned} \delta_{i-1} &= 0.5 \eta_n (F(x_n(t)) - w_{i-1}) \\ \delta_i &= \eta_n (F(x_n(t)) - w_i) \\ \delta_{i+1} &= 0.5 \eta_n (F(x_n(t)) - w_{i+1}) \end{aligned} \quad (11)$$

If n_i is on the decision boundary and currently in the same class as n_{i+1} , it must be closer in weight space to n_{i+1} than n_{i-1} . To change this behavior, we simply need the new weight vector to be closer to n_{i-1} . Quantitatively, this means:

$$\|(w_i + \delta_i) - (w_{i-1} + \delta_{i-1})\|_2 < \|(w_i + \delta_i) - (w_{i+1} + \delta_{i+1})\|_2 \quad (12)$$

Substituting (11) into (12), we obtain the following:

$$\begin{aligned} \|(1 - \eta_n)w_i - (1 - 0.5 \eta_n)w_{i-1} + 0.5 \eta_n F(x_n(t))\|_2 < \\ \|(1 - \eta_n)w_i - (1 - 0.5 \eta_n)w_{i+1} + 0.5 \eta_n F(x_n(t))\|_2 \end{aligned} \quad (13)$$

This is our second constraint. Let \mathcal{X}_2 be the feasible set of signals $x_n(t)$ that satisfy this constraint. If $\mathcal{X}_1 \cap \mathcal{X}_2 \neq \emptyset$, then a signal exists that will alter the decision boundaries.

B. Experimental Example

In this section we demonstrate the efficacy of decision boundary movement for a self-organizing map with a two-dimensional map space. The simulation uses the MATLAB Neural Network Toolbox to implement the self-organizing maps.

We create a network with a 2-dimensional map space and 3-dimensional weight space. The weight space could be arbitrarily large, but for the purposes of visualization, 3 dimensions were used. The weight space spans values $[0, 10]$ in each dimension. Input data samples are taken from three Gaussian probability distributions with means $\mu_1 = (3, 3, 3)$, $\mu_2 = (7, 7, 7)$, and $\mu_3 = (5, 8, 7)$. The standard deviation for all dimensions is 0.25. The input points are shown as blue dots in Figure 2(a). The network is trained to the input samples and neuron weights are shown in red. The associated neuron densities are depicted in Figure 2(b). There is a clear decision boundary in dark blue that separates the primary from secondary users.

In a second trial, we have the same input points, but we also add chaff points. Chaff points have means collinear with μ_1 and μ_3 , at random points in between the two. Their goal is to confuse the classifier and cause points with mean μ_3 to be in the same output class as points with mean μ_1 , rather than μ_2 .

Figure 2(c) shows the input points complete with chaff points, and the trained network of neurons. Notice that neurons are spread evenly along the chaff points, with few neurons in between. In Figure 2(d), we can see the densities have shifted significantly. There is a more homogeneous density across the area where chaff points were added, and the decision boundary between the two output classes has shifted such that points with mean μ_2 are now output as one class, and everyone is output as another.

This simple example demonstrates that given the ability to supply points with known feature values, the decision boundaries in a self-organizing map can be manipulated. In the next section we analyze this using features from signal data.

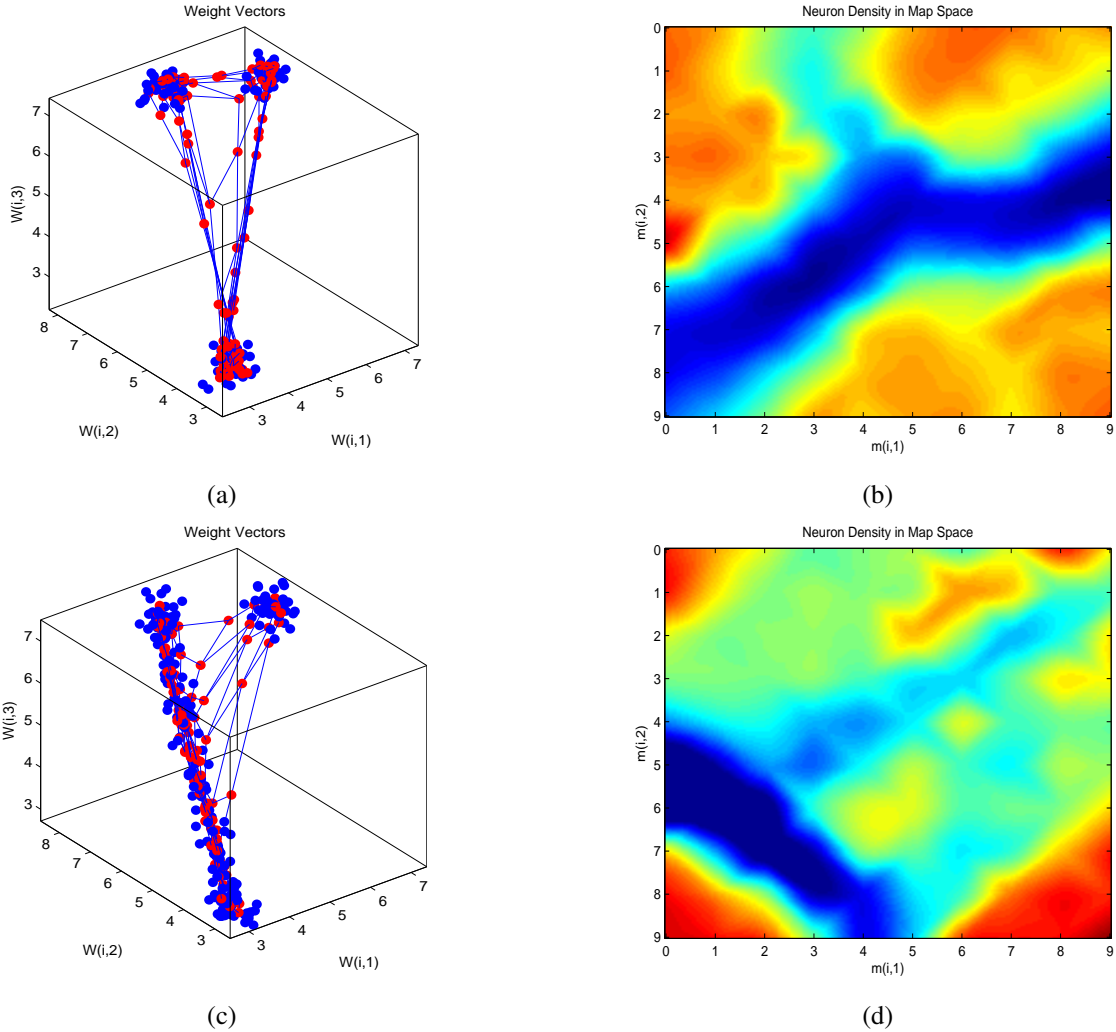


Fig. 2. (a) neuron structure for a self-organizing map with three classes, where two classes are relatively close in weight space; (b) associated node densities used for determining decision regions, note the obvious separation between classes; (c) neuron structure for the same classes after an attacker adds chaff points to manipulate neuron weights; (d) modified decision regions after attack

IV. EXPERIMENTAL RESULTS

In this section we demonstrate the efficacy of the attack on features generated from real signal data. Our scenario is that secondary users wish to distinguish the analog primary signal from the digital secondary signals.

We assume the primary user is an frequency-modulated analog signal, secondary users are binary phase-shift keying (BPSK) and adversarial users are 16-point quadrature amplitude modulation (16QAM). Representative power spectra for these signals is shown in Figure 3.

While many different types of feature extractors are possible, we selected three that were simple to compute and could be implemented as simple FIR filters. The first feature is the standard deviation of the time-domain signal itself. The second feature is the standard deviation of the time-averaged time-domain signal, averaged over 10 samples. The third feature is the standard deviation of the derivative of the signal. In the end we compute the standard deviation of signals filtered with

taps $[1]$, $[0.1, 0.1, \dots, 0.1]$, and $[1, -1]$.

For the simulation, we included small, random sampling and carrier frequency offsets of the input signals, consistent with an oscillator stability of 20ppm. As a point of comparison, the USRP and USRPv2 respectively have oscillator stabilities of 50ppm and 2ppm [14]. We also assumed the signal SNR has been normalized to approximately 10 dB, and included a scaling variance of 0.1. These random fluctuations in the input signal increase the variance of the input signal feature distributions in weight space.

Figure 4 shows the self-organizing map that was trained to our signal inputs. The three classes are discernible, though the BPSK and 16QAM classes are close in weight space. In the associated density plot, boundaries are present between all three classes, but the strongest boundary is between the FM signals and the BPSK/16QAM signals. This result is an extension of [3] and demonstrates it is feasible to use neural networks with unsupervised learning and simple features to

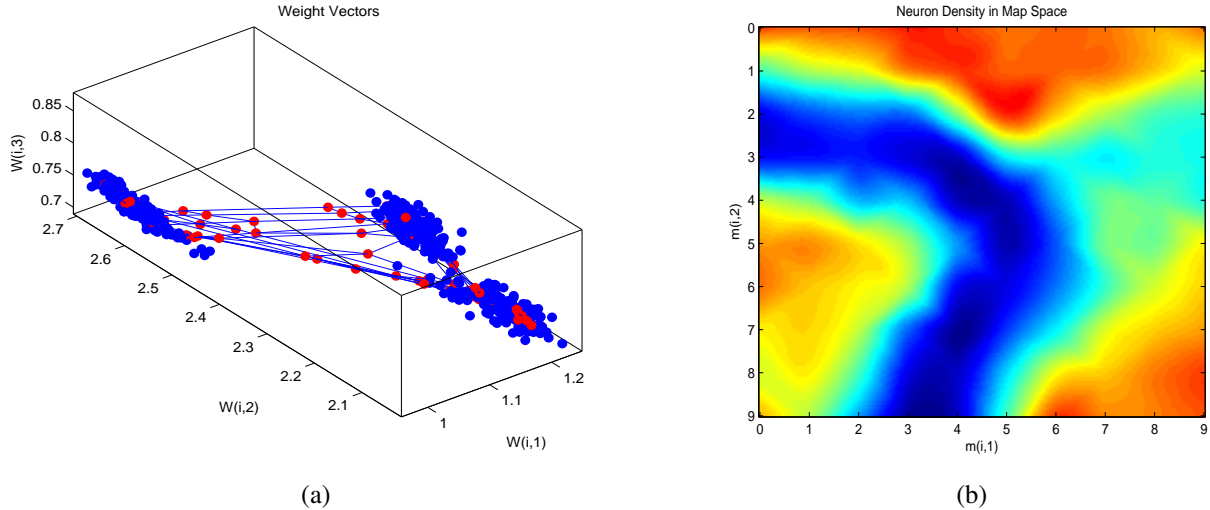


Fig. 4. (a) neuron structure for the three signal types, notice the BPSK and 16QAM classes are close but discernible; (b) associated map densities, showing a clear decision boundary between the primary and secondary users

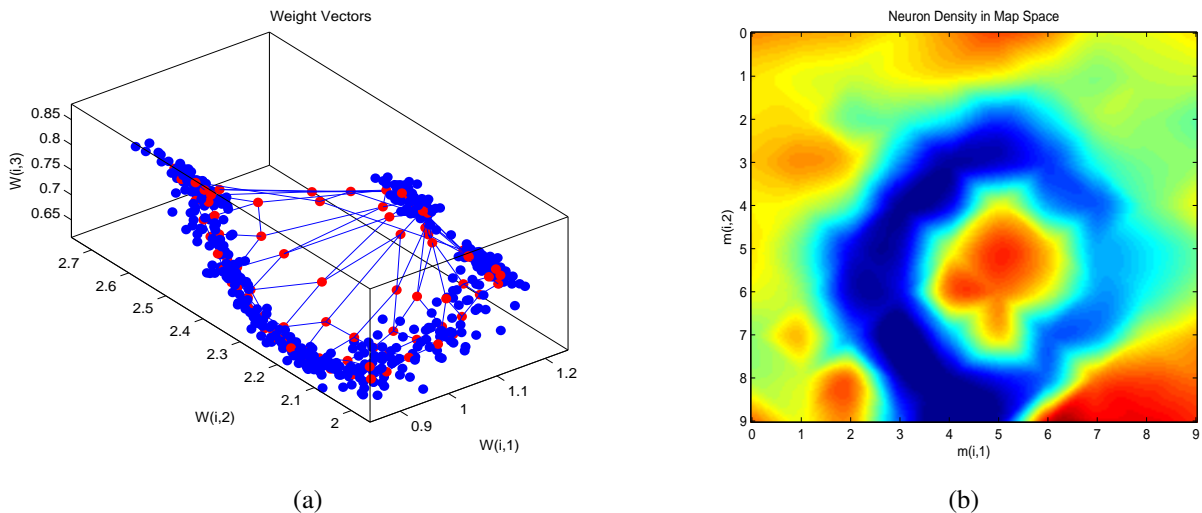


Fig. 5. (a) neuron configuration for the input signals with many input chaff signals used to connect the 16-QAM cluster with the FM cluster; (b) density showing that the PSK cluster has become an island and is now a unique class, distinct from the FM/QAM class

classify signals.

In our next experiment, we demonstrate that the attacker can create input signals to manipulate the neuron structure. By fabricating signals with the appropriate feature vectors, the attacker can cause the 16QAM signals to be lumped in with the FM signals.

Given that with the exception of the computation of standard deviation, our feature transforms are linear, we use the following rough approximation.

$$F(x_1(t) + x_2(t)) \approx F(x_1(t)) + F(x_2(t)) \quad (14)$$

Therefore, if we wish to create chaff points to connect two signal classes, we simply need to take random linear combinations of their time-domain signals. In particular if $x_{FM}(t)$ is the time-domain representation of our primary signal, and $x_{QAM}(t)$ is the time-domain representation of our adversarial

secondary users, then we can create many signals that will generate the appropriate chaff by computing the following for random $\delta \sim \text{unif}(0, 1)$.

$$x_{\text{chaff}}(t) = \delta x_{FM}(t) + (1 - \delta) x_{QAM}(t) \quad (15)$$

Figure 5 shows this attack. Note that due to the non-linearity of the standard deviation computation, the chaff points do not linearly connect the FM and 16-QAM clusters, but they achieve their end goal, none the less. In fact, the round-about nature of the points demonstrates the ability to create trails of chaff points that go around other clusters in feature space.

With this attack, we have demonstrated the ability to manipulate decision boundaries and cause clusters of signals to be misclassified in a deterministic way. This results in a primary user emulation attack where the secondary user need not mimic the primary user's signal.

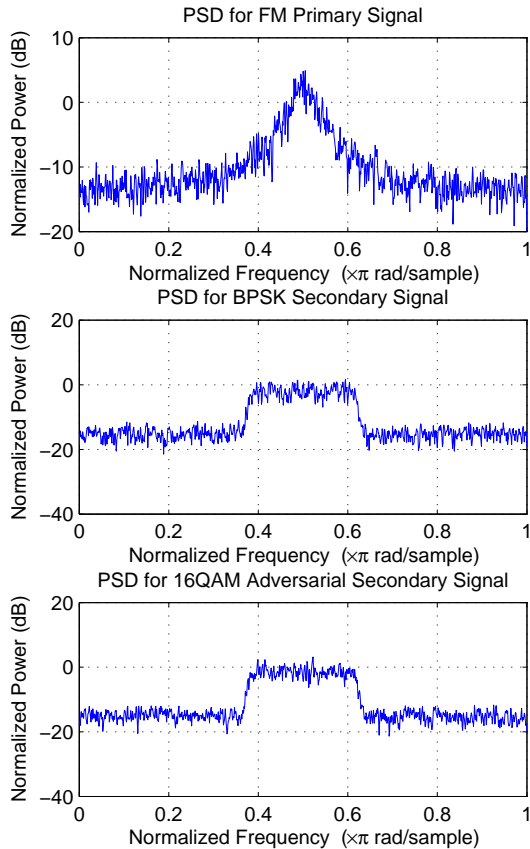


Fig. 3. Power spectral density for the three signal classes in our environment; notice the secondary and adversarial secondary users have a nearly identical power spectrum, and in spite of this the adversarial users will be able to have their signals misclassified as primary users.

V. MITIGATION AND CONCLUSION

Mitigating these types of attacks can be difficult. There are two primary areas of attack: feature extractors and classification engines. In order to defend against adversarial elements in the spectral environment, a cognitive radio needs both robust feature extraction algorithms and a sound classification engine.

While linear features are simple to compute, they are also the easiest for an attacker to manipulate. Features need to balance both ease of computation and ease of manipulation.

As for classification engines, use of completely unsupervised learning can lead to manipulation. As discussed in [6], learning in cognitive radio networks should be constrained. Broad, static principles should govern global optimization, while allowing emergent behavior only in local optimizations. In our case, decision boundaries and neural network structure should consist of both a static component derived during a controlled learning process using data with annotated classes, and a dynamic component that can tailor a system to its current environment through unsupervised learning. By maintaining the static components of the network, the effect an attacker can

have on the overall system performance can be constrained. The balance between static and dynamic neural network components must be carefully selected to balance performance and security.

This work is ongoing, and formulation of concrete mitigation of these attacks, and the evaluation of their efficacy is an open problem. Recent work [15] generalizes the attacks and builds a full signal classification engine out of self-organizing maps and shows how different clustering techniques can be affected by different placements of chaff points.

Overall, we have demonstrated a new form of primary user emulation attack that specifically targets the classification engine in a signal classification system. We show how malicious elements in a network can redraw the decision boundaries between primary and secondary users. Understanding these types of attacks is extremely important to building the next generation of secure, robust communications systems.

REFERENCES

- [1] D. Cabric, S. Mishra, and R. Brodersen, "Implementation issues in spectrum sensing for cognitive radios," in *IEEE Asilomar Conference on Signals, Systems, and Computers (Asilomar'04)*, pp. 772–776, November 2004.
- [2] A. Fehske, J. Gaeddert, and J. Reed, "A new approach to signal classification using signal correlation and neural networks," in *New Frontiers in Dynamic Spectrum Access Networks (DySPAN'05)*, November 2005.
- [3] B. Le, T. Rondeau, D. Maldonado, and C. Bostian, "Modulation identification using neural networks for cognitive radios," in *SDR Forum Technical Conference (SDR'05)*, November 2005.
- [4] T. O'Shea, T. Clancy, and H. Ebeid, "Practical signal detection and classification in gnu radio," in *SDR Forum Technical Conference (SDR'07)*, November 2007.
- [5] A. Wagstaff and N. Merricks, "A subspace-based method for spectrum sensing," in *SDR Forum Technical Conference (SDR'07)*, November 2007.
- [6] T. Clancy and N. Goergen, "Security in cognitive radio networks: Threats and mitigation," in *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (Crowncom'08)*, May 2008.
- [7] J. Burbank, "Security in cognitive radio networks: The required evolution in approaches to wireless network security," in *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (Crowncom'08)*, May 2008.
- [8] T. Brown and A. Sethi, "Potential cognitive radio denial-of-service vulnerabilities and protection countermeasures: A multi-dimensional analysis and assessment," in *International Conference on Cognitive Radio Oriented Wireless Networks and Communications (Crowncom'07)*, pp. 456–464, August 2007.
- [9] R. Chen, J. Park, and J. Reed, "Defense against primary user emulation attacks in cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 25–37, January 2008.
- [10] G. Jakimoski and K. Subbalakshmi, "Denial-of-service attacks on dynamic spectrum access networks," in *IEEE International Conference on Communications Workshops*, pp. 524–528, May 2008.
- [11] T. Newman and T. Clancy, "Security threats to cognitive radio signal classifiers," in *Virginia Tech Wireless Personal Communications Symposium*, June 2009.
- [12] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, pp. 1–6, November 1998.
- [13] J. Hartigan and M. Wong, "A K-means clustering algorithm," *Applied Statistics*, vol. 28, pp. 100–108, 1979.
- [14] M. Ettus, "Building software radio systems: The usrp product family," http://www.ettus.com/downloads/er_broch_trifold_v5b.pdf.
- [15] A. Khawar and T. Clancy, "Signal classifiers using self-organizing maps: Performance and robustness," in *SDR Forum Technical Conference (SDR'09)*, December 2009.