3

Modulation Classification in Cognitive Radio

Adalbery R. Castro, Lilian C. Freitas, Claudomir C. Cardoso,
João C. W. A. Costa and Aldebaro B. R. Klautau
Signal Processing Laboratory (LaPS) and Applied Electromagnetism Laboratory (LEA) – Federal University of Pará (UFPA), Belém – PA
Brazil

1. Introduction

The automatic modulation classification (AMC) problem aims at identifying the modulation scheme of a given communication system with a high probability of success and in a short period of time. AMC has been used for decades in military applications in which friendly signals should be securely transmitted and received, whereas hostile signals must be located, identified and jammed (Gardner, 1988). More recently, the interest for AMC has been renewed by the research on cognitive radios (Haring et al., 2010; Wang & Wang, 2010; Xu et al., 2011), where AMC plays an important role in spectrum sensing (Haykin et al., 2009).

The AMC approaches are typically organized in *likelihood* and *feature-based* methods (Dobre et al., 2007). Alternatively, the AMC methods are distinguished in this chapter by the corresponding learning algorithm: *generative* (also called *informative*) or *discriminative* (Rubinstein & Hastie, 1997). One advantage of this nomenclature is to benefit from the insights accumulated in the machine learning community with respect to the generative and discriminative approaches (Long & Servedio, 2006).

Generative algorithms perform the classification based on probabilistic models that are typically constructed by estimating probability distributions for each class separately. Examples are Naïve Bayes (Tan et al., 2006), hidden Markov models obtained with maximum likelihood estimation (Bremaud, 2010) and methods that uses likelihood ratio tests such as the average likelihood ratio test (ALRT) (Su et al., 2008), generalized likelihood ratio test (GLRT) (Xu et al., 2011) and the hybrid likelihood ratio test (HLRT) (Polydoros, 2000).

Discriminative algorithms are used to learn classifiers that focuses in class boundaries, not on modeling distributions. Examples include support vector machines (SVM) (Cortes & Vapnik, 1995) and neural networks (Krose & van der Smagt, 1996).

There are research of AMC (Dobre et al., 2007; Xu et al., 2011) and spectrum sensing (Haykin et al., 2009; Yucek & Arslan, 2009). Instead of providing an overview of alternatives, this chapter compares two AMC methods: one that uses the well-established features based on cyclostationary analysis (Gardner & Spooner, 1992; Haykin et al., 2009) and the recently proposed *CSS* (concatenated sorted symbols) front end (Muller et al., 2011). The idea is to contrast the characteristics of these methods and emphasize practical aspects. In the experiments, SVM is adopted as the learning algorithm given its performance in many classification tasks. Moreover, the implementation on a field-programmable gate array

(FPGA) of an AMC system using CSS and SVM is discussed and suggested the hardware requirements of an AMC module.

2. The modulation classification problem

An AMC system consists of a *front end* and a *back end* or *classifier*. The *front end* converts the received signal $r(t)$ to a vector $\mathbf{x}[k], k = 1, \ldots, N$ composed of N elements. Having $\mathbf{x}[k]$ as input, the *classifier* decides the class $y \in \{1, \ldots, C\}$ among C pre-determined modulation schemes. The process is depicted in the diagram below:

$$r(t) \text{ (signal)} \rightarrow \boxed{\text{front end}} \rightarrow \mathbf{x}[k] \text{ (parameters)} \rightarrow \boxed{\text{classifier}} \rightarrow y \text{ (class)}$$

There are several options to implement the front end (Mishali & Eldar, 2011), but in order to be concrete, the following alternative will be assumed:

$$r(t) \rightarrow \boxed{\text{FILT/DOWN}} \rightarrow c(t) \rightarrow \boxed{\text{A/D}} \rightarrow c[n] \rightarrow \boxed{\text{DSP}} \rightarrow s[n] \rightarrow \boxed{\text{parameter extraction}} \rightarrow \mathbf{x}[k]$$

where FILT/DOWN denotes operations such as filtering, down-conversion and signal conditioning, $c(t)$ is the input to the analog-to-digital (A/D) converter, $s[n]$ is obtained from $c[n]$ via digital signal processing (DSP) and the last block converts the time-domain signal $s[n]$ into the parameters of interest.

A classifier in a likelihood-based AMC (Su et al., 2008) assumes that $s[n]$ is a sequence of N received *symbols* (Proakis, 2001) and $\mathbf{x}[k] = s[n]$. Hence, these classifiers are considered here to be a special case of feature-based classifiers. The adoption of symbols as features is restricted to the case where the modulation is digital and linear, such as QAM, PSK, etc. (Proakis, 2001). Hence, the likelihood-based AMC can be implemented as

$$r(t) \rightarrow \boxed{\text{FILT/DOWN}} \rightarrow c(t) \rightarrow \boxed{\text{A/D}} \rightarrow c[n] \rightarrow \boxed{\text{DSP}} \rightarrow \mathbf{x}[k],$$

where $c[n]$ is the *complex envelope* (Proakis, 2001). In this case, the DSP block samples the complex envelope according to the signaling rate. This kind of model is widely adopted in the AMC literature (Mak et al., 2007).

In practice, several impairments must be mitigated to have $\mathbf{x}[k]$ as a reasonable approximation of the transmitted symbols, such as lack of synchronism, carrier frequency offset and channel noise. The front end that perfectly recovers the transmitted symbols in a digital modulation is called here *canonical*. Several works assume such front end and then contaminate $\mathbf{x}[k]$ by additive white Gaussian noise (AWGN) and other impairments. Alternatively, a front end based on cyclostationarity can extract features other than the symbols. This chapter focuses on the two distinct front ends for AMC and one classification technique, which are discussed in the sequel.

2.1 Front end: CSS

The CSS front end has been recently proposed by Muller et al. (2011) and uses the symbols of the constellations (Proakis, 2001) as input parameters for the classifier. The CSS front end takes the magnitude and the phase of the received symbols, normalizes them and sorts them

separately. The two ordered vectors (magnitude and phase) are concatenated, generating a new vector with length $D = 2N$, which should reflect an individual *signature* of the corresponding constellation. Recall that N is the number of symbols.

For example, Fig. 1(a) and Fig. 1(b) represent the constellations of noise-free (ideal) 16QAM and 8PSK modulations, respectively. An example of two possible vectors of parameters

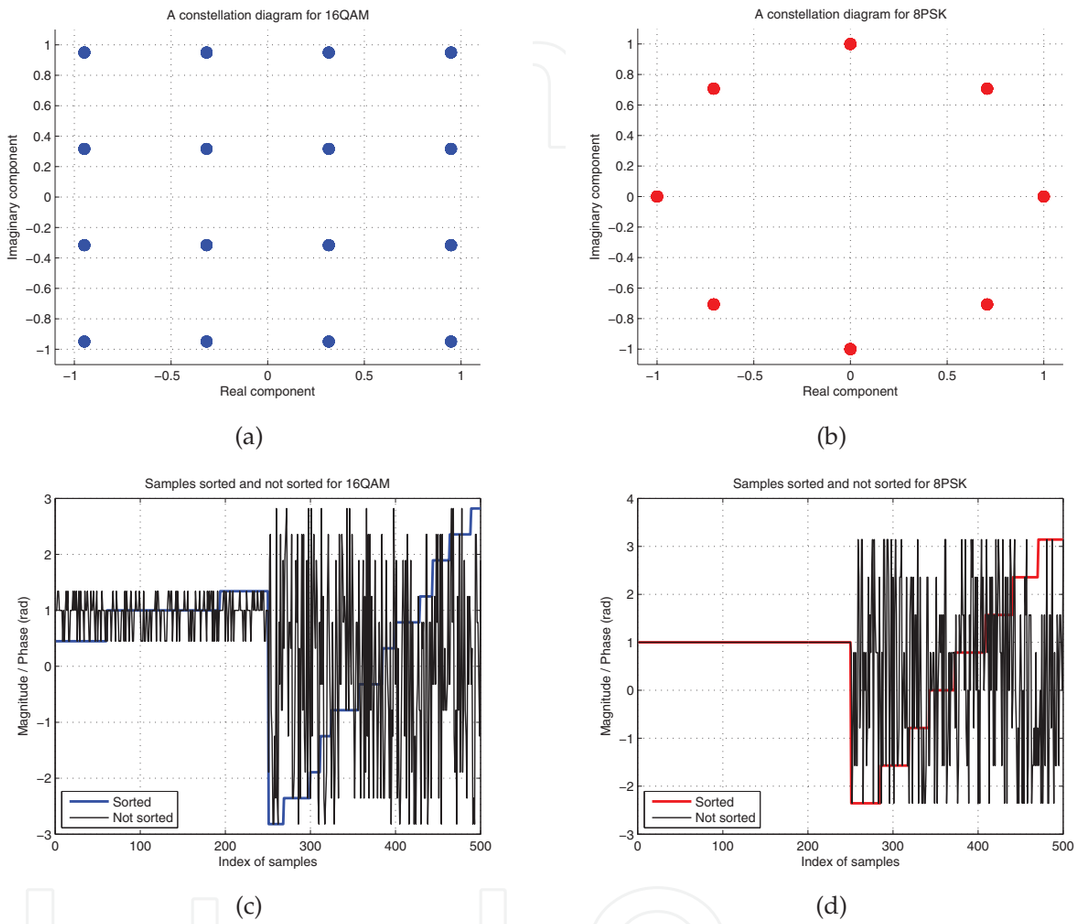


Fig. 1. Examples of vector of parameters with $D = 2N = 500$ for a 16QAM and 8PSK modulation, without noise. (a) A constellation diagram for 16QAM. (b) A constellation diagram for 8PSK. (c) Samples sorted and not sorted for 16QAM. (d) Samples sorted and not sorted for 8PSK.

representing a 16QAM and 8PSK modulations is also illustrated. There is no noise and both curves of received symbols, before and after ordering, are shown in Fig. 1(c) and Fig. 1(d). Each vector is composed of $D = 2N = 500$ features (corresponding to 250 magnitudes and 250 phases). It is observed that ordering creates a pattern that can be used to identify the type of modulation used in the generation of the received signal. For example, one can note from Fig. 1(d) that all first $N = 250$ normalized and ordered symbols corresponding to the magnitude of the 8PSK modulation are equal to one. For improved clarity, Fig. 2 summarizes the information in Fig. 1 by showing the comparison between the signatures of the 16QAM

and 8PSK modulations provided by the CSS front end. For obtaining Fig. 3 and Fig. 4, white

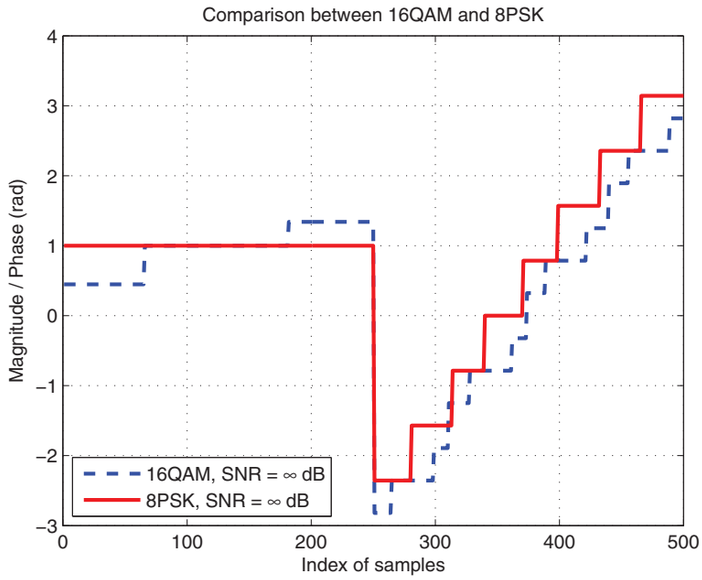
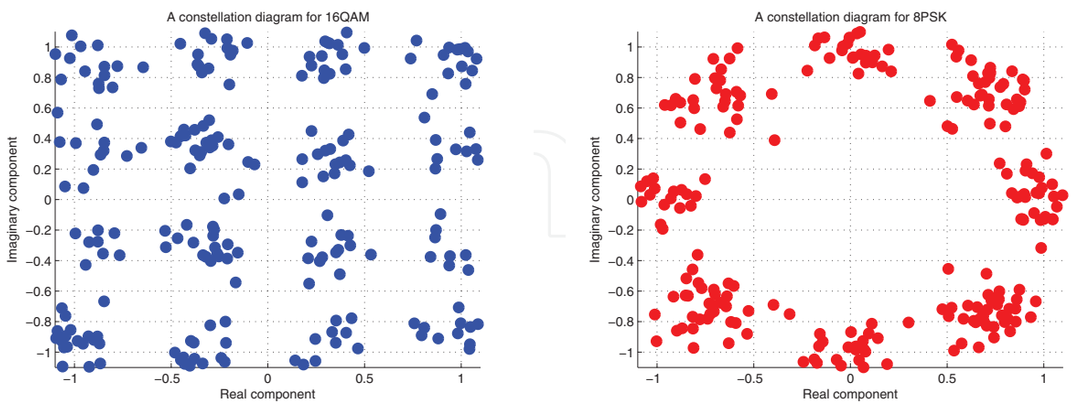


Fig. 2. Examples of vector of parameters with $D = 2N = 500$ for comparison between 16QAM and 8PSK under ideal conditions.

Gaussian noise was added to achieve signals with SNR = 15 dB. It can be observed that the signatures are modified with respect to the ideal case and their differences are less visible. However, in Muller et al. (2011) it was shown that even with noise, it is possible to distinguish these two modulations when the SNR is large enough by using, for example, an SVM classifier.

The next subsection presents an alternative front end, which will be used for performance comparisons.



(a) (b)

Fig. 3. 16QAM and 8PSK modulation, with SNR = 15 dB. (a) A constellation diagram for 16QAM. (b) A constellation diagram for 8PSK.

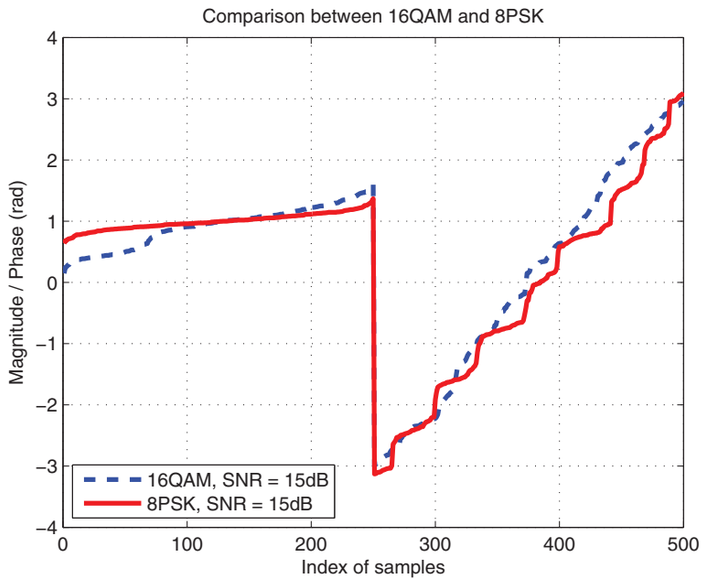


Fig. 4. Examples of vector of parameters with $D = 2N = 500$ for comparison between 16QAM and 8PSK. SNR = 15 dB.

2.2 Front end: Cyclostationarity

Cyclostationary analysis has been increasingly considered for use in a large range of applications, including signal detection, classification, synchronization and equalization. In Gardner & Spooner (1992), a large number of advantages of cyclostationary analysis are identified when compared to radiometric approaches (which are based in the measured energy of the received signal). Among its advantages are the reduced sensibility to noise and interfering signals, and also its ability to extract signal parameters such as the carrier frequency and the symbol rate.

In spite of its robustness, it is well-known that the computational cost of a cyclostationary analysis is high. According to the work of Lin & He (2008), a discrete-time signal $x[n]$ is defined to be cyclostationary if its autocorrelation function is invariant in relation to time shifts to all integer m multiple of T_o, that is

$$R_x(n + mT_0, 1) = R_x(1). \tag{1}$$

Two mathematical functions are used to characterize cyclostationary signals. The first is the cyclic autocorrelation function (CAF) (Castro, 2011), defined by

$$R_x^{\alpha}(1) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} \{x[n+l]e^{-j2\pi\alpha(n+1)}\}\{x[n]e^{-j2\pi\alpha n}\}^* \tag{2}$$

where R_x^{α} denotes the CAF for a discrete-time signal $x(n)$ and α is referred to as the cyclic frequency. When the signal $x(n)$ is cyclostationary, its CAF is nonzero at some time delay l and cyclic frequency $\alpha \neq 0$ (Wang et al., 2010).

The second function, the spectral correlation density (SCD), or cyclic spectral density, is calculated from the Fourier transform of the CAF:

$$S_x^{\alpha}(k) = \sum_{\mathrm{l}=-\infty}^{\infty} R_x^{\alpha}(\mathrm{l}) e^{-j2\pi k \mathrm{l}}, \tag{3}$$

where k is the frequency. When $\alpha = 0$, the CAF and SCD represent the autocorrelation and power spectral density functions, respectively.

2.2.1 Estimation of the SCD

When the AMC is based on cyclostationary features, estimating the SCD is fundamental to extract features that should distinguish modulated signals. Two algorithms to estimate the SCD were proposed in Schnur (2009): FFT Accumulation Method (FAM) and Strip Spectral Correlation Algorithm (SSCA). The FAM algorithm is considered to be more computationally efficient than the SSCA and is adopted in the sequel.

Assuming discrete-time processing, the FAM algorithm calculates

$$S_x^{\alpha}(k) \approx S_x^{\alpha}(n,k) = \frac{1}{N} \sum_{n=0}^{N-1} \left[\frac{1}{N'} X_{N'}(n, k + \frac{\alpha}{2}) X_{N'}^{*}(n, k - \frac{\alpha}{2}) \right] \tag{4}$$

where N is the number of time samples within the range of observation of the signal, k and α are the frequency and cyclic frequency, respectively. The parcels $X_{N'}(n, k \pm \frac{\alpha}{2})$ represent the complex envelope of the spectral component of $x[n$ and can be computed in the following way:

$$X_{N'}(n,k) = \sum_{r=-N'/2}^{N'/2} a[r] x[n-r] e^{-j2\pi k (n-r) T_s}, \tag{5}$$

where $a[r]$ is the data taper window (for instance Hamming window) and T_s is sampling period. In this method, the complex envelopes are estimated by means of sliding N' FFT points, followed by a downshift in frequency to baseband. For improved clarity, Fig. 5 represents the diagram implementation of this method. Fig. 5 pictorially represents the steps for estimating the SCD, which are:

- The input sample sequence $x[n]$ of length N is divided into P blocks, where each block containing N' samples, and L is the overlap factor;
- A Hamming window is applied across each block;
- The FFT of each block is computed;
- The complex envelopes $X_{N'}(n,k)$ are downshift in frequency to baseband;
- The SCD function is estimated by multiplying $X_{N'}(n,k)$ by its complex conjugate;
- The smoothing operation of the product sequences is executed by means of P-points FFT.

The value of L is configured to be equal to $N'/4$, because this value is a good trade off between computational efficiency and minimizing cycle leakage and cycle aliasing. The value of N' is

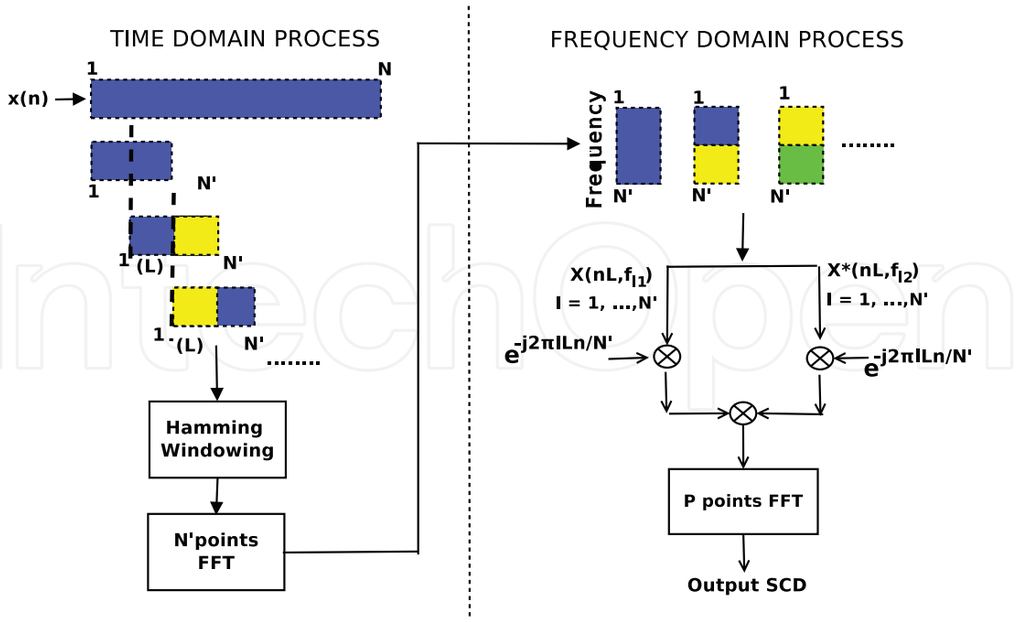


Fig. 5. Implementation of the method FAM.

computed according to the desired resolution Δk, and is defined by:

$$N' = \frac{f_s}{\Delta k}. \tag{6}$$

The value of P is determined according to the desired cyclic frequency resolution $\Delta\alpha$, and is given by:

$$P = \frac{f_s}{L\Delta\alpha}. \tag{7}$$

To perform AMC by means of cyclostationarity features it is typical to normalize the SCD. This normalization can be obtained by:

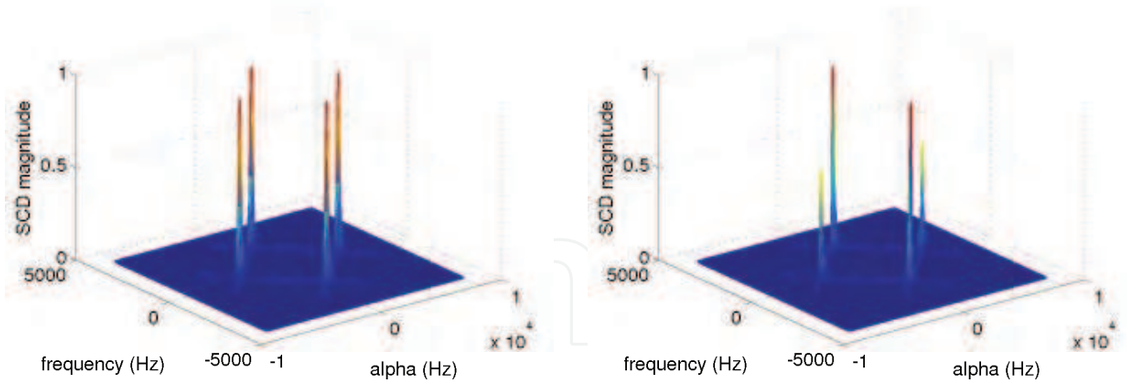
$$C_x^\alpha(k) = \frac{S_x^\alpha(k)}{[S_x^0(k+\alpha/2)S_x^0(k-\alpha/2)]^{1/2}}, \tag{8}$$

where $C_x^\alpha(k)$ is the spectral autocoherence function.

Fig. 6 shows the estimation of the SCD for BPSK and QPSK modulations respectively. This example adopted a sampling frequency $f_s = 8192$ Hz, carrier frequency $K = 2048$ Hz, cyclic frequency resolution $\Delta\alpha = 20$ Hz and frequency resolution $\Delta k = 80$ Hz. It can be noticed that the SCD is three-dimensional. The features that will distinguish each modulation, that is, the cyclic domain profile (CDP), is obtained from the SCD. The CDP $I(\alpha)$ uses only the peak values in the SCD and is obtained by

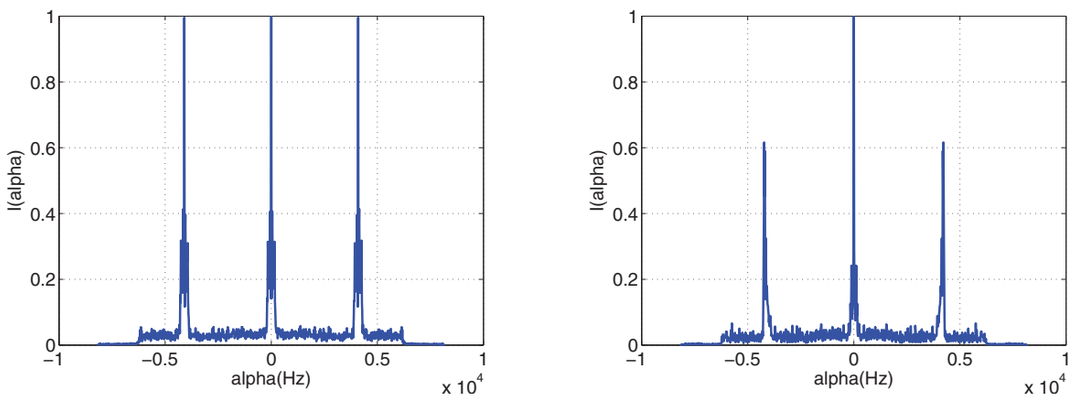
$$I(\alpha) = \max_k |C_x^\alpha(k)|. \tag{9}$$

The CDP for the BPSK and QPSK modulations of Fig. 6 are shown in Fig. 7.



(a) (b)

Fig. 6. Spetral Cyclic Density. (a) BPSK. (b) QPSK



(a) (b)

Fig. 7. Cyclic Perfil. (a) BPSK. (b) QPSK

2.3 Back end: SVM classifier

Support vector machine (SVM) is a class of learning algorithms based on the statistical learning theory, which implements the principle of the structural risk minimization (Vapnik, 1998). A basic idea of SVM is to map the input space into a feature space. This mapping can be done linearly or not, according to the kernel function used for the mapping. In the literature, various possibilities for SVM kernels are presented in applications involving pattern recognition such as: linear kernel, polynomial kernel, gaussian kernel and radial basis network (Burges, 1998).

In the feature space, an SVM builds a *maximum margin* hyperplane **w** to separate classes while minimizing the classification error. The hyperplane can be written as a combination of few points (training examples) in the feature space, called the *support vectors* of the optimal hyperplane.

Maximum margin is defined as the shortest distance that separates the training examples of different classes in relation to the hyperplane, as seen in Fig. 8. The distance of any point $\mathbf{x}_i$ to the hyperplane is given by Equation 10, where $||\mathbf{w}||$ is the norm of the vector.

$$d = \frac{\langle \mathbf{w}, \mathbf{x} \rangle + \mathbf{b}}{||\mathbf{w}||} \tag{10}$$

An SVM is a binary classifier given by

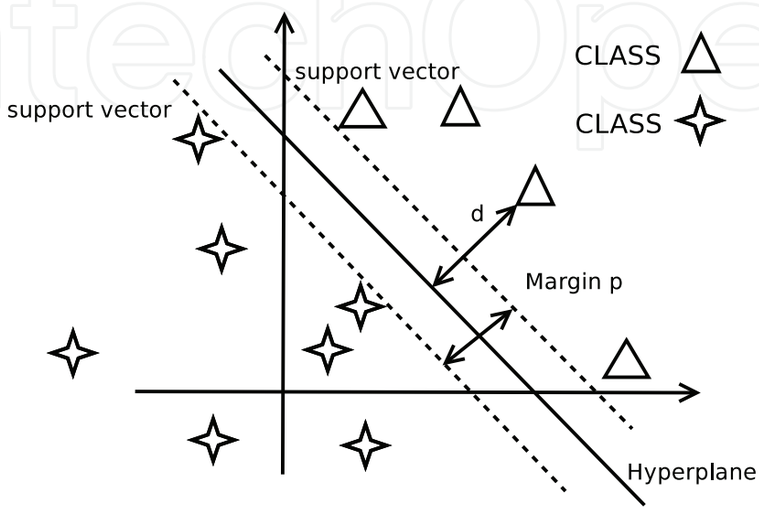


Fig. 8. Examples of the margin and support vectors.

$$f(\mathbf{x}) = \sum_{m=1}^{M} \gamma_m \mathcal{K}(\mathbf{x}, \mathbf{x}_m) + c, \tag{11}$$

where $\mathcal{K}(\mathbf{x}, \mathbf{x}_m)$ is the kernel function between the test vector $\mathbf{x}$ and the m-th training example $\mathbf{x}_m$, with $c, \gamma_m \in \Re$. The effectively used examples have $\gamma_m \neq 0$ and are called *support vectors*. The number $V \leq M$ of support vectors can be large and impact the computational cost. An SVM with a linear kernel $\mathcal{K}(\mathbf{x}, \mathbf{x}_m) = \langle \mathbf{x}, \mathbf{x}_m \rangle$ given by the inner product between $\mathbf{x}$ and $\mathbf{x}_m$, can be converted to a perceptron $f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{x} \rangle + c$, where $\mathbf{a} = \sum_{m=1}^{M} \gamma_m \mathbf{x}_m$ is pre-computed.

Therefore, linear SVMs were adopted in this chapter due to their lower computational cost when compared to non-linear SVMs with kernels such as the Gaussian (Cristianini & Shawe-Taylor, 2000). To combine the binary SVMs $f_b(\mathbf{x}), b = 1, \ldots, B$, to obtain $F(\mathbf{x})$ this work adopted the *all-pairs* error-correcting output code (ECOC) matrix with Hamming decoding (Allwein et al., 2000), where the winner class is the one with the majority of "votes". Note that an alternative to all-pairs, which uses $B = 0.5C(C-1)$ SVMs, is the one-vs-all ECOC that uses $B = C$ SVMs (Klautau et al., 2003).

3. Results

3.1 Simulation results

Simulations were performed to compare the performance of the CSS and cyclostationarity front ends to classify the modulations BPSK, 4-PAM, 16-QAM and 8-PSK in channels with

additive white Gaussian noise (AWGN). For these simulations, linear SVM classifiers were trained using several integer SNR values in the range [-5, 15] dB. The training and test sets used in all simulations were made disjoint and each had 500 examples. All the constellations were normalized to have unitary energy. For the CSS front end, $N = 250$ symbols were used per training / test instance. For the cyclostationarity front end, all signals were generated with sampling frequency $f_s = 8192$ Hz, carrier frequency $K = 2048$ Hz, cyclic frequency resolution $\Delta\alpha = 20$ Hz and frequency resolution $\Delta k = 80$ Hz.

The results are shown in Fig. 9. Note that the approach based on the CSS front end considers that the received signal was properly demodulated, whereas the cyclostationarity front end does not require demodulation and, consequently, demand less knowledge about the input signal. In fact, the cyclostationary analysis itself can be used to estimate important parameters for the demodulation task, such as the carrier frequency and symbol rate. Because of these aspects, a complete AMC may use more than one complementary front ends. It is observed

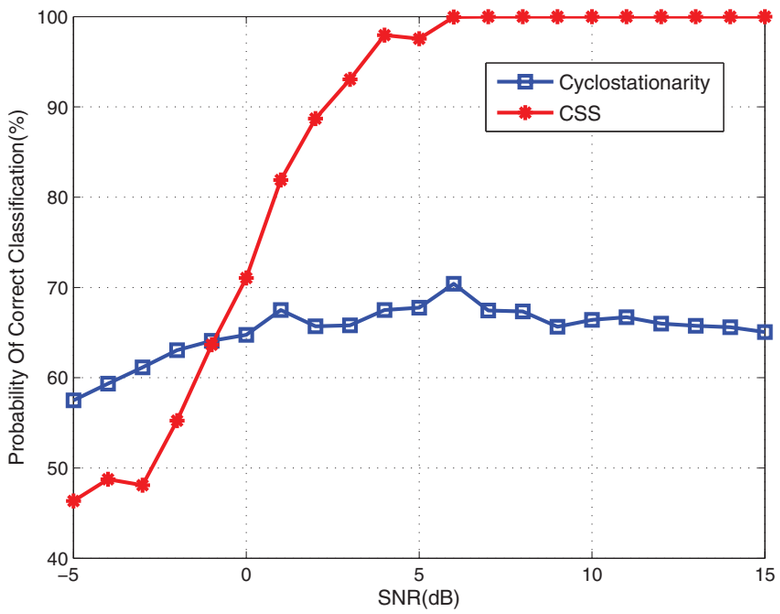


Fig. 9. Probability of correct classification of the cyclostationarity and CSS front ends.

from Fig. 9 that for the adopted AMC problem, the CSS outperformed the cyclostationarity front end. This is due to the difficulty of correctly separating cyclostationarity pair of modulations [BPSK, 4PAM] and [16QAM, 8PSK]. For both pairs, the cyclical characteristics are similar, as shown by the confusion matrix in Table 1 and discussed in A. Fehske (2005). For SNR = -1 dB, the performance of both techniques is similar. The confusion matrix shown in Table 2 and Table 3 illustrate that both front ends do not distinguish the same pairs of modulations. For SNR values larger than -1 dB, the CSS presents better results than the cyclostationarity. Although the cyclostationarity front end has not presented good results for the set of adopted modulations (BPSK, 4PAM, 8PSK and 16QAM), it is capable of producing good results for others (A. Fehske, 2005; da Silva et al., 2007; Haykin et al., 2009). Also, the CSS is restricted to linear digital modulations while the cyclostationarity is not. In spite of

classified as ->	BPSK	4PAM	16QAM	8PSK
BPSK	291	209	0	0
4PAM	157	344	0	0
16QAM	0	1	331	168
8PSK	0	0	164	336

Table 1. Confusion matrix for cyclostationarity. SNR = 15 dB

classified as ->	BPSK	4PAM	16QAM	8PSK
BPSK	391	109	0	0
4PAM	219	279	2	0
16QAM	6	1	275	218
8PSK	5	0	144	351

Table 2. Confusion matrix for cyclostationarity. SNR = -1 dB

classified as ->	BPSK	4PAM	16QAM	8PSK
BPSK	436	64	0	0
4PAM	204	295	0	1
16QAM	0	0	316	184
8PSK	0	0	274	226

Table 3. Confusion matrix for CSS. SNR = -1 dB

these two aspects, the presented results confirm that CSS is a competitive technique for AMC. Thus, the CSS front end was implemented in a FPGA for investigating its real-time processing capabilities, as described in the sequel.

3.2 Implementation results

Most of the AMC research is based on computer simulations and does not target the hardware implementation. In addition, there are few commercial devices aimed at detecting empty channels or classify modulations. But the available equipment is proprietary, directed generally to the military. Thus, there is great interest in academia and also in the industry to implement and test algorithms for AMC and spectrum sensing (Mishali & Eldar, 2011) to be easily embedded in devices such as FPGA or DSP (digital signal processor).

The next subsection describes the implementation of a CSS-SVM classifier for AMC starting by the classifier.

3.2.1 Architecture of the programmable SVM classifier

For the implementation of an SVM classifier in FPGA using the all-pairs ECOC, an architecture was designed in which the test instance (input parameters $\mathbf{x}[k]$) is continuously classified and the coefficients of the SVMs can be changed on-the-fly. This is a programmable architecture proposed for multiclass classification, using binary classifiers. The training of the classifiers is performed offline.

Equation 12 represents the function for the decision problem between two classes used by the binary SVM. Where $\mathbf{w}$ and $\mathbf{b}$ are the coefficients of the classifier and $\mathbf{x}$ is the vector test. The sign of $\mathbf{f(x)}$ indicates the result of the classifier, with $\mathbf{f(x)} = 0$ the decision threshold between

the two classes for which the classifier was trained.

$$\mathbf{f(x)} = \sum_{i=1}^{n} \mathbf{w}_i \mathbf{x}_i + \mathbf{b}. \tag{12}$$

To combine binary SVMs it was used matrix all-pairs ECOC with decoding Hamming, where the winner class was chosen by having most of the "votes". In other words, each one of binary SVM is trained to distinguish a pair of classes and, in the test phase, the chosen class was the one that had the largest number of binary SVMs, indicating the winner (Muller et al., 2011).

Fig. 10 shows the process of programming the FPGA with the SVM coefficients and reading the input test data. The structure uses a demultiplex followed by groups of shift registers. The value of the input **i** is stored in a shift register, and several registers are disposed in sequential arrangements with the information moved by the circuit until all the registers are updated. In Fig. 10, n is the number of features of the test data. Considering y classes, recall that the number B of binary classifiers for the all-pairs ECOC is $B = (y(y-1))/2$. The registers **w** and **b** store the coefficients of the classifiers and **x** stores the data for testing. After updating the coefficients, the test values are stored and classification can be started. If necessary to

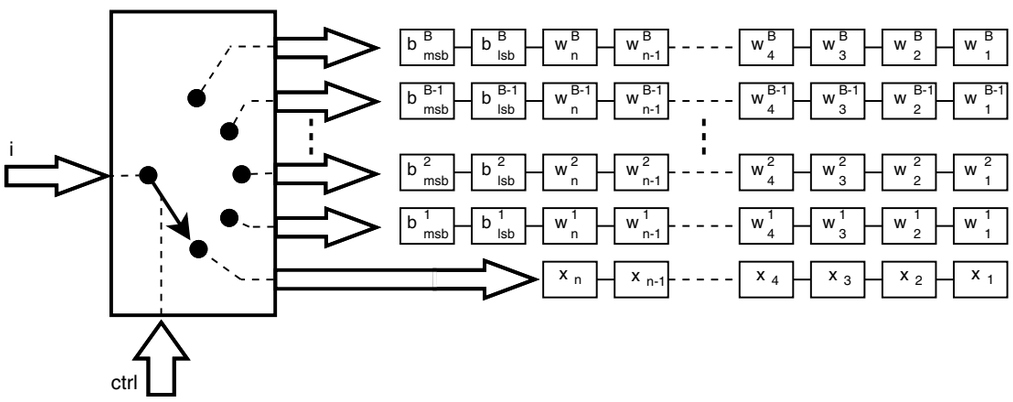


Fig. 10. Diagram representing the on-the-fly programming of SVM coefficients and input data. The row at the bottom corresponds to the input data and the other rows to SVM coefficients. The ctrl input controls the demultiplexer.

update the SVM coefficients, the flow of test data will be suspended for the time necessary. In these cases, the input **i** will be directed to the corresponding SVM register set and not to the registers that store the test data.

The proposed SVM architecture uses four steps, which are depicted in Fig. 11. The implementation uses a state machine and can be understood as:

- **The first step** makes the multiplication of elements of the vectors **w** and **x** shown in Equation 12 and stores the results in other sets with the same number of registers. However, to be able to store the multiplication results without roundoff errors, the registers that receive the results have twice the number of bits of the registers that store the vectors **w** and **x**. For each SVM classifier, a new set of registers is necessary to receive the results of multiplications.
- **The second step** performs, for each classifier, the sum of all values from the first step, added to the coefficient **b** (the "y-intercept" of a linear SVM).

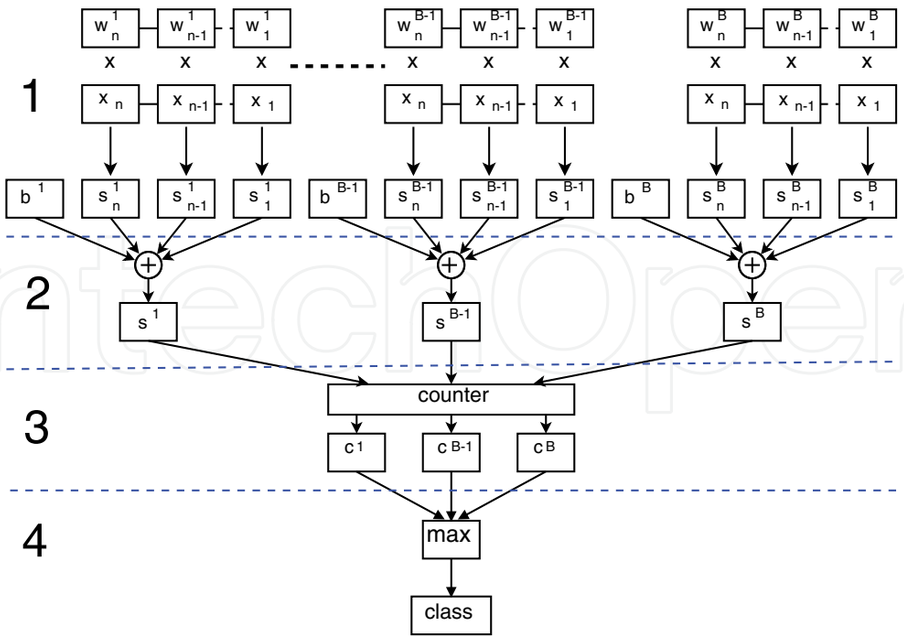


Fig. 11. Diagram that performs the classification.

- **The third step** checks the result of each classifier. As binary classifiers, the result of a classifier can be considered as a vote given to the one of the classes for which it was trained. At the end of the third step, we have totaled the number of votes for each of the classes.
- **The fourth step** verifies the class that received the largest number of votes counted in the third step. The result of this step is the output of the classification.

It is important to note that this architecture is tailored to implementation in a FPGA, where it is possible to describe hardware that performs various operations on each clock cycle (clock). In this case, each of the steps to perform the calculation of the SVM should occur in one clock period. In the first step, for example, in a single clock cycle, all the multiplications needed to compute the inner product between the test vector and the coefficients of binary classifiers are performed. The disadvantage of this architecture is the large amount of FPGA resources used, which make this approach feasible only when the number of coefficients is small. This is because the architecture, as described, uses only registers and logic elements to execute several multiplications in one clock period. A better use of the FPGA resources is obtained by using its RAM memory, as will be described together with the implementation of the CSS.

3.2.2 Architecture of the CSS-SVM modulation classifier

The front end CSS proposed by Muller et al. (2011) uses the symbols represented by magnitude and phase. For a set of received symbols, an ordered vector with the magnitude is concatenated with a vector with ordered phases, generating a third vector x with twice the number of samples.

For implementing the CSS front end in a FPGA, the VHDL language (*IEEE Standard VHDL Language Reference Manual*, 2009) was adopted within the Altera Quartus II development platform (Altera, 2011). In order to explain the proposed architecture, first a high-level

description will be provided, which treats the CSS-SVM as a processing block. Emphasis is placed on its inputs (symbols for being classified and SVM coefficients) and output (the classification result). Fig. 12 depicts the implemented block. The symbols can be entered in the same sequence they are received. The implementation assumes that demodulation has been performed and the CSS implementation corresponds to ordering the magnitudes and phases of the symbols. This ordering and the SVM-based classification are performed by the cited block. The signals in Fig. 12 have the following role:

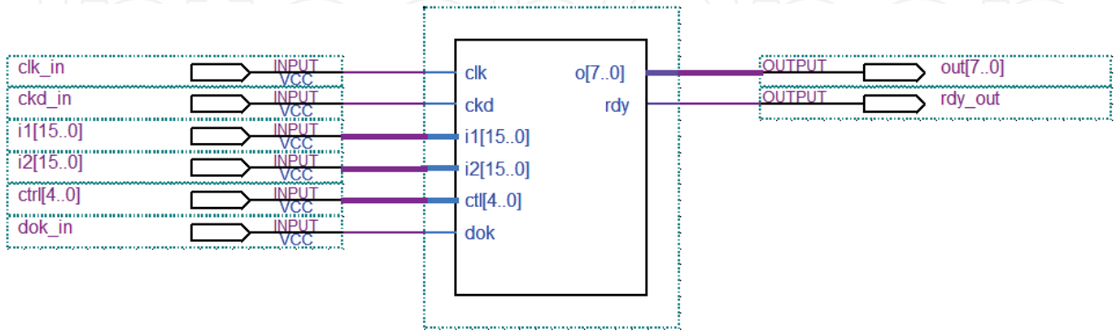


Fig. 12. Representation of the implemented CSS-SVM block with its inputs and outputs signals.

- **clk** represents the clock and is the base time system.
- **ckd** indicates the arrival of a new sample for classification or a new coefficient for the classifiers.
- **i1** and **i2** are the inputs that can be used both for input symbols that will be classified and for the SVM coefficients. The symbols or coefficients values should reach the block with a rate defined by the signal **ckd**. In this implementation, the rising clock edge is used to update the signals **i1** and **i2**. For these signals it was used the two's complement notation, with 16 bits for magnitude and 16 bits for phase for each of symbols that will be classified.
- **dok** indicates that a whole set of *N* test symbols has been informed and a new classification can begin.
- **Ctrl** is a control signal used with the inputs **i1** and **i2** that can be used to provide the symbols or the SVM coefficients. Obviously, all the SVM coefficients should be informed before symbols are received. When zero is the value given in **Ctrl**, **i1** and **i2** must contain, respectively, the magnitude and phase of a new symbol. For other values, it is understood that the value given in **i1** is a coefficient. In this case, the value of **Ctrl** tells to which classifier the coefficient corresponds and **i2** the position of the coefficient.

As explained, mapping all the processing into registers, logic elements and multipliers consume too many resources of the FPGA, because all values of samples and coefficients need to be available for accomplishing many multiplication operations in a single clock cycle. Another way to implement the CSS-SVM classifier is to direct all coefficients and symbols to be stored in RAM, which are available in FPGAs such as Altera's Cyclone II.

In this improved architecture, the SVM coefficients are separated into different blocks in the FPGA memory. The disadvantage is that only one position of each memory block can be accessed in a clock cycle and, consequently, a greater number of clock cycles is required for

performing a classification. In the architecture described in Section 3.2.1, the rate of symbols input could be the same as the clock, but in the improved architecture, the clock rate must be higher than the rate of input symbols. For $N = 250$, the clock rate should be approximately 130 times the symbol rate and for clk = 50 MHz (1/clk = 20 ns) took 627.62 us to perform the classification of a set of symbols.

Fig. 13 represents the different memory blocks used in the classifier. The upper lines (**w**) represent the memory blocks for the SVM coefficients and they have $D + 2$ positions due to the fact that storing the coefficient **b** consumes twice the number of bits used by the other coefficients. The explanation is that, during the simulations, the coefficients **b** were an order of magnitude larger than the other coefficients. Therefore, they are represented using two memory positions. The lower (smaller) blocks represent the memory space to store the

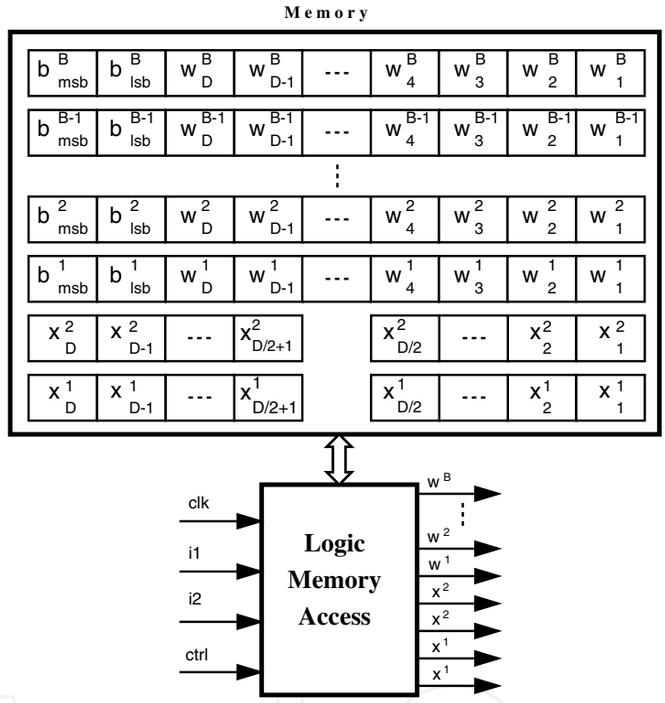


Fig. 13. Representation of the memory blocks.

symbols to be classified and that will compose the vector (**x**). As the FPGA design allows accessing memory blocks one element at a time, the set of test symbols are stored in two separate memory blocks: one for the magnitude and another block for the phase.

For supporting a steady stream of symbols, there are two sets with two memory blocks each. The classification process works on a set of previously received symbols while new symbols are received and stored.

The signal **dok** indicates that a new classification should start. A sorting algorithm is used to sort the values of magnitude and phase before being classified through the SVM classifier. During this process, new symbols received are directed to another memory block.

After sorting, the multiplication of vectors (inner product between) **w** and **x** is performed according to Equation 12, for each classifier. This stage requires D clock cycles because each

cycle is used to perform the multiplication of a value of vector **x** with a coefficient of each classifier, which means that each cycle makes B multiplications. There are B registers with the SVM output values **f(x)**, one for each SVM classifier. The next step is to count the votes of the classifiers. Finally, the result of the CSS-SVM classifier is the modulation that received the majority of votes.

Table 4 shows the used resources of an Altera Cyclone II FPGA model EP2C20F484C7 to program a CSS-SVM classifier, assuming $N = 250$ symbols for classification.

Resources	QTY	%
Estimated total logic elements	1,824	10%
Total combinational functions	1,586	8%
Dedicated logic registers	704	4%
Total memory bits (RAM)	64,000	27%
Embedded multiplier 9-bit elements	24	46%

Table 4. Used resources of an Altera Cyclone II FPGA model EP2C20F484C7 to implement the CSS-SVM classifier.

It is observed that the proposed architecture is relatively efficient with respect to the use of FPGA resources. It uses memory and multipliers available in the FPGA, which releases registers and logic elements to other algorithms such as demodulation and functions other than AMC. If this is not the case, the first architecture can be adopted.

4. Conclusions

This chapter discussed the AMC task in cognitive radio. Two front end techniques were contrasted: CSS and cyclostationarity. The adoption of SVMs as the base classifiers for AMC systems is showed and experimental results were presented.

The results shown in this chapter were obtained for classification of the BPSK, 4PAM, 8PSK and 16QAM modulations. The CSS front end combined with the linear kernel SVM classifier achieved good results for the set of modulations adopted, and proved to be feasible for implementation in a FPGA, processed in real time. In addition, a synthesizable CSS-SVM architecture was proposed, which presented satisfactory results for AMC with a relatively efficient use of the available FPGA resources.

5. References

A. Fehske, J. Gaeddert, J. R. (2005). A new approach to signal classification using spectral correlation and neural networks, *DySPAN* pp. 144–150.

Allwein, E., Schapire, R. & Singer, Y. (2000). Reducing multiclass to binary: A unifying approach for margin classifiers, *Journal of Machine Learning Research* pp. 113–141.

Altera (2011). Altera corporation. `http://www.altera.com/`.

Bremaud, P. (2010). *Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues*, Springer.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery* 2(2): 1–43.

Castro, M. E. (2011). *Cyclostationary detection for ofdm in cognitive radio systems*, Master's thesis, Faculty of The Graduate College at the University of Nebraska.

Cortes, C. & Vapnik, V. (1995). Support-vector networks, *Machine Learning* 20(3): 273–297. URL: *citeseer.nj.nec.com/cortes95supportvector.html*

Cristianini, N. & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*, Cambridge Univ. Press.

da Silva, C. R. C., Choi, B. & Kim, K. (2007). Distributed spectrum sensing for cognitive radio systems, *Information Theory and Applications Workshop*, pp. 120–123.

Dobre, O. A., Abdi, A., Bar-Ness, Y. & Su, W. (2007). Survey of automatic modulation classification techniques: Classical approaches and new trends, *IET Commun.* pp. 137–156.

Gardner, W. (1988). Signal interception: a unifying theoretical framework for feature detection, *IEEE Transactions on Communications* 36, Issue 8: 897 – 906.

Gardner, W. A. & Spooner, C. M. (1992). Signal interception: Performance advantages of cyclic-feature detectors, *IEEE Transactions on Communications* 40: 149–159.

Haring, L., Chen, Y. & Czylwik, A. (2010). Automatic modulation classification methods for wireless OFDM systems in TDD mode, *IEEE Transactions on Communications* 58: 2480 – 2485.

Haykin, S., Thomson, D. & Reed, J. (2009). Spectrum sensing for cognitive radio, *Proceedings of the IEEE* 97: 849–877.

IEEE Standard VHDL Language Reference Manual (2009). *IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002)* pp. c1 –626.

Klautau, A., Jevtić, N. & Orlitsky, A. (2003). On nearest-neighbor ECOC with application to all-pairs multiclass SVM, *J. Machine Learning Research* 4: 1–15.

Krose, B. & van der Smagt, P. (1996). *An introduction to Neural Networks*. URL: *"citeseer.nj.nec.com/article/schapire97using.html"*

Lin, Y. & He, C. (2008). Subsection-average cyclostationary feature detection in cognitive radio, *Neural Networks and Signal Processing, 2008 International Conference on*, pp. 604 –608.

Long, P. M. & Servedio, R. A. (2006). Discriminative learning can succeed where generative learning fails, *19th Annu. Conf. Learning Theory*, pp. 319–334.

Mak, P.-I., U, S.-P. & Martins, R. P. (2007). Transceiver architecture selection: Review, state-of-the-art survey and case study, *IEEE Circuits and Systems Magazine*, pp. 6–25.

Mishali, M. & Eldar, Y. C. (2011). Wideband spectrum sensing at sub-nyquist rates, *IEEE Signal Processing Magazine* 28: 102–135.

Muller, F. F., Cardoso-Jr., C. & Klautau, A. (2011). A front end for discriminative learning in automatic modulation classification, *IEEE Communications Letters (in Press)* .

Polydoros, P. P. A. A. A. (2000). Likelihood ratio tests for modulation classification, *21st Century Military Communications Conference Proceedings MILCOM 2000*, Vol. 2, pp. 670 – 674.

Proakis, J. G. (2001). *Digital Communications*, 4th edn, McGraw-Hill.

Rubinstein, Y. & Hastie, T. (1997). Discriminative vs informative learning, *Proc. 3rd Int. Conf. Knowledge Discovery Data Mining*, pp. 49–53.

Schnur, S. R. (2009). *Identification and Classification of OFDM Based Signals Using Preamble Correlation and Cyclostationary Feature Extraction*, PhD thesis, Naval Postgraduate School.

Su, W., Xu, J. L. & Zhou, M. (2008). Real-time modulation classification based on maximum likelihood, *IEEE Commun. Lett.* 12: 801–803.

Tan, P.-N., Steinbach, M. & Kumar, V. (2006). *Introduction to Data Mining*, Addison-Wesley, chapter Classification: Basic Concepts, Decision Trees, and Model Evaluation.

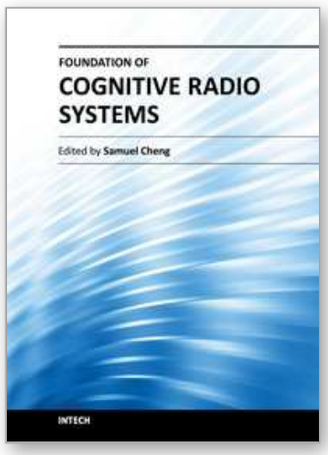
Vapnik, V. (1998). *Statistical learning theory*, John Wiley and Sons.

Wang, F. & Wang, X. (2010). Fast and robust modulation classification via kolmogorov-smirnov test, *IEEE Trans. Commun.* 58(8): 2324–2332.

Wang, R., Hou, C. & Chen, D. (2010). Blind separation of instantaneous linear mixtures of cyclostationary signals, *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, pp. 492 –495.

Xu, J. L., Su, W. & Zhou, M. (2011). Likelihood-ratio approaches to automatic modulation classification, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 41: 455–469.

Yucek, T. & Arslan, H. (2009). A survey of spectrum sensing algorithms for cognitive radio applications, *Communications Surveys Tutorials, IEEE* 11(1): 116 –130.



Foundation of Cognitive Radio Systems

Edited by Prof. Samuel Cheng

ISBN 978-953-51-0268-7
Hard cover, 298 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

The fast user growth in wireless communications has created significant demands for new wireless services in both the licensed and unlicensed frequency spectra. Since many spectra are not fully utilized most of the time, cognitive radio, as a form of spectrum reuse, can be an effective means to significantly boost communications resources. Since its introduction in late last century, cognitive radio has attracted wide attention from academics to industry. Despite the efforts from the research community, there are still many issues of applying it in practice. This books is an attempt to cover some of the open issues across the area and introduce some insight to many of the problems. It contains thirteen chapters written by experts across the globe covering topics including spectrum sensing fundamental, cooperative sensing, spectrum management, and interaction among users.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen