sensors

Article

A Strong Machine Learning Classifier and Decision Stumps Based Hybrid AdaBoost Classification Algorithm for Cognitive Radios †

Siji Chen [1], Bin Shen [1,*], Xin Wang [1] and Sang-Jo Yoo [2]

[1] School of Communication and Information Engineering (SCIE), Chongqing University of Posts and Telecommunications (CQUPT), Chongqing 400-065, China; cqchensj@outlook.com (S.C.); s170101160@stu.cqupt.edu.cn (X.W.)

[2] Department of Information and Communication Engineering, Inha University, Incheon 402-751, Korea; sjyoo@inha.ac.kr

* Correspondence: shenbin@cqupt.edu.cn; Tel.: +86-151-7889-9509

† This paper is an extended version of our paper published in Chen, S.; Shen, B.; Wang, X.; Wu, H. SVM and Decision Stumps Based Hybrid AdaBoost Classification Algorithm for Cognitive Radios. In Proceedings of the International Conference on Advanced Communications Technology (ICACT), PyeongChang, Korea, 17–20 February 2019.

Received: 24 September 2019; Accepted: 17 November 2019; Published: 20 November 2019

Abstract: Machine learning (ML) based classification methods have been viewed as one kind of alternative solution for cooperative spectrum sensing (CSS) in recent years. In this paper, ML techniques based CSS algorithms are investigated for cognitive radio networks (CRN). Specifically, a strong machine learning classifier (MLC) and decision stumps (DS) based adaptive boosting (AdaBoost) classification mechanism is proposed for pattern classification of the primary user's behavior in the network. The conventional AdaBoost algorithm only combines multiple sub-classifiers and produces a strong weight based on their weights in classification. Taking into account the fact that the strong MLC and the weak DS serve as different sub-classifiers in classification, we propose employing a strong MLC as the first-stage classifier and DS as the second-stage classifiers, to eventually determine the class that the spectrum energy vector belongs to. We verify in simulations that the proposed hybrid AdaBoost algorithms are capable of achieving a higher detection probability than the conventional ML based spectrum sensing algorithms and the conventional hard fusion based CSS schemes.

Keywords: machine learning; classifier; decision stump; AdaBoost; energy vector; cooperative spectrum sensing; cognitive radio network (CRN)

1. Introduction

Cognitive radio (CR) is commonly viewed as a promising technology that tremendously alleviates the increasing pressure on current rigid spectrum resource allocation regimes, by enabling dynamic access to the licensed spectrum in an opportunistic manner [1]. The opportunistic spectrum acquisition capability of CR systems generally relies on the spectrum sensing technologies that they adopt to identify the licensed spectrum status [2,3]. Cognitive users, also known as secondary users (SUs), are permitted to utilize the licensed spectrum only if they can assure themselves that the licensed spectrum is not temporarily occupied by the primary users (PUs). Generally, single user based spectrum sensing technologies can be categorized into blind and knowledge aided approaches [4]. Among various spectrum sensing methods, energy detection (ED) is an extensively used technique that requires no a priori knowledge of the PU signal and the channel environment. Capturing the received signal energy within a certain frequency band, a simple threshold test could agilely indicate whether the PU signal exists in the licensed spectrum or not. However, it is well known that the easy-to-implement energy detectors are vulnerable to noise power uncertainty effects and could not perform well under low signal to noise ratio (SNR) conditions, especially in severe multipath fading and shadowing environments.

In order to tackle the problem of single user based spectrum sensing, cooperative spectrum sensing (CSS) has been widely investigated, where multiple SUs collaborate to make a global decision on the licensed spectrum status. During each sensing interval, all SUs will report their spectrum observations or decisions to the fusion center (FC) and subsequently the FC makes a final decision based on its predefined decision strategy or fusion criterion [5]. Typical hard decision fusion based CSS schemes can be found in [6,7], in which logical operations are proposed for fusing the individual decisions collected from the cooperative SUs in the network.

Apart from the conventional ED based spectrum sensing schemes, machine learning (ML) based classification methods have been attracting attention in recent years, where the problem of spectrum status identification is solved by classifying the collected spectrum observations as the PU signal component contained data or the noise-only data [8–10]. For the purpose of spectrum sensing, ML approaches are classifiers to improve the detection performance of CSS and spectrum prediction, which is a part of spectrum sharing approaches [11]. ML based classifiers have the function of automatic learning over a large amount of training data and are hence able to make spectrum status predictions on the test data, which in some general way allays the nuisance effects of multipath fading and shadow, encountered by threshold-test based traditional spectrum sensing strategies. Therefore, the ML based classifiers usually perform better than energy detectors and the logical criteria based CSS schemes. As for the implementation of ML algorithms, they can be roughly divided into two types: unsupervised learning, e.g., K-means, and supervised learning, e.g., support vector machine (SVM) and K-nearest neighbors (KNN) [12–14]. The unsupervised methods directly model the input training data without exactly known labels for each training data point. In other words, they can not be sure whether the classification result of the training samples is correct or not [15]. The feature of these algorithms is beneficial in that they demand no a priori knowledge of the PU behavior and only rely on the training samples to automatically find their potential labels according to the training samples themselves.

Unlike unsupervised learning, the supervised learning methods analyze the relationship between the training data and the corresponding labels readily available for them, and map the input test data to the appropriate prediction or decision. Specifically, when the training phase is completed, the generalized model is applied to the new test data and their labels are thus accordingly predicted. Compared with unsupervised learning, the supervised learning methods are characterized by demanding more a priori knowledge from the input training samples and the corresponding true labels.

As a typical classifier, SVM is studied in [16] for CSS in detail, with an aim to find a linearly separable hyperplane with the help of support vectors by maximizing the margin of the classifier while minimizing the sum of classification errors. In [17,18], KNN is introduced and proved to be not only one of the simplest machine learning algorithms, but also one of the most basic, and best text categorization algorithms in case-based learning methods. However, to correctly apply KNN, we need to choose an appropriate value for K, because the success of classification is highly dependent on this value. In this sense, it may be concluded that the KNN method is biased by K, and there are many ways of choosing the K value, among which a simple method is to run the algorithm many times with different K values and choose the one with the best performance [19]. However, this might not be a satisfactory plug-and-play solution for spectrum sensing in practice. Essentially as a most well-known clustering algorithm, the K-means algorithm clusters the data points with high similarity into the same cluster according to the principle of similarity [20]. Due to its simplicity and efficiency, it has become the most widely used clustering algorithm [21]. Since different classifiers have various advantages and drawbacks, combining different individual classifiers into a comprehensive one with better performance is a straightforward solution. In [22,23], the adaptive boosting (AdaBoost) algorithm is introduced, for which the constructed classifier is composed of multiple weaker models that are independently trained and whose predictions are combined to make the overall prediction.

Noticing that employing ML techniques for CSS may be currently a novel solution in discerning the licensed spectrum status, we study the potentiality and feasibility of adopting ML algorithms in CSS, where the FC handles the spectrum data in a totally different manner compared to the conventional hard decision fusion based CSS schemes. To be more specific, we propose a hybrid AdaBoost classification algorithm which combines a strong ML algorithm (e.g., SVM, K-Means, and KNN) and multiple weak ML classifier (e.g., DS) for CSS in the cognitive radio network (CRN). Taking into account the fact that the ML classifier and DS serve as relatively strong and weak sub-classifiers respectively, the proposed algorithm employs one of the common ML classification algorithms (SVM, KNN, and K-Means) as the first-stage classifier and DS as the second-stage classifier to eventually determine the class that the spectrum observations belong to. For the scenario that there is one single PU and multiple SUs in the CRN, the proposed hybrid AdaBoost classification algorithm yields higher detection probability than the pure ML classification algorithms themselves and the DS based AdaBoost classification algorithm as well. In our proposed schemes, the SVM based hybrid AdaBoost performs the best. When compared with the traditional CSS schemes, e.g., AND and OR criteria based hard decision fusion algorithms, the proposed hybrid AdaBoost classification algorithm achieves better performance too. All these verifications prove that the proposed algorithms could be utilized in practice as a newfangled solution for CSS.

The remainder of this paper is organized as follows. In Section 2, we present the system model and some assumptions. Section 3 introduces the conventional spectrum sensing methods. Section 4 investigates various typical ML classifiers for CSS. In Section 5, we propose and describe the hybrid AdaBoost classification algorithm in detail. In Section 6, performance evaluation results of the proposed hybrid AdaBoost classification algorithm are given and compared with the conventional algorithms. Finally, Section 7 concludes the paper.

2. System Model

We consider a CRN consisting of a single PU transmitter and N SU receivers indexed by $n = 1, 2, \cdots, N$. In the CRN, the n-th SU is located at the geographic coordinate $\mathbf{C}_{\mathrm{SU}}^{n} = [x_n, y_n]^T$. We assume that the probability of the PU being active over the licensed spectrum is P_{ON} and there are two hypotheses, H_0 and H_1 with probability $(1 - P_{\mathrm{ON}})$ and P_{ON}, respectively.

The received signal at the n-th SU is expressed as

$$y_n(i) = \begin{cases} r_n(i) & H_0, \\ g_n \sqrt{P_{\mathrm{tx}}} x(i) + r_n(i) & H_1, \end{cases} \tag{1}$$

where g_n denotes the channel gain from the PU transmitter to n-th SU, $x(i)$ is the PU's transmitted signal, P_{tx} is the PU's transmitting power, and $r_n(i)$ is the additive white gaussian noise (AWGN) with zero-mean and variance $\eta = E[|r_n(i)|^2] = \sigma_0^2$. If the time duration of each spectrum sensing interval is denoted as τ and the spectrum bandwidth is W, each SU usually grasps $2W\tau$ samples of its received signal during the sensing duration τ.

The noise variance normalized energy statistic of the n-th SU in the first CSS phase (It is assumed that multiple SUs collect their spectrum observations in the first phase of CSS and the second phase is for global decision making in the FC.) can be denoted by Y_n:

$$Y_n = \frac{1}{\eta} \sum_{i=1}^{2W\tau} |y_n(i)|^2. \tag{2}$$

In the second CSS phase, all SUs report their spectrum decision D_n or energy observation Y_n to the FC as:

$$\begin{aligned}\mathbf{D} &= [D_1, D_2, \cdots, D_N]^T, \\ \mathbf{Y} &= [Y_1, Y_2, \cdots, Y_N]^T,\end{aligned} \tag{3}$$

where the FC operates over **D** or **Y** via different criteria to make the global decision on the spectrum status.

The power attenuations between the PU and the SU depend on the channel coefficient g_n in that $G_n = |g_n|^2$, where the power attenuation coefficient G_n can be expressed as:

$$G_n = PL(\|\mathbf{C}_{\mathrm{PU}} - \mathbf{C}_{\mathrm{SU}}^n\|_2) \cdot \psi_n \cdot v_n, \tag{4}$$

where $\|\cdot\|_2$ stands for the Euclidean distance or equivalently the L2-norm of the vector $\mathbf{C}_{\mathrm{PU}} - \mathbf{C}_{\mathrm{SU}}^n$, $PL(d) = d^{-\rho}$ is the pathloss for the relative distance d with the propagation loss exponent ρ, $\mathbf{C}_{\mathrm{PU}} = [x_{\mathrm{PU}}, y_{\mathrm{PU}}]^T$ denotes the coordinate of the PU tranmitter, ψ_n is the shadow fading coefficient, and v_n the multipath fading coefficient.

3. Conventional Spectrum Sensing Methods

3.1. Single User Based Energy Detection

According to the well-known ED, the n-th SU measures the strength of the received signal y_n and obtains the test statistic Y_n and the spectrum decision D_n via the threshold-test:

$$Y_n \gtrless \lambda_n \begin{array}{l} \nearrow D_n = 1, \\ \searrow D_n = 0, \end{array} \tag{5}$$

where λ_n is the pre-calibrated threshold, depending on the desired false alarm probability under the criterion of constant false alarm probability (CFAP), and the binary decision $D_n = 0$ and $D_n = 1$ indicates the cases of PUs being idle and active in the CRN, respectively.

3.2. Multi-User Based Hard Decision Fusion

On obtaining the spectrum decisions D_n, the n-th SU may cooperate with multiple SUs in its neighborhood to strengthen the sensing reliability through the hard decision fusion criteria, e.g., AND, OR, and Vote, as:

$$\Lambda_n = \sum_{t \in \mathbb{Q}_n, t \neq n} D_t + D_n \gtrless \lambda_T \begin{array}{l} \nearrow \tilde{D}_n = \hat{\mathcal{H}}_1, \\ \searrow \tilde{D}_n = \hat{\mathcal{H}}_0, \end{array} \tag{6}$$

where D_t is the spectrum decision of the t-th SU, $\mathbb{Q}_n$ is the index set of the n-th SU's neighbouring SU with the set cardinality of $|\mathbb{Q}_n|$, and λ_T is an integer threshold for the hard decision fusion schemes. The hard decision fusion scheme described in Equation (6) may boil down to the AND scheme for $\lambda_T = 1$, the OR scheme for $\lambda_T = |\mathbb{Q}_n| + 1$, and the $\lambda_{T,0}$-out-of-$(|\mathbb{Q}_n| + 1)$ (Vote) scheme for $\lambda_T = \lambda_{T,0}$, respectively. Decisions $\hat{\mathcal{H}}_0$ and $\hat{\mathcal{H}}_1$ are actually the final decisions for the n-th SU indicating the cases of PUs being idle and active in the CRN, respectively.

The spectrum sensing performance is usually evaluated in terms of detection probability $P_{\mathrm{D}} = Pr(\hat{\mathcal{H}}_1 \mid \mathcal{H}_1)$ and the false alarm probability $P_{\mathrm{FA}} = Pr(\hat{\mathcal{H}}_1 \mid \mathcal{H}_0)$. The detection probability P_{D} is a function of the SNR and the P_{FA} is usually predetermined as the desired performance for spectrum sensing.

4. Typical ML Classifiers for CSS

In this section, we investigate and propose some typical ML classifiers for the proposed CSS model. For ML method based CSS, a sufficiently large number of training energy vectors are demanded as training samples to train the classifiers. Let $\mathbf{x}^{(l)} = [X_1^{(l)}, X_2^{(l)}, \cdots, X_N^{(l)}]^T$ denote the l-th training energy vector obtained as $X_n = \frac{1}{\eta}\sum_{i=1}^{2W\tau} |y_n(i)|^2$ and $c^{(l)}$ the spectrum availability flag (or label) corresponding to $\mathbf{x}^{(l)}$. Thus, the training set of energy vectors are $\bar{\mathbf{X}} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(L)}\}$ and the spectrum availability labels are represented by $\bar{\mathbf{c}} = \{c^{(1)}, c^{(2)}, \cdots, c^{(L)}\}$, where L is the number of training samples (energy vectors) in the training set. After the classifier is successfully trained over the given training set $\bar{\mathbf{X}}$, it is able to classify the input test energy vector $\mathbf{x}^{*(m)}$ with label $c^{*(m)}$, denoting its true spectrum availability label. In addition, the set of test energy vectors can be denoted as $\mathbf{X}^* = \{\mathbf{x}^{*(1)}, \mathbf{x}^{*(2)}, \cdots, \mathbf{x}^{*(M)}\}$ and the set of corresponding spectrum availability labels is $\mathbf{c}^* = \{c^{*(1)}, c^{*(2)}, \cdots, c^{*(M)}\}$, where M is the number of test samples in the test set. If $\hat{c}^{(m)}$ is used to denote the m-th test sample's spectrum availability label decided by the classifier, it is either the spectrum available label (i.e., $\hat{c}^{(m)} = -1$) or the spectrum unavailable label (i.e., $\hat{c}^{(m)} = 1$). If the test energy vector $\mathbf{y}^{*(m)}$ is classified and labeled as spectrum available, it means that there is no PU in the active state and the licensed spectrum is available for the SU to access; otherwise, the SU cannot gain spectrum opportunity when the label is drawn as spectrum unavailable.

Therefore, the spectrum availability is correctly determined in the case that $\hat{c}^{(m)} = c^{*(m)}$, while mis-detection (or false alarm) occurs in the case that $\hat{c}^{(m)} = -1$ and $c^{*(m)} = 1$ (or $\hat{c}^{(m)} = 1$ and $c^{*(m)} = -1$). Thus, the prediction error (P_e) is defined as the probability of mis-prediction of the status of the licensed spectrum:

$$P_\mathrm{e} = \lim_{M\to\infty} \frac{1}{M}\sum_{m=1}^{M} I(\hat{c}^{(m)} \neq c^{*(m)}), \tag{7}$$

where $I(.)$ is the indicator function that takes the value 1 if its argument is true and 0 otherwise.

4.1. K-Means Clustering

The K-means clustering algorithm operates in an unsupervised manner and partitions a set of the training energy vectors (i.e., $\bar{\mathbf{X}} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(L)}\}$) into K disjoint clusters, which represents the K categories of training vectors. In other words, the training energy vectors can be assigned into K subsets of spectrum observations denoted by cluster labels C_k, $k = 1, 2, \cdots, K$. Normally, if we purposely set the number of clusters as two in advance, we can obtain two clusters by using the K-means algorithm without knowing the true labels that these two clusters actually correspond to. In spectrum sensing, it is applicable to set cluster 1 and cluster 2 as the cluster with channel availability label -1 and the cluster with channel availability label 1, respectively. In this way, the K-means clustering algorithm serves as an ML classifier, equivalently. The core idea of K-means is each cluster has a centroid that is defined as the mean of all training energy vectors within itself. Let $\boldsymbol{\mu}_k$ be the centroid of cluster k and $\gamma_{lk} \in \{0, 1\}$ be the indicative variable. We define $\gamma_{lk} = 1$, if the energy vector $\mathbf{x}^{(l)}$ is assigned to cluster k, otherwise, $\gamma_{lk} = 0$. Through several iterations of distance calculation, the values of $\boldsymbol{\mu}_k$ are updated sample by sample as follows:

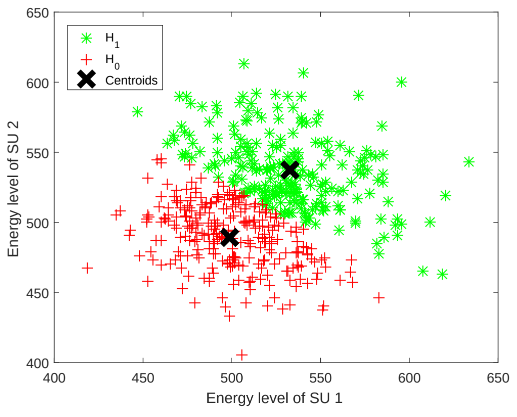
$$\boldsymbol{\mu}_k = \frac{\sum\limits_{l=1}^{L} \gamma_{lk}\mathbf{x}^{(l)}}{\sum\limits_{l=1}^{L} \gamma_{lk}}, \tag{8}$$

The objective function of K-means is to minimize the squared error in clustering:

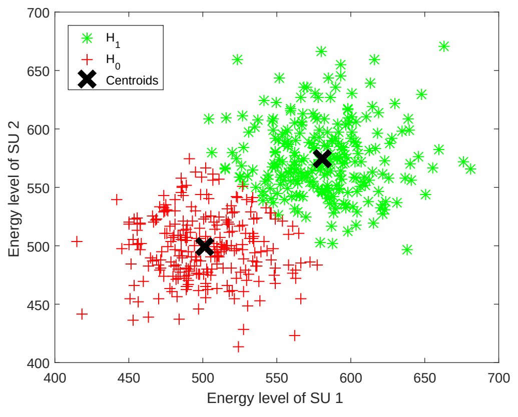
$$J(\boldsymbol{\gamma}, \bar{\boldsymbol{\mu}}, \bar{\mathbf{X}}) = \sum_{l=1}^{L}\sum_{k=1}^{K} \gamma_{lk} \|\mathbf{x}^{(l)} - \boldsymbol{\mu}_k\|_2^2, \tag{9}$$

where $\boldsymbol{\gamma} = [\gamma_{11}, \gamma_{21}, \cdots, \gamma_{L1}, \gamma_{12}, \cdots, \gamma_{L2}, \cdots \gamma_{LK}]^T$ and $\bar{\boldsymbol{\mu}} = [\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_K]^T$.

Figure 1 shows the two-user based classification results of the K-means clustering algorithm on the training samples when SNR = 1 dB and SNR = 6 dB respectively. We can see the farther the distance between samples belonging to two different labels, the better the classification result that can be obtained.



(**a**) Signal to noise ratio (SNR) = 1 dB



(**b**) SNR = 6 dB

Figure 1. Scatter plot of energy vectors classified by K-means.

4.2. Support Vector Machine

The SVM provides a binary model in machine learning which strives to find a linearly separable hyperplane with the help of support vectors (i.e., energy vectors that lie closest to the decision surface) by maximizing the margin of the classifier while minimizing the sum of classification errors [24]. As shown in Figure 2, the learning strategy of SVM is to maximize the margin and its learning goal is to find a hyperplane in the N-dimensional sample space.

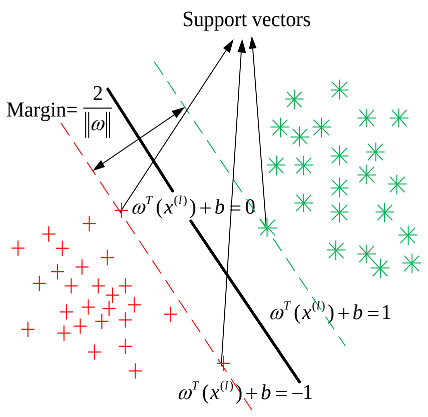


Figure 2. Support vector machine (SVM) model.

The hyperplane equation can be expressed as

$$\boldsymbol{\omega}^T\mathbf{x}^{(l)} + b = 0, \tag{10}$$

where $\boldsymbol{\omega}$ is the weighting vector and b is the bias. Based on $\boldsymbol{\omega}$, we need to minimize the vector norm of $\boldsymbol{\omega}$ so as to maximize the category margin, and hence the objective function is

$$\begin{aligned} &\text{Min} \quad \tfrac{1}{2}\|\boldsymbol{\omega}\|^2 \\ &\text{s.t.} \quad c^{(l)}(\boldsymbol{\omega}^T\mathbf{x}^{(l)} + b) \geq 1. \end{aligned} \tag{11}$$

Therefore, the SVM should satisfy the following condition for all $l \in \{1, 2, \cdots, L\}$:

$$c^{(l)} = \begin{cases} 1, & \text{if} \quad \boldsymbol{\omega}^T \mathbf{x}^{(l)} + b \geq 1, \\ -1, & \text{if} \quad \boldsymbol{\omega}^T \mathbf{x}^{(l)} + b \leq -1. \end{cases} \tag{12}$$

In practice, when a test energy vector $\mathbf{x}^{*(m)}$ is fed into the SVM model, the SVM can determine which class it belongs to through the following rules:

$$\hat{c}^{(m)} = \begin{cases} 1, & \text{if} \quad \boldsymbol{\omega}^T \mathbf{x}^{*(m)} + b \geq 1, \\ -1, & \text{if} \quad \boldsymbol{\omega}^T \mathbf{x}^{*(m)} + b \leq -1. \end{cases} \tag{13}$$

However, in practice for most of the time, the test samples are usually not linearly separable. For this case, the hyperplane satisfying such conditions does not exist at all. Then, we need to find a fixed nonlinear feature mapping function ϕ to map the non-linear samples into a new feature space and use a linear SVM in the feature space [25]. Hence, the non-linear SVM should satisfy the following condition for all $l \in \{1, 2, \cdots, L\}$:

$$c^{(l)} = \begin{cases} 1, & \text{if} \quad \boldsymbol{\omega}^T \phi(\mathbf{x}^{(l)}) + b \geq 1, \\ -1, & \text{if} \quad \boldsymbol{\omega}^T \phi(\mathbf{x}^{(l)}) + b \leq -1, \end{cases} \tag{14}$$

and the decision rule for the non-linear SVM is given as:

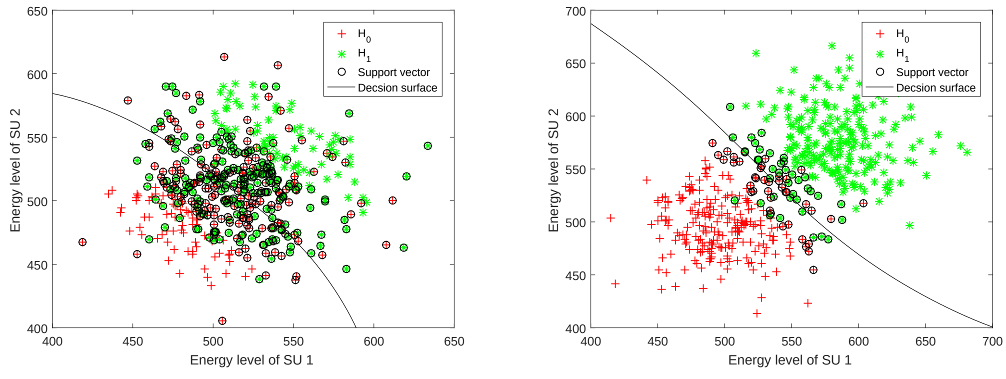
$$\hat{c}^{(m)} = \begin{cases} 1, & \text{if} \quad \boldsymbol{\omega}^T \phi(\mathbf{x}^{*(m)}) + b \geq 1, \\ -1, & \text{if} \quad \boldsymbol{\omega}^T \phi(\mathbf{x}^{*(m)}) + b \leq -1. \end{cases} \tag{15}$$

Although the training energy vectors have been mapped into a higher dimensional feature space, practically we cannot achieve a perfect linearly separable hyperplane that satisfies the condition in Equation (15) for each $\mathbf{x}^{(l)}$. Hence, we rewrite the optimization problem as a convex optimization problem as follows:

$$\begin{aligned} \text{Min} \quad & \tfrac{1}{2}\|\boldsymbol{\omega}\|^2 + C \sum_{l=1}^{L} \xi^{(l)} \\ \text{s.t.} \quad & c^{(l)}(\boldsymbol{\omega}^T \mathbf{x}^{(l)} + b) \geq 1 - \xi^{(l)}, \ l = 1, 2, \cdots, L \\ & \xi^{(l)} \geq 0, l = 1, 2, \cdots, L, \end{aligned} \tag{16}$$

where C is the soft margin constant, for which a larger C means the assignment of a higher penalty to errors, and $\xi^{(l)}$ is the slack variable.

Figure 3 shows the training samples classified by SVM-rbf. We can notice that the decision surface divides the energy vectors in each class as clearly as possible, which leads to improved detection performance. Also, the decision surface can separate energy vectors more accurately as the SNR increases.



(**a**) SNR = 1 dB (**b**) SNR = 6 dB

Figure 3. Scatter plot of energy vectors classified by SVM.

4.3. K-Nearest-Neighbor

The supervised K-nearest neighbors is a case-based learning method, which keeps all the training samples for classification. Being a lazy learning method prohibits it in many applications such as dynamic web mining for a large repository, as the KNN classifier requires storing the whole training set and may be too costly when this set is too large. One way to improve its efficiency is to find some representatives of the whole training data for classification, building an inductive learning model from the training samples, and classifying the test samples based on a similarity measure (e.g., distance functions), for example:

$$\text{Euclidean}: \left\|\mathbf{x}^{*(m)} - \mathbf{x}^{(l)}\right\|_2 = \sqrt{\sum_{n=1}^{N} |x_n^{*(m)} - x_n^{(l)}|^2}, \quad l = 1, 2, \cdots, L, \tag{17}$$

$$\text{Manhattan}: \left\|\mathbf{x}^{*(m)} - \mathbf{x}^{(l)}\right\|_1 = \sum_{n=1}^{N} |x_n^{*(m)} - x_n^{(l)}|, \quad l = 1, 2, \cdots, L. \tag{18}$$

According to the given distance measure, K samples with the nearest neighbor of test energy vector $\mathbf{x}^{*(m)}$ are found in the training set $\bar{\mathbf{x}}$ and the domain of covering these K samples is N_K. Among them, the setting of K value is generally lower than the square root of the number of samples, and it must be an odd number. Then the spectrum availability label for the spectrum data $\mathbf{x}^{*(m)}$ can be predicted based on classification decision rules (e.g., majority voting) as:

$$\hat{c}^{(m)} = \arg\max_{c^{(k)}} \sum_{k=1}^{K} I(c^{(k)} = \chi), \tag{19}$$

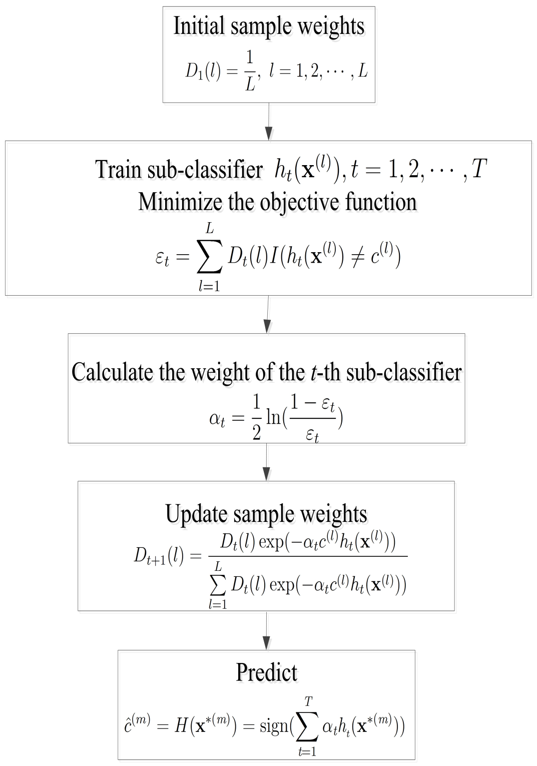
where $\chi \in \{c^{(1)}, c^{(2)}, \cdots, c^{(L)}\}$ and $c^{(k)}$ is the training label of the k-th nearest neighbor in N_K.

4.4. AdaBoost Algorithm

As for the weak classifier DS, it is a one-dimension decision tree, that uses only a single attribute of the spectrum data for splitting and makes only one judgment on each attribute (i.e., one of the test sample's dimensions). It is also well known that the decision stumps (DS) are often used as sub-classifiers in ensemble methods.

In addition to the weak DS and strong ML classifiers, AdaBoost (adaptive boosting) algorithm is an ensemble learning algorithm composed of plenty of sub-classifiers to overcome the drawbacks of poor classification of individual sub-classifiers [26,27]. The core idea is to train different sub-classifiers $h_t, t \in \{1, 2, \cdots, T\}$ on the same training samples. As one kind of weak classifier, the DS is often used as sub-classifier in the AdaBoost algorithm [28,29], and these weak sub-classifiers are grouped together to construct a final stronger classifier. In Figure 4, we present the flowchart of the AdaBoost algorithm.

As a combination of multiple sub-classifiers, the AdaBoost algorithm can be used for classification by choosing suitable sub-classifiers. The conventional AdaBoost algorithm adopts DS as sub-classifiers and it is proved to be efficient for many applications. However, complexity of this pure DS based structure is $\mathcal{O}(LT)$, and clearly it would spend more time to train and predict with the increase of the number of training samples and iteration rounds.



Initial sample weights
$D_1(l) = \frac{1}{L},\ l = 1,2,\cdots,L$
Train sub-classifier $h_t(\mathbf{x}^{(l)}), t = 1,2,\cdots,T$
Minimize the objective function
$\varepsilon_t = \sum_{l=1}^{L} D_t(l) I(h_t(\mathbf{x}^{(l)}) \neq c^{(l)})$
Calculate the weight of the t-th sub-classifier
$\alpha_t = \frac{1}{2}\ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
Update sample weights
$D_{t+1}(l) = \frac{D_t(l)\exp(-\alpha_t c^{(l)} h_t(\mathbf{x}^{(l)}))}{\sum_{l=1}^{L} D_t(l)\exp(-\alpha_t c^{(l)} h_t(\mathbf{x}^{(l)}))}$
Predict
$\hat{c}^{(m)} = H(\mathbf{x}^{*(m)}) = \mathrm{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}^{*(m)}))$

Figure 4. Flowchart of the adaptive boosting (AdaBoost) classification algorithm.

5. Hybrid AdaBoost Classification Algorithm

In this section, we propose a hybrid AdaBoost algorithm which uses a strong ML algorithm and multiple DS as sub-classifiers. The philosophy of employing both the strong ML algorithm and the DS lies in the feature of non-misclassification on training samples of decision stumps based AdaBoost algorithm and the feature of high classification accuracy of the strong ML algorithm. The hybrid AdaBoost algorithm adopts a strong ML algorithm to be a first sub-classifier and it is supposed to obtain the same classification result as that of many DS classifiers jointly operating together. Therefore, it decreases the number of iteration rounds and demands lower computational complexity than the conventional AdaBoost algorithm. As expected and verified, the proposed hybrid structure of the sub-classifiers in AdaBoost helps increase the detection probability and reduces the time consumed in operations. In our work, the aforementioned typical ML based strong classifiers (e.g., SVM and KNN) and clustering algorithm (e.g., K-Means) are first considered for serving as sub-classifiers in constructing a comprehensive classification algorithm to achieve better performance. We must point out that only cluster 1 or cluster 2 are obtained by K-Means and are not able to get the labels of data. But K-Means can still be used here in AdaBoost as the first sub-classifier, as long as we set cluster 1 and cluster 2 as label -1 and label 1 respectively.

To be more specific, we assign the initial weight D_1 to each training sample as $1/L$ in the first step. A strong ML classifier is adopted as the first sub-classifier and multiple DS are used as the following sub-classifiers (i.e., $h_t = h_{\mathrm{DS}}, t = 2,3,\cdots,T$). After hybrid AdaBoost classification is performed, we can obtain the weight of each sub-classifier as follows:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right), \tag{20}$$

where ε_t is the classification error at the t-th AdaBoost round by using the t-th sub-classifier and it is defined as:

$$\varepsilon_t \triangleq \sum_{l=1}^{L} D_t(l) I(h_t(\mathbf{x}^{(l)}) \neq c^{(l)}), \tag{21}$$

where $h_t(\mathbf{x}^{(l)})$ is the predicted class of $\mathbf{x}^{(l)}$ by classifier h_t and $D_t(l)$ denotes the l-th training sample's weight at the t-th AdaBoost round.

In the proposed hybrid AdaBoost algorithm, the correctly classified sample's weight increases and the falsely classified sample's weight decreases at the end of each AdaBoost round, with the weights updated as:

$$D_{t+1}(l) = \frac{D_t(l) \exp(-\alpha_t c^{(l)} h_t(\mathbf{x}^{(l)}))}{Z_t}, \tag{22}$$

where Z_t presents the normalization factor which is defined as:

$$Z_t \triangleq \textstyle\sum_{l=1}^{L} D_t(l) \exp(-\alpha_t c^{(l)} h_t(\mathbf{x}^{(l)})). \tag{23}$$

Since a strong ML classifier's classification performance is generally better than that of the single DS, the weight of the strong ML classifier α_1 classifier is usually greater than that of the other single DS classifiers $\{\alpha_2, \alpha_3, \cdots, \alpha_T\}$. Finally, these sub-classifiers are linearly combined to obtain a final classifier to predict the spectrum availability label on the basis of the test samples $\mathbf{x}^{*(m)}$. The classification result for the CSS is then obtained as:

$$\hat{c}^{(m)} = H(\mathbf{x}^{*(m)}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}^{*(m)})), \tag{24}$$

where $H(.)$ represents the hybrid AdaBoost operator.

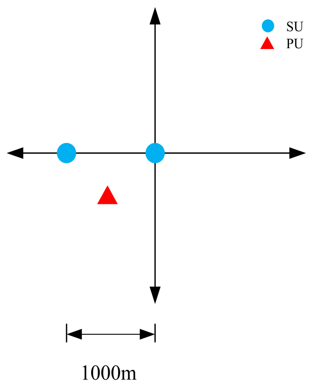
Specifically, the proposed AdaBoost algorithm based on the hybrid structure of a strong ML classifier and multiple DS is described in Algorithm 1.

Algorithm 1 A Strong Machine Learning Classifier and Decision Stumps Based Hybrid Adaboost Classification Algorithm.

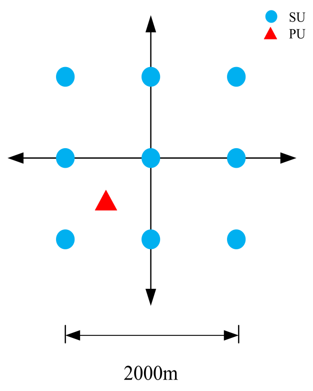
```
Require: $\bar{\mathbf{X}}$, $\{c^{(1)}, c^{(2)}, \cdots, c^{(L)}\}$, $\mathbf{X}^*$, $T$(number of sub-classifiers).
Ensure: $\hat{c}^{(m)}, m \in 1, 2, \cdots, M$.
 1: Initialize sample weight   $D_1(l) = \frac{1}{L}$
 2: $\varepsilon_1 = \sum_{l=1}^{L} D_1(l) I(h_1(\mathbf{x}^{(l)}) \neq c^{(l)})$
 3: $\alpha_1 = \frac{1}{2} \ln(\frac{1-\varepsilon_1}{\varepsilon_1})$
 4: $D_2(l) = \frac{D_1(l) \exp(-\alpha_1 c^{(l)} h_1(\mathbf{x}^{(l)}))}{Z_1}$
 5: for $t = 2$ to $T$ do
 6:     $\varepsilon_t = \sum_{l=1}^{L} D_t(l) I(h_t(\mathbf{x}^{(l)}) \neq c^{(l)})$
 7:     $\alpha_t = \frac{1}{2} \ln(\frac{1-\varepsilon_t}{\varepsilon_t})$
 8:     $D_{t+1}(l) = \frac{D_t(l) \exp(-\alpha_t c^{(l)} h_t(\mathbf{x}^{(l)}))}{Z_t}$
 9: end for
10: Predict the class of the $m$-th test sample   $\hat{c}^{(m)} = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}^{*(m)})\right)$
```

6. Simulations Results

In this section, the performance of the proposed hybrid AdaBoost classification algorithm is evaluated in computer simulations. We consider both a small scale CRN and a large scale CRN and the layout of a small scale CRN with two SUs participating in CSS is shown in Figure 5a, where the SU are located respectively at (−1, 0) km and (0, 0) km and the PU at (−0.6,−0.6) km. The large scale layout with nine SU is shown in Figure 5b, where all the SU are uniformly scattered within a 2.0 km × 2.0 km square area and the single PU is located at (−0.6,−0.6) km.



(**a**) Small scale CRN



(**b**) Large scale CRN

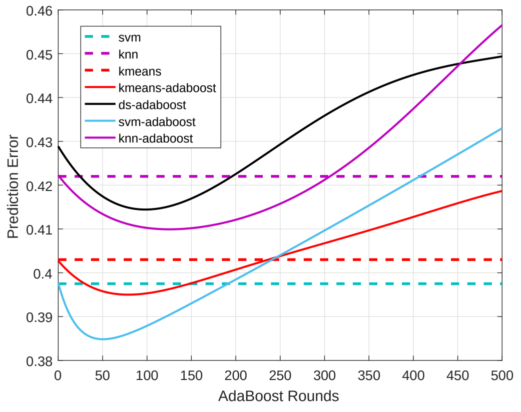
Figure 5. Layout of the cognitive radio networks (CRN).

The PU signal bandwidth W is fixed as 5 MHz, the sensing duration τ is set as 100 µs, and the path-loss exponent ρ is chosen as 4. The shadow fading and the multi-path fading components are set as normal random variables with $E[\psi_n] = E[v_n] = 0$ dB, $\text{Var}[\psi_n] = 1$ dB, and $\text{Var}[v_n] = 5$ dB. In addition, the number of training samples L is 1000.

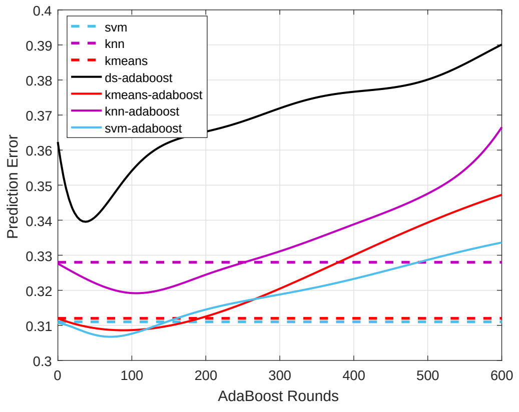
6.1. Prediction Error for Different Classifiers

The prediction errors of different ML technique based CSS algorithms for different sets of cooperative SUs are evaluated and compared in Figures 6 and 7, respectively. It is shown that the prediction error of the proposed hybrid AdaBoost (e.g., SVM-AdaBoost, K-Means-AdaBoost, and KNN-AdaBoost) and the conventional DS-AdaBoost algorithms deteriorate when the number of AdaBoost rounds increases. Although the prediction errors of SVM, KNN, and K-Means algorithms remain consistent, they are inferior to the hybrid AdaBoost algorithms respectively. The prediction error performance is closely related to the number of SUs participating in CSS, since it is shown to be unacceptably deteriorating when the number of the cooperative SUs is as small as two. However, it is apparent that for the number of AdaBoost rounds being approximately 50, the SVM-AdaBoost algorithm achieves the best performance among all the algorithms, regardless of the number of SUs participating in the CSS.

It is important to note that the DS-AdaBoost algorithm and the proposed hybrid AdaBoost algorithms both encounter the overfitting problem when the number of AdaBoost rounds is unnecessarily increased, which makes the prediction error increase. In particular, the overfitting point of both the DS-AdaBoost and the hybrid AdaBoost are approximately around 100 rounds. Through this case, it is suitable to choose the number of iterations less than 100 in hybrid AdaBoost. It is verified that the SVM-AdaBoost algorithm achieves the lowest prediction error among all hybrid AdaBoost algorithms, outperforming all the other algorithms investigated in this paper.

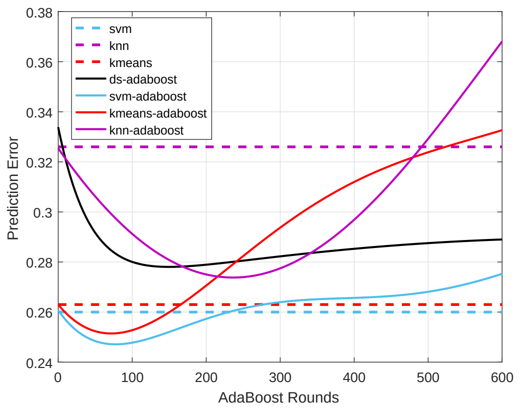


(**a**) SNR = −2 dB

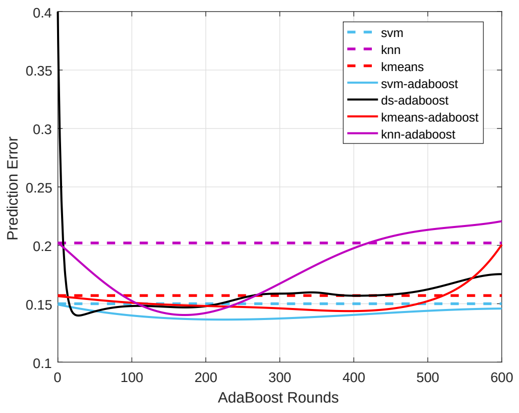


(**b**) SNR = 1 dB

Figure 6. Prediction error when two secondary users (SUs) participate in cooperative spectrum sensing (CSS). K-nearest neighbors (KNN), decision stumps (DS).



(**a**) SNR = −2 dB



(**b**) SNR = 1 dB

Figure 7. Prediction error when nine SUs participate in CSS.

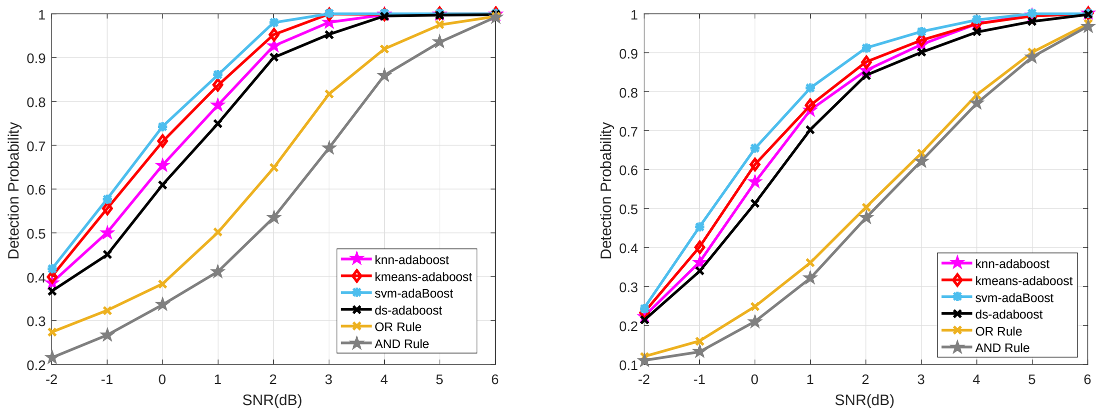
6.2. Detection Probability for Different Classifiers

In addition to the prediction error, we also compare the detection probability performance of the conventional CSS schemes, e.g., hard decision fusion based AND and OR methods, and the hyrbid AdaBoost classifiers based CSS algorithms, as shown in Figure 8. The results are obtained, with 10-fold cross-validation, for a desired false alarm probability of 10% when nine and two SUs participate in CSS, respectively. Simulation results show that all the ML classifiers based CSS schemes studied in this paper outperform the traditional logical based CSS methods. Moreover, the proposed SVM-AdaBoost scheme performs the best among all the CSS schemes.

6.3. Training Duration and Prediction Duration

The training duration for different classifiers with different numbers of training samples are shown in Table 1. The training duration is reasonably increased with the increase of the number of training samples. The DS-AdaBoost shows the longest training duration (5.6157 s for 1000 training samples) among all these algorithms, whereas the proposed hybrid AdaBoost algorithms take relatively lower training durations. The proposed hybrid AdaBoost algorithms take less training time than the DS-AdaBoost algorithm, due to the high classification accuracy of the strong classifier that replaces the role of some DS in DS-AdaBoost and accelerates the training processing. The K-means classifier has the capability of detecting the spectrum availability more agilely in comparison to other ML classifiers.

The times taken for deciding the spectrum availability for different classifiers with different numbers of test samples are shown in Table 2. It is clear to observe that for the same test samples readily given, the prediction durations of all classifiers increase more or less with the increase of the number of test samples and the KNN algorithm takes the longest time for determining the spectrum availability. In Table 2, it is also easy to notice that the SVM-AdaBoost algorithm takes the lowest time and the KNN-AdaBoost takes the longest time among the proposed hybrid AdaBoost algorithms when the number of test samples is the same.



(**a**) Nine SUs participate in detection (**b**) Two SUs participate in detection

Figure 8. Detection probability with desired false alarm probability 10% and multiple SUs in CSS.

Table 1. Training duration for different classifiers.

Classification Method	Number of Training Samples					
	100	200	300	400	500	1000
SVM	0.0203	0.0254	0.0320	0.0391	0.1207	0.1466
K-means	0.0145	0.0168	0.0171	0.0178	0.0205	0.0542
KNN	0.0421	0.0441	0.0466	0.0502	0.0543	0.0967
DS-AdaBoost	5.0670	5.1078	5.1955	5.2169	5.3562	5.6157
SVM-AdaBoost	4.9518	4.9538	4.9640	4.9767	4.9857	5.0912
K-means-AdaBoost	4.9343	4.9423	4.9478	4.9498	4.9585	5.0053
KNN-AdaBoost	5.1258	5.1299	5.1325	5.1386	5.1756	5.2287

Table 2. Prediction duration for different classifiers.

Classification Method	Number of Test Samples					
	100	200	300	400	500	1000
SVM	1.156×10^{-3}	1.069×10^{-3}	1.219×10^{-3}	1.267×10^{-3}	1.752×10^{-3}	3.146×10^{-3}
K-means	2.12×10^{-2}	2.12×10^{-2}	2.12×10^{-2}	2.13×10^{-2}	2.13×10^{-2}	2.15×10^{-2}
KNN	1.75×10^{-1}	1.76×10^{-1}	1.77×10^{-1}	1.78×10^{-1}	1.82×10^{-1}	2.31×10^{-1}
DS-AdaBoost	4.94×10^{-2}	8.50×10^{-2}	9.74×10^{-2}	1.45×10^{-1}	2.19×10^{-1}	3.35×10^{-1}
SVM-AdaBoost	1.65×10^{-2}	1.65×10^{-2}	1.66×10^{-2}	1.66×10^{-2}	1.68×10^{-2}	1.7×10^{-2}
K-means-AdaBoost	2.78×10^{-2}	2.78×10^{-2}	2.78×10^{-2}	2.80×10^{-2}	2.80×10^{-2}	2.83×10^{-2}
KNN-AdaBoost	1.76×10^{-1}	1.76×10^{-1}	1.77×10^{-1}	1.79×10^{-1}	1.83×10^{-1}	2.32×10^{-1}

7. Conclusions

In this paper, hybrid AdaBoost classification algorithms are proposed on the basis of different sub-classifiers, composed of a strong ML classifier and multiple weak DS, for cooperative spectrum sensing in cognitive radio networks. During spectrum sensing operation, the energy vectors collected from the cooperative SUs are used as the feature vectors to determine the spectrum availability for the SU. The proposed hybrid AdaBoost algorithms achieve lower prediction errors than the conventional DS based AdaBoost algorithm. Meanwhile, among various hybrid AdaBoost algorithms, SVM-AdaBoost exhibits the best performance in terms of prediction error and detection probability, in comparison with other hybrid AdaBoost algorithms and with conventional cooperative spectrum sensing methods. With the salient performance and technical merits, the SVM-AdaBoost algorithm may serve as a practical machine learning based solution for cooperative spectrum sensing.

Author Contributions: Conceptualization, B.S.; Data curation, S.C.; Formal analysis, S.C.; Investigation, B.S. and S.-J.Y.; Software, S.C. and X.W.; Supervision, B.S.

Funding: This research was funded by The National Nature Science Foundation of China (61571073).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Marzetta, T. L. Noncooperative cellular wireless with unlimited numbers of base station antennas. *IEEE Trans. Wireless Commun.* **2010**, *9*, 3590–3600. [CrossRef]
2. Axell, E.; Leus, G.; Larsson, E.G. Overview of spectrum sensing for cognitive radio. In Proceedings of the IEEE International Workshop on Cognitive Information Processing, Elba, Italy, 14–16 June 2010; pp. 322–327.
3. Axell, E.; Leus, G.; Larsson, E.G.; Poor, V.H. Spectrum Sensing for Cognitive Radio: State-of-the-Art and Recent Advances. *IEEE Signal Process Mag.* **2012**, *29*, 101–116. [CrossRef]
4. Awin, F.; Esam, A.-R.; Kemal, T. Blind spectrum sensing approaches for interweaved cognitive radio system: A tutorial and short course. *IEEE Commun. Surv. Tutorials* **2019**, *21*, 238–259. [CrossRef]
5. Mosleh. S.; Tadaion, A.A.; Derakhtian, M. Performance analysis of the Neyman-Pearson fusion center for spectrum sensing in a Cognitive Radio network. In Proceedings of the IEEE EUROCON, St.-Petersburg, Russia, 18–23 May 2009; pp. 1420–1425.
6. Ghasemi, A.; Sousa, E.S. Spectrum sensing in cognitive radio networks: The cooperation-processing tradeoff. *Wirel. Commun. Mob. Comput.* **2007**, *7*, 1049–1060. [CrossRef]
7. Peh, E.C.Y; Edward, C.Y.; Guan, Y.L.; Zeng, Y. Optimization of cooperative sensing in cognitive radio networks: A sensing-throughput tradeoff view. *IEEE Trans. Veh. Technol.* **2009**, *58*, 5294–5299. [CrossRef]
8. Shilton, A.; Palaniswami, M.; Ralph, D.; Tsoi, A. C. Incremental training of support vector machines. *IEEE Trans. Neural Networks* **2005**, *16*, 114–131. [CrossRef] [PubMed]
9. Agostini, A.; Celaya, E. Reinforcement learning with a Gaussian mixture model. In Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN), Barcelona, Spain, 18–23 July 2010; pp. 1–8.
10. Mangasarian, O.L.; Musicant, D. R. Active support vector machine classification. In Proceedings of the International Conference on Neural Information Processing Systems, Denver, CO, USA, 27 November–2 December 2000; pp. 556–562.
11. Awin, F.A., Yasser, M. Alginahi, E.A.-R.; Kemal, T. Technical issues on cognitive radio-based Internet of Things systems: A survey. *IEEE Access* **2019**, *7*, 97887–97908. [CrossRef]
12. Xiong, H.; Wu, J.; Chen, J. K-Means clustering versus validation measures: A data-distribution perspective. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2009**, *39*, 318–331. [CrossRef] [PubMed]
13. Alam, S.; Moonsoo, K.; Jae-Young, P.; Kwon, G. Performance of classification based on PCA, linear SVM, and Multi-kernel SVM. In Proceedings of the 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), Vienna, Austria, 5–8 July 2016; pp. 987–989.
14. Zhang, S.; Li, X.; Zong, Z.; Zhu, X.; Wang, R. Efficient kNN classification with different numbers of nearest neighbors. *IEEE Trans. Neural Networks Learn. Syst.* **2018**, *29*, 1774–1785. [CrossRef] [PubMed]
15. Duda, R.O.; Hart, P.E.; Stork, D.G. *Pattern Classification*; John Wiley & Sons: New York, NY, USA, 2001; pp. 203–205.
16. Thilina, K.M.; Choi, K.W.; Saquib, N.; Hossain, E. Machine learning techniques for cooperative spectrum sensing in cognitive radio networks. *IEEE J. Sel. Areas Commun.* **2013**, *31*, 2209–2221. [CrossRef]
17. Guo, Y.W. A Survey on spectrum sensing techniques in cognitive radio networks. *Zte Commun.* **2010**, *1*, 34–36.
18. Lu, S.; Yu, H.; Wang, X. Clustering method of raw meal composition based on PCA and Kmeans. In Proceedings of the 2018 37th Chinese Control Conference (CCC), Wuhan, China, 25–27 July 2018; pp. 9007–9010.
19. Hand, D.; Mannila, H.; Smyth, P. *Principles of Data Mining*; MIT Press: Cambridge, MA, USA, 2001; pp. 50–52.

20. Thilina, K.M.; Choi, K.W.; Saquib, N.; Hossain, E. Pattern classification techniques for cooperative spectrum sensing in cognitive radio networks: svm and w-knn approaches. In Proceedings of the Global Communications Conference (GLOBECOM), Anaheim, CA, USA, 3–7 December 2012; pp. 1260–1265.
21. Manjusha, M.; Harikumar, R. Performance analysis of KNN classifier and K-means clustering for robust classification of epilepsy from EEG signals. In Proceedings of the 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), Chennai, India, 23–25 March 2016; pp. 2412–2416.
22. Pandey, P.; Prabhakar, R. An analysis of machine learning techniques (J48 & AdaBoost)-for classification. In Proceedings of the IEEE India International Conference on Information Processing, Delhi, India, 31 July–3 August 2016; pp. 1–6.
23. Wang, R. AdaBoost for feature selection, classification and its relation with SVM, a review. In Proceedings of the Second International Asia Symposium on Intelligent Interaction & Affective Computing & Second International Conference on Innovation Management (ASIA-ICIM), Wuhan, China, 4–5 December 2010; pp. 800–807.
24. Last, M. Kernel Methods for Pattern Analysis. *J. Am. Stat. Assoc.* **2004**, *101*, 1730. [CrossRef]
25. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
26. Schapire, R. A breif introduction to boosting. In Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 6–8 May 1999; pp. 1401–1406.
27. Cheng, S.T.; Bing, X.; Hua, C.L. The application of the AdaBoost algorithm in the text classification. In Proceedings of the 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Xi'an, China, 25–27 May 2018; pp. 1792–1796.
28. Xie, J.; Lu, Y.; Lei Z.; Hui, X. Boosting decision stumps to do pairwise classification. *Electron. Lett.* **2014**, *50*, 866–868.
29. Kawaguchi, S.; Nishii, R. Hyperspectral image classification by bootstrap AdaBoost with random decision stumps. *IEEE Trans. Geosci. Remote Sens.* **2007**, *45*, 3845–3851. [CrossRef]