Abstract—We propose a novel RF signal classification method based on sparse coding, an unsupervised learning method popular in computer vision. In particular, we employ a convolutional sparse coder that can extract high-level features by computing the maximal similarity between an unknown received signal against an overcomplete dictionary of matched filter templates. Such dictionary can be either generated or trained in an unsupervised fashion from signal examples labeled with no ground truths. The computed sparse code then is applied to train SVM classifiers to discriminate RF signals. As a result, the proposed approach can achieve blind signal classification that requires no prior knowledge (e.g., MCS, pulse shaping) about the signals present in an arbitrary RF channel. Since modulated RF signals undergo pulse shaping to aid the matched filter detection by a receiver for the same radio protocol, our method can exploit variability in relative similarity against the dictionary atoms as the key discriminating factor for SVM. We present an empirical validation of our approach. The results indicate that we can separate different classes of digitally modulated signals from blind sampling with 70.3% recall and 24.6% false alarm at 10 dB SNR. If a labeled dataset were available for supervised classifier training, we can enhance the classification accuracy to 87.8% recall and 14.1% false alarm.

I. INTRODUCTION

Cognitive radios have emerged over the last decade as a new means to share radio spectrum, the most expensive resource to build a wireless network. For commercial applications, Dynamic Spectrum Access (DSA) [1] presents a compelling model to improve the utility of radio spectrum resources. Much of contemporary research has viewed cognitive radios as the secondary user of a licensed channel and focused on developing the mechanism to opportunistically access the channel to its maximal spectral efficiency.

While commercial opportunities are promising, the applicability of cognitive radios for tactical networking seems even more adequate. The primary advocate for tactical cognitive radio systems is intelligent decision making that can enhance resiliency against a hostile, fiercely competing radio environment. There has been significant amount of previous work devoted to algorithmic approaches for a cognitive strategy layer, including game-theoretic frameworks [2]–[5] to sequential decision making [6]–[8].

These approaches have provided a strong foundation for cognitive tactical radio systems, yet their performance highly depends on the lower layer capability such as sensing, detection, and inference of radio signals. In order to operate the cognitive strategy layer, our claim is that we require intelligent sensing mechanisms enabled by learning. In this paper, we focus on the development of such mechanisms. Particularly, we use sparse coding [9], a feature learning technique widely used in machine learning, to perform blind and semi-supervised signal classification for cognitive radios.

Our methods are new and unconventional to the field of signal detection and estimation. Our methods can learn over time after bootstrapping with no prior knowledge about RF signals of interest and achieve a 72.6% recall for blind signal classification under a reasonably good SNR. If a labeled dataset were available for semi-supervised training, our classifiers would have achieved a 87.8% recall with 14.1% false alarms, all without any protocol-specific knowledge about modulation of radio signals.

The rest of the paper is organized as follows. In Section II, we provide a comprehensive background on sparse coding. In Section III, we describe a discriminative framework that employs sparse coding as the primary means to extract features from raw data in a powerful classification pipeline. Section IV presents our RF signal classification methods. We propose a method for blind signal classification before presenting a semi-supervised approach under the availability of a labeled dataset. We evaluate the proposed classification methods in Section V. In Section VI, we discuss related work, and Section VII concludes the paper.

II. SPARSE CODING BACKGROUND

This section presents a background on sparse coding and dictionary learning.

A. Sparse Coding

Sparse coding [9] is an unsupervised method to learn a dictionary of overcomplete bases that represent data efficiently. Each basis vector in the dictionary is also known as an atom. The mathematical objective of sparse coding is to describe an input vector as a sparse linear combination of the basis vectors from the dictionary.

Fig. 1 explains the sparse coding problem. Given an $N$-dimensional input $x \in \mathbb{R}^N$ and dictionary $D \in \mathbb{R}^{N \times K}$, sparse coding seeks for a sparse representation $y \in \mathbb{R}^K$ that minimizes the loss function

$$J(x, D) = \min_{y \in \mathbb{R}^K} \frac{1}{2} \|x - Dy\|_2^2 + \lambda \psi(y),$$  \hspace{1cm} (1)
Subsequently, solving problem can be written as Eq. (3), however, is not in the standard dual of a linear program. One approach for this optimization is linear programming [16].

\[
\min_{\mathbf{c}^\top} \mathbf{c}^\top \mathbf{y} \quad \text{s.t.} \quad D \mathbf{y} = \mathbf{x}, \quad \mathbf{y} \geq 0.
\]

Chen, Donoho & Saunders [11] recommend make the following translations

\[
\mathbf{y} \leftrightarrow (\mathbf{u}, \mathbf{v}), \quad \mathbf{c}^\top \leftrightarrow (\mathbf{1}^\top, 1^\top), \quad D \leftrightarrow (D, -D).
\]

Subsequently, solving

\[
\min \mathbf{u} + \mathbf{v} \quad \text{s.t.} \quad D \mathbf{u} - D \mathbf{v} = \mathbf{x}, \quad \mathbf{u}, \mathbf{v} \geq 0
\]
gives the $\ell_1$-minimization solution via linear programming.

### B. Dictionary Learning

How can we learn a dictionary for sparse coding? A dictionary is trained by an unsupervised learning algorithm such as K-means clustering. A classical approach [17] examines the projected first-order stochastic gradient descent in a sequence of updates for $D$

\[
\frac{D_t}{\Pi_C} = D_{t-1} - \frac{\mathbf{\rho}}{t} \nabla J(x_t, D_{t-1}),
\]

where $\mathbf{T}$ is the gradient step, $\Pi_C$ is the orthogonal projector on $C$, and unlabeled training examples $\{x_k\}_{k=1}$.

In principal component analysis (PCA), we learn a complete set of basis vectors—i.e., a square matrix of eigenvectors. Dictionary learning for sparse coding aims to learn an overcomplete set of basis vectors such that the column dimension of $D$ is larger than its row dimension. (Recall $D \in \mathbb{R}^{N \times K}$, so $K \ll N$.) The advantage of having an overcomplete bases is that we can better capture structures and patterns inherent in the input data more conveniently.

K-SVD [18] is a fast iterative algorithm for PCA-like basis learning. The inner loop of K-SVD has two phases. First, it performs batch sparse coding with current dictionary. Using the notation $X = [x_1 \ldots x_T]$, the batch sparse coding yield the corresponding matrix of sparse codes $Y = [y_1 \ldots y_T]$ such that $X = DY$. In the next phase, K-SVD updates each dictionary atom in $D$ by rank-1 update via singular value decomposition of residual matrix for the atom. K-SVD also updates each sparse code in $Y$ accordingly. The K-SVD optimization is given by

\[
\min_{D, Y} \|X - DY\|_F^2 \quad \text{s.t.} \quad \|y_k\|_0 \leq S \forall k.
\]

Because of the batch sparse coding phase, K-SVD requires a sparse coder. We can use OMP, $\ell_1$-minimization via linear programming or LASSO.

### III. DISCRIMINATIVE SPARSE CODING FRAMEWORK

In this section, we present a discriminative sparse coding framework to build a high-performance classification pipeline. We explain unsupervised feature learning method based on sparse coding and dictionary learning. Given the learned feature mapping, we describe how we can perform feature extraction, train classifiers, and predict a class label.

#### A. Unsupervised Feature Learning via Sparse Coding

Typically, an unsupervised method is used to learn a feature representation of raw data. Since the feature mapping should be generally applicable and descriptive of all classes of data, feature learning takes in randomly mixed, unlabeled training examples. Sparse coding and dictionary training provide an unsupervised feature learning algorithm that consists of the following steps as illustrated in Fig. 2:

1. Form input patches $x$ from measured/received signal data that are unlabeled of their classes;
2. (Optionally) apply preprocessing such as normalization and whitening;
3) Learn a feature-mapping via joint sparse coding (compute $y$) and dictionary ($D$) training.

In summary, unsupervised feature learning takes the unlabeled dataset $X = \{x_1 \ldots x_T\}$ of random input patches (each $x_i$ with dimension $N$), undergoes sparse coding and dictionary learning, and yield a function $f_{ext} : \mathbb{R}^N \rightarrow \mathbb{R}^K$. The transformation via $f_{ext}$ converts the raw data input $x$ to sparse code $y$ in the feature space learned by sparse coding and dictionary training. For classification, we use the sparse code $y$ as a feature vector whose $K$ elements are features of the input $x$ according to dictionary $D$.

![Fig. 2. Unsupervised learning via sparse coding and dictionary training](image)

**B. Supervised Classifier Training**

A representational feature mapping learned from the unsupervised method plays a crucial role for classification tasks. Having the feature mapping alone, however, is usually insufficient to classify data. Classifiers take a feature vector as the input, and they should be instructed with the ground truth class (i.e., supervision) about the feature inputted. Therefore, classifier training is typically done by a supervised method such as logistic regression [19] and support vector machine (SVM) [20]. Supervised classifier training is depicted in Fig. 3. Note the labeled input $\{x_i, l_i\}$, where $l_i$ designates the class label for an input $x_i$.

![Fig. 3. Supervised classifier training](image)

Under the context of tactical networking scenarios, it may be too optimistic to assume the availability of labeled dataset for supervised classifier training. This is because of the null a priori assumption for an adversarial radio network. Signal examples of the adversary could be hard to acquire for pre-analysis before a field operation. However, we can assume a plenty of signal examples for the friendly network. With only friendly network signal examples, one can employ one-class classifier [21] instead.

**C. Subsampling Features with Max Pooling**

In Fig. 3, feature vectors (i.e., sparse code $y$) go through one more processing step known as max pooling before being inputted to a classifier under training. If all feature vectors resulted from a stream of input vectors were used straightforwardly for classification, we could overwhelm the classifier training. The dimensionality of feature vectors is highly correlated with the complexity of classifiers. Usually, a complex classification model leads to classifier overfit, which is the discrepancy in the classification results between the training and test datasets. It is therefore customary to reduce the number of extracted features by subsampling.

Pooling, popular in convolutional neural networks [22], operates over multiple (sparse) feature representations and aggregates to a higher level of features in reduced dimension. Pooling is by no means to discard any useful information. An important property of the pooled feature representation is translation invariance. Max pooling [23] takes the maximum value for the elements in the same position over a group of feature vectors. For example, consider max pooling of $L$ sparse codes $\{y_1, y_2, \ldots, y_L\}$ that yields the pooled feature vector $z$ as in Fig. 4. Noting $y_k = [y_{k,1} \ldots y_{k,K}]$ and $z = [z_1 \ldots z_K]$, max pooling operation is given by $z_j = \max(y_{1,j}, y_{2,j}, \ldots, y_{L,j})$.

![Fig. 4. Max pooling of $L$ sparse codes](image)

**IV. RF Signal Classification with Convolutional Sparse Coder**

This section introduces a new method for RF signal classification based on feature extraction via sparse coding. We consider two case scenarios. In the first scenario, we consider that there is no labeled dataset for supervised classifier training. Here, we completely rely on unsupervised learning by sparse coding and dictionary training. The first scenario can be considered as blind source separation in the feature domain. In the second scenario, our approach is based on the semi-supervised learning framework.

**A. Sparse Coding Stage**

Sparse coding is a customizable feature extractor. The learned features represent a high-level structure of the raw signal examples. The sparse coding setup in Fig. 1 is a realization based on matching or basis pursuits that emphasize reconstructive representation with the regularization on sparsity. For discriminative purposes, an OMP sparse coder evaluates the membership of a given input $x$ to each dictionary atom with the inner product. Conceptually, this is equivalent to the way that K-means clustering evaluates the Euclidean
distance between data and a cluster centroid, or that the Gaussian mixture model computes the posterior probabilities given \( x \).

Since our eventual goal is classification, we want to optimally configure the sparse coding framework with the most suitable metric that examines the correlation of a received RF measurement to our dictionary atoms. We propose a convolutional sparse coder that maps an input vector \( x \in \mathbb{C}^N \) (samples of received signal) to the feature \( y \in \mathbb{R}^R \) with respect to the matched filter templates in dictionary \( D \). The \( i \)th element in \( y \) is given by

\[
y_i = \max \left| x \ast d_i \right|,
\]

where \( \ast \) denotes the convolution operator, and \( d_i \) the \( i \)th dictionary atom. We impose the similar regularization for \( y \), leaving only the \( S \) largest values of \( y \) as are and setting the rest zeros.

The underlying principle behind our setup is matched filtering. Mathematically, the nonzero element in the convolutional sparse code \( y \) reflects the maximum correlation between the input \( x \) and the corresponding dictionary atom, which is some matched filter.

What are the matched filter templates that constitute the dictionary? Radio protocols employ specific pulse shaping functions to aid effective detection of known signals for the receiver. According to detection and estimation theory, an optimal matched filter design leverages a priori knowledge about the pulse shaping function to maximize SNR.

**B. Signal Classification Stage**

Essentially, our approach is a semi-supervised pipeline. We have performed unsupervised feature learning via sparse coding and dictionary training. Partly, we generate dictionary atoms by taking variations of a well-known pulse shape filter such as rectangular, Gaussian, and raised cosine. The rest of the dictionary atoms come from training with example signals. These examples are received from various RF channels used to transmit modulated wireless signals. We use K-SVD [18], a generalization of the K-means clustering algorithm, to train the sparse coding dictionary. We note that this unsupervised dictionary learning step essentially performs blind (signal) source separation in the signal’s feature (i.e., sparse code) domain.

Given the learned dictionary, we train linear 1-vs-all SVM classifiers. Assuming a multiclass classification problem with \( M \) classes, each SVM is trained to classify signal class \( j \) against class \( k \neq j \) \( \forall j, k = 1, \ldots, M \). During the runtime test, we take sample measurements, perform sparse coding and subsampling of sparse codes by max pooling, and predict the signal class label using the pooled sparse code with the trained SVMs.

**V. Evaluation**

We evaluate the proposed blind and semi-supervised signal classification methods in MATLAB.

**A. Signals**

We assume target signals S1 and S2, and non-target signals S3 and S4, mixed in radio channels of a given bandwidth. Blind sampling at the receiver is triggered by an energy level above certain threshold. We sample the channel according to its bandwidth and stores the measured samples for further processing. By target signals, we assume that we have sufficient amount of received examples for S1 and S2. More importantly, we do not have the actual specification (e.g., modulation scheme, pulse shaping) about S1 and S2. If the exact knowledge about pulse shaping were given, we would have performed matched filter detection. The target signals are defined below.

- (S1) Single-carrier: QPSK with rectangular pulse;
- (S2) OFDM: QPSK modulated on-carriers with raised cosine pulse.

The non-target signals S3 and S4 have the following specifications. Once again, we do not have any knowledge about S3 and S4. Moreover, there is no assurance that we have sufficient number of examples for S3 and S4.

- (S3) Single-carrier: QPSK with unknown custom pulse \( p(t) = \frac{1}{2}[1 - \cos(\frac{\pi t}{T})] \);
- (S4) OFDM: BPSK, QPSK, 16-QAM modulated on-carriers with \( p(t) \).

**B. Experimental Methodology**

1) Generation and transmission of signals: To generate signals, we have first generated random data bit stream \( b_k \). The baseband signals are generated according to the following digital (I-Q) modulation schemes.

- BPSK: \( d_{\text{BPSK}}(t) = \sum_k b_k p(kT_b) \)
- QPSK: \( d_{\text{QPSK}}(t) = \sum_k b_{2k} p(kT_s) + \sum_k b_{2k+1} p(t - kT_s) \)
- 16-QAM: \( d_{\text{QAM}}(t) = \sum_k i_k p(kT_s) + \sum_k q_k p(kT_s) \)
- OFDM: generated by \text{comm.OFDMModulator} method in MATLAB

For 16-QAM, \( i_k, q_k \) are the in-phase and quadrature amplitudes, taking values \( \pm 1, \pm 3 \).

As mentioned earlier, we use rectangular, raised cosine, square-root raised cosine, and custom pulse functions for baseband pulse shaping of the baseband modulated waveforms. The final carrier-modulated signal is given by

\[
s(t) = A_c [d_I(t) \cos(2\pi f_c t) + d_Q(t) \sin(2\pi f_c t)],
\]

where \( f_c \) is the carrier frequency, and \( A_c \) the carrier amplitude gain. The in-phase and quadrature components \( d_I(t), d_Q(t) \) are generated according to one of the I-Q modulation schemes above.

We transmit \( s(t) \) through the AWGN channel at 20 dB and 0 dB SNR. Hence, the measurement at a receiver constitutes noisy samples. For each signal class, we generate two datasets \text{train} and \text{eval}. We use \text{train} for sparse coding and SVM training, and \text{eval} to test classification performances. There are 1,000 training examples in \text{train}, whereas \text{eval} contains 10,000 test examples for each signal class. Thus, we
are evaluating a scenario where the size of training dataset is substantially smaller than the test set.

2) Data processing and classification pipeline: The data processing and classification pipeline is depicted in Fig. 5. The measured signal samples are vectorized in a size \( N = 64 \). Note that an I-Q modulated signal is complex-valued, hence the received samples are also complex, i.e., \( x \in \mathbb{C}^{64} \). We can train the dictionary using the received samples via unsupervised K-SVD algorithm (without knowing what ground-truth classes are). We summarize how we partition the sparse coding dictionary \( D \), which has generated and learned dictionary atoms.

1) \( D \) has \( K = 256 \) dictionary atoms (i.e., \( D \in \mathbb{C}^{64 \times 256} \))
2) Each atom \( d_i \) has matching size \( N = 64 \) and is complex-valued
3) \( D \) is divided to 4 regions—first 30 atoms belong to the family of rectangular pulses, second 30 atoms to raised cosine family, third 30 atoms to square-root raised cosine; the last 166 atoms are trained

We use a convolutional sparse coder whose operation is described by Eq. (6) given a patch \( x \) of the received signal samples. Sparse code has a dimension \( K \) and is real-valued, i.e., \( y \in \mathbb{R}^{256} \). We use the max pooling factor \( M = 10 \). Note that the pooled feature vector \( z \) has the same dimension as \( y \) and is also real.

In summary, the feature transformation \( x \mapsto y \mapsto z \) takes place by sparse coding of sequentially-fed raw input patches followed by max pooling. The pooled feature vector \( z \) is used for classification.

3) SVM classifier training: Fig. 6 explains 1-vs-all SVM training. We have trained two linear SVM classifiers using the pooled feature vectors \( z \). The first SVM classifies the signal class \( S_1 \) against all others. For this, we prepare labeled datasets \( \{z_{S_1}^{(j)}, +1\}_{j=1}^T \) and \( \{z_{S_2,S_3,S_4}^{(j)}, -1\}_{j=1}^{T'} \). (Note that \( T \) and \( T' \) are the number of examples for target and non-target signals, respectively, for the SVM.) Similarly, the second SVM that classifies \( S_2 \) against all others are trained with labeled datasets \( \{z_{S_2}^{(j)}, +1\}_{j=1}^T \) and \( \{z_{S_1,S_3,S_4}^{(j)}, -1\}_{j=1}^{T'} \).

4) Evaluation metric: We compute recall or true positive rate (TPR) and false alarm rate (FPR) to evaluate classification accuracy.

\[
\text{Recall} = \frac{\sum \text{True positives}}{\sum \text{True positives} + \sum \text{False negatives}}
\]

\[
\text{False alarm} = \frac{\sum \text{False positives}}{\sum \text{False positives} + \sum \text{True negatives}}
\]

\[
\{z_{S_1}^{(j)}, +1\} \quad \text{Linear SVM}
\]

\[
\{z_{S_2}^{(j)}, +1\} \quad \text{Linear SVM}
\]

\[
\{z_{S_2,S_3,S_4}^{(j)}, -1\}, -1\} \quad \text{Linear SVM}
\]

C. Results and Discussion

For blind classification, we try to see whether or not K-means clustering in the pooled feature (\( z \)) domain gives natural separation. Setting the number of clusters \( K = 4 \) for K-means (not to be confused with the number of atoms \( K \) in dictionary \( D \)), we have been able to see the separation that we seek for. By counting the cluster majority signal class and minorities, we have computed the recall and false alarms. Table I summarizes the accuracy of the blind signal classification via K-means.

The blind classification performance is reasonably good considering that we do not use any prior knowledge about these signals, yet we can achieve up to a 70.3% recall at 24.6% false alarm at SNR = 10 dB. If a labeled dataset were available for semi-supervised training, our classifiers would have achieved a 87.8% recall with 14.1% false alarms, all without any protocol-specific knowledge about modulation of radio signals. In Fig. 7, we present the confusion matrix for 1-vs-all SVM classifier trained using signal sampled received at SNR = 10 dB. Similarly, the confusion matrix for SMV classifier trained under SNR = 0 dB is presented in Fig. 8.

Fig. 6. 1-vs-all SVM training for semi-supervised approach
VI. RELATED WORK

Our signal classification methods are inspired by the way that sparse representations of raw image, audio, and text data are used in computer vision and pattern recognition. Wright et al. [24] have developed a recognition system that can classify an image of human face using sparse representations of image segments, which is a similar idea to ours. Pooling multiple sparse features to make an aggregate representation is widely studied in computer vision. The original idea of spatial pooling techniques dates back to Riesenhuber and Poggio [25]. Heisele, Ho, and Poggio [26] explain useful techniques of applying SVM for multi-class classification such as training a 1-vs-all classifier in our semi-supervised approach.

VII. CONCLUSION

We have introduced a blind signal classification method based on sparse coding. Our method is motivated by an active area of research in sparse representation learning [27]. With no prior knowledge or assumptions on a blindly sampled signal, we take advantage of correlating it to an overcomplete dictionary of known matched filter templates, which can be pregenerated or trained by an unsupervised learning algorithm. This coding process yields a discriminative feature that captures the variability of correlations measured by convolving the signal with respect to each dictionary atom.

As our goal is to build a discriminative framework for classification tasks, we have exploited sparsity by regularizing over the convolutional filter outputs and leaving only the largest several values as are. The empirical results for classification are promising. We have designed a simulated experiment for blind classification similar to blind source separation and found that our method can achieve up to a 70.3% recall at 24.6% false alarm rate at a reasonable SNR of 10 dB without any protocol-specific knowledge about simulated radio signals. If a labeled dataset were available for supervised training, our classifiers would have achieved a 87.8% recall with 14.1% false alarm.

REFERENCES