Classification of Modulations in DSA Networks

Manel Davins
Universitat Politèncinca de Catalunya
Barcelona, Spain
Email: manel.davins@estudiantat.upc.edu

Joan Manel Càrdenas
Universitat Politèncinca de Catalunya
Barcelona, Spain
Email: joan.manel.cardenas@estudiantat.upc.edu

***Abstract*—In this paper we explore several Machine Learning (ML) techniques for classifying modulation types in Dynamic Spectrum Access (DSA) networks. Utilizing a dataset with signals modulated in four different ways and subjected to various channel effects. Utilizing a dataset from real measurements, including signals with AM-DSB, BPSK, GFSK, and QPSK modulations, we address the challenge of accurate modulation classification.**

I. Introduction

The increasing demand for wireless communication services and the scarcity of available frequency bands have led to the rise of DSA as a key solution. DSA enables the dynamic and adaptive use of radio frequency spectrum, with a critical aspect being the detection of primary transmitters to prevent interferences and maintain regulatory compliance. In this paper we will focus on leveraging Machine Learning to identify modulation types in wireless transmissions, a crucial step in effective DSA implementation. This research aims to overcome the limitations of traditional transmitter detection techniques, offering a more accurate and efficient approach for spectrum management in DSA networks.

In our study, we utilize a dataset provided by the UPC from real measurements in a testbed environment. This dataset contained signals modulated using one of four techniques: AM-DSB, BPSK, GFSK, or QPSK, all subjected to various channel effects like multipath fading or noise. The dataset is divided into two parts: a training set with 9600 labeled examples and a test set with 2400 unlabeled examples.

II. Feature Analysis

In this section, we explore the dataset used for this study, focusing on understanding the most important features for recognizing the modulation and assessing potential feature engineering or selection techniques.

A. Dataset description

In this work, the dataset derived from real measurements, where different signals, modulated with one of four modulations are exposed to varying non-deterministic channel effects.Each sample comprises 128 in-phase and 128 quadrature (I/Q) samples, captured over a 128 µs time window, with varying symbol characteristics and noise levels. Notably, the training dataset is completely balanced across these four modulation types, ensuring that each class is equally represented. Additionally, each sample also includes the measured signal-to-noise ratio (SNR).

In Figure 1, we show an example of the in-phase and quadrature samples of received signals for the four different modulations.

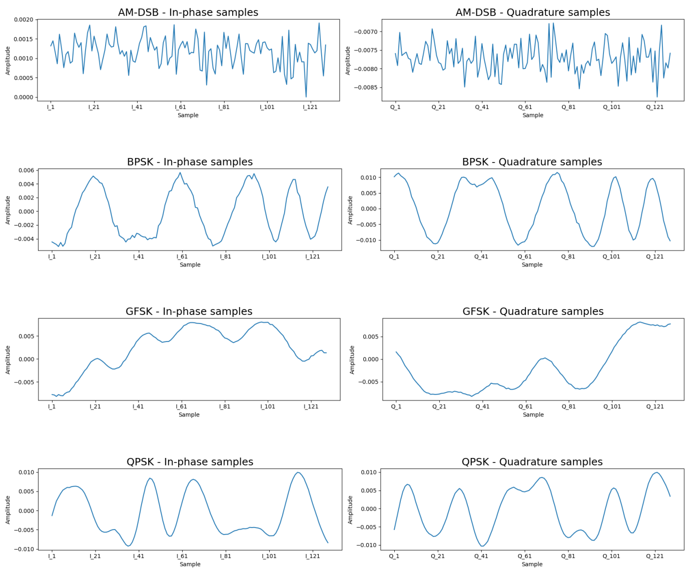

Fig. 1: In-phase and quadrature sample examples for the four modulation types.

As previously mentioned, apart from the in-phase and quadrature samples the dataset also provides the SNR from the signal, which measures the ratio between the signal level to the noise level of the received signal and is a fundamental parameter to assess the quality of the communication link. To study the importance of this feature in the classification, we analyzed its distribution among the four modulations types. We observed that the SNR are limited to the specific values of 10 dB, 16 dB or 18 dB and they are equally distributed among the four classes. This indicates that the SNR may not provide significant discriminatory information for distinguishing between different modulation types and that the I/Q samples will play a more important role in the classification task.

B. Feature engineering

In addition to the original features provided in the dataset, we have applied feature engineering to further expand the original data with informative attributes that can help in the task at hand. The raw I/Q samples represent the signal in the time domain. While this representation captures the signal's temporal dynamics, it does not explicitly reveal frequency-based characteristics. By transforming these I/Q samples into the frequency domain, we can create new features that may help to better identify modulation of the signal.

We have employed the Fast Fourier Transform (FFT) to convert the time-domain I/Q samples into their frequency-domain representation. This method allows us to extract the amplitude of the spectrum and the phase of the signal, and example of which is shown in Figure 2.

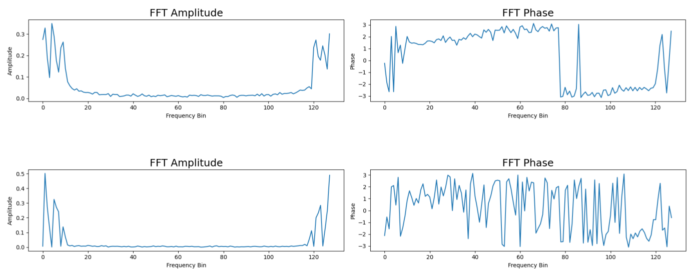

Fig. 2: Examples of resulting amplitude and phase.

C. Feature reduction

After the addition of features based on the Fast Fourier transform, the dimensionality of the dataset has doubled, leading to a large feature space and potential problems such as overfitting or added complexity. In order to simplify the dataset and reduce the computational complexity of training the classification models, we have applied Principal Component Analysis (PCA). PCA is a technique that reduces the dimensionality while preserving as much variance as possible. This process identifies the directions of maximum variance of the original data, and projects the data into these new axes.

With the application of PCA our objective is to have a more manageable dataset while preserving its core information. By configuring PCA to retain 99% of the dataset's variance, we ensure that the transformed data captures the essential characteristics and patterns from the original dataset and, at the same time, it is more approachable and compact. As a result of this process, the size of our dataset has been effectively reduced by half.

III. Classification methods

As shown in the "Feature Analysis" section, the dataset was completely balanced and features did not fit a Gaussian PDF. Therefore, our first approach was to use the methods that provide better results for this classification task which work with an unknown density function for classes such as Support Vector Machines (SVM), Neural Networks (NN) and Decision Trees (DT).

A. Support Vector Machines

Support Vector Machines (SVM) utilize linear discriminants, g(x), to measure and maximize the margin, C, between data points and a separating hyperplane for optimal classification. Support Vectors are the data points closest to the hyperplane and are critical in defining the margin's width. The classification goal is to maximize C, ensuring all Support Vectors lie beyond it. The SVM predicts class labels by the sign of the discriminant function, which is a weighted sum of the input features plus a bias term. For non-linearly separable problems, SVM applies a non-linear kernel transformation to project the data into a higher-dimensional space, facilitating the discovery of non-linear boundaries. The kernel essentially captures the dot product of transformed feature vectors, allowing boundary formation without explicitly knowing the transformation. Decision making in non-linear SVM follows the same principle as the linear case but involves the kernel's computation, leading to the decision function:

$$y = \text{sign}(\boldsymbol{\alpha}^T \mathbf{S} + w_0) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + w_0\right)$$

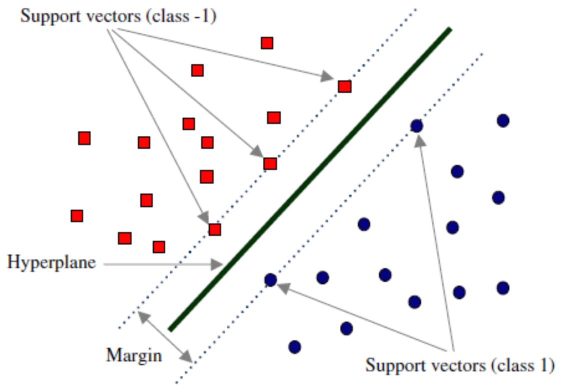

Fig. 3: Support Vector Machine model.

B. Neural Networks

A multi-layer Neural Network, as depicted in Figure 4, functions as a nonlinear entity processing inputs through learned parameters (weights and biases) derived from the training data. The input data are forwarded through various hidden layers before reaching the output layer. This layer is characterized by a softmax activation function, providing normalized probability values for each class.

Additionally, nonlinear functions at each neuron facilitate the creation of complex decision regions. The network employs a loss function, such as logarithmic loss or mean squared error (MSE), to quantify the deviation between predicted outputs and actual labels. To minimize this loss, a backward propagation process adjusts the network's weights and biases, refining the model's accuracy.

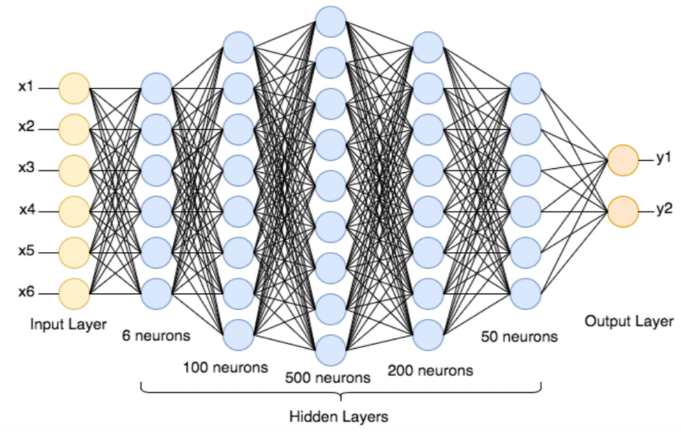

Fig. 4: A deep neural network architecture.

C. Decision trees

Decision Trees are a non-metric classification method that differs from those using real-valued vectors and distance metrics. They are constructed from an initial labeled dataset, forming a tree that iteratively partitions the data into subsets based on feature evaluations, as shown in Figure 5. The goal at each node is to achieve the purest possible split, guided by impurity measures like Entropy, Gini, or misclassification rates.

A subset with uniform class labels becomes a leaf node, signaling a stop in further splitting. Tree growth can be controlled by setting impurity thresholds or applying the Minimum Description Length (MDL) principle. To prevent overfitting, trees may be pruned using various criteria, such as Quinlan's or Reduced Error pruning, especially if the tree becomes too complex.

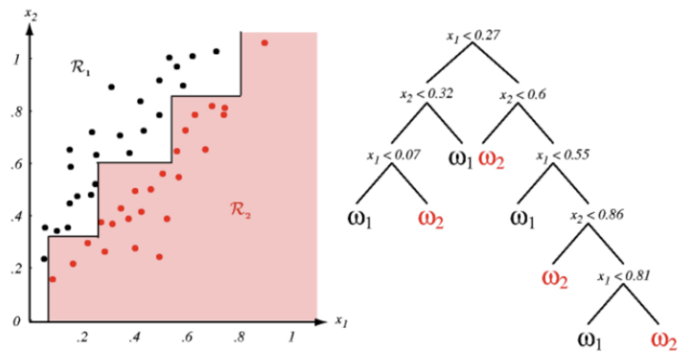

Fig. 5: Simple example of a binary decision tree.

IV. Evaluation metrics

To assess the performance of our classification models, we have primarily used the macro averaged F1-score as the principal evaluation metric, in accordance with the Kaggle competition's standards. This metric, a harmonic mean of precision and recall, provides an averaged evaluation of the precision and recall of each class. The F1-score is defined as:

$$\text{F1-score} = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}}$$

Additionally, we have also used the general accuracy and the confusion matrix as complementary metrics to assess the overall performance of the models and to gain insights into class-level details.

V. Experiments

In this section, we enumerate and briefly describe the experiments that have been done with the base classifiers presented in the 'Classification methods' section. Before delving into the specifics of each experiment, we must remark on some previous steps. The resulting dataset after feature engineering and PCA has been splitted into train and test sets following a proportion of 80% and 20% respectively. Additionally, samples have been scaled following a standardization method, which consists of the samples to have zero mean and the standard deviation of 1. With this process, we remove the influence of scale of the different features and we favor the algorithms to converge faster.

A. Support Vector Machine

To train using an SVM model, we decided to tune hyperparameters C, γ, and the kernel using the GridSearch class from the Scikit-Learn library. This tool systematically evaluates various parameter combinations to determine the most effective one, based on a chosen performance metric. As mentioned in the "Evaluation metrics" section, for this study, we used the macro-averaged F1-score. At the end, the hyperparameter tuning process identified $C = 10$, $\gamma =$ 'scale', and the kernel as 'rbf' as the optimal values.

B. Neural network

During the exploration of neural network architectures, the best results were achieved with convolutional neural networks (CNN) using 1-dimensional convolutions. In this case, to be able to correctly assess the performance of the model, the training set was further partitioned into a validation set, representing 20% of the original training data. In terms of the architecture, we opted to use four convolutional blocks with a final fully connected layer. Additionally, we have used batch normalization to improve the training process and dropout to avoid overfitting. To conclude, with the experimentation details, we used the categorical cross-entropy as the loss function and ADAM [5] as the optimizer.

C. Random Forest

After training using Decision Tree models, we decided to use a different approach with Random Forest models seeking better overall performance. Random Forest models select a subset d' of the d features of the samples and develop a Decision Tree with them using replacement. We used the

Random Forest Classifier implementation of the Scikit-Learn library. Although training with the default parameters showed great results, we used grid-search and cross-validation to tune hyperparameters as mentioned in previous sections with the same evaluation metrics too. The hyperparameters that we tuned were, and its optimal values were:

- `n_estimators`: 250
- `max_depth`: 'None'
- `min_samples_split`: 5
- `min_samples_leaf`: 1

D. Boosting

Boosting is an ensemble technique that builds a series of models in a sequential manner, with each new model focusing on correcting the errors made by the previous ones. In this work, we focused on XGBoost [3], a variant of boosting that has historically provided high performance in a diverse set of tasks. Particularly in this experiment, the best results were achieved using the original dataset and the feature-engineered attributes, without PCA nor feature normalization. This model is capable of handling and leveraging the complexity of the dataset. To optimize the model, we conducted a grid search using cross-validation to determine the most effective hyperparameters, and the obtained parameters are:

- `learning_rate`: 0.1
- `n_estimators`: 300
- `subsample`: 0.9

E. Ensemble

In our final experiment, we evaluated the efficacy of the Ensemble method. This technique combines multiple models, treated as weak learners, to produce a single predictive model more powerful and accurate than any of the individuals. The idea is that by combining different models, the ensemble can balance out individual weaknesses of each component model leading to better overall performance. We employed soft voting, which computes the average estimated class probabilities of each classifier and picks the class with highest probabilities.

VI. Results

In this section, we present the outcomes of the experiments conducted in this study. Table I provides a detailed breakdown of the F1-scores for the best-performing models.

As detailed in Table I, all the models obtain an F1-score that exceeds 0.95 in the test set, which indicates generally high performance. Specifically, the SVM and RF models, despite achieving perfect scores during the training phase, suffer a small drop in test performance. These results suggest a strong learning capacity but a slight lack of generalization. In terms of the NN, the results show a small improvement, showcasing its ability to capture complex data patterns. However, the XGBoost classifier outperforms the other individual models on the test set, achieving an F1-score of 0.992, highlighting its efficiency and robustness.

The best results are achieved through the ensemble approach, obtaining an F1-score of 0.993 in the test set. This method, employing a soft voting strategy, effectively combines the strengths of the individual models, leading to an improved performance.

TABLE I: Model Performance

Model	Train F1-score	Test F1-score
SVM	1.00	0.952
RF	1.00	0.965
NN	0.988	0.973
XGBoost	1.00	0.992
Ensemble	1.00	0.993

VII. Conclusions

Throughout this study, we have successfully demonstrated the effectiveness of various machine learning methods, including SVM, NN, RF, XGBoost, and Ensemble techniques, in the task of modulation classification.

A notable observation was the difficulty in accurately classifying QPSK samples, particularly when they closely resembled BPSK samples. Despite exploring strategies like data augmentation, incorporating additional features such as modulation constellations, training on specific subsets or weighted approaches, we observed no significant improvement in the classification of these two classes. This highlights a potential area for further research, possibly focusing on more sophisticated feature extraction methods or specialized classifiers for these particular modulations.

Moreover, a primary challenge lies in the variance of our models, despite having almost no bias. To address this, we propose the exploration of methods, such as Bagging, as they can potentially reduce the variance without increasing the bias, improving the overall performance. Additionally, considering alternative evaluation metrics might provide new insights into the model's effectiveness.

References

[1] S. Ramjee, S. Ju, D. Yang, X. Liu, A. El Gamal, and Y. C. Eldar, "Fast Deep Learning for Automatic Modulation Classification," *arXiv preprint arXiv:1901.05850*, 2019.

[2] J. L. Xu, W. Su, and M. Zhou, "Likelihood-Ratio Approaches to Automatic Modulation Classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41

[3] T. Chen, C. Guestrin, "XGBoost: A Scalable Tree Boosting System," arXiv preprint arXiv:1603.02754, 2016.

[4] X. Liu, "Deep Neural Network Architectures for Modulation Classification," Master's thesis, Purdue University, May 2018.

[5] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980, 2014.