Machine Learning Approaches for QAM-16 Demodulation: Evaluating Decision Trees and Random Forests as Hardware Alternatives

Illinois Institute of Technology
Vishnu Vijay, Maureen Stillman, Vijay Gurbani, Aneesh Bargaje, Carol Davids, James Kinney and Sasrika Ghosh
8-5-2025

Problem Statement

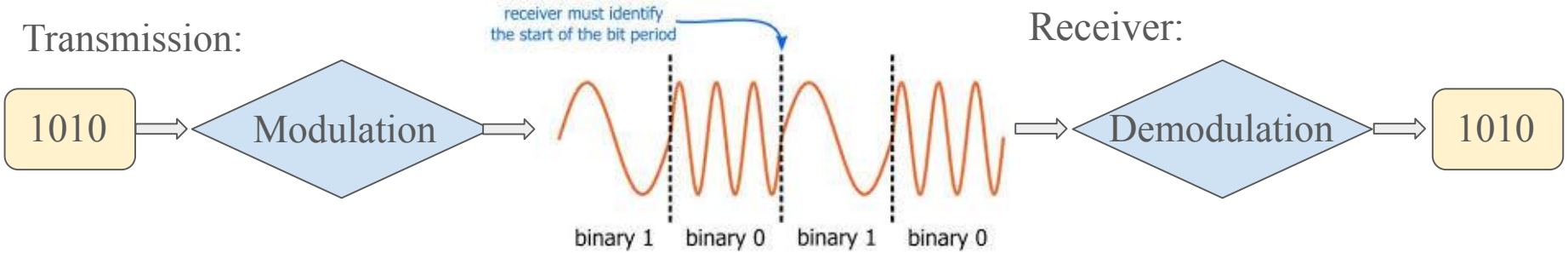
Modern communication systems rely on hardware (ASICS) to transmit and receive information

- Hardware takes a long time to develop and deploy, and can be expensive and inflexible

Our project aims to use Machine Learning to decode modulated signals into byte streams of 0s and 1s and replace that hardware

We use 802.11 as this is commonly found in laptops and cell phones

Goal - Correctly decode the signal at the receiver



Related Work

Papers related to machine learning with signal demodulation are often overview papers discussing types of Machine Learning models that can potentially be deployed

- Examples include K nearest neighbors, decision trees, and random forests

These are high level surveys that don’t apply these models nor analyze the models with different datasets

We apply two specific models: decision trees and random forests

- Our models take an input feature set of eight specific values that are derived from the complex numbers of the point being demodulated
- Our algorithm provides an analysis of the results with the accuracy and prediction time for each model

Overview papers do not provide this level of detail or analysis into any of the models we tested

Related Work

Some papers use ML to separate signals into discrete classes through a modulation classifier

- These papers use decision trees to separate the incoming transmissions as QAM-16, QAM-64, DPSK, and QPSK
- Others use cumulants to perform the modulation classification

Rather than trying to classify the modulation scheme of the transmitted point, we coded a model to demodulate signals with a known modulation scheme (QAM-16)

Another paper focuses on using a neural network to decode transmissions of the QPSK modulation, which encodes 2 bits of data (4 points)

- Our work used more explainable decision trees of depths 8, 12 and 16, and random forests with 10 trees and 50 trees
- We also focused on QAM-16, which encodes 4 bits of data (16 points)

What happens when the decoding is wrong?

The answer depends on the transport layer that you are using

TCP/IP is a protocol that checksums packets - reliable transport protocol

- If the checksum is wrong then it asks for a retransmit
- This keeps the data accurate

What about voice and video?

- These use the UDP protocol and allow errors - don’t need 100% accuracy
- UDP does not get retransmitted
- Can be undetected by humans or might hear some noise or see some jitter
- QUIC runs on UDP and reduces latency by combining the transport and cryptographic handshakes into a single, faster process and requires TLS 1.3

Summary - Transport layer protocols - TCP and UDP, QUIC

What is QAM-16?

Quadrature Amplitude Modulation (QAM) is used to map a sequence of n bits represented by a sequence of 0s and 1s into a phase and amplitude which is applied to a carrier wave

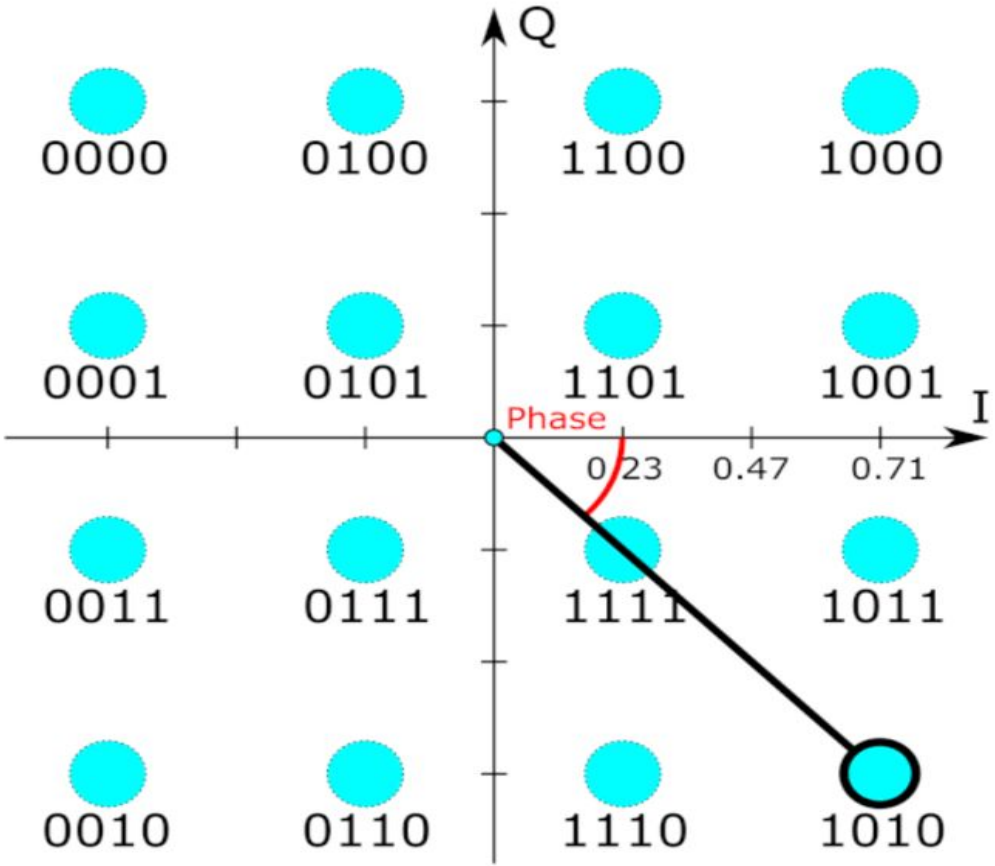
QAM-16 means that n is equal to 4, so 4 bits at a time are transmitted with possible values: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

There is also QAM-64 and QAM-256 which encode 8 and 16 bits respectively

The phase and amplitude associated with each value above is called a constellation

This is represented by a complex number a+bj

QAM-16 constellation bit mapping example



Amp	Phase	Data
0.33	135°	0101
0.33	45°	1101
0.75	161.6°	0001
1.00	135°	0000
1.00	-45°	1010

I - real axis
Q - imaginary axis

Mapping QAM-16 to complex numbers

Byte sequence	Complex number
0000	-0.9489 -0.9489j
0001	-0.9489 -0.3162j
0010	-0.9489 +0.3162j
0011	-0.9489 +0.9489j
0100	-0.3162-0.9489j
0101	-0.3162-0.3162j
0110	-0.3162+0.3162j
0111	-0.3162+0.9489j

Byte Sequence	Complex number
1000	0.3162-0.9489j
1001	0.3162-0.3162j
1010	0.3162+0.3162j
1011	0.3162+0.9489j
1100	0.9489 -0.9489j
1101	0.9489 -0.3162j
1110	0.9489 +0.3162j
1111	0.9489 +0.9489j

Signal anomalies

When signal is received, it can be subject to white noise and/or impulse noise

- If the signal is perfect then the complex number is one of the 16 in the prior table

In the real world signals are not perfect, therefore our complex number will have a range of values that correspond to the same bit mapping

There are many ways for a signal to be distorted and the model must either handle them or throw them away - we classified them

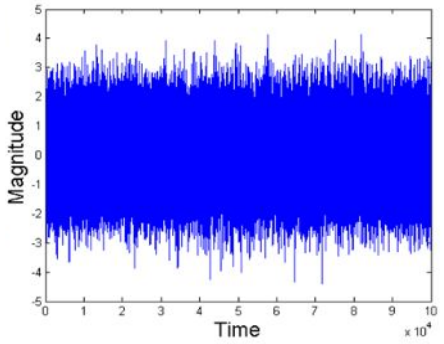
Machine learning is used to map the complex numbers to 4 bit pattern (QAM-16)

Gaussian and Impulse Noise

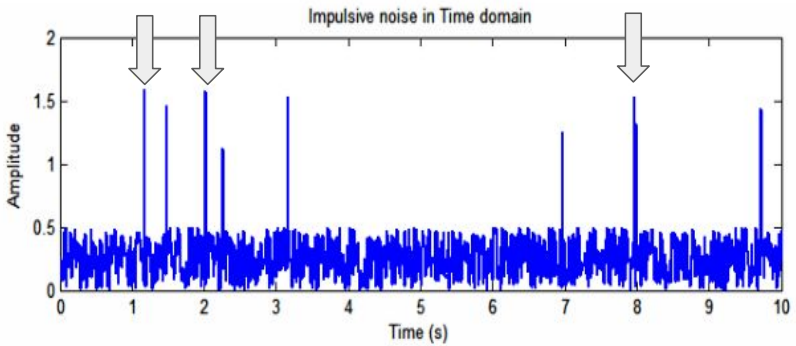
Noise on the receiver side can disrupt signals and result in different outputs than transmitted

- Gaussian noise is white noise (a fan or humming) and causes small discrepancies
- Impulse noise (loud bangs such as car backfire or lightning) causes huge spikes in amplitude

These present a challenge for demodulation as they can interfere with data transmission and result in faulty outputs



Gaussian noise (small variations)



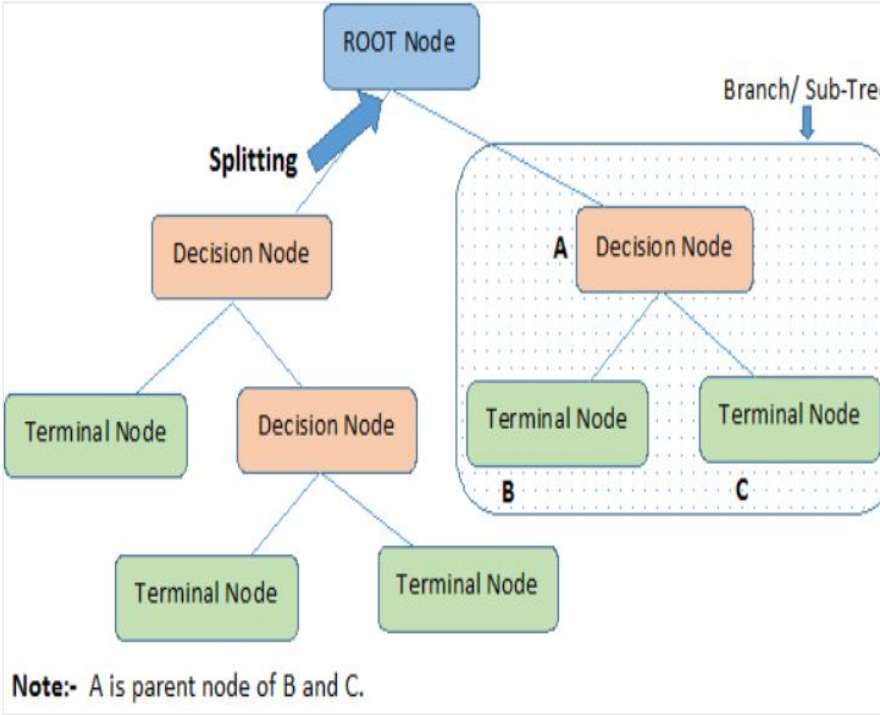
Impulse noise (visible spikes)

Model choice - Decision Tree

A Decision Tree consists of nodes that represent different decisions that lead to a final outcome or classification.

Root Node: First decision or question
Branches: Represent outcomes of decisions
Leaf Nodes: Final outcomes or predictions

- Easy to understand and interpret
- Handles both numerical and categorical data
- No need for data normalization (but in our case we did normalize).
- Works well for classification and regression tasks



Decision tree - features and hyperparameters

Used the scikit-learn Python library to implement the decision tree:

```
DecisionTreeClassifier(
        max_depth=8,
        min_samples_split=8,
        min_samples_leaf=4,
        random_state=42
    )
X_train, X_test, y_train, y_test = train_test_split(
        X_features, y, test_size=test_size, random_state=42
    )
```

- Hyperparameters: max_depth = 8, 12 and 16 prevents overfitting; random_state ensures same result every time
- Features - math formulas derived from input X

Real
Imaginary
Phase
Magnitude
Real Rank
Imaginary Rank
Quadrant
Log Magnitude

Decision tree - features and hyperparameters

- Features 1-4 - Complex number - X real and imaginary parts, phase, magnitude
- Features 5-8 - derived from mathematical functions (see table)

Features represent the relative position of values rather than absolute positions, making them less sensitive to extreme values

Features:

- Real Rank - percentile in all real numbers of the dataset
- Imaginary Rank - percentile in all imaginary numbers of the dataset
- Quadrant (1 - 4) - which quadrant the complex number is in
  - Even with extreme noise, the quadrant of a point often remains informative
- Log Magnitude - a - real component b - imaginary component; see formula below
  - Using logarithmic transformation of magnitude reduces impact of impulse spikes

Real
Imaginary
Phase
Magnitude
Real Rank
Imaginary Rank
Quadrant
Log Magnitude

$$\ln|z| = \ln\left(\sqrt{a^2 + b^2}\right)$$

Model architecture - Supervised learning

Takes a complex number as input (modulated signal)

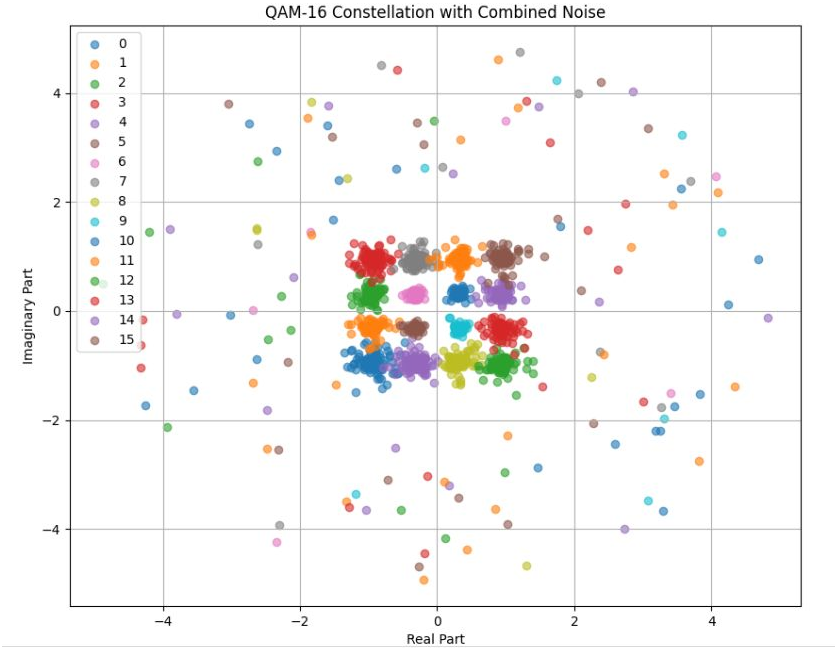
Maps the complex number to one of 16 outputs (demodulated signal, or the binary data): 0000 to 1111

Process

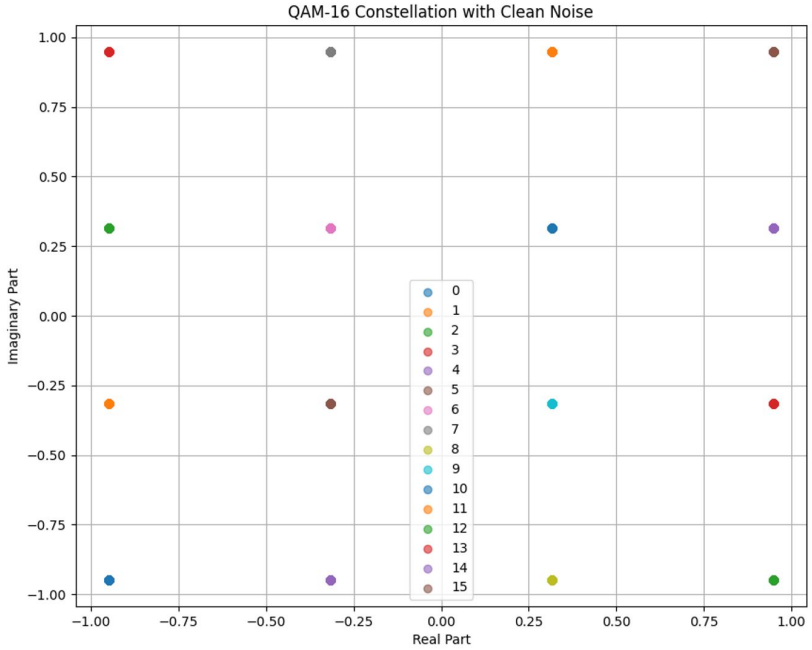
1. Generate the data using simulation with known X and y (inputs and outputs) - 20,000 total points with 8 input values
2. Dataset is split into 2 parts - 80% training and 20% test
3. Model is trained on training data - 16,000 known inputs X and outputs y
4. Evaluated on test data (remaining 4,000 points) to determine its accuracy
5. New data is presented to the model as a complex number and it maps the output y using the model that was built in step 3

$$\text{Accuracy} = \frac{\text{Number of correctly identified samples}}{\text{Total number of samples}}$$

Results



QAM-16 Square Constellation with Gaussian and Impulse noise



QAM-16 Square Constellation with no noise

Results - accuracy and prediction time

- Decision trees are faster than random forests in predicting one point, but accuracy wise are worse
  - Since prediction time is important in real time communications, decision trees are more feasible to use than random forests
- Decision tree of depth 8 had the best accuracy of the three tested

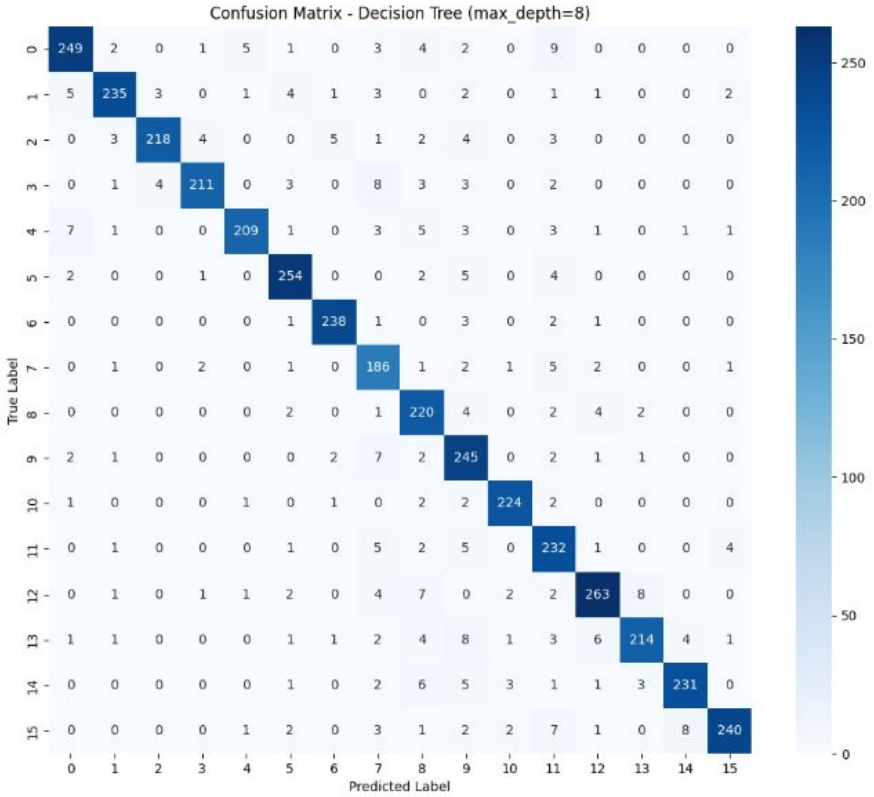
Model	**Max-depth/trees**	**Accuracy (%)**	**Prediction time**
Decision tree	8	91.72	.384 ms
Decision tree	12	91.38	.427 ms
Decision tree	16	91.50	.397 ms
Random Forest	10	92.05	1.732 ms
Random Forest	50	92.33	4.120 ms

Performance comparison of ML models

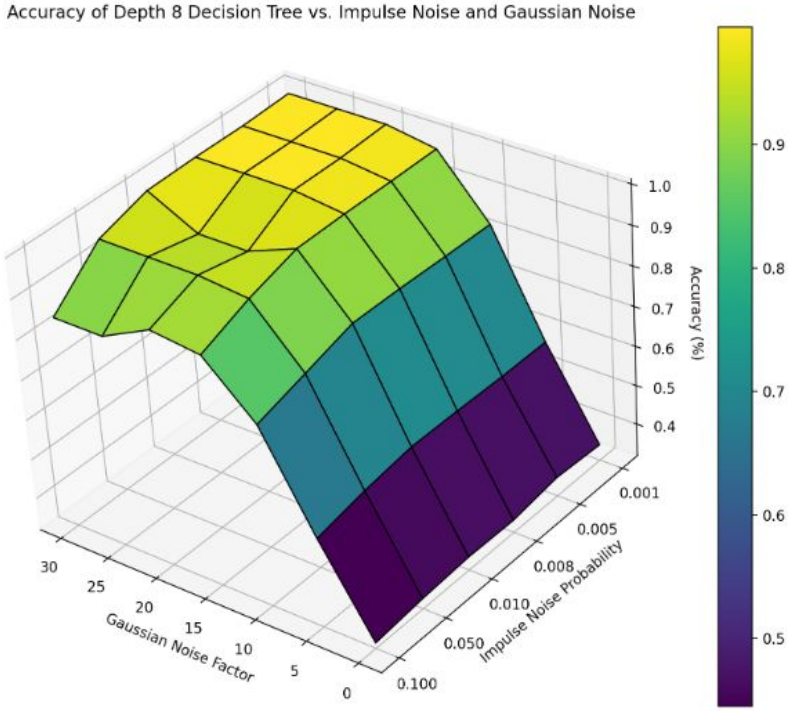
Results - Confusion matrix max depth 8

A confusion matrix tells us where the misclassifications of the model are and if there are any recurring errors

- Since the few errors are random, this means the model isn't making consistent mistakes



SNR vs Impulse vs accuracy - max depth 8



As the signal to noise ratio (SNR, labeled Gaussian Noise factor in figure) increases, so does the accuracy of the model

- The inverse is true for impulse noise probability

The highest accuracy can be achieved when there is no impulse noise and a gaussian noise factor of 30

Future work

Compare performance to hardware with real-world data

- Instead of using computer generated / simulated data, we hope to use real transmitted signals to train and test our models to see how they fare in actual applications

QAM-64 and QAM-256 accuracy

- To transmit more data at a time, we want to look into QAM-64 and QAM-256 (6 and 8 bits respectively) to see if they are plausible systems of data transmission

Investigate other models

- We will look into SVMs (Support Vector Machines) and determine their accuracy and prediction time

References - 1

Kumar A, Majhi S, Gui G, Wu HC, Yuen C. A Survey of Blind Modulation Classification Techniques for OFDM Signals. Sensors (Basel). 2022 Jan 28;22(3):1020. doi: 10.3390/s22031020. PMID: 35161766; PMCID: PMC8840120.

Maria Wikstrom Swedish Defense Research Agency, A survey of modern classification methods for QAM Signals, March 2005 https://www.foi.se/rest-api/report/FOI-R--1616--SE

Xiaoyong Sun Modulation Classification Using Compressed Sensing and Decision Tree–Support Vector Machine in Cognitive Radio System, March 2020, *Sensors* **2020**, *20*(5), 1438; https://doi.org/10.3390/s20051438

Kevin D. Martinez Zapata and Jhon J. Granada Torres, "Asymmetric demodulation using decision trees in gridless WDM systems," Appl. Opt. 63, 6253-6262 (2024) https://opg.optica.org/ao/viewmedia.cfm?uri=ao-63-23-6253&seq=0&html=true

Hussain A, Alam S, Ghauri SA, Ali M, Sherazi HR, Akhunzada A, Bibi I, Gani A. Automatic Modulation Recognition Based on the Optimized Linear Combination of Higher-Order Cumulants. Sensors (Basel). 2022 Oct 2;22(19):7488. doi: 10.3390/s22197488. PMID: 36236583; PMCID: PMC9571176. *Electronics* **2022**, *11*(17), 2764; https://pmc.ncbi.nlm.nih.gov/articles/PMC9571176/

References - 2

Andersson, A. ”Utilizing neural networks to adaptively demodulate and decode signals in an impulsive environment” (Master’s thesis, Uppsala University). Uppsala University. 2023 June https://www.diva-portal.org/smash/get/diva2:1765577/FULLTEXT01.pdf.

J¨urgen Schmidhuber, ”Deep learning in neural networks: An overview, ”Neural Networks, vol. 61, pp. 85–117, 2015.

T.M. Cover and P.E. Hart, ”Nearest neighbor pattern classification,” IEEE Transactions on Information Theory, vol. 13, no. 1, pp. 21–27, January 1967.

Detlof von Winterfeldt and Ward Edwards, ”Decision trees”. Decision Analysis and Behavioral Research. Cambridge University Press. pp.63—89, 1986.

D. Heath, S. Kasif, and S. Salzberg, ”k-DT: A multi-tree learning method,” In Proceedings of the Second Intl. Workshop on Multistrategy Learning, pp. 138–149, 1993.