MPA-MLF - Miniproject

Classification of wireless transmitters

Date: 16.3.2023

1 Task description

The task will be to classify wireless transmitters based on data from measurement. You can download the dataset from your e-learnings. We will use a multi-layer perceptron machine learning model to complete this exercise. The dataset consists of 19200 samples and 12 features in total. The features represent the main radio frequency impairments such as:

- **cfo_meas** - Carrier Frequency Offset (CFO) between transmitter and receiver. Measured via vector spectrum analyzer, in Hz
- **cfo_demod** - Carrier Frequency Offset (CFO) between transmitter and receiver, measured after demodulation, in Hz
- **gain_imb** - Gain imbalance of modulator defined as:

$$gain\_imb = 20\log_{10}\left(\frac{g_i}{g_q}\right), \tag{1}$$

  where g_i, g_q are gains in I and Q path

- **iq_imb** - combination of gain imbalance and quadrature imbalance aggregated into one parameter. More information can be found e.g. in a document from Rhode & Schwarz company FSQ-K70/FSMR/FSU-B73 Vector Signal Analysis Software Manual (page 87)
- **or_off** - Origin offset, often known as DC offset. Represents how the constellation diagram is shifted from point [0+0j], expressed in dB's
- **quadr_err** - quadrature error imbalance - deviation of phase shift between I and Q components from 90 degrees
- **m_power** - Measured signal power. JUST FOR YOUR INFORMATION - DO NOT USE IT FOR TRAINING NOR TESTING
- **ph_err** - represents phase difference between received $Y(n)$ and ideal $X(n)$ constellation points:

$$ph\_err = \arg(Y(n) - X(n)) \tag{2}$$

- **mag_err** - Magnitude error between received $Y(n)$ and ideal $X(n)$ constellation points in QAM constellation:

$$mag\_err(n) = ||Y(n)| - |X(n)|| \tag{3}$$

- **evm** - Error Vector Magnitude measurements, representing the RMS (root mean square) error between received $Y(n)$ and ideal $X(n)$ constellation points in QAM constellation (see figure 1.) The EVM is computed for N total symbols such as:

$$\text{EVM} = \frac{\sum_{n=1}^{N}(Y(n) - X(n))^2}{\sum_{n=1}^{N} X(n)^2} \tag{4}$$

  EVM is an aggregated feature, and several transceiver impairments can contribute to it. The most important source of EVM increase is power amplifier nonlinearity.

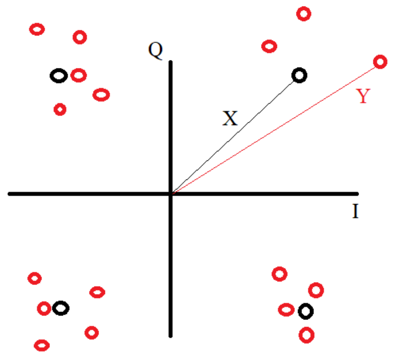

Figure 1: Original (black) and distorted (red) constellation points

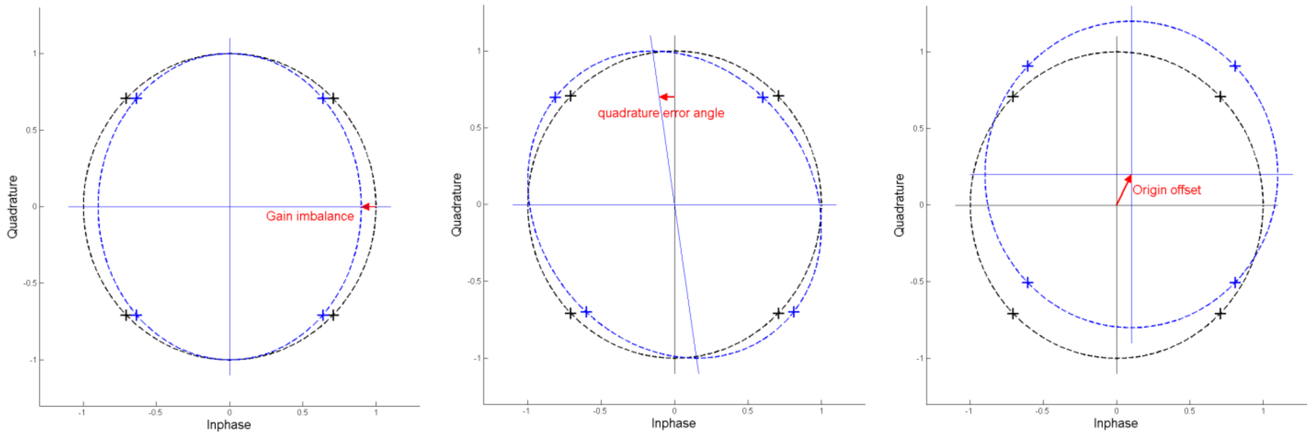

Figure 2: Effect of selected imbalances on QPSK constellation diagram

- **Tosc** - Temperature of the software-defined radio, measured at the local oscillator. JUST FOR INFORMATION - DO NOT USE FOR TRAINING NOR TESTING
- **Tmix** - Temperature of the software-defined radio, measured at the RF mixer. JUST FOR INFORMATION - DO NOT USE FOR TRAINING NOR TESTING

The effect of gain imbalance, quadrature error and origin offset on the QPSK constellation is illustrated in figure 2. The target value consists of 8 possible labels, marked by numbers from 1-8, where each number represents one transmitter.

You are provided with four different *.csv* files:

1. **x_train.csv** - training dataset,
2. **y_train.csv** - ground truth values for the training data
3. **x_test.csv** - data for testing
4. **submission_example.csv** - example of data format that is accepted by kaggle

2 Steps

2.1 Mandatory steps

You are expected to perform the following steps:

- **Data examination** - Examine what data types you have in your dataset. What preprocessing steps appear to be the best ones?
- **Data preprocessing** - Perform the preprocessing,
- **Model building and model training** - build multilayer perceptron model, set appropriate loss_function, optimizes and metric. Experiments with the number of hidden layers and with the number of neurons in hidden layers. Set the proper activation function for your output layer (hint: check *softmax* activation function). Do not forget to split your training data into train and validation. Check how the train and validation loss changes during the training epochs.
- **Performance tunning** How does your model perform on validation data? What did you do to improve the performance of the model? What regularization techniques did you use?
- **Model evaluation** - check how well the model performs on the testing dataset

2.2 Voluntary steps

- **Compare MLP to SVM** - compare your MLP model to Support Vector Machine. Compare the achieved result and the training time
- **Implement a hyperparameter tuning algorithm** - Implement any automatic hyperparameter tuning algorithm. Describe how to algorithm works.

3 Submission and grading

The deadline for submission is **31.3.2023**. You are expected to write a report(a maximum of 5 pages) that describes the essential steps and strategies you used to build a model with the highest possible testing accuracy. You can receive 15 points maximum.
Your solutions will be submitted in three different ways:

- **Report, e-learning**. You will upload your report into the e-learning. Please upload your report in .pdf format
- **Model predictions, kaggle** You are required to test your results in the Kaggle competition, link: *https://www.kaggle.com/t/d069d084fb8843aabb37ade7ca78e65f*. You are limited to 15 submissions a day, so please start early. The correct format of your submissions can be seen in the *submission example.csv* file
- **Code, GitHub, e-learning**. Please create a new folder in the repo you have used for MPA-MLP. Do not push the input dataset to your GitHub repo.

4 General comments

- You are required to do all of your coding in python. You have to use one of these frameworks to implement the ML algorithm: *Keras*, *PyTorch*, *Scikit-learn*.
- Usage of Google Collab is strongly recommended but not required.
- If your testing accuracy is not the highest, do not worry. You can still receive the maximum points.