Experiment Report

## Introduction

This report discusses an experiment of used Multi-layer perceptron as a learning model to predict the true signal symbol. This report has include the experiment from 10/29/20 to 11/09/20, and the data set is generated by a simulation algorithm from MATLAB.

## About Data

In each data set, there are (M*N) samples, (M) denoted as the number of blocks and (N) denoted as the number of the samples in each blocks. In each block, we will have 2 major modules (4QAM and 16QAM), and the main variable in the experiment are SNR(Signal), INR(Interference), P_INT(Probability of the Interference being present in each block), and N (Noise). The distribution of the data set in the experiment are the following:

Data set 1: SNR = 60, INR = 5, P_INT = 0.8, N = 1
Data set 2: SNR = 60, INR = 30, P_INT = 0.8, N = 1
Data set 3: SNR = 30, INR = 30, P_INT = 0.8, N = 1
Data set 4: SNR = 60, INR = 5, P_INT = 0.8, N = 30

In each dataset, a sample X is a complex number, and a label Y is symbol.There are 20 labels in this data set (0-3 denoted as the the symbols from 4QAM module, and 4-19 denoted as the symbols from 16QAM module).

## About Learning Module

The machine learning module in the experiment is a Multiple layer perceptron, and it has 1 input layer (with 2 nodes), and 5 hidden layer (with 128 nodes respectively), and 1 output layer (with 20 nodes). The output of this module is an array with 20 numbers and each number denoted as the probability of its index symbol, and the highest probability of the index is the final prediction $\hat{y}$ of the module. The learning rate is 0.001, and the epoch is 20.

## Measurement

The Accuracy rate is the total success prediction (y = $\hat{y}$) the divided by the size the of number which formulated as

$$a = \frac{1}{n}\sum_{0}^{n} 1_{\hat{y}=y} .$$

In a similar way, the error rate r = 1 - a.

## Result

The model performance has converge with 5 hidden layers and each layer has 128 nodes. The following table shows the model performance in data set 4. Also, the different learning rate in different data set has a different performance. Take a model has 5 hidden layer and each layer has 128 nodes as an example model. In the learning rate of 0.01, the model has accuracy rate with 0.87 in data set 2 and 0.82 accuracy in data set 4. However, with the same model and learning rate of 0.001, the model has accuracy rate with 0.97 in data set 2 but only 0.74 in data set 4. Therefore, the setting of learning rate is different base on the data distribution.

| **Layers/Nodes** | **128** |
|---|---|
| **3** | 0.76 |
| **4** | 0.79 |
| **5** | 0.82 |
| **6** | 0.81 |

The module show a large gap on performance of different types of data distribution. The module performs best on the data set 1, which the accuracy rate is 1, and the data set 2 is follow with around an 98% accuracy. Data set 4 has 82% accuracy, and the module show the worse performance on data set 3 with only 81% accuracy rat.

In a same way, the performance on 4QAM and 16QAM also show a large difference. Taking data set 4 as an example, within 10 experiments the module has the mean accuracy rate of a = 95%, but on predicting 16QAM  mean accuracy rate of a =  51%. However, the model is 100% accurate on predicting the sample whether from 4QAM or 16QAM.

When the data size is (50*1000), the increasing of data size does not affect the performance of the data. Giving Data set 4 as a distribution model, I generated a sequence of 10 data set, which has (50*1,000), (50*2,000) …, (50*10,000) respectively. In the same leaning model, the accuracy rate table as shown in the below.

| Increased size test | |
|---|---|
| Data set | Accuracy |
| 1 | 0.727 |
| 2 | 0.731 |
| 3 | 0.733 |
| 4 | 0.716 |
| 5 | 0.732 |
| 6 | 0.733 |
| 7 | 0.735 |
| 8 | 0.722 |

| | |
|---|---|
| 9 | 0.732 |
| 10 | 0.739 |

As a result, the accuracy rate does not affected by the increasing size of testing set or training set when the data set has (50*1000) samples.

The accuracy rate of each symbols is steady in the experiment. Using data 4 as a distribution module, within 10 iteration, the standard deviation of each symbols is listed as the following. Therefore, the accuracy rate of each symbols is steady.

| Standard Deviation of each Symbols (SYM=symbols) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| SYM | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| STD | 0.01 | 0.01 | 0.01 | 0.01 | 0.04 | 0.02 | 0.02 | 0.03 | 0.02 | 0.02 |
| SYM | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| STD | 0.02 | 0.03 | 0.03 | 0.02 | 0.02 | 0.02 | 0.03 | 0.04 | 0.03 | 0.03 |

Using a training model from data set (I) to test a different distribution data set (II), the error rate of each symbol is close to the result of using training model from data set (II). Giving data set 3 and 4 as the distribution model, this cross testing experiment using the Euclidean distance to measure the distance between 2 accuracy rate with 10 iteration. Therefore, the error rate of each symbols is affect by its data distribution.

| | Result_cross3 | Result_cross4 |
|---|---|---|
| Result_ori3 | 0.34 | 0.27 |
| Result_ori4 | 0.039 | 0.05 |

Result_ori means using the training model to test its corresponding data distribution, and result_cross means using training model to test other distribution.

## Conclusion

The current experiment used the a basic learning model (multiple layer perceptron), but it still have a well performance in some easy data distribution. However, as the scale of interference or noise increased, the performance drop badly. Especially with dealing with the 16 QAM module, the average accuracy will be 0.50 or lower. Our next step of the machine learning experiment will focus on improve the accuracy rate by applying a higher and more complex model.