A Deep Learning approach in estimation of maximum data rate and bit error rate in spatial modulation scheme

Reddy Vishwas, Mukesh Gandhi, E.S Gopi

[1] National Institute of Technology, Tiruchirappalli, Tamilnadu, India.

Abstract. This paper proposes a method to estimate the number of active antennas that can be chosen to maximize the data rate in Spatial Modulation (SM) technique and to predict the Bit error rate for QAM symbols transmission through Spatial modulation. For reducing the computational complexity, we propose two machine learning algorithms, one to predict the number of active antennas for a specified number of antennas and modulation scheme M and the other to predict the Bit error rate for the specified values of number of active antennas, QAM modulation scheme M and SNR. We then compare the accuracy obtained in various optimization algorithms.

Keywords: Spatial Modulation, QAM, MIMO

1 Introduction

1.1 Spatial modulation

Spatial Modulation is a transmission technology developed for MIMO systems to increase the channel capacity. In this technique, we can transmit information using both active antennas index and constellation symbols, an antenna switching mechanism is used to switch the antennas between on and off state based on index. The incoming data will be divided into two parts as shown in Fig. 1, one part is for the symbol in constellation and the other part is for the index of the antennas. There are number of ways in which we choose the number of active antennas,

To find the number of spatial bits that can be transmitted, first we need to find the number of combinations that are possible for a given number of antennas n_t. Let us choose n_r number of antennas for transmitting a symbol, then total number of symbols that possible is given by ${}^{n_t}C_{n_r}$. Now the number of bits in spatial domain is given by $log_2\left({}^{n_t}C_{n_r}\right)$. If each selected antenna is transmitting a M-ary modulated symbols, then the number of bits transmitted in constellation domain is given by $n_r\, log_2\, M$. The total channel capacity 'R' is given by,

$$R = log_2\left({}^{n_t}C_{n_r}\right) + n_r\, log_2\, M \tag{1}$$

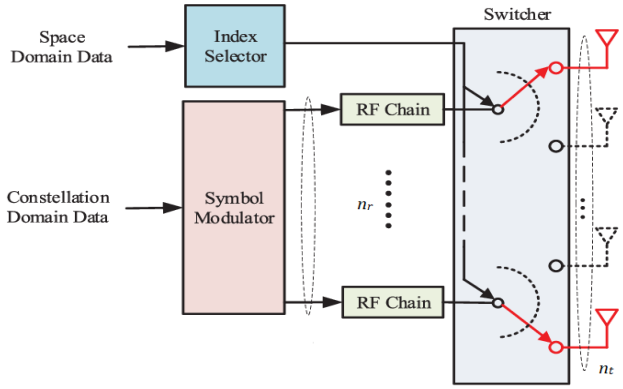

Fig. 1. A block diagram showing Spatial modulation of data.

2 Motivation

We see that it is difficult to maximize (1) by using differential calculus as the equation is in discrete form. According to the Universal approximation theorem, A feedforward neural network can be used to approximate any real valued function. In this paper we have implemented deep learning framework to approximate a function which predicts n_r for given values of n_t and M for which R is maximized and to obtain the Bit error rate (BER)

3 Analyzing and Predicting Data Rate

3.1 Analyzing the Data Rate equation

In this section we mathematically analyze the data rate equation (1). One way to do is obtaining the plot by varying the values of n_r for different values of n_t and M. The plots can be used to interpret that the data rate function is not a monotonic increasing function and has global maximum.

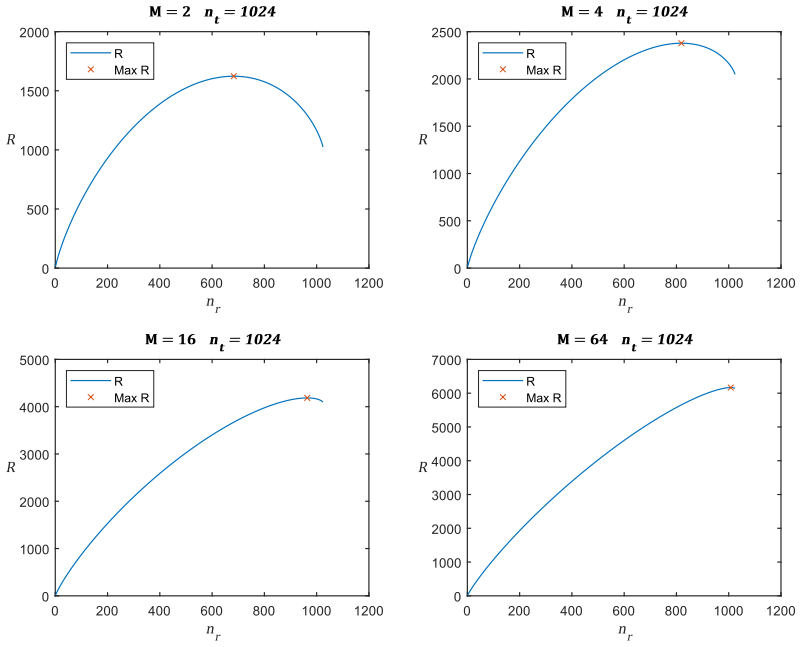

Fig. 2. Plots representing R vs n_r for different values of n_t and M.

From Fig. 2 we can interpret that the maxima exist at a value of n_r and as the number of points in the constellation increases the maxima shifts towards n_t.

A function to calculate n_r at which R is maximum.

Data rate function (m, n_t)

$$y = \max_{i \in \{1,2,\ldots,n_t\}} \left(\sum_{j=1}^{n_t} \log_2(j) - \sum_{j=1}^{n_t - i} \log_2(j) + \sum_{j=1}^{i} \log_2(j) + i \log_2(m) \right)$$

return y

Here summation of logarithmic values is used to have the values within the computational memory unit range, but it takes a lot of time to compute. We can reduce this time by using a trained neural network.

3.2 Training the neural network

We can decide the architecture and train Neural networks using a MATLAB tool called Neural Net Fitting but first we need to generate the dataset.

In the generated dataset first column represents the value of M; second column represents the value of n_t and last column represents the value of n_r and each row is a sample. We need to separate the inputs and outputs into two datasets and feed them to the Neural net fitting tool. In the tool we will select the architecture as 10 hidden neurons, after fixing the architecture we will use the Bayesian Regularization technique for training the network. The training will be automatically done by the MATLAB tool till

the error reaches a minimum value as shown in Fig. 3. After the training we can use the tool to generate a MATLAB function of the network with the trained biases and weights. We can use this function to predict the values of n_r for a given value of n_t and M. This network function approximates the function of n_r in terms of n_t and M for which R will be maximum.

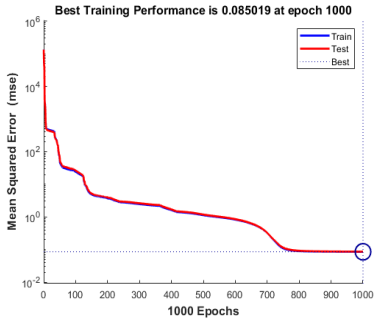

Fig. 3. Performance plot for 1000 epochs.

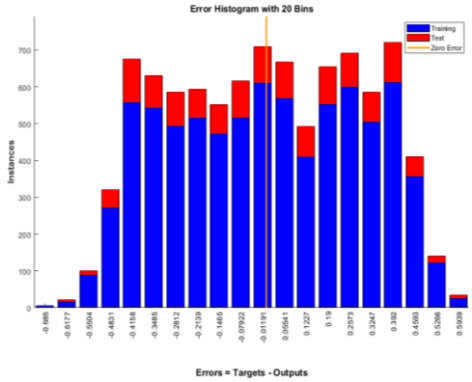

Fig. 4. Error histogram plot for training and testing data

4 Analyzing and Predicting Bit Error Rate

4.1 Transmitter and Receiver Model

We use the Quadrature Amplitude Modulation technique for generation of constellation data, these QAM symbols are then transmitted using the transmission model shown in Fig.1 where the channel is modeled as a Rayleigh Flat fading channel.

Consider the example where the number of transmitting antennas and receiving antennas are fixed as 64, the real and imaginary parts of the channel coefficients are assumed to be outcome of a random variable with gaussian distribution with mean 0 and variance 2.

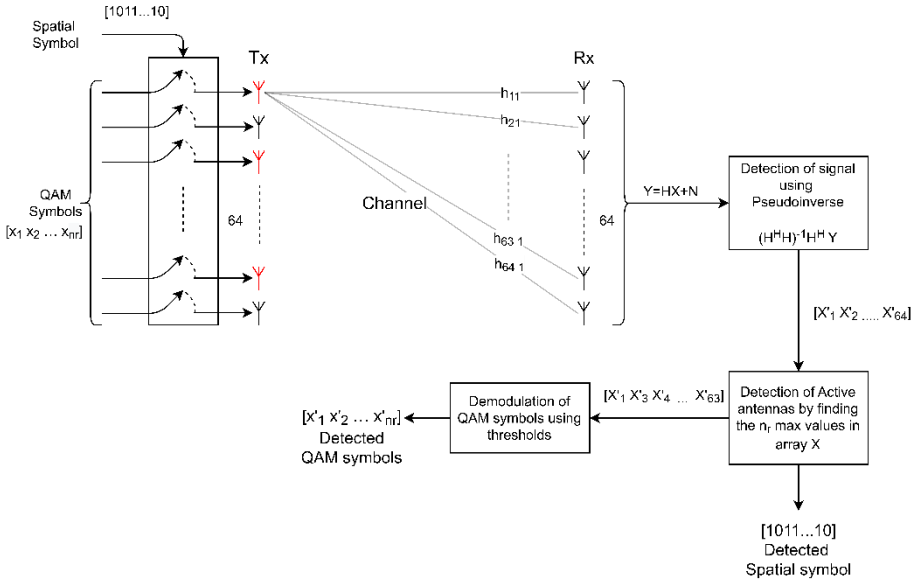

Fig. 5. Transmitter and receiver model

The bits transmitted are divided in two parts, first part contains $n_r \log_2 M$ bits which are modulated using QAM constellation and second part contains $log_2\left({}^{64}C_{n_r}\right)$ bits which is used for active antenna index. These modulated symbols are transmitted through a Rayleigh channel with channel matrix denoted as H. Let the channel also add noise to each symbol which is gaussian distributed with mean = 0 and variance = σ_N^2.

$$H = \begin{bmatrix} h_{11} & \cdots & h_{1\,64} \\ \vdots & \ddots & \vdots \\ h_{64\,1} & \cdots & h_{64\,64} \end{bmatrix}$$

At the receiver the received signal is $Y = HX + N$ where X is transmitted signal vector and N is Noise vector. In our model we assume that the perfect channel state information is known so at the receiver pseudoinverse technique is used to detect the signal.

$$\hat{X} = (H^H H)^{-1} H^H Y$$

From the detected signal $\hat{X}$ the largest n_r elements in the magnitude vector of $\hat{X}$ are taken and their indices are the demodulated the spatial bits. The detected n_r values from the $\hat{X}$ vector are then sent to a QAM demodulator which uses thresholds to detect the symbols.

4.2 Bit Error Rate

Bit error rate is the ratio of number of symbols detected wrong to the number symbols transmitted. To calculate this, we will randomly select n_r QAM symbols of Modulation scheme M and randomly select n_r transmit antennas and store their indices, the selected QAM symbols are transmitted through the selected transmit antennas, in the channel noise is added whose power is obtained from given SNR value. At the receiver these symbols are detected as described in section 4.1. After detection the number of symbols

that are mismatched are counted including the spatial symbol, this value is divided with $n_r + 1$ to get the Bit Error rate. This process is repeated for thousand times for a given n_r , M and SNR values and the average of the Bit Error rate is calculated. Thus, obtained BER is plotted by varying SNR for fixed values of n_r and M.

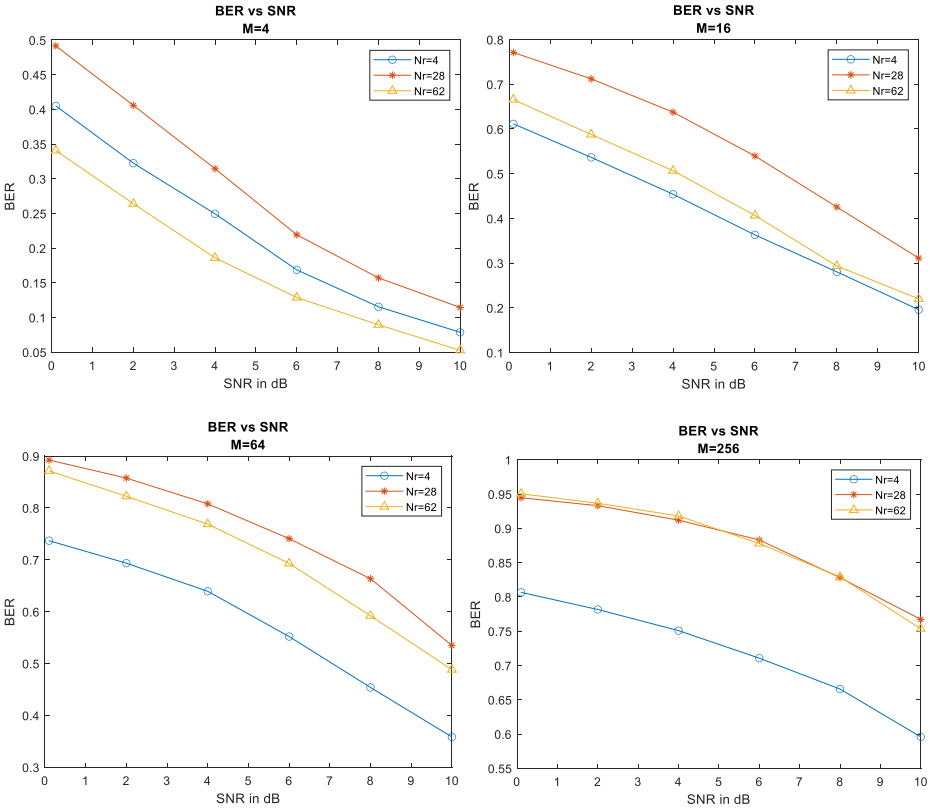

Fig. 6 Plots representing BER vs SNR for different values of n_r and M.

4.3 Training the neural network to predict BER

The dataset consisting of "number of bits", "number of antennas", "SNR", "BER" was obtained from the methodology described in Section 4. The open source library Pytorch was used to design and train the neural network. The network was trained in Google Colab's Tesla K80 for 1000 epochs. Batch normalization and dropouts were added to prevent the overfitting of model. The neural network consisted of 5 hidden layers with 50 neurons/layer.

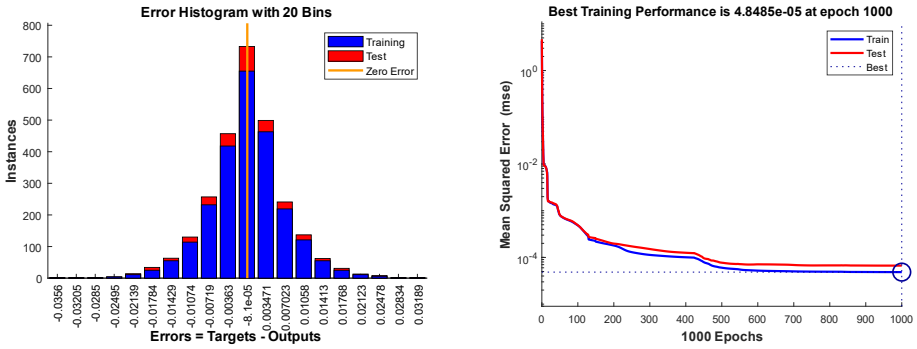

Fig. 7. Performance plot for 1000 epochs and error histogram

The test and training loss were plotted while training to optimize the model and prevent overfitting. The model was trained without overfitting as the test the loss was decreasing with increasing training epochs. The error histogram depicts the no of data inputs that were predicted with their respective error buckets. Almost 70% of data inputs had an error of 8.1e-05.

5 Results:

5.1 Neural Network Architecture and Accuracy for Data Rate

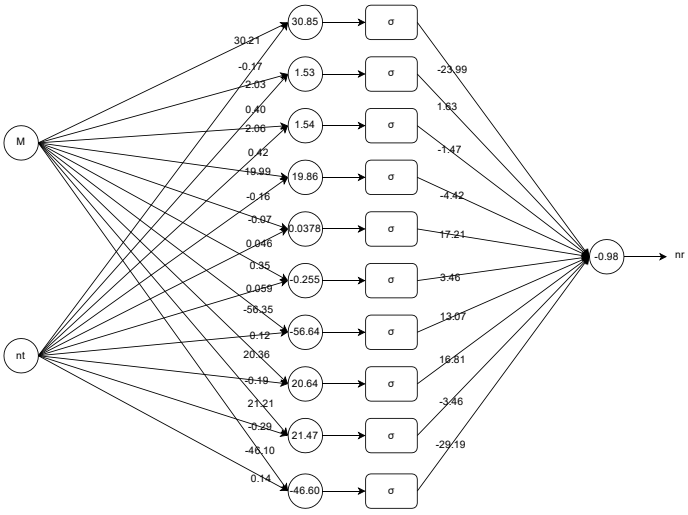

Fig. 8. Neural Network architecture with trained weights and biases for predicting Data rate

Using the neural network, we can predict the values of n_r with error ± 0.6, this is shown in Fig. 4. The computing time is also reduced drastically, for example to calculate n_r for m=64 and $n_t = 1024$ using Data rate function it takes 0.080329 seconds, but by using the neural network it takes 0.005821 seconds. It is almost fourteen times faster than the Data rate function. We can see the network predicted values are very closer to the actual values from the plots in Fig. 6.

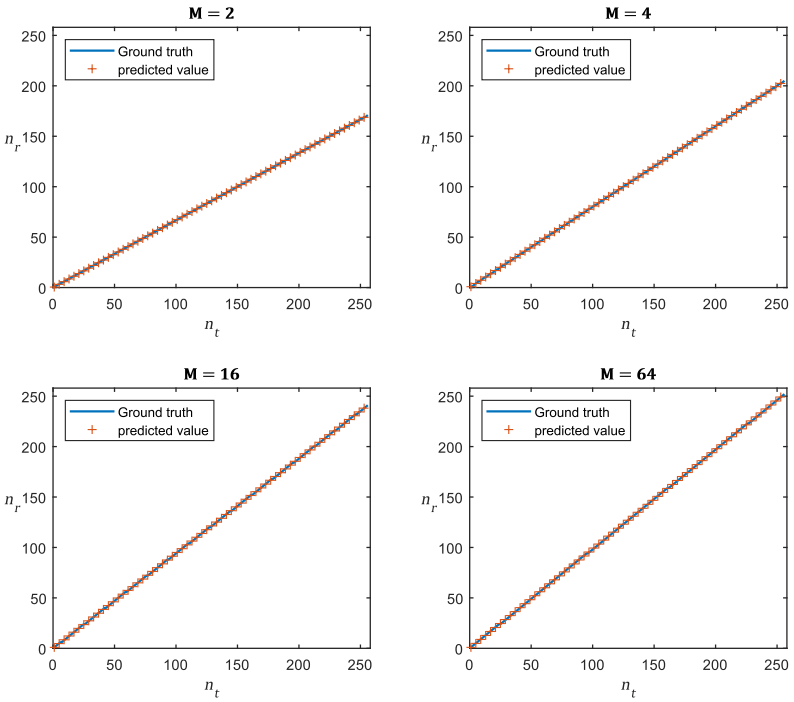

Fig. 9. Predicted and actual data plots of n_r vs n_t at different values of M.

5.2 Neural Network Accuracy for Predicting BER.

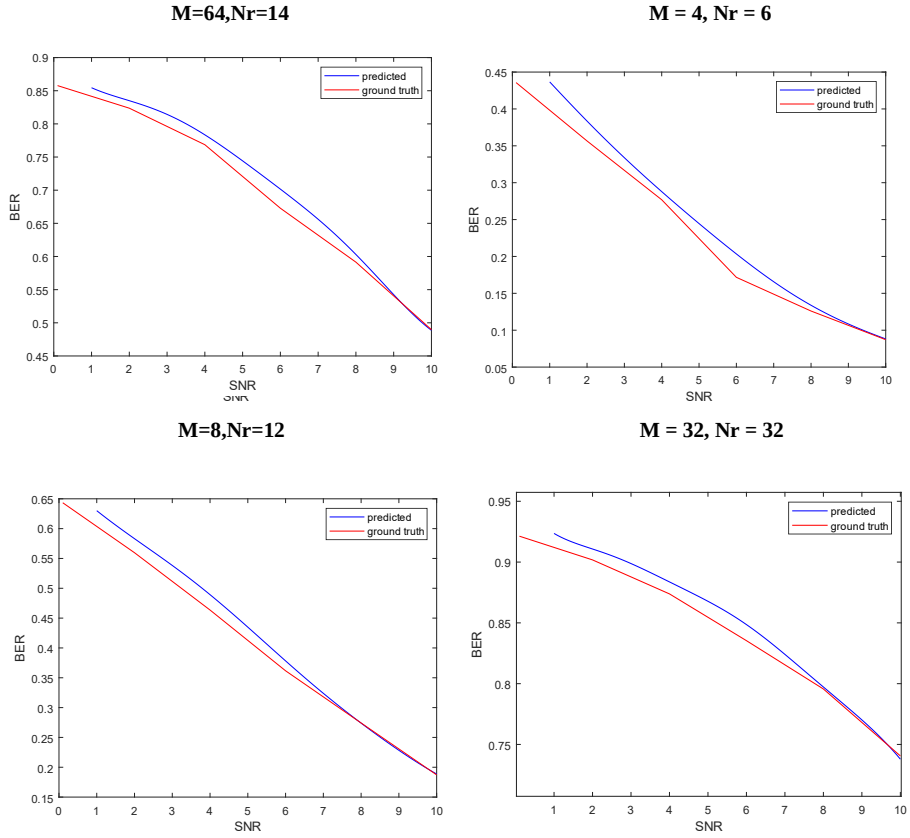

Fig. 10. Predicted and actual data plots of $BER\ vs\ SNR$ at different values of M and Nr.

The Neural network trained obtained an average loss in order of 10e-04 as shown in convergence graph in fig 7. The model was generalized and satisfactory results for various combination of inputs was obtained. Fig 10 shows the plot at different SNR for a fixed value of M and Nr. The prediction was nearly co-incidental with the actual value at higher SNR. For SNR values greater than 7 the error is reduced drastically.