Spring 2020

Sphere decoding based on Deep Neural Network

Yangyue Ma

Deep Learning Based Sphere Decoding

Yangyue Ma

February 20, 2020

Abstract

Machine learning is wildly used nowadays in communication area. Also, 5G will become more general in the future, an effective decoder for massive MIMO channel is important. In my creative component, I combined sphere decoding algorithm with deep neural network, to generate a low complexity decoder.

Background

Problem need to solve

$$y = Hs + e$$

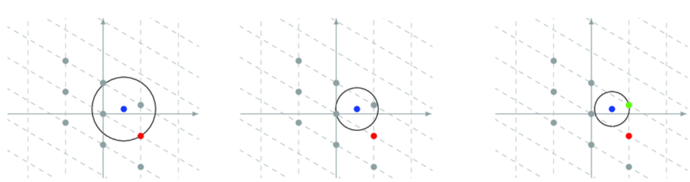
In this MIMO system, y is $m \times 1$ received signal; s is a $n \times 1$ transmitted signal; H is $m \times n$ channel matrix; e is $m \times 1$ Additive white Gaussian noise.

Maximum Likelihood algorithm

$$\hat{s}_{ML} = \underset{s}{\operatorname{argmin}} \|y - Hs\|_2$$

Maximum Likelihood algorithm is trying to find the optimal solution by enumerating all s, it has high complexity when n is large.

Sphere Decoding



(a) A radius is chosen based on a initial candidate solution.

(b) The tightening of the sphere around the transformed unconstrained solution.

(c) The sphere is tightened un one point lies within the which is the optimal solution.

Figure 1: Principle of Sphere Decoding[1]

Fincke and Pohst[2] firstly introduced Sphere Decoding that is restricting lattice points searching in an appropriate hypersphere, to reduce the complexity. As figure shown above, we will firstly choose a hypersphere on the skewed lattice, then try to search if there is any point inside the hypersphere. If there are multiple points in the hypersphere, we will decrease the radius of the hypersphere until there is only one point inside. If there is no point in the hypersphere, we will increase the radius and search again.

Deep Neural Network

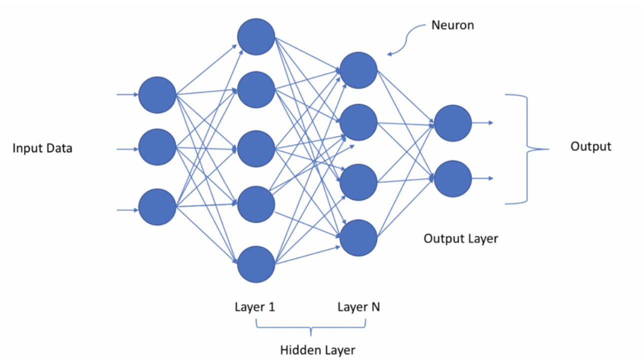


Figure 2: A Deep Neural Network with N hidden layers[3]

A DNN is a fully connected feedforward neural network (NN) composed of several hidden layers and the neurons between the input and output layer.[4]The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship.

[1] Dorfling, Martinus and Mouton, Hendrik and Karamanakos, Petros and Geyer, Tobias. (2017). Experimental evaluation of sphere decoding for long-horizon direct model predictive control. P.1-P.10.10.23919/EPE17ECCEEurope.2017.8099126.

[2] Fincke, U., and Pohst, M. (1985). Improved Methods for Calculating Vectors of Short Length in a Lattice, Including a Complexity Analysis. Mathematics of Computation, 44(170), 463-471. doi:10.2307/2007966

[3] Moolayil, Jojo John. "A Layman's Guide to Deep Neural Networks." Medium, Towards Data Science, 25 July 2019, https://towardsdatascience.com/a-laymans-guide-to-deep-neural-networks-ddcea24847fb.

[4] Mohammadkarimi, M., Mehrabi, M., Ardakani, M., and Jing, Y. (2018). Deep Learning-Based Sphere Decoding. IEEE Transactions on Wireless Communications, 18, 4368-4378.

Sphere decoding based on DNN

Purpose

The purpose of sphere decoding is to reduce the complexity comparing with Maximum Likelihood algorithm. Introducing DNN into sphere decoding is trying to find the initial radius for the hypersphere. This will further reduce complexity.

Point is inside of hypersphere or not

QR factorization[5]

$$H = QR$$

where Q is unitary and R is upper-trapezoidal, then

$$y = Hs + e \rightarrow y = QRs + e$$

Pre-multiply with Q^H, since $Q^H Q = I$, we have

$$Q^H y = Rs + Q^H e$$

Set

$$Q^H y = \hat{y}$$

then we need to solve

$$\|\hat{y} - Rs\| \leq r$$

where r is the given radius of a hypersphere. The problem actually is finding s inside the hypersphere, which is

$$\sum_{i=1}^{n}(\hat{y}_i - \sum_{j=1}^{n} R_{ij}s_j)^2 \leq r^2$$

Implementation

I generated 10000 samples as training set, 1000 samples as testing set. In each sample, the transmitted signal is a $n \times 1$ vector which elements are drawn from ± 1, channel matrix is a $m \times n$ matrix which elements are $i.i.d. \mathcal{N}(0,1)$. Received signal is also a $m \times 1$ vector. Noise is a $m \times 1$ Additive White Gaussian noise with $i.i.d. \mathcal{N}(0, \mu^2)$.
We rerange the received signal and channel matrix as input

[5] Boyd and Vandenberghe 2018, 10.4

$$x = [y_1 \ y_2 \ ... \ y_m \ h_{11} \ h_{12} \ ... \ h_{mn}]^T$$

The Euclidean distance between q optimal solutions and received signal as output

$$\hat{r} = [r_1 \ r_2 \ ... \ r_q]$$

A three layers DNN with one hidden layer is considered for the training phase, where the numbers of neurons in each layers are n(m+1), 128, and q. We choose ReLU as the activation function in second layer, no activation function for third layer.It should be mentioned that an SNR dependent DNN, since the noise variance influences radius a lot. Thus, for different SNR, we need to train a different model. I implemented this step in Python and output of the neural network is one radius, which I will use it as initial radius in sphere decoding procedure. The sphere decoding was implemented in MATLAB and I modified Xiaoyong Guo's[6] MATLAB code to complete the sphere decoder.

Result

I tried to implement this algorithm in Python and used TensorFlow, and found that this algorithm really helped, the initial radius can significantly reduce the complexity. I have listed the probability that there is one and only one lattice point in the least predicted radius hypersphere at different SNR.

Table 1: Probability that there is one and only one lattice point in the initial hypersphere with $SNR = 20dB$

	[-1,1]	[-3,-1,1,3]
2*2MIMO	90%	83.3%
4*4MIMO	97.6%	91.7%
8*8MIMO	70%	64%

[6] Xiaoyong Guo (2019). Sphere Decoder for MIMO Systems (https://www.mathworks.com/matlabcentral/fileexchange/22890-sphere-decoder-for-mimo-systems), MATLAB Central File Exchange. Retrieved December 23, 2019.

Table 2: Probability that there is one and only one lattice point in the initial hypersphere with $SNR = 17dB$

	[-1,1]	[-3,-1,1,3]
2*2MIMO	80.8%	73.9%
4*4MIMO	93.6%	83.2%
8*8MIMO	54%	65.1%

Table 3: Probability that there is one and only one lattice point in the initial hypersphere with $SNR = 10dB$

	[-1,1]	[-3,-1,1,3]
2*2MIMO	64.4%	48.2%
4*4MIMO	47.5%	42%
8*8MIMO	36%	31.3%