CONSTELLATION DESIGN FOR MULTI-USER COMMUNICATIONS WITH DEEP LEARNING

CONSTELLATION DESIGN FOR MULTI-USER COMMUNICATIONS WITH DEEP LEARNING

BY

YILIN SUN, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2019) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Constellation Design for Multi-user Communications with Deep Learning
AUTHOR:	Yilin Sun B.Eng., (Industrial Electronics and Control Engineering), Liverpool John Moores University
SUPERVISOR:	Dr. Jian-Kang Zhang
NUMBER OF PAGES:	xiv, 70

Abstract

In the simple form, a communication system includes a transmitter and a receiver. In the transmitter, it transforms the one hot vector message to produce a transmitted signal. In general, the transmitter demands restrictions on the transmitted signal. The channel is defined by the conditional probability distribution function. On receiving of the transmitted signal with noise, the receiver appears to apply the transformation to generate the estimate of one hot vector message. We can regard this simplest communication system as a specific case of autoencoder from a deep learning perspective. In our case, autoencoder used to learn the representations of the one hot vector which are robust to the noise channel and can be recovered at the receiver with the smallest probability of error.

Our task is to make some improvements on the autoencoder systems. We propose different schemes depending on the different cases. We propose a method based on optimization of softmax and introduce the $L_{1/2}$ regularization in MSE loss function for SISO case and MIMO case, separately. The simulation shows that both our optimized softmax function method and $L_{1/2}$ regularization loss function have a better performance than the original neural network framework.

To my parents and love

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Dr. Jian-Kang Zhang. In the past two years, he gave me patient guidance, encouragement and care. His wide knowledge of communications and neural networks formed the basis of this thesis. I am very thankful and lucky to have him as my supervisor. Without him, there is no possibility for me to finish this thesis.

Next, I would like to thank Dr. Jun Chen and Dr. Dong-Mei Zhao for being my defense members and giving valuable advice to me.

Last but not least, I would like to thank my parents and girlfriend for their support and love.

To them, I dedicate this thesis.

Contents

List of Figures

List of Tables

Notation and Abbreviations

Notation

$\mathbf{W}$	Matrices are denoted by boldface characters with uppercase
$\|x\|$	The Euclidean norm of x
w_{ij}	The (i, j)-element of $\mathbf{W}$
$\bigtriangledown$	The gradient operator
$\mathbb{R}$	The sets of real numbers

Abbreviations

AI	Artificial Intelligence
NN	Neural Network
CNN	Convolutional Neural Networks
ML	Machine Learning
DL	Deep Learning

SISO	Single Input and Single Output
MIMO	Multiple Input and Multiple Output
CSI	Channel State Information
AWGN	Additive White Gaussian Noise

Chapter 1

Introduction

Artificial intelligence (AI) has lately witnessed an explosion in global interest. The interest is according to the reality that AI allows computers to study from experience, summarise their acts, and execute tasks that are usually associated with human intelligence. The general public is very familiar with some AI applications, for instance, won masters in chess and fingerprint identifications. Other technologies are less well known to people, like detecting fraud, finding features in big quantities of information, and controlling complicated manufacturing procedures. All these technologies are according to the same principles from deep learning.

Neural network is a part of the AI field. One of the dominant kinds of Neural Networks (NN) used for a multi-dimensional signal processing is deep learning. The word *deep* generally relates to networks ranging from a "few" to more convolutional layers, and deep learning applies to method for training data to learn their functional parameters automatically using statistics representative of a particular issue area of interest [1]. Deep learning currently uses in a wide range of applications, which all have the same aim of automatically learning characters from data sets and generalizing

their reactions to situations not happen in the training process. Finally, the learned characteristics can be used to distinguish the kinds of signals that are supposed to be processed by Neural network.

The basic building blocks of a communication system includes a transmitter, a channel and a receiver. Although the process seems straightforward, many complications arise especially in the physical layer and there are diminishing results in terms of performance in this layer. The machine learning and deep learning approaches have shown significant improvements in performance in domains such as Computer Vision where it is impossible to arrive at a robust solution due to the inconsistency and unpredictability of inputs (as in case of handwritten digit recognition) [2, 3]. This might also be true in some cases in the field of the communications where some complex scenarios are difficult to be described in mathematical models. So, the concepts of Deep Learning (DL) can be applied to this domain as long as the communication models do not sufficiently capture the real effects. The transmitter, channel, and receiver are represented as a NN which can be regarded as an Autoencoder. The performance of such a system is comparable to the current transmitter-receiver system which means that the same performance can be obtained using the neural networks and they can be used to replace the highly complicated architectures of the present-day systems [4].

1.1 Potential of Deep Learning in Communication Systems

There are some explanations for why DL might obtain better performance than current communications algorithms.

First, most communication methods for signal processing have strong statistical and information theory foundations. Generally, the mathematical models are linear and stationary. However, there are some imperfections and non-linearities [5] in the practical systems. A system based on DL does not require a mathematically controllable model.

Next, NNs are universal approximators [6] and the latest research has demonstrated that Algorithmic learning with Recurrent Neural Networks [7] which is called Turing-complete [8] has a notable capacity. Since the implementation of NNs on simultaneous architectures can be extremely parallel and simply applied with small precision data types [9], it is a proof that learned algorithms can be calculated more quickly and have reduced energy costs than the manually "programmed" competitors.

Last, the architectures with large amount of parallel processing are highly energy effective and capable of impressive computing performance when they are fully exploited by concurrent algorithms [10].

1.2 MIMO Communications

In recent years, MIMO wireless systems are commonly used in 4G cellular network systems and local area network systems to boost throughput and bandwidth by exploiting the channel's multi-path features. It is possible to improve data in different

channel circumstances by encoding data across multiple antenna components which use spatial diversity or range of multiplexing or throughput. Typical MIMO communication models are split into two classifications [11], open-loop (without the transmitter's Channel State Information (CSI) data) and closed-loop (with the transmitter's CSI fed back from the receiver). These MIMO systems depend on rigid encoding-beamforming and decoding systems acquired analytically and are not known to be accurate in general. In addition, closed-loop spatial multiplexing methods depend on CSI and channel prediction and transmitting this data to the transmitter results in prediction and feedback (i.e. quantization) mistakes that further complicate the ability for these schemes to achieve optimality.

Machine Learning (ML) applications have been suggested to be incorporated into communication systems in latest years. Authors in [12] suggest using the Vector Support Machines (SVM) to allow the classification of machine learning to determine coding scheme and modulation. Authors in [13] introduce a blind detection structure that conducts information symbol detection in a receiver without knowing the CSI. These methods use ML as an extra functional unit to accomplish a specific task in conventional communications.

Recent study in machine learning for SISO channel [4] shows that by studying the scheme in the training process, autoencoders can achieve the results by existing near-optimal baseline modulation and coding schemes.

1.3 Main Contributions

In this thesis, we keep using autoencoders in SISO and MIMO systems, and our principle achievements are as follows:

In SISO systems, we propose a method which transforms the softmax activation function to an optimization problem and then find the optimal solution. Through the experiment, our method gets the better results of BER in binary case than the baseline.

We know that $L_{1/2}$ regularization has a perfect performance in signal recovery. Therefore, we introduce the $L_{1/2}$ regularization into the MSE loss function as the penalty term in our training process. Through the simulation, our MSE loss function with $L_{1/2}$ regularization has better performance than the baseline method.

Chapter 2

Preliminaries

In this section, we will discuss the principle of Convolutional Neural Networks (CNN) and introduce $L_{1/2}$ regularization in our loss function.

2.1 Convolutional Neural Networks

Using computers for automatic image recognition can be traced back to about fifty years ago. Between mid 1950s and early 1960s, the problem of learning machines [14] received a significant attention in machine learning field. During that period, it was found that, when trained with linearly independent data sets , the *perceptrons* (i.e. the basic computer units) could converge to one solution in a finite iterations. This solution is based on the hyper-plane which could correctly separate data classes in feature hyperspace. However, basic perceptrons are not adequate for accomplishing practical tasks. Authors in [15] attempt to develop perceptron itself by combining multiple layers of devices lacked effective training algorithms. With the development

of *backpropagation* [16], the aforementioned state of the art changes. Backpropagation is a method of training neural network, which consists of several layers of perceptron-like units. In the context of what we now call deep CNN [17], backpropagation was first applied to two-dimensional signals in 1989. In the next 20 years, similar efforts was still at a relatively low level. Fortunately, there has been a rapid development in neural networks [18, 19] since the publication of the 2012 Imagenet Challenge showed the power of deep CNN. In recent years, CNN is the preferred method to solve complex image recognition and signal processing tasks.

Machine pattern recognition includes four basic steps: acquisition, preprocessing, feature extraction and classification. Acquisition is about generating the original input patterns (e.g. the original signals). The tasks of preprocessing are about something like noise reduction and geometric correction. Classification is the process of assigning the input patterns to one of the predefined classes. Feature extraction is about computing attributes and it is the fundamental for distinguishing patterns. It requires extensive engineering to define and proof a set of appropriate characters for a given case. This is the most difficult problem to solve. CNN provides an alternative method for automatically feature learning by using large sample databases (i.e. training sets).

The problem is to determine an approach based on CNN for automatically extracting characters from a big training sets and deploy these characters to correctly recognize images from both the training sets and test sets. CNNs are presently being effectively implemented in a variety of other fields, including speech recognition and handling of natural language [20].

A CNN structure includes convolution, activation, and pooling layers. The data

after CNN framework is then transmitted into a fully connected neural network (FCN) with the aim of mapping a number of 2-D characteristics for each input database into a class label. Fundamental to this method is the ability to use training samples to study functional coefficients in all layers. We deploy propagation algorithm to adjust the weights (also known as parameters) in each network layers by doing the loop operation to the training data.

2.1.1 Fully Connected Neural Networks

A Fully connected neural network with J layers explains a mapping $f(\mathbf{r}_0;\alpha) : \mathbb{R}^{N_0} \mapsto \mathbb{R}^{N_J}$ of an input vector $\mathbf{r}_0 \in \mathbb{R}^{N_0}$ to an output vector $\mathbf{r}_J \in \mathbb{R}^{N_J}$ via J iterations:

$$\mathbf{r}_j = f_j(\mathbf{r}_{j-1};\alpha_j), \qquad j = 1, ..., J \tag{2.1.1}$$

where $f_j(\mathbf{r}_{-1};\alpha_j) : \mathbb{R}^{N_{j-1}} \mapsto \mathbb{R}^{N_j}$ is the mapping on the jth layer. The mapping relies on the output vector $\mathbf{r}_{j-1}$ from the earlier layer as well as a series of parameters α_j. We define $\boldsymbol{\alpha} = \alpha_1, ..., \alpha_J$ to represent the sets of parameters in the each layer of all J layers. The jth layer named dense layer or fully-connected layer if $f_j(\mathbf{r}_{j-1};\alpha_j)$can be written as this form

$$f_j(\mathbf{r}_{j-1};\alpha_j) = h(\mathbf{W}_j\mathbf{r}_{j-1} + \mathbf{b}_j) \tag{2.1.2}$$

where $\mathbf{W}_j \in \mathbb{R}^{N_j \times N_{j-1}}$, $\mathbf{b}_j \in \mathbb{R}^{N_j}$, and $h(\cdot)$ is the activation function. The parameters in the dense layer are $\alpha_j = \{\mathbf{W}, \mathbf{b}\}$. Table 2.1 shows a few of layer types with mapping functions and parameters. All J layers with random mappings produce a new random mapping. For instance, the noise layer involves adding a zero mean and

variance matrix $\beta \mathbf{I}_{N_{j-1}}$ Gaussian noise to the input. Therefore, different output will be generated each time with the same input. The activation function $h(\cdot)$ in (2.1.2) is a non-linear function. Without this non-linearity, heaping multiple layers on top of each other would have no advantages. The activation function is commonly applied to each entry of its input vector individually. Table 2.2 lists some frequently used activation functions. In general, NNs using labeled data to train, i.e., a pair of input and output vectors $(\mathbf{r}_{0,k}, \mathbf{r}^*_{J,k})$, $k = 1, ..., K$, where $\mathbf{r}^*_{J,k}$ is the target output of NN when $\mathbf{r}_{0,k}$ is the input. In the training process, our aim is to attain the minimized loss

$$L(\alpha) = \frac{1}{K} \sum_{k=1}^{K} l(\mathbf{r}^*_{J,k}, \mathbf{r}_{J,k}) \tag{2.1.3}$$

where $l(\mathbf{u}, \mathbf{v}) \ : \ \mathbb{R}^{N_J} \times \mathbb{R}^{N_J} \ \mapsto \mathbb{R}$ is the loss function, $\mathbf{r}_{J,k}$ is the prediction of the NN when input is $\mathbf{r}_{0,k}$. Table 2.3 shows some relevant loss functions. The most common algorithm for finding suitable sets of coefficients is the stochastic gradient descent method (SGD), which begins with some initial random values of $\alpha = \alpha_0$ and then iteratively modify α as

$$\alpha_{t+1} = \alpha_t - \gamma \bigtriangledown \tilde{L}(\alpha_t) \tag{2.1.4}$$

where γ is the learning step. When K is large, $\tilde{L}(\alpha_t)$ is the result of the loss function that calculated from a random small size of batch of the training sets $K_t \subset 1, 2, ..., K$ in each training epoch, i.e.,

$$\tilde{L}(\alpha) = \frac{1}{K_t} \sum_{k=1}^{K_t} l(\mathbf{r}^*_{J,k}, \mathbf{r}_{J,k}). \tag{2.1.5}$$

Table 2.1: A list of layer types

Name	$f\ (x; \alpha)$	α
Dense	$h(W \cdot x\ +\ b)$	W, b
Noise	$x + \eta$	None
Normalization	$\frac{x}{\sqrt{E \cdot \lvert x \rvert^2}}$	None

Table 2.2: A list of activation functions

Name	Function $[h(z_i)]$	Range
Linear	z_i	$(-\infty,\ \infty)$
ReLU [21]	$max(0,\ z_i)$	$[0,\ \infty)$
Softmax	$\frac{e^{z_i}}{\sum_j e^{z_j}}$	$(0,\ 1)$
sigmoid	$\frac{1}{1+e^{-z_i}}$	$(0,\ 1)$
tanh	$\tanh(z_i)$	$(-1,\ 1)$

Table 2.3: A list of loss functions

Name	$l(\mathbf{y}, \hat{\mathbf{y}})$
MSE	$\lVert \mathbf{y} - \hat{\mathbf{y}} \rVert_2^2$
Categorical cross-entropy	$-\sum_i y_i log(\hat{y}_i)$

Using the small size of K_t, the calculation complexity will be reduced compared with whole samples K. One thing needs to mention is that some of SGD algorithms can modify the learning step dynamically to promote the convergence [22]. In our experiment, we choose the Adam [23] for the optimizer method to update the parameters.

2.2 L 1/2 Regularization

The sparsity problems in past years have got a lot of attention, with the aim of finding a sparse solution to a representation. The sparsity problem can be defined as: given an $M \times N$ matrix $\mathbf{H}$, and an operation to generate $\mathbf{y}$ such as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \boldsymbol{\eta}$$

where $\boldsymbol{\eta}$ is the noise. Our task is to estimate $\mathbf{x}$ from $\mathbf{y}$ when $\mathbf{x}$ is the sparsest (i.e., $\mathbf{x}$ has the lowest non-zero elements). Then the problem can be formed as

$$\min_{\mathbf{x}\in\mathbb{R}^N} \|\mathbf{x}\|_0 \qquad s.t. \qquad \mathbf{y} = \mathbf{H}\mathbf{x} + \boldsymbol{\eta}$$

where $\|\mathbf{x}\|_0$ is the number of nonzero elements in $\mathbf{x}$, normally called L_0 norm. Sparsity issues can often be converted into the following problem called the regularization of L_0:

$$\min_{\mathbf{x}\in\mathbb{R}^N} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_0 \right\}$$

where $\|\cdot\|_2$ refers to the Euclidean norm, $\mathbf{x} = (x_1, ..., x_N)^T \in \mathbb{R}^N$, and $\lambda > 0$ is a parameter of regularization.

The difficulty of the L_0 regularization model is equivalent to the amount of variables, and it is usually difficult to solve the model, especially when N is massive[24]. To overcome this difficulty, many scientists [25, 26, 27, 28] recommended relaxing the regularization of L_0 and considering the regularization of L_1 as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^N} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 \right\}$$

where $\|\mathbf{x}\|_1$ is the L_1 norm. The $L1$ regularization issue can be converted into an equivalent problem for convex quadratic optimization and can thus be solved very effectively. It could also lead to a sparse solution to the problem being considered, with a guarantee that, under some slight circumstances, the resulting solution will coincide with one of the L_0 regularization (L_0/L_1 equivalence) solutions [27, 28]. As a result, the L1 regularization is gaining popularity and has been adopted as a very helpful instrument for solving sparsity issues. However, the regularization of L_1 may result in inconsistent selections [29] when applied to variable choice in certain conditions. It often adds extra bias in prediction [30] and in compressed sensing [31, 32], and it can not retrieve a signal with the least measurements. A further modification is therefore necessary. Among these attempts, the suggestion of using L_p regularization [31, 33, 32, 34, 35] is a very natural enhancement

$$\min_{\mathbf{x}\in\mathbb{R}^N} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_p^p \right\}$$

where $0 < p < 1$, and $\|\mathbf{x}\|_p$ is defined by $\|\mathbf{x}\|_p = (\sum_{i=1}^{N} |x_i|^p)^{1/p}$

Generally speaking, L_p regularization is hard to have a comprehensive theoretical knowledge and effective algorithms to solve. Furthermore, even if solvable, it is also a problem where p should be chosen to achieve the best result. These problems have been partly solved by recent studies in [33, 36, 37, 32]. In [33], for image deconvolution, Krishnan and Fergus have proved the regularization of $L_{1/2}$ and $L_{2/3}$ have a very high efficiency. In [37], a phase diagram research is performed, showing the adequacy of $L_{1/2}$ regularization between all L_p regularization for p in (0, 1). The results literally demonstrated that L_p regularization can certainly produce more sparse results than L_1 regularization, and yet the index 1/2 plays a typical character

somehow: if $p \in [1/2, 1)$, when the p is smaller, the solution is sparser, and, if $p \in (0, 1/2)$, the results do not have a clear difference whatever p is chosen. Therefore, the unique significance of $L_{1/2}$ regularization as follows is suggested from these researches.

$$\min_{\mathbf{x}\in\mathbb{R}^N} \left\{ \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_{1/2}^{1/2} \right\}$$

There are two reasons why we decide to use $L_{1/2}$ regularization. Firstly, even though the sparsest parameters can be attained by L_0 regularization , it is an NP-hard problem [24] to solve the L_0 regularization. In order to achieve relative sparse coefficients, Tibshirani [38] launches the Lasso algorithm (L_1 regularization), which is much simpler to solve because L_1 regularization is part of the convex problem. Moreover, they demonstrate that L_0 regularization on the certain condition of constraint is equivalent to the L_1 regularization.

In the practical implementation, however, we discover that L_1 regularization is generally not capable of producing the sparsest solution, so a question arises as to whether we can design a new regulated parameter that offers a sparser solution than L_1 regularization. Luckily, the Xu's experimental discoveries [36] have shown that $L_{1/2}$ regularization can generate sparser representation than L_1 regularization, which is also demonstrated by its geometry property. The Fig.2.1 shows that different regularization problems have different shapes, and the solution of the L_p regularization problems is the point of intersection of the regular term and the loss function. For instance, it can be seen that the solution for L_2 is not sparse, and because the coordinates of regular term for $L_{1/2}$ is easier to meet the loss function, we can say that the solution for $L_{1/2}$ is sparser than that for L_1.

Even though $L_{1/2}$ regularization is part of non-convex optimization issues, we can

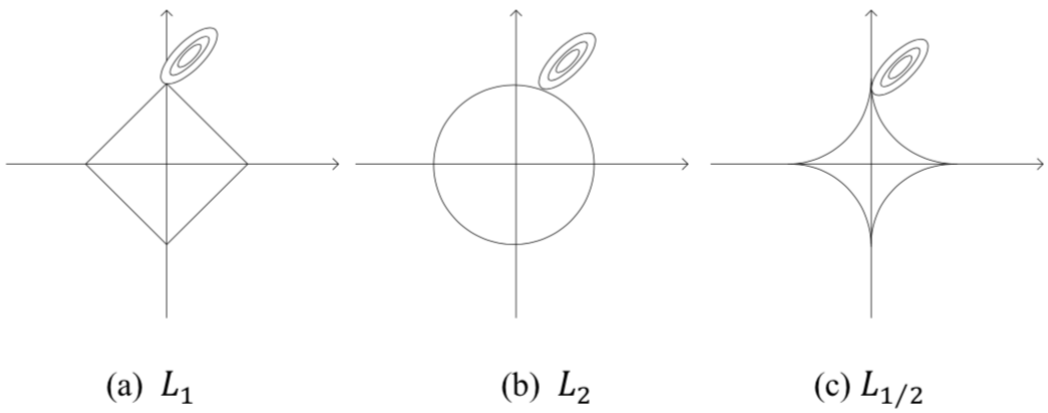

Figure 2.1: The possibility of sparse solution for L1, L2 and L1/2

turn it into a set of weighted L_1 regularization that is also easy to solve using current techniques. In addition, based on Xu's studies [36], $L_{1/2}$ regularization is more robust than L_1 regularization, so that is more appropriate for signal processing to eliminate the effects of noise.

The second is although we wish to find other possibilities for L_p regularization, e.g., $0 < p < 1/2$ or $1/2 < p < 1$, Xu's experiments [36] clearly show that the results of sparse representation via $L_{1/2}$ regularization are greater than other L_p regularization. Thus, $L_{1/2}$ regularization can totally substitute for L_p regularization ($0 < p < 1$).

In our experiment, $L_{1/2}$ regularization has a better result than L_1 regularization in signal recovery problems, and our research can also be seen as a signal recovery problem with noise. Therefore, we decide to add $L_{1/2}$ regularization in the MSE loss function for our autoencoder system. The original MSE is

$$MSE = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2$$

where $\mathbf{y}$ is the target vector and $\hat{\mathbf{y}}$ is the prediction vector. We combine the MSE

and $L_{1/2}$ regularization, then we get our loss function

$$Loss = \|\mathbf{y} - \hat{\mathbf{y}}\|_2^2 + \lambda \|\hat{\mathbf{y}}\|_{1/2}^{1/2}.$$

Chapter 3

Technical Approach

In this section, we will represent a communication system as an autoencoder implemented in DL. We also introduce a method to overcome the problem for softmax activation function does not have the optimal value.

3.1 Single Input and Single Output Case

In this thesis, we use a CNN-based method to automatically extract features from the training sets and then use these features to classify signals. CNN has been also successfully applied in many other fields, including semantics image segmentation and natural language processing[20]. Though the structure of CNN in each different case may be different, the basic operation principles are the same with those applied in our thesis.

In the simple form, a communication system includes a transmitter, a channel and a receiver. In the transmitter, it transforms the signal s to produce a transmitted signal $x = f(s) \in \mathbb{R}^n$. In general, the transmitter demands restrictions on x, (i.e.,

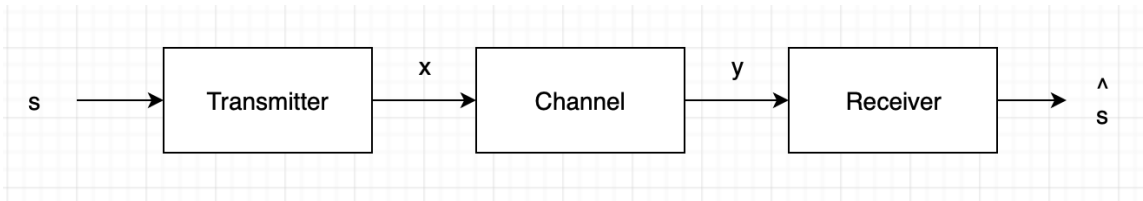

Figure 3.1: A simple communication system

the constraint for average power $\mathbb{E}[|x|^2]$. The channel is defined by the conditional probability distribution function $p(y \mid x)$, where $y \in \mathbb{R}^n$ indicates the received signal. The receiver appears to generate the estimate $\hat{s}$ from received signal y.

We can regard this simplest communication system as a specific case of autoencoder from a deep learning perspective. In our case, autoencoder used to learn the representations of the one hot vector which are robust to the noise channel and can be recovered at the receiver with the smallest probability of error.

The transmitter here includes a Neural Network with two dense layers and a normalization layer. The input is a $one-hot\ vector\ 1_s \in \mathbb{R}^M$, (i.e., an M dimensional vector, the sth entry is one and other entries are zero). The Additive white Gaussian noise (AWGN) channel is defined by a zero mean and variance $\beta = (\log_2(M)E_b/N_0)^{-1}$ additive noise layer, our experiment uses a fixed value of $E_B/N_0 = 10$db, where E_B/N_0 indicates the energy per bit (E_B) to noise power spectral density (N_0) ratio. The receiver also includes a Neural Network with two dense layers. Each entry in ouput $\hat{s}$ is a probability. In quantization, we set the biggest probability entry to one and other entries to zero.

Fig. 3.2 shows an example of such an autoencoder. It can be seen that the transmitter end has three layers: two dense layers and one normalization layer. After the noise channel, the receiver part has two dense layers. The output dimensions for

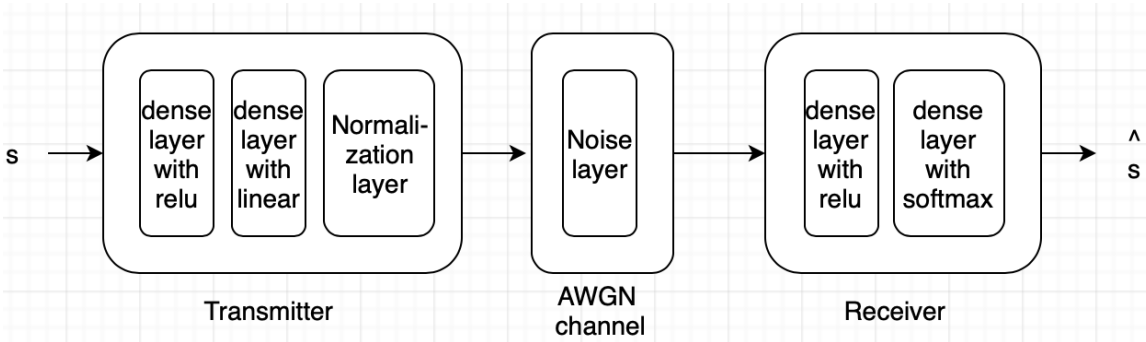

Figure 3.2: A communication system over an AWGN channel represented as an autoencoder

our autoencoder system are shown in Table. 3.1. The SISO network structure can be regarded as extracting a 2-dimensional feature from the message s, and receiver makes the classification of the feature. The dimension of feature is 2 because each point in the constellation can be considered as a two-dimensional vector.

Table 3.1: Output dimensions in autoencoder

Layer	Output dimensions
Input	M
Dense with ReLU	M
Dense with linear	2
Normalization	2
Noise	2
Dense with ReLU	M
Dense with softmax	M

First, let us only consider about the receiver part. (Fig. 3.3). As we can see, the output layer is a dense layer (also called fully connected layer) with softmax activation function $S_i = \frac{e^{u_i}}{\sum_j e^{u_j}}$. We know that the weights in each dense layer will be updated through Back-propagation algorithm, in that case we need to find the derivative of

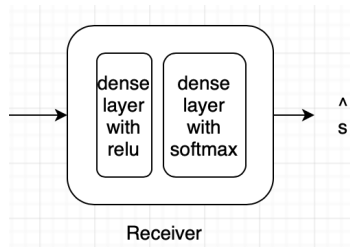

Figure 3.3: Receiver

softmax.

$$\frac{\partial S}{\partial u_i} = \sum_j \frac{\partial S_j}{\partial u_j} \tag{3.1.1}$$

when $j \neq i$:

$$\begin{aligned}
\frac{\partial S_j}{\partial u_i} &= \frac{\partial(\frac{e^{u_j}}{\sum_j e^{u_j}})}{\partial u_i} \\
&= -e^{u_j} \cdot \frac{1}{(\sum_j e^{u_j})^2} \cdot e^{u_i} \\
&= -\frac{e^{u_j}}{\sum_j e^{u_j}} \cdot \frac{e^{u_i}}{\sum_j e^{u_j}} \\
&= -S_j \cdot S_i
\end{aligned} \tag{3.1.2}$$

when $j = i$:

$$\begin{aligned}
\frac{\partial S_j}{\partial u_i} &= \frac{\partial S_i}{\partial u_i} \\
&= \frac{\partial(\frac{e^{u_i}}{\sum_j e^{u_j}})}{\partial u_i} \\
&= \frac{e^{u_i} \cdot \sum_j e^{u_j} - (e^{u_i})^2}{(\sum_j e^{u_j})^2} \\
&= \frac{e^{u_i}}{\sum_j e^{u_j}} \cdot \frac{\sum_j e^{u_j} - e^{u_i}}{\sum_j e^{u_j}} \\
&= \frac{e^{u_i}}{\sum_j e^{u_j}} \cdot (1 - \frac{e^{u_i}}{\sum_j e^{u_j}}) \\
&= S_i \cdot (1 - S_i)
\end{aligned} \tag{3.1.3}$$

It is known that if we want to find the local maximum of a function, obviously the first order derivative should be zero, however, (3.1.2) and (3.1.3) cannot be equal to 1. It means the softmax function does not have the local minimum point, and the weights in output layer can not update to the optimal.

3.1.1 Binary Case

We propose a method to get the optimal solution of the softmax function. We set all the bias b in the receiver end are 0. Figure 3.4 shows the diagram of the receiver end.

In this case, $M = 2, N = 2$, so the inputs should be $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$. We define the output for the first receiver layer is $\begin{bmatrix} z_{11} \\ z_{12} \end{bmatrix}$ and $\begin{bmatrix} z_{21} \\ z_{22} \end{bmatrix}$, the weight on last layer is $\mathbf{W}$, and the predicted outputs in the last layer before softmax activation function are

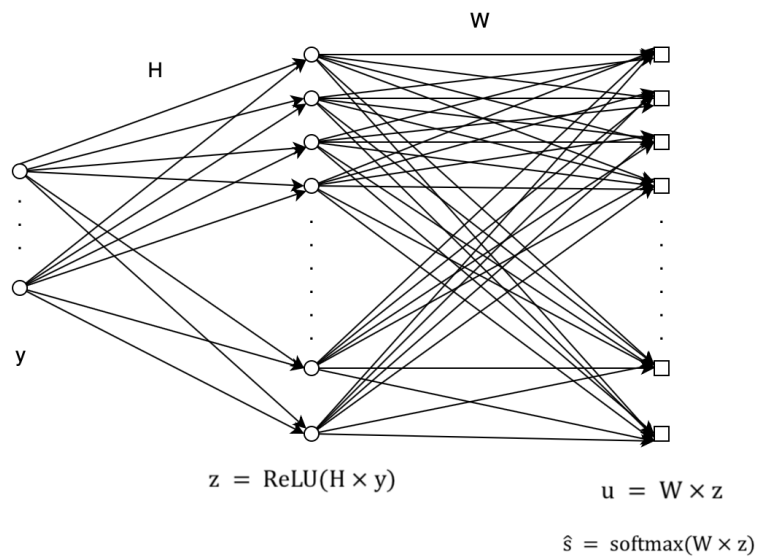

Figure 3.4: A diagram of the network structure in receiver end

$\begin{bmatrix} u_{11} \\ u_{12} \end{bmatrix}$ and $\begin{bmatrix} u_{21} \\ u_{22} \end{bmatrix}$. Then the loss function becomes

$$\begin{aligned} L &= -\log \frac{e^{u_{11}}}{e^{u_{11}} + e^{u_{12}}} - \log \frac{e^{u_{22}}}{e^{u_{21}} + e^{u_{22}}} \\ &= \log(1 + e^{u_{12}-u_{11}})(1 + e^{u_{21}-u_{22}}). \end{aligned} \tag{3.1.4}$$

We set $P = (1 + e^{u_{12}-u_{11}})(1 + e^{u_{21}-u_{22}})$, and we know that $\binom{u_{11}}{u_{12}} = \mathbf{W}\binom{z_{11}}{z_{12}}$ and $\mathbf{W} = \binom{w_1}{w_2}$, then we attain

$$u_{12} - u_{11} = (w_2 - w_1)z_1 = \Delta w z_1.$$

In a similar way, we attain

$$u_{21} - u_{22} = (w_1 - w_2)z_2 = -\Delta w z_2.$$

Therefore, we have

$$\begin{aligned} P &= (1 + e^{\Delta w z_1})(1 + e^{-\Delta w z_2}) \\ &= 1 + e^{\Delta w \Delta z} + e^{\Delta w z_1} + e^{-\Delta w z_2} \end{aligned} \tag{3.1.5}$$

where $\Delta z = z_1 - z_2$, our target is to make P as small as possible, hence we want to find the minimum value of P. We set the first derivative equal to zero

$$P' = e^{\Delta w \Delta z} \cdot \Delta z + e^{\Delta w z_1} \cdot z_1 + e^{-\Delta w z_2} \cdot -z_2 = 0.$$

Then we attain

$$(e^{\Delta w \Delta z} + e^{\Delta w \Delta z_1}) \cdot z_1 = (e^{-\Delta w z_2} + e^{\Delta w \Delta z}) \cdot z_2.$$

Assuming that $z_1 = C \cdot z_2$, where $C > 0$, we can get

$$\begin{aligned} C &= \frac{(z_2)^T z_1}{\|z_2\|^2} \\ &= \frac{e^{-\Delta w z_2} + e^{\Delta w \Delta z}}{e^{\Delta w \Delta z} + e^{\Delta w \Delta z_1}} \\ &= \frac{1 + e^{-\Delta w z_1}}{1 + e^{\Delta w z_2}} \end{aligned}$$

then we have

$$\begin{aligned} & 1 + e^{\Delta w z_2} = C^{-1} \cdot (1 + e^{-\Delta w z_1}) \\ \Leftrightarrow \quad & e^{\Delta w z_2} = \frac{1 + e^{-\Delta w z_1}}{C} - 1 \\ \Leftrightarrow \quad & e^{-\Delta w z_2} = \frac{C}{1 + e^{-\Delta w z_1} - C}. \end{aligned}$$

We set $x = e^{\Delta w z_1}$, hence, we have

$$\begin{aligned} P &= 1 + x + \frac{C}{1 + x^{-1} - C} + x \cdot \frac{C}{1 + x^{-1} - C} \\ &= 1 + x + \frac{C \cdot x}{(1 - C) \cdot x + 1} + \frac{C \cdot x^2}{(1 - C)x + 1} \\ &= 1 + \frac{x^2 + (1 + C)x}{(1 - C)x + 1}. \end{aligned}$$

We want to find the minimum value of P. We set the first derivative equal to zero:

$$\begin{aligned} & P' = (2x + (1 + C))((1 - C) \cdot x + 1) - (x^2 + (1 + C) \cdot x)(1 - c) = 0 \\ \Leftrightarrow \quad & (1 - C) \cdot x^2 + 2x + (1 + c) = 0 \\ \Leftrightarrow \quad & (1 - C)(x + 1)(x - \frac{C + 1}{C - 1}) = 0. \end{aligned}$$

When $C = 1$, we attain $z_1 = z_2$,

$$\begin{aligned} P &= 1 + e^{\Delta w \cdot 0} + e^{\Delta w z_1} + e^{-\Delta w z_2} \\ &= 2 + e^{\Delta w z_1} + e^{-\Delta w z_2} \\ &>= 4 \end{aligned}$$

hence we get $P_{min} = 4$.

When $C > 1$, there are two different situations. Firstly, when $-1 < x < \frac{C+1}{C-1}$, according to $x = e^{\Delta w z_1} > 0$, hence, when $0 < x < \frac{C+1}{C-1}$, P increases monotonically with the the increase of x. Secondly, when $x > \frac{C+1}{C-1}$, P decreases monotonically with the the increase of x. Therefore, when $x \to 0$ or $x \to \infty$, P gets the minimum value, then we attain $P > 1$.

When $0 < C < 1$, P increases monotonically with the the increase of x, therefore,

when $x \to 0$, P gets the minimum value, then we attain $P > 1$.

In summary, we get

$$P \begin{cases} > 1, & 0 < C < 1 \\ \geq 4, & C = 1 \\ > 1, & C > 1 \end{cases}$$

Now we consider the case that $z_1 \neq C \cdot z_2$, assuming that only one $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and one $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ send to the transmitter as the training sample, we know that $P = (1 + e^{\Delta wz_1}) \cdot (1 + e^{-\Delta wz_2})$ and our goal is to find the infimum of P, hence we need to let the exponent of e becomes small, in that case Δwz_1 and $-\Delta wz_2$ should be less than 0. Therefore we attain $\Delta wz_1 < 0$ and $\Delta wz_2 > 0$. Our goal is to let Δwz_1 becomes much less than 0 and at the same time Δwz_2 becomes much larger than 0. It is known that $\Delta wz_1 = \Delta w_1 \cdot z_{11} + \Delta w_2 \cdot z_{12}$ and $\Delta wz_2 = \Delta w_1 \cdot z_{21} + \Delta w_2 \cdot z_{22}$, hence, our task transforms to

$$\Delta w_2 < \frac{-\Delta w_1 \cdot z_{11}}{z_{12}}$$
$$\Delta w_2 > \frac{-\Delta w_1 \cdot z_{21}}{z_{22}}.$$

Therefore, we should modify the weights in the previous layers to make sure $\frac{-\Delta w_1 \cdot z_{11}}{z_{12}} > \Delta w_2 > \frac{-\Delta w_1 \cdot z_{21}}{z_{22}}$. Fig. 3.5 shows an example, what we need is to make our choice locate in the shadow area and far from these two lines. So, we calculate the angle bisector of the included angle of the two lines and choose a point which is at the angle bisector and far from lines for $\Delta w_2 = \frac{-\Delta w_1 \cdot z_{11}}{z_{12}}$ and $\Delta w_2 = \frac{-\Delta w_1 \cdot z_{21}}{z_{22}}$. The coordinate of this point is the value of Δw_1 and Δw_2.

In the experiment, in order to balance the effects of noise, we need to set the

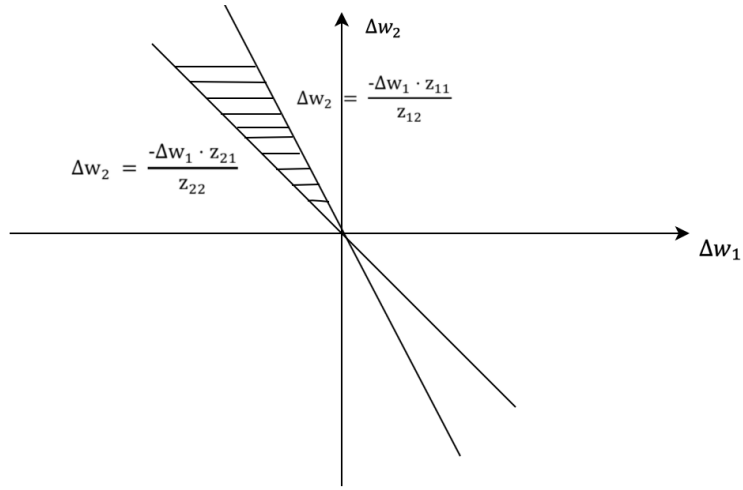

Figure 3.5: A diagram of the choice for Δw_1 and Δw_2

batch size N more than 10^3, hence, we define the outputs from first dense layer in the receiver are

$$z_i^{(1)} = \begin{pmatrix} z_{i1}^{(1)} \\ z_{i2}^{(1)} \end{pmatrix}$$

and

$$z_i^{(2)} = \begin{pmatrix} z_{i1}^{(2)} \\ z_{i2}^{(2)} \end{pmatrix}$$

where $i = 1, 2, ...N$. Then we attain

$$\Delta w_2 < \frac{-\Delta w_1 \cdot z_{i1}^{(1)}}{z_{i2}^{(1)}}$$
$$\Delta w_2 > \frac{-\Delta w_1 \cdot z_{i1}^{(2)}}{z_{i2}^{(2)}}$$

we define that $\frac{z^{(1)}_{min,1}}{z^{(1)}_{min,2}} = \min\left\{\frac{z^{(1)}_{i1}}{z^{(1)}_{i2}}\right\}_{i=1}^{N}$ and $\frac{z^{(2)}_{max,1}}{z^{(2)}_{max,2}} = \max\left\{\frac{z^{(2)}_{i1}}{z^{(2)}_{i2}}\right\}_{i=1}^{N}$, hence, we get

$$\begin{aligned} \Delta w_2 &< \frac{-\Delta w_1 \cdot z^{(1)}_{min,1}}{z^{(1)}_{min,2}} \\ \Delta w_2 &> \frac{-\Delta w_1 \cdot z^{(2)}_{max,1}}{z^{(2)}_{max,2}} \end{aligned} \tag{3.1.6}$$

Then similar with the one sample case, we find the point in the angle bisector of the included angle. For reducing the computation time and complexity, we set the weights in layers for transmitters and first layer in receiver randomly picked from $\{-1, 0, 1\}$, and the simulation results are shown in Chapter 4.

3.2 Multiple Input and Multiple Output Case

In this section, we introduce $L_{1/2}$ regularization to the MIMO cases. We design the two-user and three-user cases. We introduce the $L_{1/2}$ regularization in loss functions. Through the simulations, we have found that the loss function with $L_{1/2}$ regularization has the better results than the original categorical cross-entropy.

3.2.1 Two-input and Two-output Case

For two users situation, we consider a two-user disturbance channel. Each user has one base station which includes one transmitting antenna and one receiving antenna. The channel is the typical uplink multi-access channel. Fig. 3.6 shows the diagram of

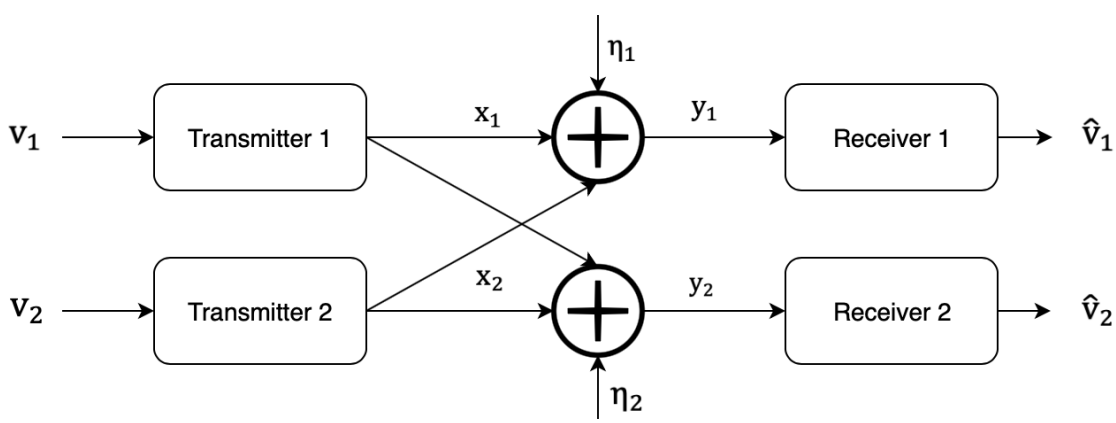

Figure 3.6: A diagram of two users channel

two disturbing autoencoders which try to regenerate their respective messages. The first transmitter sends message $\mathbf{v_1} \in \mathbf{M}$ to the first receiver, and at the same time, the second transmitter sends message $\mathbf{v_2} \in \mathbf{M}$ to the second receiver. This two users channel is also implemented by Neural Networks. The differences between single user

and two users are that the transmitted signals $\mathbf{x_1}$ and $\mathbf{x_2}$ disturbing at the receivers with the noise. In other words, each receiver receive the signals from not only its own transmitter but also from the others:

$$\mathbf{y_1} = \mathbf{x_1} + \mathbf{x_2} + \boldsymbol{\eta_1}$$

$$\mathbf{y_2} = \mathbf{x_2} + \mathbf{x_1} + \boldsymbol{\eta_2}$$

where $\boldsymbol{\eta_1}$ and $\boldsymbol{\eta_2}$ are Gaussian noise. We define L_1 and L_2 represent the loss of the first autoencoder and the second autoencoder, separately. In the training process, it is unclear that how to train this two paired autoencoders with disturbing objectives. One strategy is to minimize a weighted sum of the two losses, i.e.,

$$L = \alpha L_1 + \beta L_2$$

where $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$. However, setting same ratio of weight for this two losses, i.e., $\alpha = \beta = 0.5$ usually makes unoptimal results. Therefore, we introduce the dynamic weights for α and β in each mini-batch t:

$$\alpha_{t+1} = \frac{L_1(t)}{L_1(t) + L_2(t)}$$
$$\beta_{t+1} = \frac{L_2(t)}{L_1(t) + L_2(t)}$$

where $\alpha_0 = \beta_0 = 0.5$. Using this approach, our experiments get the perfect solutions for Symbol Error Rate for both autoencoders users in same average power cases.

3.2.2 Three-input and Three-output Case

The three users case is similar with the two-user case. Each user has one base station which includes one transmitting antenna and one receiving antenna. The channel uses typical up-link multi-access channel. Fig. 3.7 shows the diagram of three disturbing autoencoders which try to regenerate their respective messages. This three transmitters send the messages $\mathbf{v_1}$, $\mathbf{v_2}$ and $\mathbf{v_3}$ at the same time, and the transmitters receive the interference and transmitted signals $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$ with the noise.

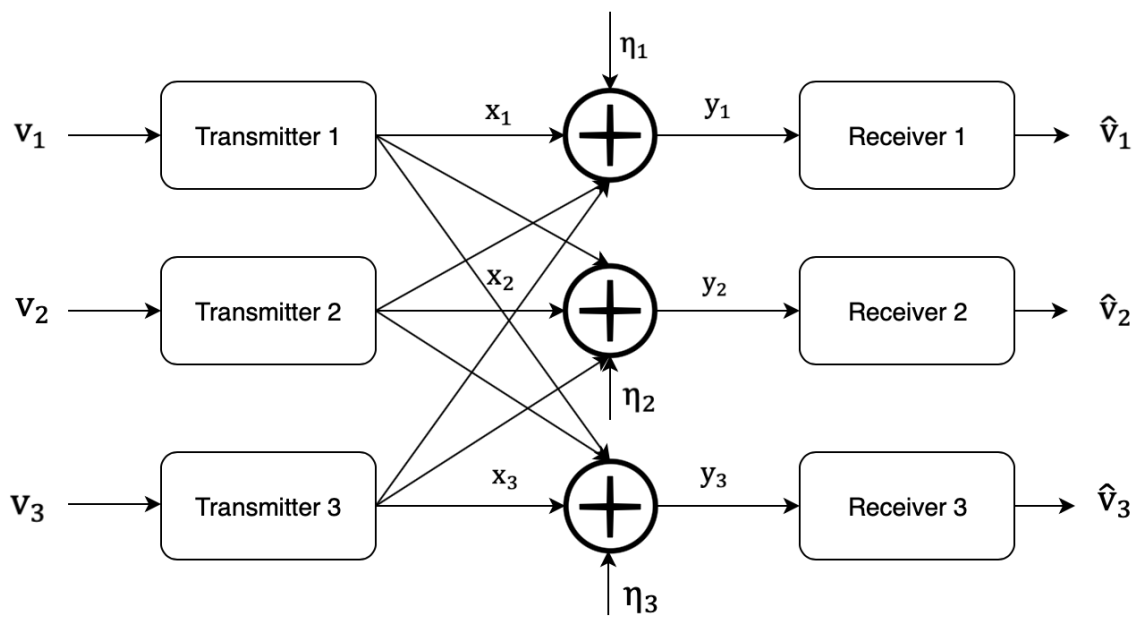

Figure 3.7: A diagram of three users channel

$$\mathbf{y_1} = \mathbf{x_1} + \mathbf{x_2} + \mathbf{x_3} + \boldsymbol{\eta_1}$$

$$\mathbf{y_2} = \mathbf{x_2} + \mathbf{x_1} + \mathbf{x_3} + \boldsymbol{\eta_2}$$

$$\mathbf{y_3} = \mathbf{x_3} + \mathbf{x_1} + \mathbf{x_2} + \boldsymbol{\eta_3}$$

where $\boldsymbol{\eta_1}$, $\boldsymbol{\eta_2}$ and $\boldsymbol{\eta_3}$ are Gaussian noise. This time we define L_1, L_2 and L_3 of the

loss for each autoencoder system, respectively. Hence the whole loss is

$$L = \alpha L_1 + \beta L_2 + \gamma L_3$$

where $\alpha, \beta, \gamma \in [0, 1]$ and $\alpha + \beta + \gamma = 1$. Similarly, the dynamic weights in this three-user channel are

$$\alpha_{t+1} = \frac{L_1(t)}{L_1(t) + L_2(t) + L_3(t)}$$
$$\beta_{t+1} = \frac{L_2(t)}{L_1(t) + L_2(t) + L_3(t)}$$
$$\gamma_{t+1} = \frac{L_3(t)}{L_1(t) + L_2(t) + L_3(t)}$$

where $\alpha_0 = \beta_0 = \gamma_0 = \frac{1}{3}$.

Chapter 4

Simulations

4.1 Single Input and Single Output Case

In this section, for the experiment, we send the two-dimensional one-hot vectors through two channels (i.e., $(M, n) = (2, 2)$). We compare the block error rate between our method and autoencoder network training with the Categorical cross-entropy loss. It is clear to see that our method has better results than the autoencoder network with the Categorical cross-entropy loss.

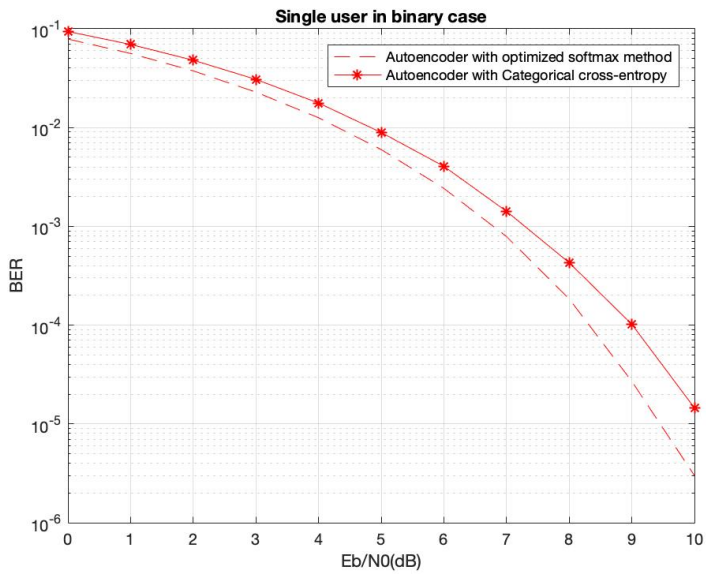

Figure 4.8: Symbol error rate comparison between Categorical cross-entropy loss and our method

4.2 Multiple Input and Multiple Output Case

In all multi-users cases, we fix the output dimension for the transmitters to $n = 2$, and that means at each time, transmitters send a 2×1 vector to the receivers.

4.2.1 Two-Input and Two-Output Case

Two-Input and Two-Output in the Same Average Power Constraint Case

In the same average power cases, we fix the average power for both users to $\mathbf{E}[|x_i|^2] = 2$.

Four Points Simulation

Fig. 4.9 and Fig. 4.10 show the representation x for transmitters in each users for $L_{1/2}$ norm loss and Categorical cross-entropy loss, separately. Fig. 4.11 shows the sum constellation for $L_{1/2}$ norm loss and Categorical cross-entropy loss. It can be seen that the two sum constellations are similar. The d_{min} of $L_{1/2}$ norm loss sum constellation is 1.232 and the d_{min} of Categorical cross-entropy loss sum constellation is 1.222. From the symbol error rate graph (Fig. 4.12), we can see that the four lines almost coincide with each other, only at 10dB, $L_{1/2}$ norm loss has a slight advantage.

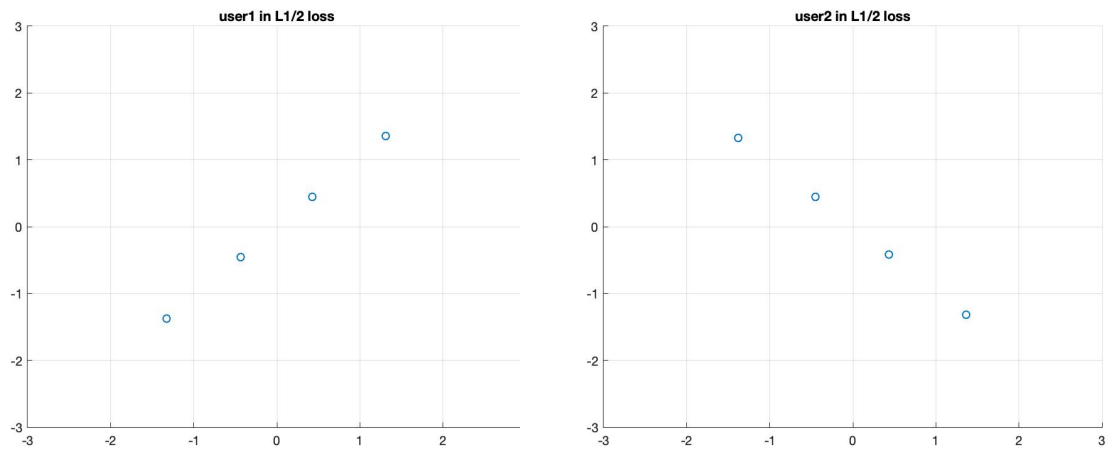

Figure 4.9: User 1 and user 2 constellations in $L_{1/2}$ norm loss

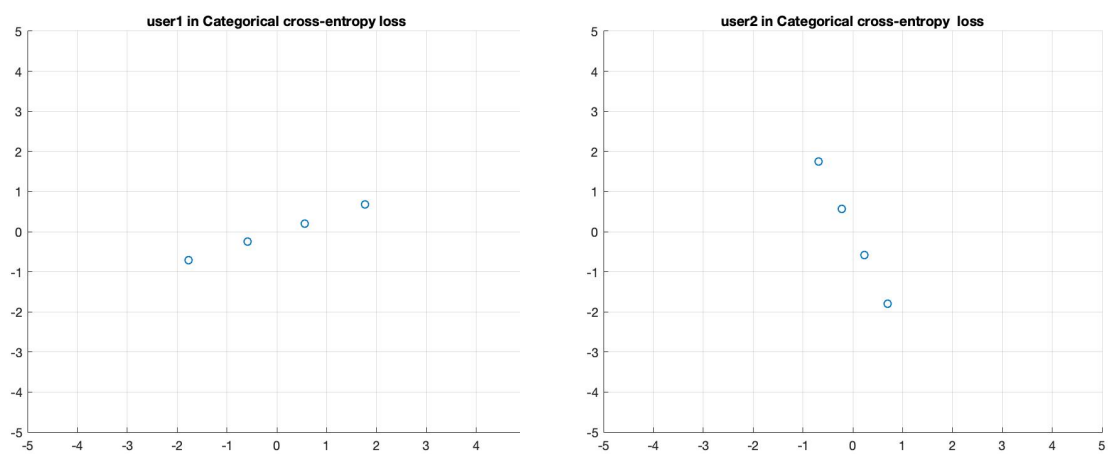

Figure 4.10: User 1 and user 2 constellations in Categorical cross-entropy loss

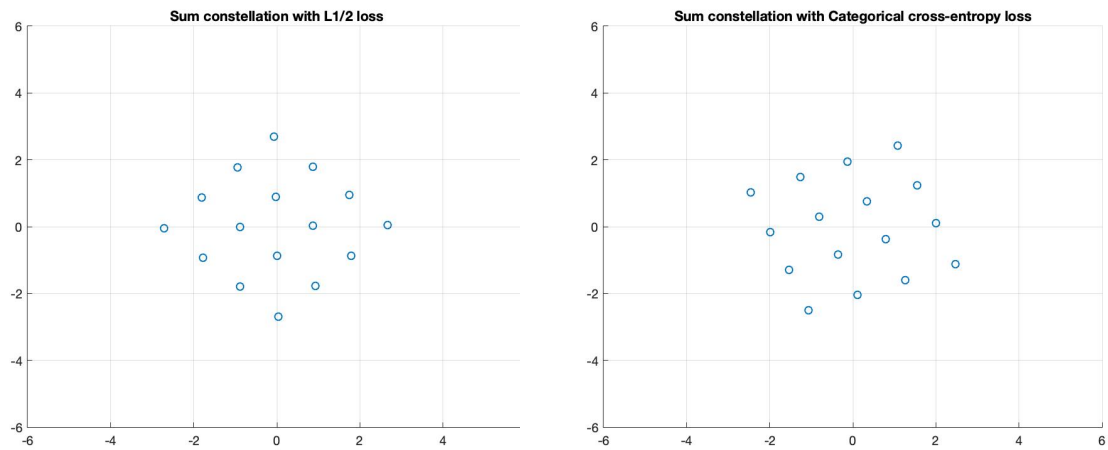

Figure 4.11: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

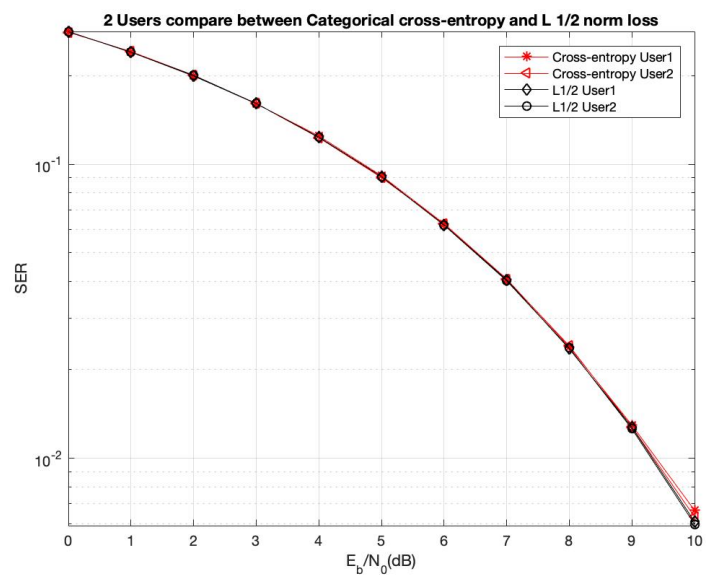

Figure 4.12: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in two users case

Eight Points Simulation

Fig. 4.13 and Fig. 4.14 show the representation x for transmitters in each users for $L_{1/2}$ norm loss and Categorical cross-entropy loss, separately. Fig. 4.15 shows the sum constellation for $L_{1/2}$ norm loss and Categorical cross-entropy loss. It can be seen that the two sum constellations are similar. The d_{min} of $L_{1/2}$ norm loss sum constellation is 0.561 and the d_{min} of Categorical cross-entropy loss sum constellation is 0.552. From the symbol error rate graph (Fig. 4.16), we can see that the four lines almost coincide with each other, that means two autoencoders have almost same performance.

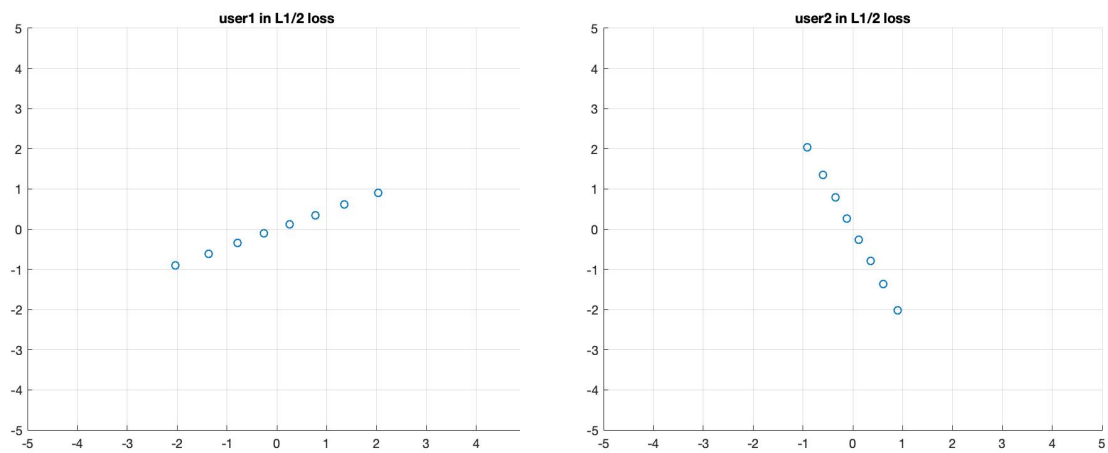

Figure 4.13: User 1 and user 2 constellations in $L_{1/2}$ norm loss

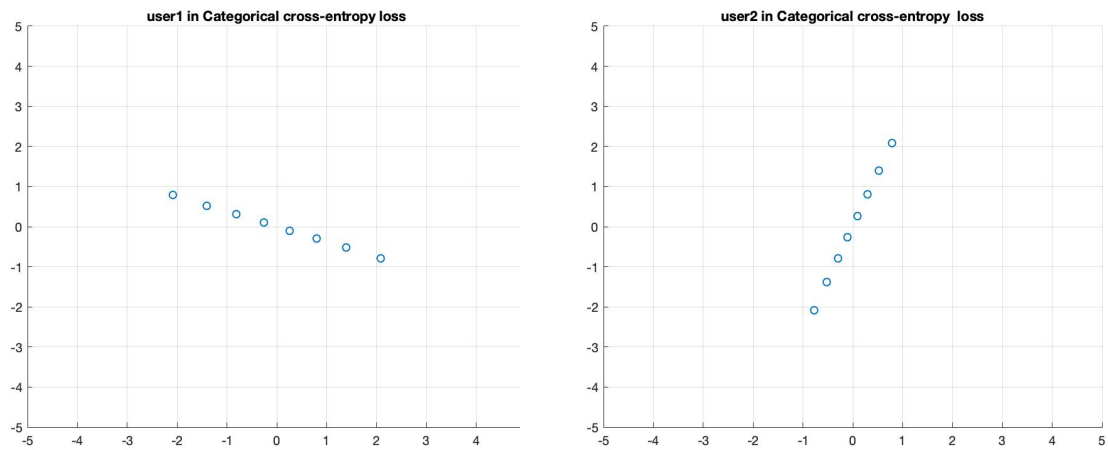

Figure 4.14: User 1 and user 2 constellations in Categorical cross-entropy loss

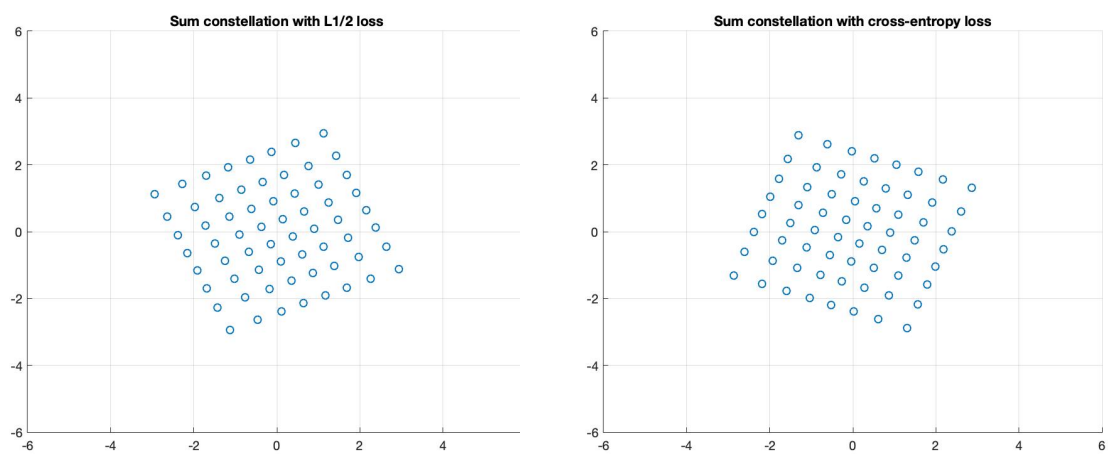

Figure 4.15: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

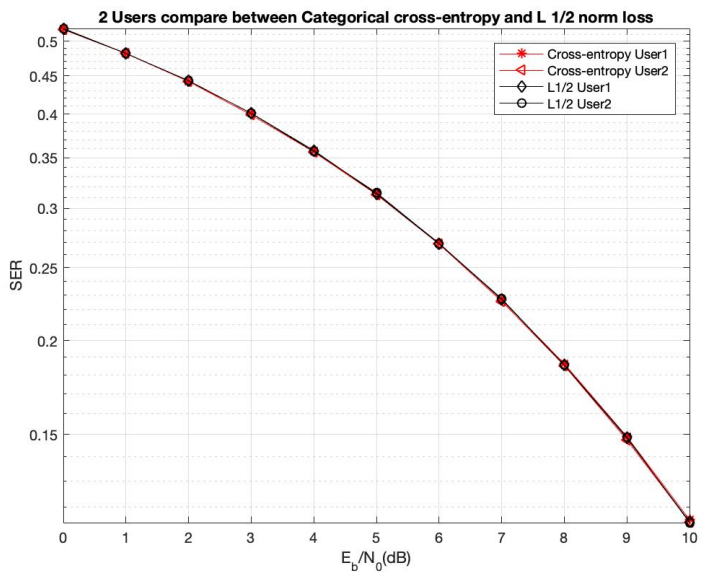

Figure 4.16: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in two users case

Sixteen Points Simulation

Fig. 4.17 and Fig. 4.18 show the representation x for transmitters in each users for $L_{1/2}$ norm loss and Categorical cross-entropy loss, separately. Fig. 4.19 shows the sum constellation for $L_{1/2}$ norm loss and Categorical cross-entropy loss, it can be seen that the two sum constellations are similar. The d_{min} of $L_{1/2}$ norm loss sum constellation is 0.238 and the d_{min} of Categorical cross-entropy loss sum constellation is 0.146, From the symbol error rate graph (Fig. 4.32), we can see that the the four lines almost coincide with each other. $L_{1/2}$ norm loss has a slight advantage from 7dB to 10dB.

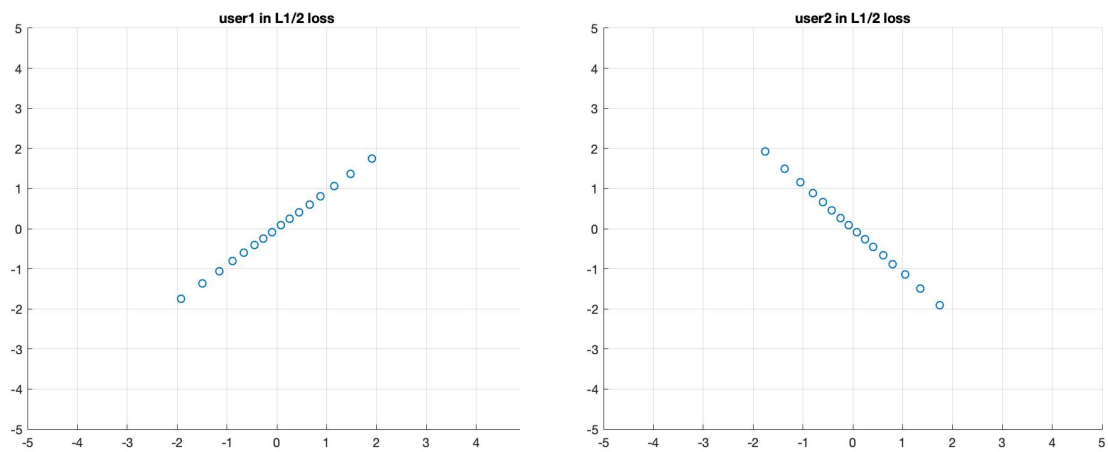

Figure 4.17: User 1 and user 2 constellations in $L_{1/2}$ norm loss

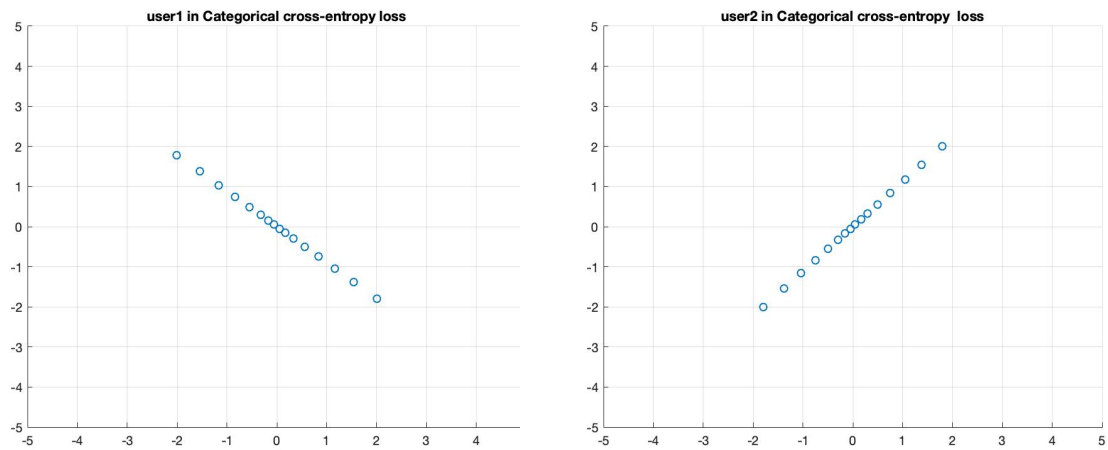

Figure 4.18: User 1 and user 2 constellations in Categorical cross-entropy loss

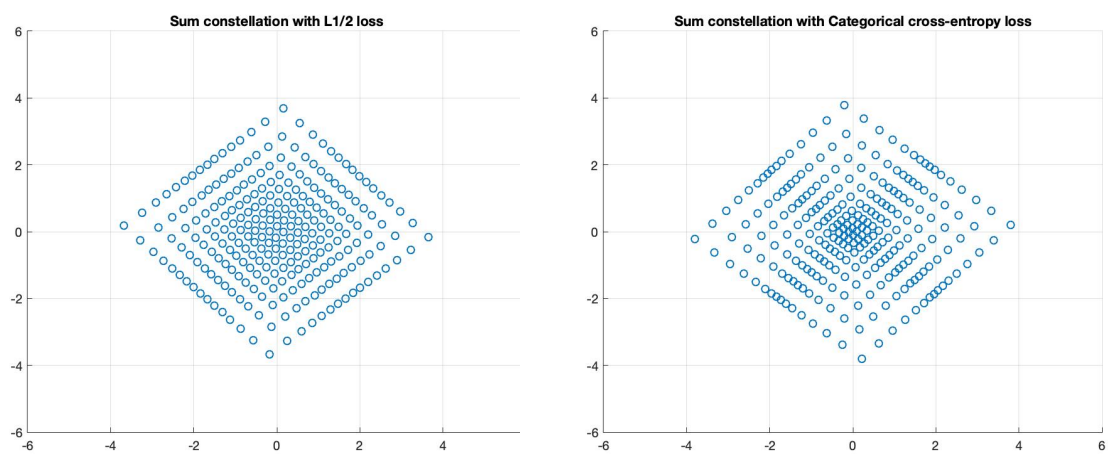

Figure 4.19: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

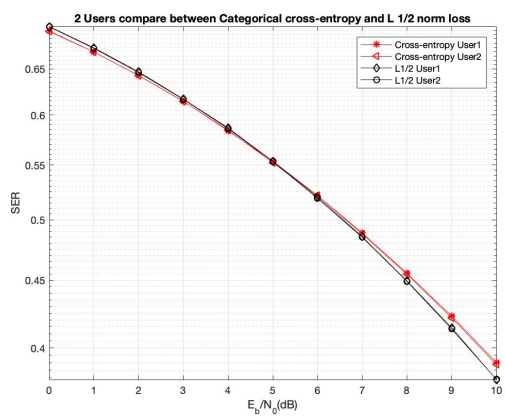

Figure 4.20: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in two users case

Two-Input and Two-Output in Different Average Power Constraint Case

In this experiment, we set the average power constraint in user 1 to $\mathbf{E}[|x_i|^2] = 2$ and set the average power constraint in user 2 to $\mathbf{E}[|x_i|^2] = 8$.

Four points simulation

In this experiment, we compare the symbol error rate with $L_{1/2}$ norm loss and Categorical cross-entropy loss. Fig. 4.21 and Fig. 4.22 show the studied representations of x which are sent by transmitters as the constellations for each users in $L_{1/2}$ norm loss and Categorical cross-entropy loss. Fig. 4.23 shows the representations of sum constellation in $L_{1/2}$ norm loss and Categorical cross-entropy loss, the d_{min} of sum constellation in $L_{1/2}$ norm loss is 1.238 and d_{min} of sum constellation in Categorical cross-entropy is 1.226. Fig. 4.12 shows the symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss. It can be seen that symbol error rate for both user 1 coincide with the other one, however, user 2 for $L_{1/2}$ norm loss has the clear advantage than user 2 for Categorical cross-entropy. Therefore, the $L_{1/2}$ loss gets better performance than Categorical cross-entropy loss.

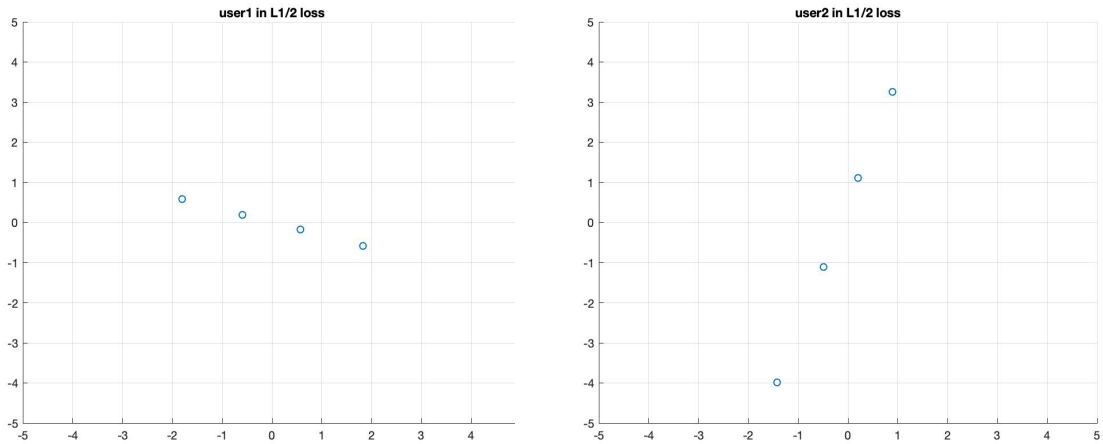

Figure 4.21: User 1 and user 2 constellations in $L_{1/2}$ norm loss

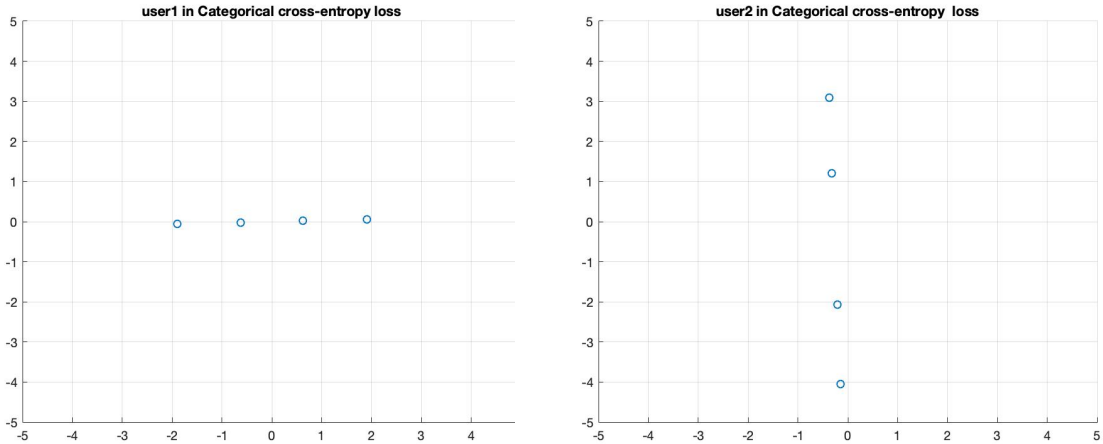

Figure 4.22: User 1 and user 2 constellations in Categorical cross-entropy loss

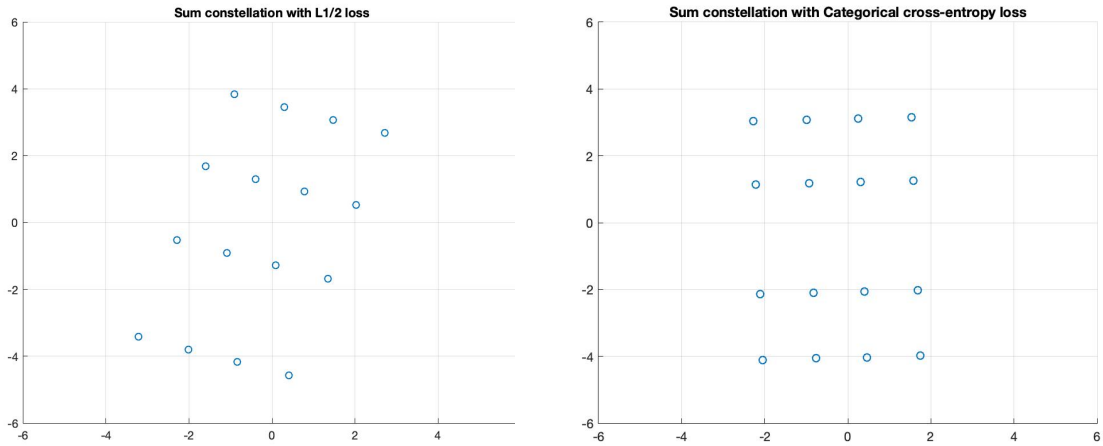

Figure 4.23: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

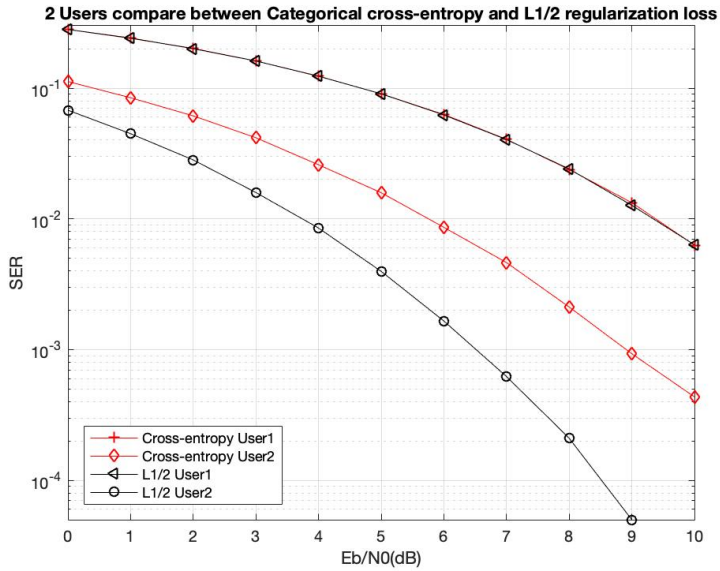

Figure 4.24: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in two users case

Eight Points Simulation

Fig. 4.25 and Fig.4.26 show the studied representations of x which are sent by transmitters as the constellations for each users in $L_{1/2}$ norm loss and Categorical cross-entropy loss. Fig. 4.27 shows the representations of sum constellation in $L_{1/2}$ norm loss and Categorical cross-entropy loss. The d_{min} of sum constellation in $L_{1/2}$ norm loss is 0.580 and d_{min} of sum constellation in Categorical cross-entropy is 0.576. Fig. 4.28 shows the symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss. It can be seen that symbol error rate for both user 1 coincide with the other one, however, user 2 for $L_{1/2}$ norm loss has the clear advantage than user 2 for Categorical cross-entropy. Therefore, the $L_{1/2}$ loss gets better performance than Categorical cross-entropy loss.

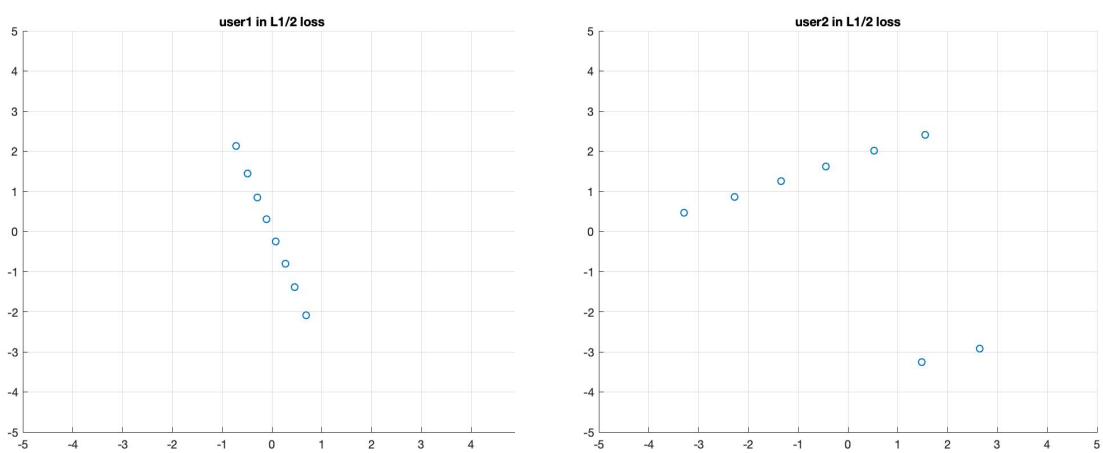

Figure 4.25: User 1 and user 2 constellations in $L_{1/2}$ norm loss

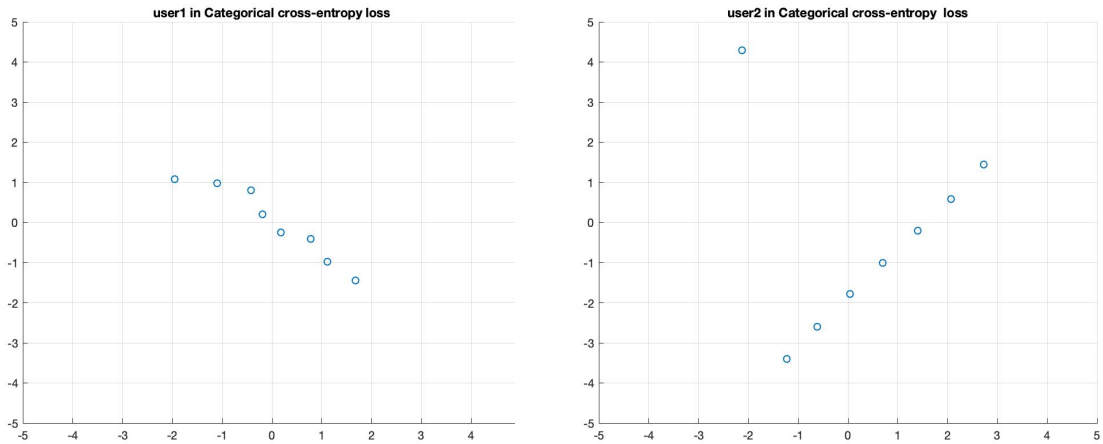

Figure 4.26: User 1 and user 2 constellations in Categorical cross-entropy loss

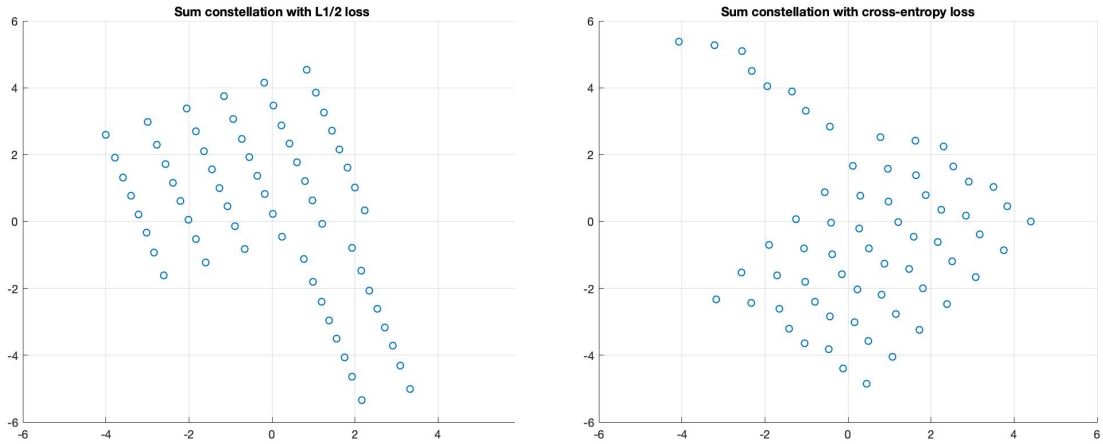

Figure 4.27: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

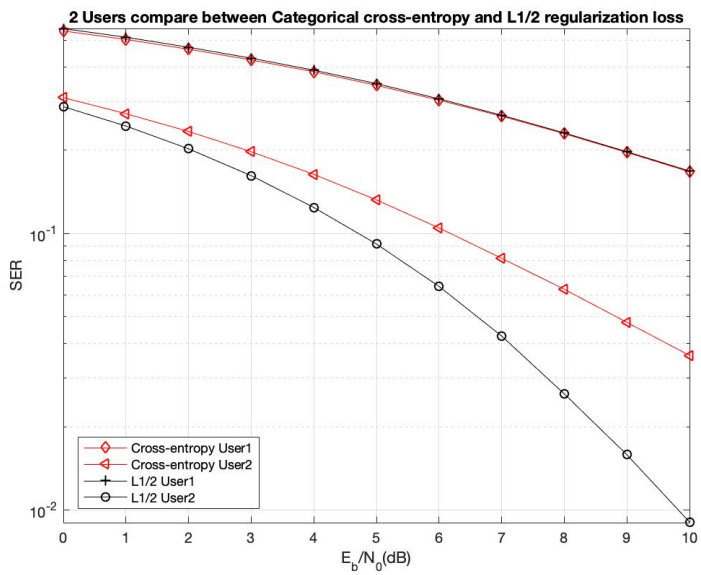

Figure 4.28: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in two users case

Sixteen Points Simulation

Fig. 4.29 and Fig. 4.30 show the studied representations of x which are sent by transmitters as the constellations for each users in $L_{1/2}$ norm loss and Categorical cross-entropy loss. Fig. 4.31 shows the representations of sum constellation in $L_{1/2}$ norm loss and Categorical cross-entropy loss, the d_{min} of sum constellation in $L_{1/2}$ norm loss is 0 and d_{min} of sum constellation in Categorical cross-entropy is 0.254. Fig. 4.32 shows the symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss. It can be seen that symbol error rate for both user 1 coincide with the other one, however, user 2 for $L_{1/2}$ norm loss has the clear advantage than user 2 for Categorical cross-entropy. Therefore, the $L_{1/2}$ loss gets better performance than Categorical cross-entropy loss.

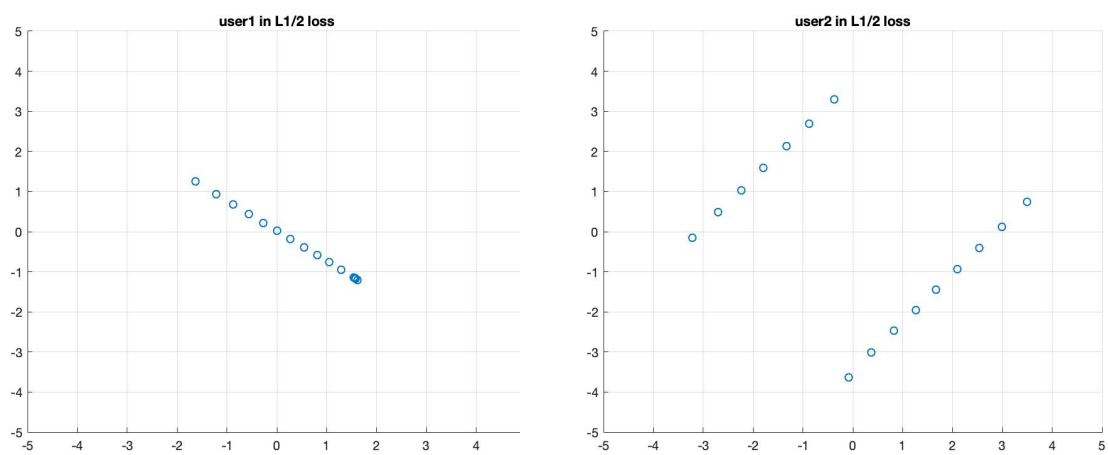

Figure 4.29: User 1 and user 2 constellations in $L_{1/2}$ norm loss

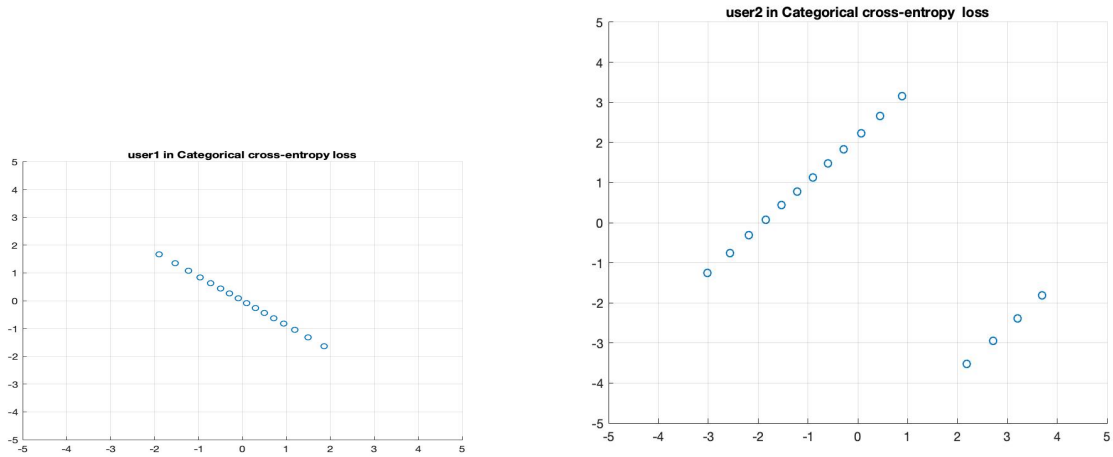

Figure 4.30: User 1 and user 2 constellations in Categorical cross-entropy loss

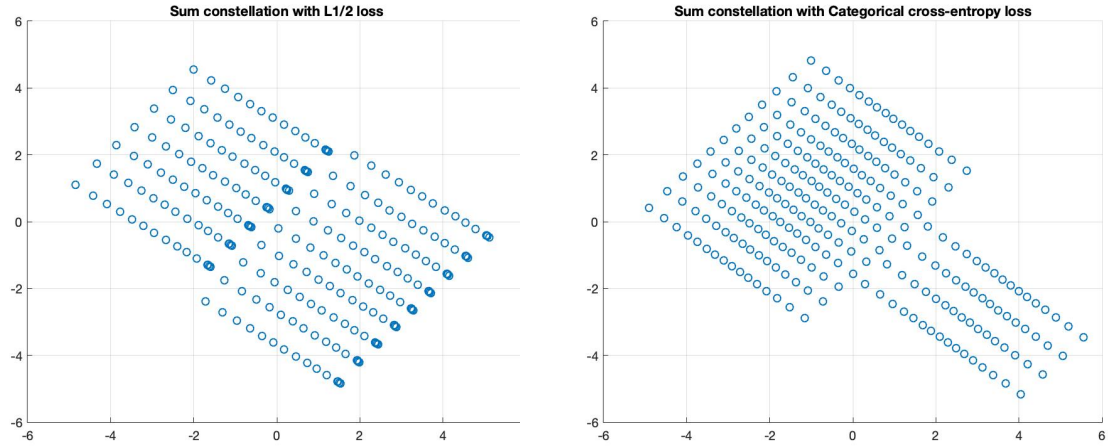

Figure 4.31: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

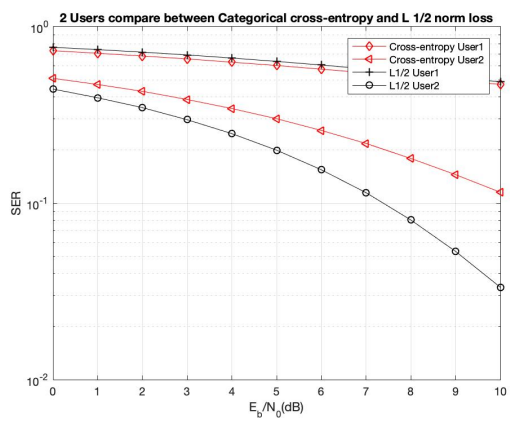

Figure 4.32: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in two users case

4.2.2 Three-input and Three-output Case

Three-input and Three-output in the Same Average Power Constraint Case

In this experiment, we set the average power constraint for all users to $\mathbf{E}[|x_i|^2] = 2$. Fig. 4.33, Fig. 4.34 and Fig. 4.35 show the learned representation of which send by transmitters as the constellation and the interference for each users in $L_{1/2}$ norm loss. It can be seen that for user 1 and user 2, both of their constellations have four points and both of their interference constellations have twelve points. However, for user 3, its constellation has three points and its interference constellations has sixteen points. Fig. 4.36, Fig. 4.37 and Fig. 4.38 show the learned representation of which send by transmitters as the constellation and the interference for each users in Categorical cross-entropy loss. It can be seen that for user 1 and user 3, both of their constellations have three points and both of their interference constellations have 12 points. However, for user 2, its constellation has four points and its interference constellations has nine points. Fig. 4.39 shows the representations of sum constellation in $L_{1/2}$ norm loss and Categorical cross-entropy loss, the d_{min} of sum constellation in $L_{1/2}$ norm loss is 0.013 and d_{min} of sum constellation in Categorical cross-entropy is 0. Fig. 4.40 shows the symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss, Fig. 4.41 shows the average symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss. It can be seen clearly that $L_{1/2}$ norm loss has the advantage than Categorical cross-entropy. Therefore, the $L_{1/2}$ loss gets better performance than Categorical cross-entropy loss.

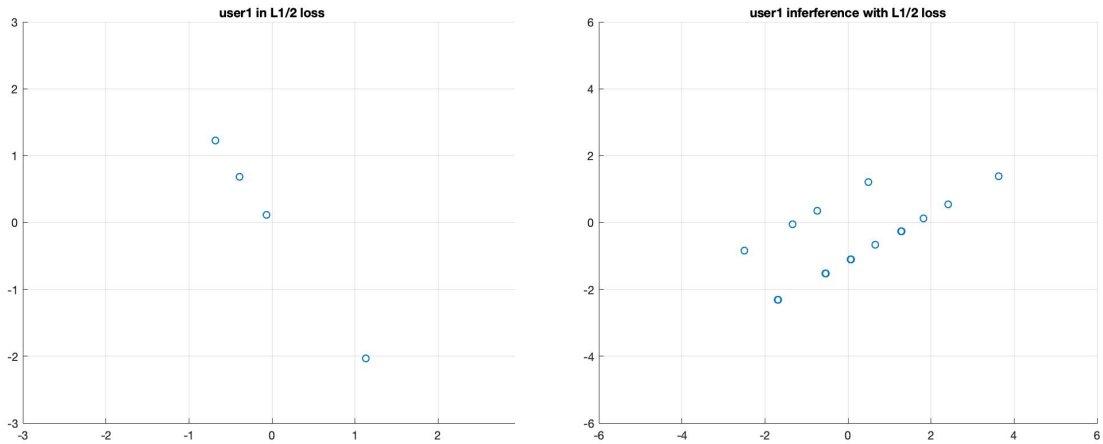

Figure 4.33: User 1 constellation and interference constellation in $L_{1/2}$ norm loss

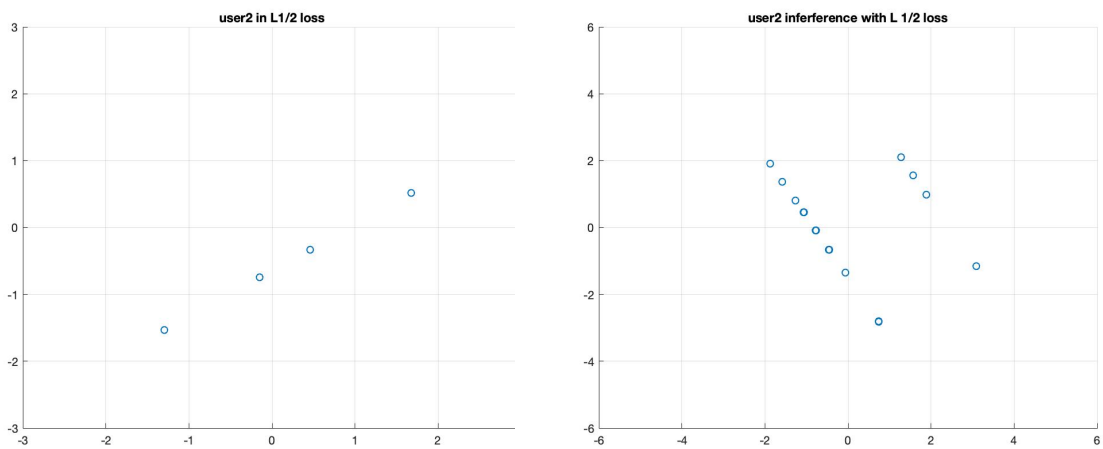

Figure 4.34: User 2 constellation and interference constellation in $L_{1/2}$ norm loss

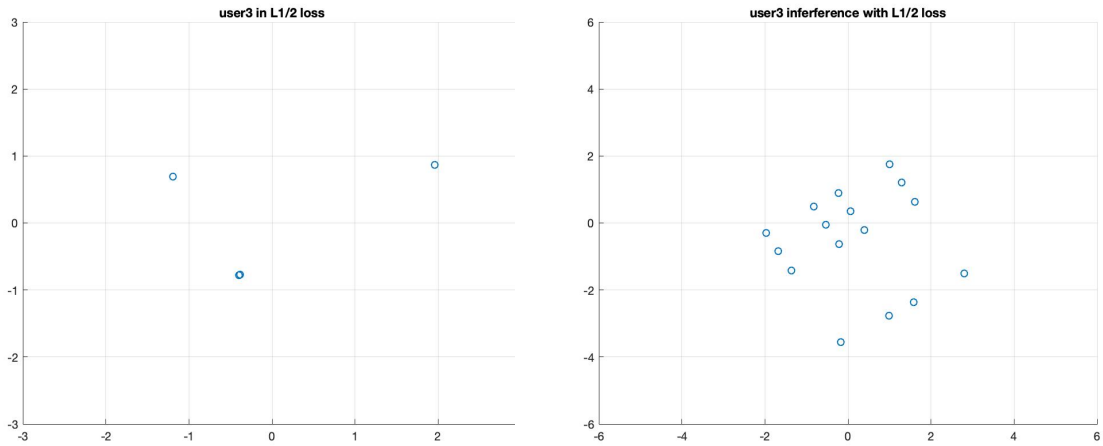

Figure 4.35: User 3 constellation and interference constellation in $L_{1/2}$ norm loss

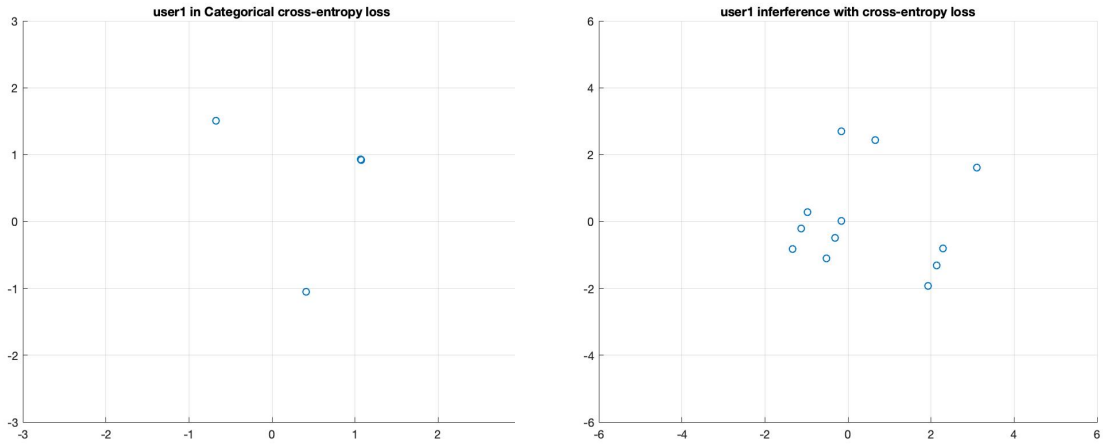

Figure 4.36: User 1 constellation and interference constellation in Categorical cross-entropy loss

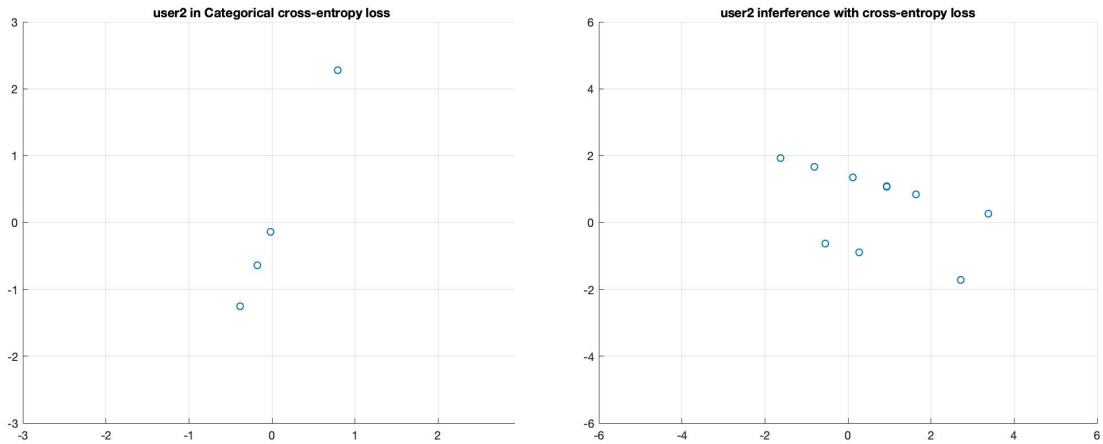

Figure 4.37: User 2 constellation and interference constellation in Categorical cross-entropy loss

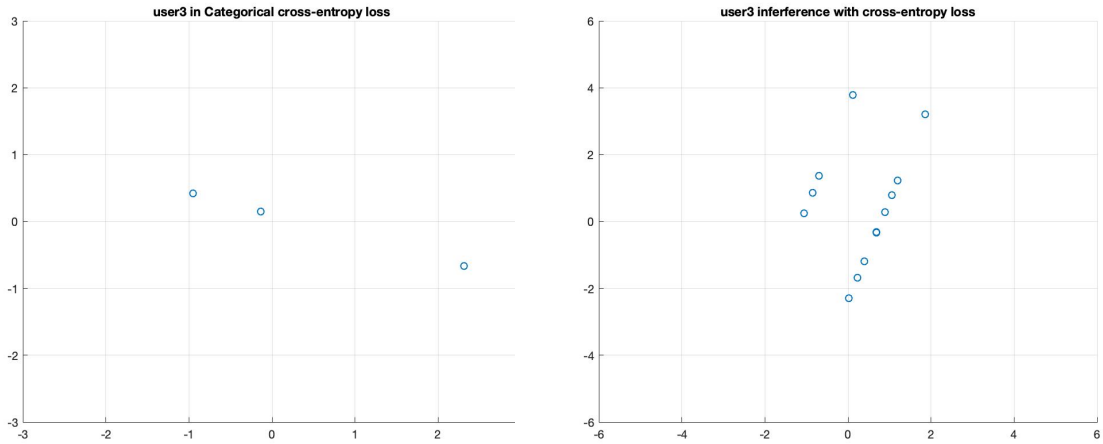

Figure 4.38: User 3 constellation and interference constellation in Categorical cross-entropy loss

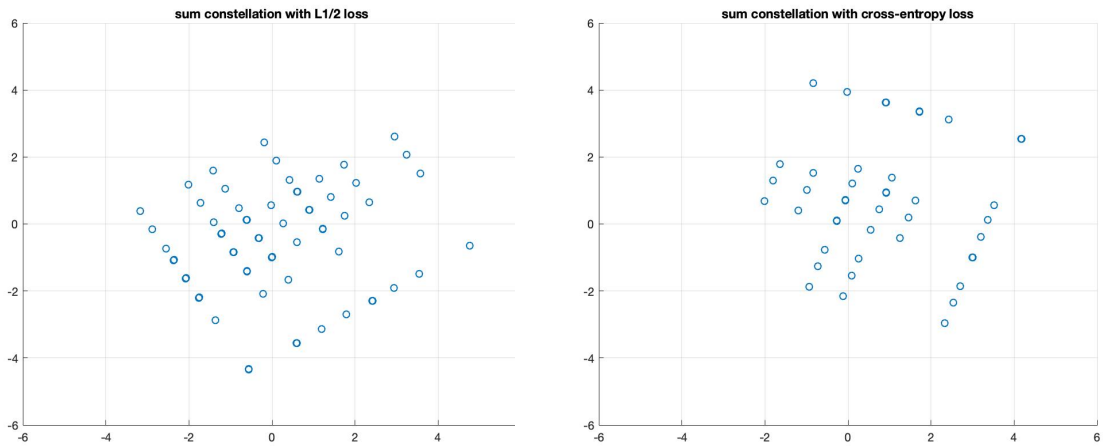

Figure 4.39: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

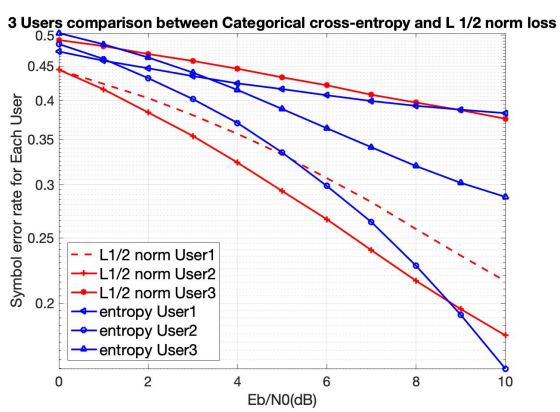

Figure 4.40: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in three users case

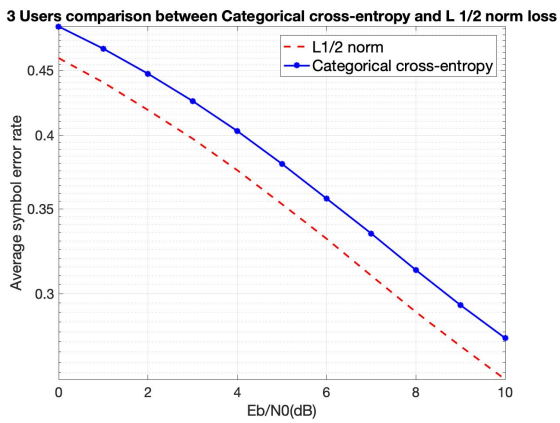

Figure 4.41: Average symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in three users case

Three-input and Three-output in the Different Average Power Constraint Case

In this experiment, we set the average power constraint for user 1 to $\mathbf{E}[|x_i|^2] = 2$, set for user 2 to $\mathbf{E}[|x_i|^2] = 8$ and set for user 3 to $\mathbf{E}[|x_i|^2] = 32$. Fig. 4.42, Fig. 4.43 and Fig. 4.44 show the learned representations of which are sent by transmitters as the constellations and the interference for each users in $L_{1/2}$ norm loss. It can be seen that for all users, their constellations have four points and both of their interference constellations have twelve points. Fig. 4.36, Fig. 4.37 and Fig. 4.38 show the learned representation of which send by transmitters as the constellation and the interference for each users in Categorical cross-entropy loss. It can be seen that for user 2 and user 3, both of their constellations have four points and both of the interference constellations have twelve points. However, for user 1, its constellation has three points and its interference constellation has sixteen points. Fig. 4.48 shows the representations of sum constellation in $L_{1/2}$ norm loss and Categorical cross-entropy loss, the d_{min} of sum constellation in $L_{1/2}$ norm loss is 0.879 and the sum constellation in Categorical cross-entropy loss is 0.008. Fig. 4.49 shows the symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss, Fig. 4.50 shows the average symbol error rate between Categorical cross-entropy and our $L_{1/2}$ norm loss. It can be seen clearly that $L_{1/2}$ norm loss has the advantage than Categorical cross-entropy. Therefore, the $L_{1/2}$ loss gets better performance than Categorical cross-entropy loss.

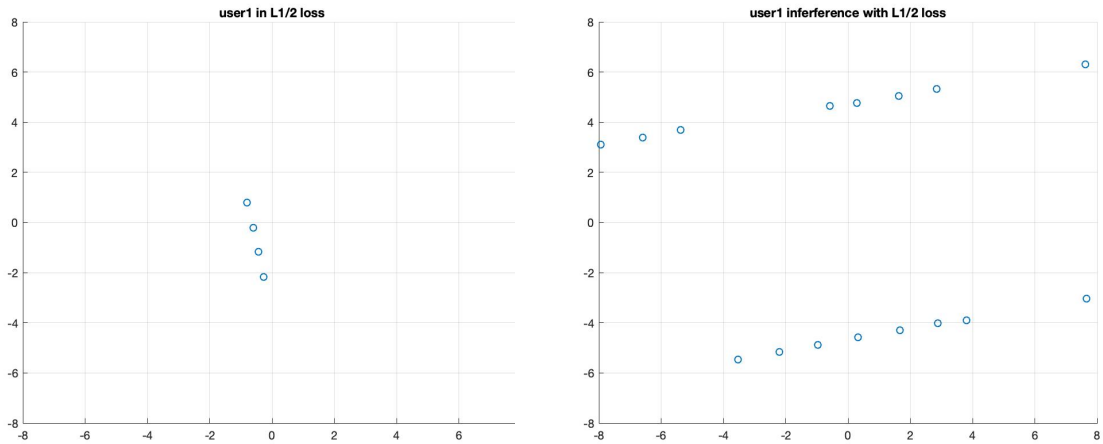

Figure 4.42: User 1 constellation and interference constellation in $L_{1/2}$ norm loss

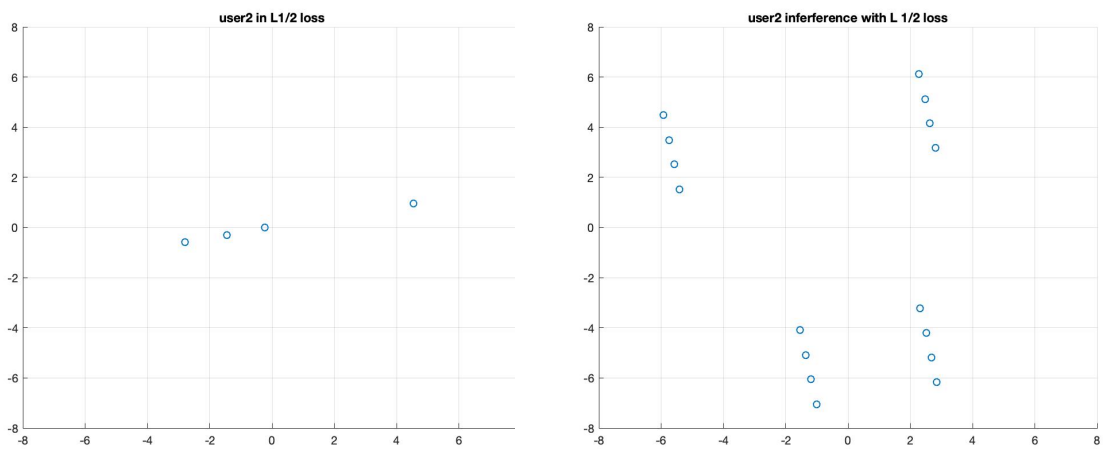

Figure 4.43: User 2 constellation and interference constellation in $L_{1/2}$ norm loss

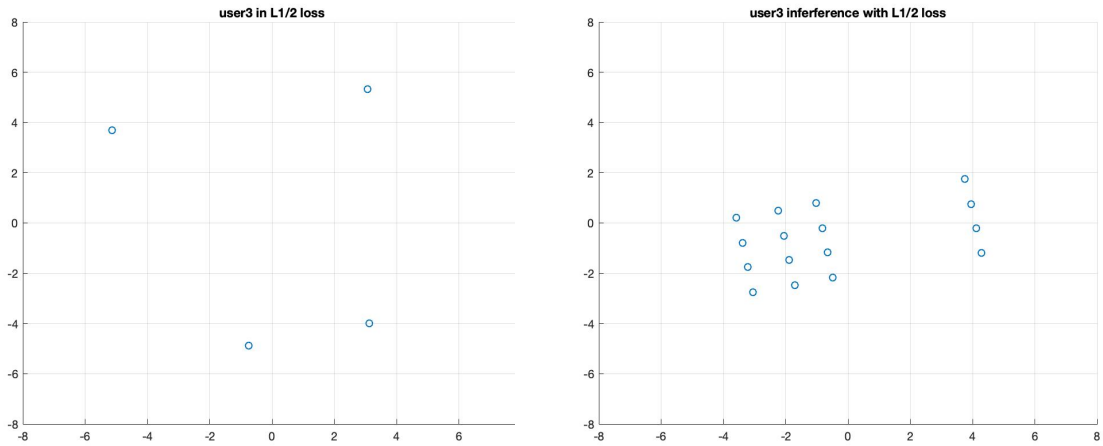

Figure 4.44: User 3 constellation and interference constellation in $L_{1/2}$ norm loss

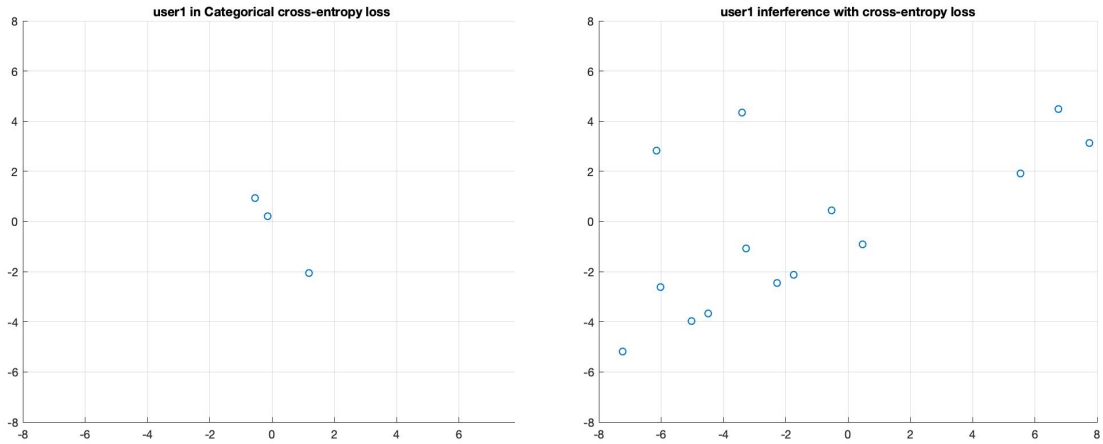

Figure 4.45: User 1 constellation and interference constellation in Categorical cross-entropy loss

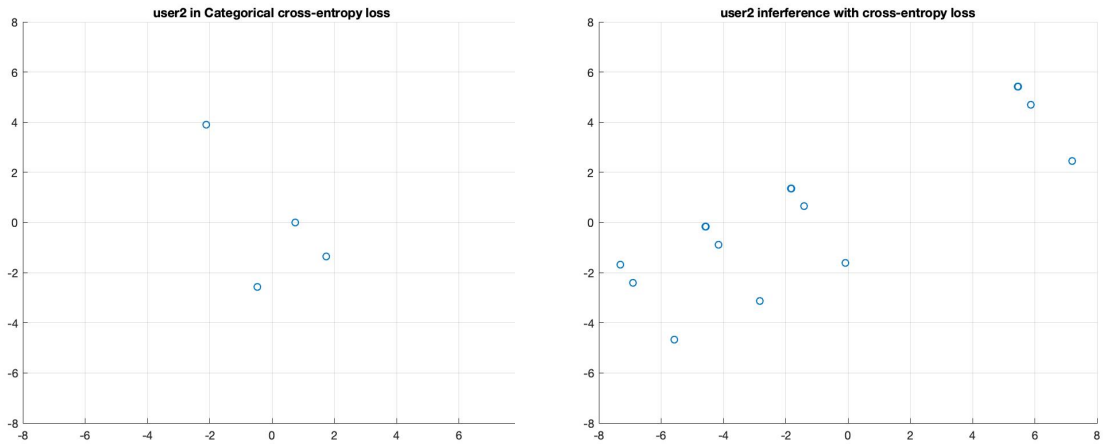

Figure 4.46: User 2 constellation and interference constellation in Categorical cross-entropy loss

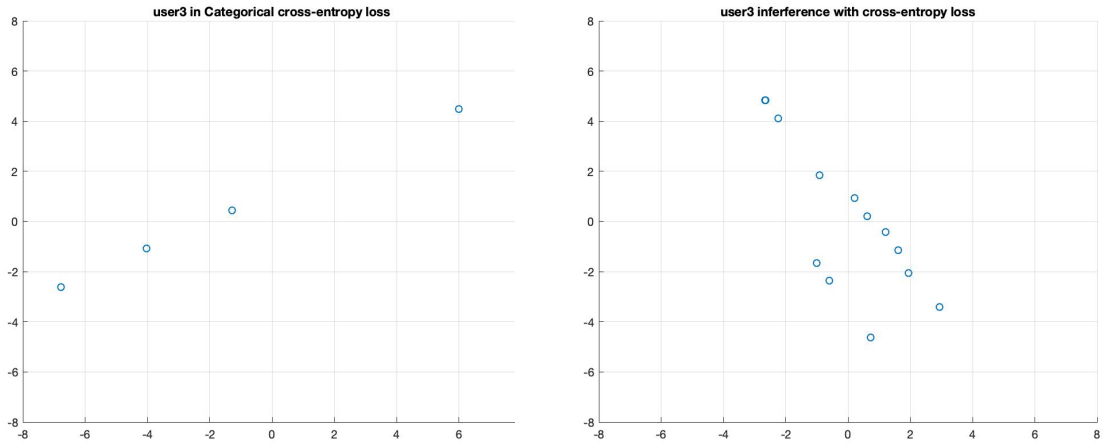

Figure 4.47: User 1 constellation and interference constellation in Categorical cross-entropy loss

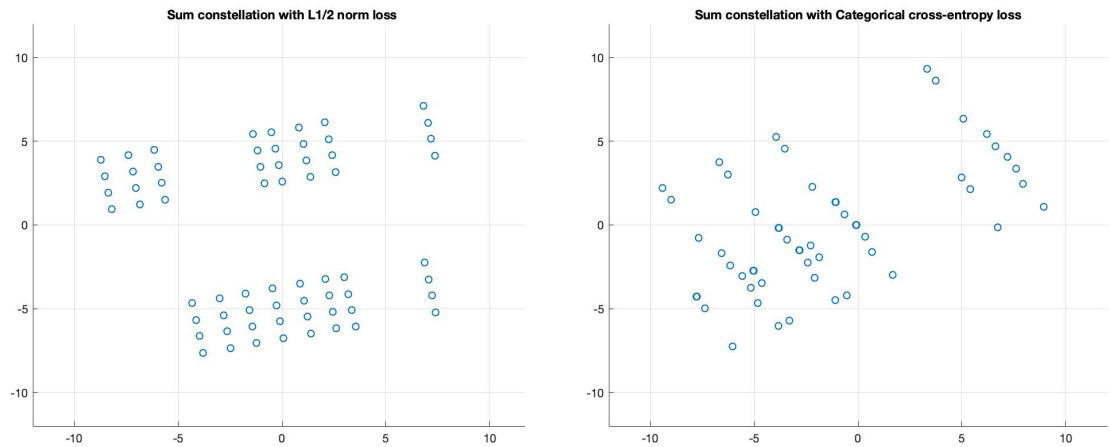

Figure 4.48: Sum constellations in $L_{1/2}$ norm loss and Categorical cross-entropy loss

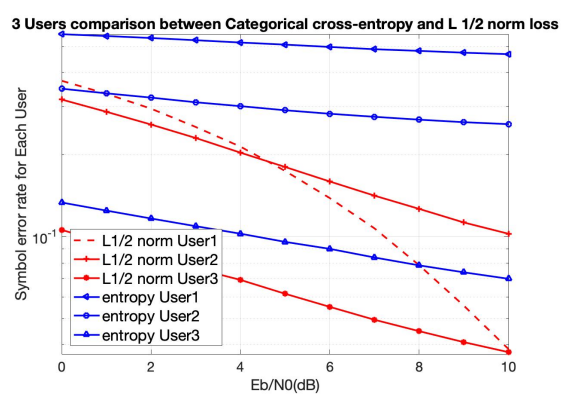

Figure 4.49: Symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in three users case

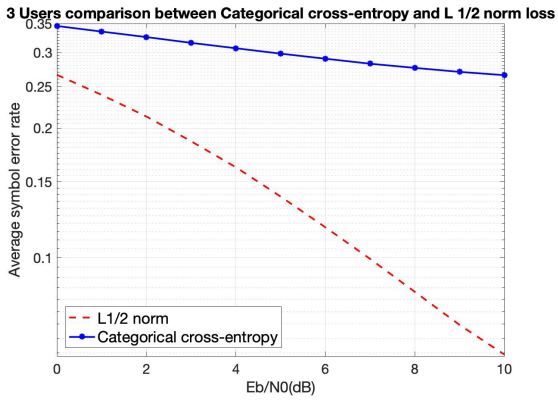

Figure 4.50: Average symbol error rate comparison between Categorical cross-entropy loss and $L_{1/2}$ regularization loss in three users case

Chapter 5

Conclusion

In this thesis, based on existing neural network technologies in autoencoder, we introduce two methods for improvement. In SISO systems, through the optimization of softmax function, we propose an optimzied softmax method. Compared with the baseline, our method reduce the computing time and complexity. Moreover, it has the better performance of symbol error rates than baseline. In MIMO systems, we introduce $L_{1/2}$ norm in the MSE loss function as the penalty term, and the new MSE loss function has better results than the Categorical cross-entropy loss function, especially in Three-Input and Three-output cases. We think both methods are valuable attempts for deep learning in autoencoders. Although, there are still some problems need to solve, we hope our idea will have some contributions for the future wireless communication network systems.

Bibliography

[1] R. C. Gonzalez, "Deep convolutional neural networks [lecture notes]," *IEEE Signal Processing Magazine*, vol. 35, pp. 79–87, 2018.

[2] Y. Lecun, *Generalization and network design strategies.* Elsevier, 1989.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *IEEE International Conference on Computer Vision (ICCV 2015)*, vol. 1502, 02 2015.

[4] T. OShea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, Dec 2017.

[5] T. Schenk, *RF imperfections in high-rate wireless systems: Impact and digital compensation*, 01 2008.

[6] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jul. 1989. [Online]. Available: http://dx.doi.org/10.1016/0893-6080(89)90020-8

[7] S. Reed and N. de Freitas, "Neural programmer-interpreters," 11 2015.

[8] H. T. Siegelmann and E. D. Sontag, "On the computational power of neural nets," in *COLT*, 1992.

[9] V. Vanhoucke, A. Senior, and M. Z Mao, "Improving the speed of neural networks on cpus," 01 2011.

[10] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 873–880. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553486

[11] T. O'Shea, T. Erpek, and T. Clancy, "Deep learning based mimo communications," 07 2017.

[12] R. Daniels and R. W. Heath, "Online adaptive modulation and coding with support vector machines," in *2010 European Wireless Conference (EW)*, April 2010, pp. 718–724.

[13] Y. Jeon, S. Hong, and N. Lee, "Blind detection for mimo systems with low-resolution adcs using supervised learning," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[14] F. Rosenblatt, *Two theorems of statistical separability in the perceptron (Project Para).*, ser. Report (Cornell Aeronautical Laboratory). Cornell Aeronautical Laboratory, Inc., 1958.

[15] J. Orbach, "Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms." *JAMA Psychiatry*, vol. 7, no. 3, pp. 218–219, 09 1962.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Learning Internal Representations by Error Propagation, pp. 318–362. [Online]. Available: http://dl.acm.org/citation.cfm?id=104279.104293

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, Dec. 1989. [Online]. Available: http://dx.doi.org/10.1162/neco.1989.1.4.541

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[20] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[21] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines vinod nair," vol. 27, 06 2010, pp. 807–814.

[22] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio, "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization," *NIPS*, vol. 27, 06 2014.

[23] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[24] B. Natarajan, "Sparse approximate solutions to linear systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995. [Online]. Available: https://doi.org/10.1137/S0097539792240406

[25] E. C, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," 2005.

[26] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theor.*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006. [Online]. Available: https://doi.org/10.1109/TIT.2006.871582

[27] D. Donoho, "Neighborly polytopes and sparse solutions of underdetermined linear equations," Tech. Rep., 2005.

[28] ——, "High-dimensional centrally symmetric polytopes with neighborliness proportional to dimension," *Discrete Computational Geometry*, vol. 35, pp. 617–652, 05 2006.

[29] H. Zou, "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1418–1429, 2006. [Online]. Available: https://doi.org/10.1198/016214506000000735

[30] N. Meinshausen and B. Yu, "Lasso-type recovery of sparse representations for high-dimensional data," *Ann. Statist.*, vol. 37, no. 1, pp. 246–270, 02 2009. [Online]. Available: https://doi.org/10.1214/07-AOS582

[31] R. Chartrand and V. Staneva, "Restricted isometry properties and nonconvex compressive sensing," *Inverse Problems*, vol. 24, no. 3, p. 035020, may 2008. [Online]. Available: https://doi.org/10.1088%2F0266-5611%2F24%2F3%2F035020

[32] Z. Xu, H. Zhang, Y. Wang, X. Chang, and Y. Liang, "L 1/2 regularization," *Science China Information Sciences*, vol. 53, 06 2010.

[33] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-laplacian priors," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1033–1041. [Online]. Available: http://papers.nips.cc/paper/3707-fast-image-deconvolution-using-hyper-laplacian-priors.pdf

[34] R. Chartrand, "Exact reconstruction of sparse signals via nonconvex minimization," *Signal Processing Letters, IEEE*, vol. 14, pp. 707 – 710, 11 2007.

[35] ——, "Fast algorithms for nonconvex compressive sensing: Mri reconstruction from very few data," 08 2009, pp. 262 – 265.

[36] Z. Xu, "Data modeling: Visual psychology approach and l1/2 regularization theory," *Proceedings of the International Congress of Mathematicians 2010, ICM 2010*, 01 2010.

[37] Z. xu, H.-L. GUO, Y. Wang, and H. Zhang, "Representative of l1/2 regularization among lq (0 ¡ q 1) regularizations: an experimental study based on phase diagram," *Acta Automatica Sinica*, vol. 38, p. 12251228, 07 2012.

[38] R. Tibshirani, "Regression shrinkage and selection via the lasso," *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, vol. 58, pp. 267–288, 1994.