CHALMERS
UNIVERSITY OF TECHNOLOGY

Data-driven Learning of Fiber-optic Communication Systems with Quantized Feedback

Master's thesis in Master Programme of Communication Engineering

Jinxiang Song

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:NN

Data-driven Learning of Fiber-optic Communication Systems with Quantized Feedback

Jinxiang Song

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Master Program of Commnunication Engineering
Chalmers University of Technology
Gothenburg, Sweden 2019

Data-driven Learning of Fiber-optic Communication Systems with Quantized Feedback
Jinxiang Song

Supervisor: Henk Wymeersch, Christian Häger, Bile Peng
Examiner: Henk Wymeersch

Master's Thesis 2019:NN
Department of Electrical Engineering
Master Program of Communication Engineering Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 704648175

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Data-driven Learning of Fiber-optic Communication Systems with Quantized Feedback

Jinxiang Song
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Conventional model based transmitter and receiver design has lead to the stable and widely applied communication systems we have today. However, optimal modulation formats and detection methods are hard to find for communication systems where the physical channels are nonlinear. Recently work shows that data-driven optimization of transmitters and receivers can reveal new modulation and detection schemes and enable physical-layer communication over unknown channels. In practice, this approach requires a feedback signal from the receiver to the transmitter.

In this thesis, the impact of quantized feedback in data-driven learning of communication over a fiber-optic channel is studied. We propose a novel quantization method that exploits the specific properties of the feedback signal and is suitable for non-stationary signal distributions. Our simulation results show that feedback quantization does not appreciably affect the learning process and can lead to excellent performance, even with 1-bit quantization. In addition, it is shown that learning is robust to noisy feedback where the quantization bits are randomly flipped.

Keywords: data-driven learning, policy optimization, unknown channel, noisy feedback, quantization

Acknowledgements

I would like to thank Prof. Henk Wymeersch for proposing this novel thesis topic and for giving me this precious opportunity to do this project.

I would like to express my gratitude to my supervisors Henk Wymeersch and Christian Häger and Bile Peng for their continuous guidance, support, patience and tolerance during my thesis work. Thank you all for the kindness and great work for the paper that we are going to publish.

In a word, I sincerely thank all the people who were involved in my thesis work. This work would been a unforgettable experience, and I do believe whatever I learnt during this thesis work will benefit my future career.

Jinxiang Song, Gothenburg, June 2019

Contents

List of Figures

List of Tables

1
Introduction

1.1 Communication system design

The fundamental problem of communication is to transmit a message at one point and either exactly or approximately reproduce it at another point [1] or, in other words, a communication system aims at reliably exchanging information between two nodes over a channel. Fig. 1.1 shows a classic representation of a communication system, where communication over a physical channel is conducted by the use of transmitter and receiver.

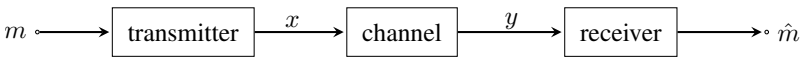


Figure 1.1: Communication system model

Here, transmitter maps messages m to complex symbols x and sent it through the channel defined by a conditional probability density function (PDF) $p(y|x)$. Receiver makes observation of channel output y and estimates the transmitted messages $\hat{m}$.

Conventionally, communication systems are designed by dividing transmitter and receiver into several processing blocks, each performing a individual task such a source encoding, channel encoding, or modulation. This approach enables independent analysis and optimization of each block, and has lead to the communication systems that are in use today [2].

1.2 Difficulties in fiber-optic communication system design

As communication system becomes more complex, this model-based approach becomes more and more difficult to analysis and optimize, especially in fiber-optic communication systems. The difficulty in designing transmitter and receiver for optical communication system mainly arises from the fact that Kerr effect [3] in the fiber-optic channel makes optimal transmission and detection method unknown. This is because that Kerr effect leads to a rotation of transmitted constellation symbols, and and symbols with higher energy rotate more. Fig.1.2 shows the constellation pattern observed when 16-QAM symbols are transmitted over a simplified fiber-optic channel where only Kerr effect and Gaussian noise exist.

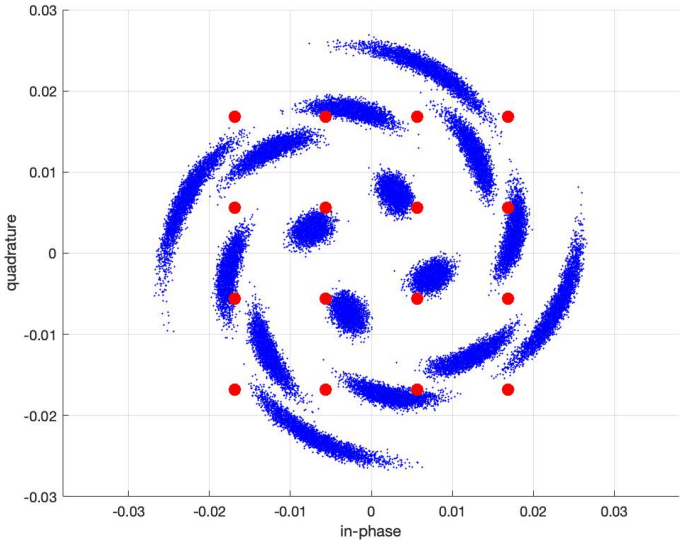


Figure 1.2: Detected 16-QAM symbols when transmitted over a simplified fiber optical channel with input power equals to -5dBm and noise power equals to -21.3dBm.

As one can see from the figure, symbols with higher energy are perturbed with stronger phase noise, which makes the design of optimal receiver an complex task. To see that, Fig. 1.3 visualizes the achieved symbol error rate (SER) when 16-QAM symbols are transmitted over a fiber-optic channel, and Euclidean distance detector are used for symbol detection. The channel noise is assumed to be Gaussian noise with fixed power.

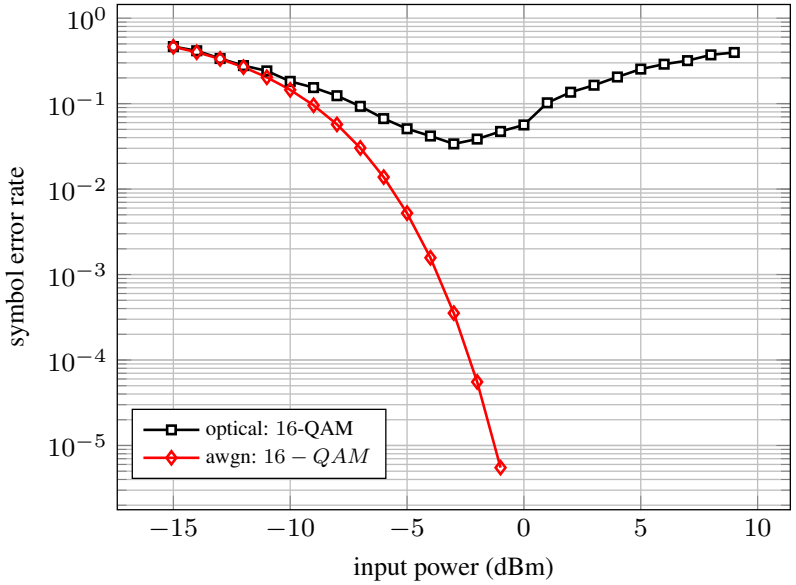


Figure 1.3: Symbol error rate achieved when 16-QAM symbols are transmitted over a simplified fiber-optical channel with Kerr effect and fixed noise power equals to -21.3dBm and an AWGN channel. Both cases use Euclidean distance detector

As one can see from the figure, the resulted SER first goes down and then goes up as

transmission power increases. This is because when transmission power is low, the main perturbation comes from the Gaussian noise. While when transmission power is high, the phase noise introduced by Kerr effect becomes more dominant than Gaussian noise. As a reference, the performance achieved from an AWGN channel, where there only exists Gaussian noise, is also shown. From the figure one can see that SER decreases as transmission power increases.

1.3 Data-driven learning of communication systems

The rapid growth of machine learning (ML) has lead to a new line of transmitter and receiver design. Recent work shows that transmitter and receiver can be jointly learned from data without introducing any block-wise structure like modulator, channel encoder and so on. This concept was first brought up in [4] and [5], in which a communication system was interpreted as an auto-encoder. Since then, numerous extensions of the original idea towards channel coding [6], joint source-channel coding [7], orthogonal frequency-division multiplexing (OFDM) [8], and multiple-input multiple-output (MIMO) [9, 10] have been made, which all demonstrate the versatility of this approach.

However, this joint optimization of transmitter and receiver is problematic in practice since it requires a known and differential channel model, which is often hardly the case. One simple approach to circumvent this limitation is to first learn transmitter and receiver based on an estimated channel model and then performs receiver fine-tuning in real channel [2]. However, with this approach, the transmitter cannot be fine-tuned, resulting in sub-optimal performance. Another approach is to first learn a generative differentiable channel model and optimize the transmitter and receiver by using the surrogate channel [8, 11]. However, the learnt channel model probably does not cover all hardware insufficiency and channel characteristics, thus the performance is highly limited by the model accuracy. A different approach is to regard transmitter learning as an reinforcement learning task, where transmitter maps a fixed message to a random variable following a certain distribution during the transmitter training process [12–14]. With this approach, the surrogate gradient can be computed without channel model, and leads to the sufficient training of transmitter.

In order to compute the surrogate gradient for training transmitter, a feedback signal must be communicated from the receiver to the transmitter over a feedback channel. And this feedback signal can be either perfect [12–15] or noisy [16]. In the latter case, it is instructive to regard the feedback transmission as a separate communication problem for which optimized transmitter and receiver pairs can again be learned. This has been done in [16], where a feedback link is learnt such that real numbers can be communicated over an additive white Gaussian noise (AWGN) channel. In practice, however, signals will be quantized to a finite number of bits, including the feedback signal. And it's not clear what influence of quantization will have on the

system training process.

1.4 Goal of the thesis

Reinforcement learning based transmitter optimization enables end-to-end learning of communication systems without knowing a differentiable channel model. Previous work shows that this approach requires a feedback signal from the receiver to the transmitter. In this thesis, we study the impact of quantized feedback signal in learning communication system over a fiber-optic channel. To estimate the performance of our proposed training approach, we provide numerical results where the system performance is measured in terms of symbol error rate (SER).

1.5 Machine learning Ethics

Machine learning has become a hot research topic and it brings many advantages to human society. However, there are still many ethical concerns about machine learning that should not be ignored by researchers.

Firstly, machine learning algorithm can be biased or discriminating [17]. On one hand, machine learning algorithm are designed by developers, which means that it's possible that the algorithm itself is designed to be discriminative against some population. on the other hand, the machine learning algorithms are usually learned from data. If the dataset is discriminative or incomplete, the trained algorithms are possible to be biased and discriminating.

Secondly, machine learning can leads privacy concern. Training of machine learning algorithm requires huge amount of data, and user data might be collected without being notification. For example, a recommending system might predict user's likeness or dis-likeness according to user's previous usage.

Thirdly, machine learning can lead to safety concerns. For example, autonomous driving car has become a hot research topic in recent years. However, we still don't know the trustness of autonomous driving system.

Last but not least, machine learning helps the growing of industrial robots design, and there has been concern about high unemployment rate caused by large-scale usage of robots.

In this thesis, machine learning is used for designing of fiber-optic communication system, the issues mentioned above won't be concerns of this work.

2
Theory

In this chapter, we first give a brief introduction to fiber-optical communication, then this is followed by a brief description of a simplified optical channel model that we will use in this thesis work. After that,the idea of quantization is introduced. In the last part of this chapter, we give an overview of concepts related to deep learning and some of the commonly used algorithms used in deep learning.

2.1 Fiber-Optic communication

Fiber-optic communication is a method of transmitting information from one point to another by sending pulses of light through an optical fiber. The light forms an electromagnetic carrier wave that is modulated to carry information [18]. The propagation of signals in a optical fiber with ideal distributed amplification is modeled by the nonlinear Schrödinger equation [19]:

$$\frac{\partial \mathbf{x}(z,t)}{\partial z} = i\gamma||\mathbf{x}(z,t)||^2 x(z,t) - i\frac{\beta_2}{2}\frac{\partial^2 \mathbf{x}(z,t)}{\partial t^2} - \mathbf{n}(z,t) \tag{2.1}$$

where $\mathbf{x}(z,t) = [\mathbf{x}_x\ \mathbf{x}_y]$ is the trasmitted signal, z and t are distance and time coordinates, γ is the nonlinearity parameter, β_2 is the group velocity dispersion coefficient, and $\mathbf{n}(z,t)$ is the Gaussian noise. The first term on the right side of the equation represents the Kerr effect, which causes a rotation of transmitted signal and the rotation is proportional to the signal's power. The second term on the right side of the equation represents dispersion, which usually leads to the inter symbol interference in fiber-optic communication systems. In this work, a simplified fiber-optic channel model is considered, in which we the dispersion is ignored. The model is described by the following recursion [20]:

$$x_{k+1} = x_k e^{jL\gamma|x_k|^2/K} + n_{k+1}, \quad 0 \leq k < K \tag{2.2}$$

where $x_0 = x$ is the complex-valued channel input, $y = x_K$ is the channel output, $n_{k+1} \in \mathcal{CN}(0, P_N/K)$ is the white Gaussian noise, L is the total link length, P_N is the noise power, and γ is the nonlinearity parameter. The model assumes ideal distributed amplification and $K \to \infty$.

2.2 Quantization

Quantization is the process of converting a continuous-valued or large set of values into a finite set of discreet values [21]. More precisely, a quantizer is a function

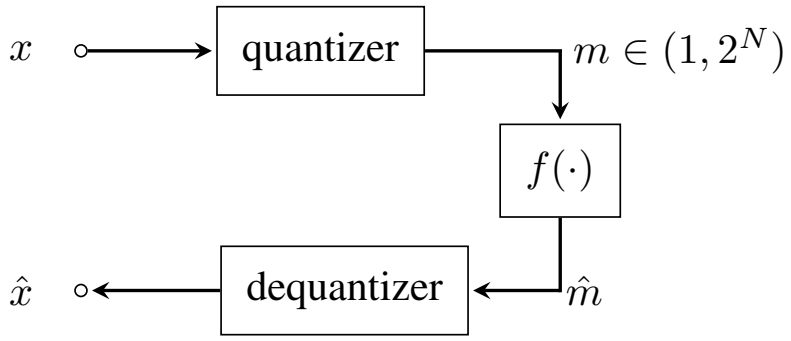


Figure 2.1: Quantization

$f_Q(\cdot)$ that maps signals x with continuous range to a discrete set of representation levels $\mathbf{r} \triangleq \{r_1, r_2, \cdots, r_M\}$ by applying signals x to a set a decision levels $\mathbf{d} \triangleq \{d_1, d_2, \cdots, d_M\}$ such that

$$f_Q(x) \to r_i,\ if\ x \in (d_{i-1}, d_i],\ i = 1, 2, 3, \cdots, M \tag{2.3}$$

where M is the total number of representation levels. Usually, M is set to $M = 2^N$ so that each input value can be represented by N bits. Fig. 2.1 illustrates the process of quantization. As it can be seen, quantization is decomposed into two distinct stages: classification stage denoted by $Q(\cdot)$ and reconstruction stage denoted by $Q^{-1}(\cdot)$. In classification stage, each input value x is mapped to an integer m such that $d_{m-1} < x \leq d_m$; In the reconstruction stage, the quantized value $\hat{x}$ is restored from m following $\hat{x} = r_m$. Note that the successful reconstruction of quantized value relies on the assumption that $Q(\cdot)$ and $Q^{-1}(\cdot)$ shares common knowledge about the decision levels $\mathbf{d}$ and representation levels $\mathbf{r}$.

Due to the fact that all signals that lie in $(d_{i-1}, i]$ are represented by a fixed value r_i, there will always be a difference between the original signal x and the quantized signal $\hat{x}$. Usually, the difference between x and $\hat{x}$ is measured by the mean-square quantization error defined by

$$D = \mathbb{E}\{(x_i - \hat{x}_i)^2\} \tag{2.4}$$

Designing optimal quantizer can be regard as finding $\hat{x}$ that minimizes the mean square quantization error.

2.2.1 Uniform quantization

Depending on how the decision levels and representation levels are spaced, quantizer can be categorized into different types. In this work, we only consider uniform quantizer in which all decision levels and representation levels are equally spaced. Assuming all input values lies within (a, b), and each value is to be quantized with N bits, the uniform quantizer would divide (a, b) into 2^N equal-size sub-ranges with size $\Delta = (b - a)/2^N$, and take the decision levels and representation levels as:

$$d_i = \frac{i\Delta}{M}, \quad i = 0, 1, 2, \cdots, M \tag{2.5}$$

$$r_i = \frac{i\Delta}{M} + \frac{\Delta}{2M}, \quad i = 0, 1, 2, \cdots, M \tag{2.6}$$

2.3 Neural network basics

Artificial neural networks are mathematical models or computing models that simulates the neuronal structures and functions of animal brains [22].And like the neural network in an animal brain, a artificial neural network is made up of several connected layers of nodes known as neuron.

2.3.1 Neuron

Neurons are the basic elements that forms an artificial neural network. Fig. 2.2 shows the structure of one single neuron.

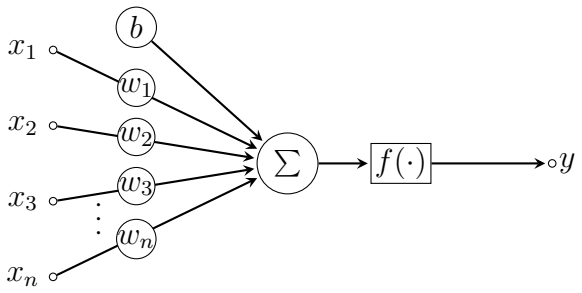


Figure 2.2: Neuron structure

Given an input vector $\mathbf{x}$, the neuron generates an output y according to

$$y = f(\mathbf{w}^T\mathbf{x} + b) \tag{2.7}$$

where $\mathbf{w}$ is vector of weights, b is a scalar called bias, and $f(\cdot)$ is an activation function. In a neural network, the input vector x are outputs of neurons in the previous layer.

2.3.1.1 Activation Function

An activation function of one neuron is a function that defines the output of that neuron [23]. Usually, using activation function aims at introducing non-linearity to the neuron , such that a neural network of more than one layer can be possibly approximate arbitrary complex functions. Consider a scenario where no activation function is used in the neural network, then each neuron in the neural network is simply a linear function and the neural network as a whole is simply a linear regression model, which has limited power in approximating a complex function. So, using activation function in a neural network enables non-linearity thus dramatically improves the system performance.

In practice, there are many different types of activation functions with different features. Table 2.1 lists some of the most common activation functions.

Table 2.1: List of some commonly used activation functions

name	$f(x) =$	range
Linear	x	$(-\infty, \infty)$
Sigmoid	$\frac{1}{1+e^{-x}}$	$(0, 1)$
tanH	$\frac{1-e^{-2x}}{1+e^{-2x}}$	$(-1, 1)$
ReLU	$max(0, x)$	$(0, \infty)$
Softmax	$\frac{e^{x_i}}{\sum_i e^{x_i}}$	$(0, 1]$

Different activation functions have different features, and these different features determines the different usage of these activation functions. Generally speaking, the desirable properties of an activation function include:

- Non-linearity: As it's mentioned before, non-linearity makes a neural network of more than one layer a universal function approximator [24]. In another world, a multiple layers neural network without non-linearity is equivalent to a single layer network, which is a simple linear regression model.
- Finite range: An activation function with finite range makes the gradient based training method more stable.
- Continuously differentiable: When the activation is not differentiable at some point, the gradient based training method could possibly get stuck at that point.
- Computationally friendly: Activation functions with lower computation complexity are more favorable then those requires high computation power.

Fig. 2.3 shows some of the commonly used activation functions. As one can see, each of these functions has different features. For instance, identity function is linear, and it's only used in a neural networks output layer. Sigmoid function saturates when the input is large, which in practice makes gradient descent based learning converge slowly. This is because when input is large, the gradient of a Sigmoid function is close to zero, which leads to a small update to the trainable weights. TanH function is similar to Sigmoid function in terms of saturating when the input is large. However, in many applications TanH function outperforms Sigmoid function due to the fact that the output of TanH function is zero-centered, while Sigmoid function always has positive output. Besides, both TanH function and Sigmoid function are computational complex due to the exponential computation. In practice, Relu function was used instead of TanH and Sigmoid in many cases since ReLu function is more computationally friendly. However, Relu can leads problems such as "Relu-dying" because Relu function has zero-graident when the input is smaller than zero.

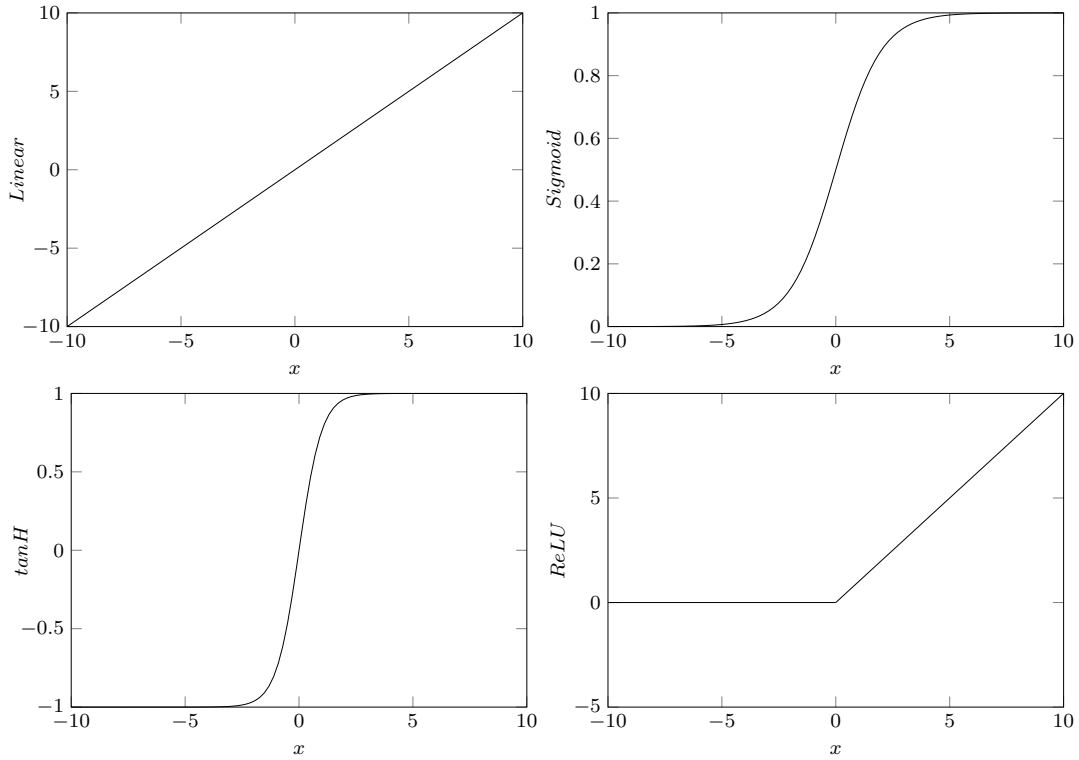


Figure 2.3: Activation functions

2.3.1.2 Bias

Bias is another parameter that helps to improve the performance of an neuron. Fig. 2.4 shows the impact of bias on a single neuron with one feature input. As one can see from the figure, when no bias is added to the neuron model, it will be trained over point passing through origin only, which is not in accordance with the real world scenarios. On the other hand, the neuron can be trained to represent any linear function in the presence of bias.

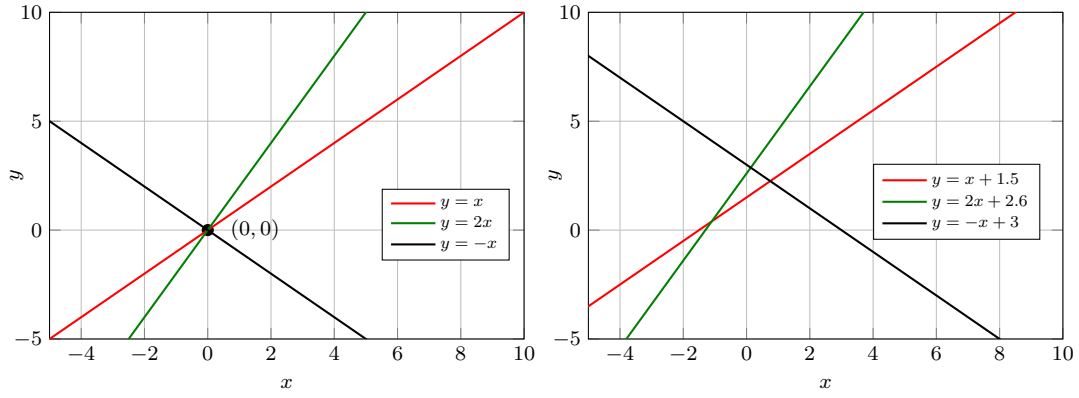


Figure 2.4: impact of bias on neuron with one input

2.3.2 Artificial Neural Network

Artificial neural networks (ANN) are computing systems vaguely inspired by the biological neural networks that constitute animal brains [25]. And like the neural network in an animal brain, a artificial neural network consists of several layers. The first layer is referred to as input layer and the last layer as output layer, the layers between input layer and output layer are called hidden layers. The structure of neural networks can vary a lot from each other and NNs in practice are divided into different categories. In this work, the simplest kind of neural network, fully-connected neural network (FNN), is considered. Fig. 2.5 shows a simple FNN with an input layer, an output layer and 3 hidden layers.

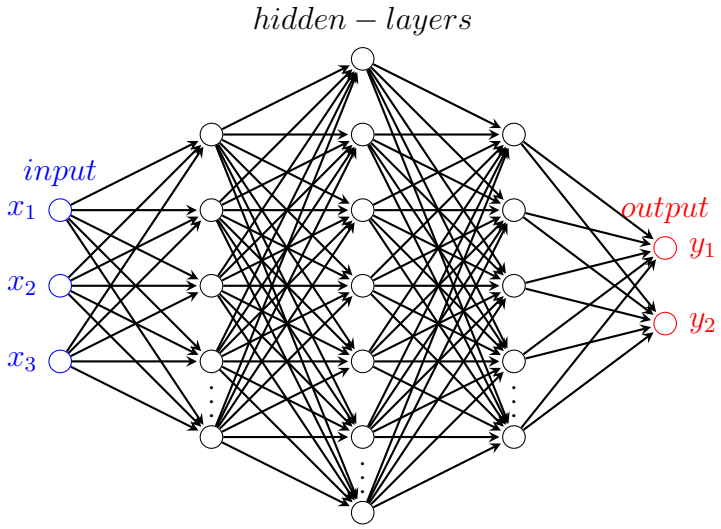


Figure 2.5: struture of an artificial neural network

As one can see from the figure, neurons in one layer of a FNN are connected to all neurons in the adjacent layers, and the computation within one layer can be described by:

$$\mathbf{y}^{[l]} = f^{[l]}(\mathbf{W}^{[l]}\mathbf{y}^{[l-1]} + \mathbf{b}^{[l]}) \tag{2.8}$$

where the superscript $[l]$ denotes the parameters associate to the l^{th} layer and $\mathbf{y}$ is a vector of outputs, $\mathbf{W}$ is a matrix of trainable weights, $\mathbf{b}$ is a vector of bias, and $f(\cdot)$ is the activation function. To the fact that data in FNN is passed through in one direction, FNN is relatively simple to maintain and it's one of the mostly commonly used neural network in practice.

2.4 Learning of neural networks

Neuron Network has been applied to solve problems in different fields, and based on those different applications, machine learning tasks can be broadly categorized

into 3 classes, which are supervised learning, unsupervised learning and reinforcement learning. In this section, we will introduce the concept of supervised learning, reinforcement learning and some commonly used algorithms related to them.

2.4.1 Objective Function

In order to train a neural network, we need a metric that tells us how good or bad that a neural network performs in modeling the given data. In machine learning context, this performance metric is referred to as objective function. And depending on what machine learning tasks that an objective function is applied to, it's usually divied into two categories: loss function (see Sec. 2.4.3.1) and reward function(see Sec. 2.4.4.1). Typically, all machine learning algorithm aims at minimizing the loss function or maximizing the reward function.

2.4.2 Gradient Based Learning

Training of a neural network is equivalent to find the optimal parameters so that the objective function is optimized. In this section, we give an overview of gradient descent, which is the most popular algorithm for optimizing objective functions.

2.4.2.1 Gradient Descent

Gradient descent [26] is one of the most common ways to minimize loss function [1]. The general idea is that to finding the minimum value of a function can be achieved by iteratively taking steps proportional to the negative gradient of the function at current point. Given a multi-variable function $J(\boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ that we want to minimize, gradient descent algorithm solves the problem by recursively updating $\boldsymbol{\theta}$ until convergence. In each update, the parameters $\boldsymbol{\theta}$ follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) \tag{2.9}$$

where α is the learning step size referred to as learning rate and the subscript t denotes the time step of optimization.

[1] Maximizing the reward function can be equally achieved by minimizing the negative reward function

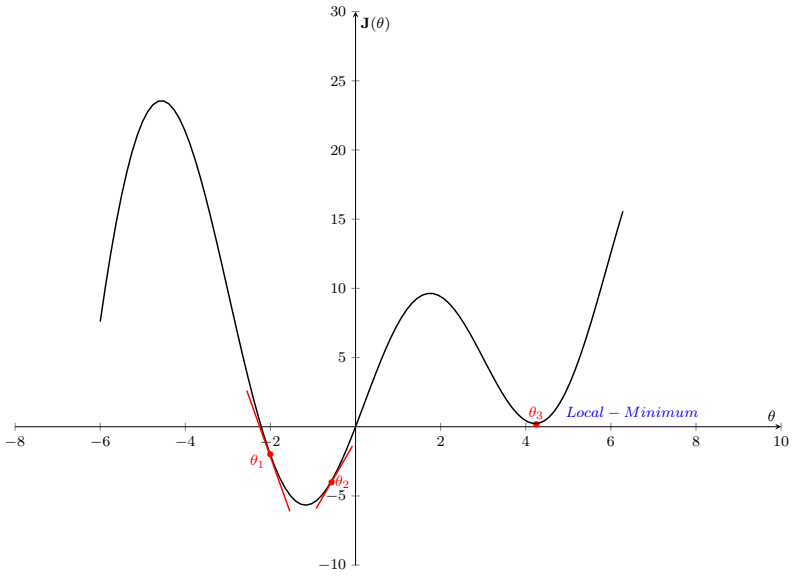


Figure 2.6: gradient descent example

Fig.2.6 shows a simple example how gradient descent works. For instance, when $\boldsymbol{\theta} = \boldsymbol{\theta}_1$, $J(\boldsymbol{\theta})$ has an negative gradient and the updating rule leads to an increased $\boldsymbol{\theta}$ which results in an reduction of objective function. When $\boldsymbol{\theta} = \boldsymbol{\theta}_2$, the gradient of $J(\boldsymbol{\theta})$ is positive, thus leading to an decrease of $\boldsymbol{\theta}$ which reduces $J(\boldsymbol{\theta})$ as well.

2.4.2.2 Gradient descent variations

Depending on how the training data set is used for updating parameters θ, there exists several variants of gradient descent. This section gives an brief description of these gradient descent variants.

Batch Gradient Descent
In BGD, the entire training data set is used for calculating the gradient of $J(\boldsymbol{\theta})$ in each update of $\boldsymbol{\theta}$, which is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \mathbb{E}\{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t; (x^{(i)}, y^{(i)}))\} \tag{2.10}$$

By averaging the gradient over the whole data set, it leads to a stable update of $\boldsymbol{\theta}$, and only a few updates can lead to loss convergence. However, when the training data is very large, computing loss gradient over the whole data set is very inefficient and it takes a long time for training. In the mean time, BGD has the drawback that it's very likely to converge to a local minimum for a non-convex function.

Stochastic Gradient Descent
Different from BGD, in each update of trainable parameters θ, SGD computes the loss gradient by using only one sample from the data set [27], that is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t; (x^{(i)}, y^{(i)}))\} \tag{2.11}$$

The good side of SGD is that loss gradient is computed in a very fast way, and computing loss gradient with one sample introduced randomness into the parameters update, which can actually help SGD jump out of local minimum. However, on the other hand, the randomness introduced in loss gradient makes training unstable, and SGD usually requires much more iteration for convergence.

Mini – batch Gradient Descent
In MBGD, parameters are updated by computing loss gradient over a mini-batch of the whole training data set, that is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{1}{N} \sum_{i=1}^{N} \{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t; (x^{(i)}, y^{(i)}))\} \tag{2.12}$$

where N is the mini-bacth size. By computing loss gradient in this way, MBGD shares the characteristics of both BGD and SGD. In each parameters update, loss gradient is computed in a fast way, and the average over a min-batch stabilizes the training process.

2.4.2.3 Optimization methods

The fact that learning steps fluctuate in SGD and MBGD makes the learning slow. In practices, there exists several extensions of SGD that speed up training of neural network. This section introduces some of the extensions of SGD.

Momentum
Different from SGD where only the gradient of the current step is used for parameters update, Momentum updates trainable parameters by taking the gradient of past steps into account [28]. The equations of gradient descent are revised as follows:

$$\boldsymbol{v}_t = \beta \boldsymbol{v}_{t-1} + (1 - \beta) \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) \tag{2.13}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \boldsymbol{v}_t \tag{2.14}$$

When $\beta = 0$, the method is equivalent to gradient descent, meaning that gradients of past steps are not taking into account. In practice, the β is usually initialized at 0.5, and gradually adjusted to 0.9 over multiple epochs.

RMSProp
Root mean square propagation (RMSProp) is another extension of SGD. Aiming at dampen the oscillations in the training process, RMSProp takes the second moment of gradient into account when updating parameters [29]. The mathematical expression is:

$$\boldsymbol{v}_t = \beta \boldsymbol{v}_{t-1} + (1 - \beta)(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t))^2 \tag{2.15}$$

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \frac{\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t)}{\sqrt{\boldsymbol{v}_t + \epsilon}} \tag{2.16}$$

where β is a hyper-parameter generally chosen to be 0.9, α is the initial learning rate, and ϵ is a hyper-parameter that avoids the gradient being divided by zero.

Adam optimization
Adam or Adaptive Moment Optimization algorithms combines the heuristics of both Momentum and RMSProp [30], and the update rule follows:

$$\boldsymbol{v}_t = \beta_1 \boldsymbol{v}_{t-1} + (1 - \beta_1)\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) \tag{2.17}$$

$$\boldsymbol{s}_t = \beta_2 \boldsymbol{s}_{t-1} + (1 - \beta_2)(\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t))^2 \tag{2.18}$$

$$\boldsymbol{v}_t^{corrected} = \frac{\boldsymbol{v}_t}{1 - \beta_1} \tag{2.19}$$

$$\boldsymbol{s}_t^{corrected} = \frac{\boldsymbol{s}_t}{1 - \beta_1} \tag{2.20}$$

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha \frac{\boldsymbol{v}_t^{corrected}}{\sqrt{\boldsymbol{s}_t^{corrected} + \epsilon}} \tag{2.21}$$

where β_1 and β_2 are a set of hyper-parameters, α is the initial learning rate. By taking the gradients of past steps as well as the second moment of gradients, Adam optimizer has the following properties:

- The actual step size taken by the Adam in each iteration is approximately bounded by the step size hyper-parameter.
- Step size of Adam update rule is invariant to the magnitude of the gradient, which helps the training a lot when going through areas with tiny gradients such as saddle points or ravines.

2.4.3 Supervised Learning

As the name suggests, supervised learning is the machine learning task that aims at learning a mapping from an input to an output based on training data set which we have prior knowledge about what the output y should be given an input x. To be precise, given training data pairs $\{x_i, y_i\}$ $(i = 1, 2, 3, \cdots, N)$, the goal of supervised learning is to find a function $\hat{y}_i = f_{\boldsymbol{\theta}}(x_i)$ that minimizes the difference between label y_i and prediction $\hat{y}_i$ for each input data x_i.

2.4.3.1 Loss Function

The differences between predictions and previously known labels in supervised learning are measured by an object function, or more precisely loss function or cost function. In practice, there are more than one type of loss function used in supervised learning depending on whether it's a regression problem or a classification problem. In this section, we introduce some of the most commonly used loss function.

For regression problem, the most commonly used loss function is known as Mean Squared Error (MSE) Loss:

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} |f_{\boldsymbol{\theta}}(x_i) - y_i|^2 \tag{2.22}$$

that computes the mean of squared differences between predictions and previous known labels. Another common loss function is Mean Absolute Error(MAE) loss:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N}|f_{\boldsymbol{\theta}}(x_i) - y_i| \tag{2.23}$$

that computes the mean of absolute differences between predictions and previous known labels. Generally, MSE loss performs better than MAE loss, since the square operation in MSE loss makes it more sensitive to samples with larger difference, thus leading to the model to be adjusted to fit that data. While in scenarios where outliers exist in the data set, MAE loss sometimes outperforms MSE loss. This is can be explained as follow: given a outlier in the dataset, The square operation in MSE loss makes it more sensative to that outlier than MAE loss does, thus model using MSE loss is more likely to fit itself to that out lying example at the expense of many other common samples. Another type of Loss that compromises MAE loss and MSE loss is Huber loss [31]. Defined by:

$$J_\delta(\boldsymbol{\theta}) = \begin{cases} \frac{1}{2N}\sum_{i=1}^{N}|f_{\boldsymbol{\theta}}(x_i) - y_i|^2 & \text{, if } |f_{\boldsymbol{\theta}}(x_i) - y_i| < \delta \\ \frac{1}{N}\sum_{i=1}^{N}\delta|f_{\boldsymbol{\theta}}(x_i) - y_i| - \frac{1}{2}\delta & \text{, otherwise} \end{cases}$$

Huber loss is much less sensative to outlier than MSE loss does since square operation only happens in a small interval.

For classification problem, since the classification model outputs a vector of probabilities, the above mentioned loss functions can no longer work and the most popular loss function used is cross entropy loss:

$$J(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} y_i \log(f_{\boldsymbol{\theta}}(x_i)) \tag{2.24}$$

Minimizing the cross entropy loss is equivalent to finding a mapping function that outputs a probability of 1 for the target label.

2.4.4 Reinforcement learning

Reinforcement learning is another type of machine learning in which the problem is modeled as an agent learn how to behave in a environment by performing actions and seeing the results [32]. The idea behind reinforcement learning is that an agent will learn from the environment by interacting with it and receiving rewards for performing actions. The figure depicts the concept of reinforcement learning.

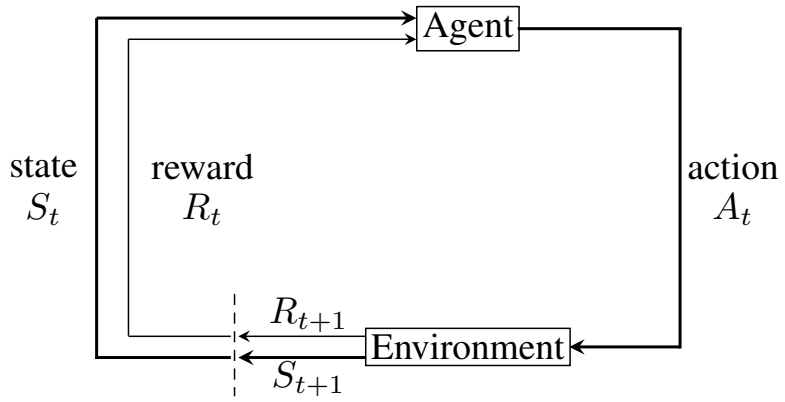


Figure 2.7: Reinforcement Learning Model

As it's shown in Fig.2.7, the agent moves in the environment and makes observation. Based on its observation of current state S_t, the agent takes an action A_t according to a tragedy called policy and transfers to a new state S'. In this transaction process, the environment returns a feedback signal R_t called reward, according to which the agent estimate the bad or good of that taken action. Reinforcement learning aims at finding a policy such that the agent can achieve maximal accumulative reward .

2.4.4.1 Reward Function

Different from supervised learning, the objective function in reinforcement learning is called reward function. The following is a commonly used objective function for reinforcement learning:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left\{ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right\} \tag{2.25}$$

where t stands for the time step, $\gamma \in (0, 1]$ is the discount factor[2], s_t is the current state of step t, a_t is the action taken according to the policy $\pi_{\boldsymbol{\theta}}$, and $r(s_t, a_t)$ is the reward received when by taking action a_t.

The reward function can be rewritten as :

$$J(\boldsymbol{\theta}) = \sum_{\tau} \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \tag{2.26}$$

where $\tau = (s_t, a_t; s_{t+1}, a_{t+1}; s_{t+2}, a_{t+2}; \cdots s_H, a_H)$ is a trajectory of actions that the agent follows under policy $\pi_{\boldsymbol{\theta}}$, $\pi_{\boldsymbol{\theta}}(\tau) = \prod_{i=t}^{H} p(s_i, a_i, s_{i+1})$ is the probability distribution of trajectory τ, $p(s_t, a_t, s_{t+1})$ is the probability of transferring from state s_t to s_{t+1} by taking action a_t and $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(a_t | s_t, \pi_{\boldsymbol{\theta}})$ is the corresponding accumulative reward of trajectory τ.

[2] Discount factor is a measure of how far ahead in time the algorithm looks. To prioritise rewards in the distant future, keep the value closer to one. A discount factor closer to zero on the other hand indicates that only rewards in the immediate future are being considered, implying a shallow look ahead. In scenarios where discount factor equals 1, rewards in the long future will be considered and the rewards goes to infinity and never converge.

The optimization problem in reinforcement learning is to find a policy $\pi_{\boldsymbol{\theta}}$ that creates a trajectory:

$$(s_t, a_t; s_{t+1}, a_{t+1}; s_{t+2}, a_{t+2}; \cdots s_H, a_H) \tag{2.27}$$

that maximizes the reward function:

$$\max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \sum_{\tau} \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \tag{2.28}$$

2.4.4.2 Policy Gradient

The optimization problem of maximizing reward function $J(\boldsymbol{\theta})$ can be achieved by doing gradient ascend[3] which recursively updates parameters θ in the direction of gradient of $J(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) \tag{2.29}$$

where the gradient of $J(\boldsymbol{\theta})$:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathbf{J}(\theta) &= \nabla_{\boldsymbol{\theta}} \sum_{\tau} \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \end{aligned} \tag{2.30}$$

is hard to compute since $\pi_{\boldsymbol{\theta}}(\tau) = \prod_{i=t}^{H} p(s_i, a_i, s_{i+1})$, where $p(s_t, a_t, s_{t+1})$ is a function of $\boldsymbol{\theta}$. Policy gradient theorem simplifies the computation of $\nabla_{\boldsymbol{\theta}} J\boldsymbol{\theta}$ by applying the likelihood ratio trick [33] to Eq. 2.30:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \nabla_{\boldsymbol{\theta}} \sum_{\tau} \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \\ &= \sum_{\tau} \nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \\ &= \sum_{\tau} \pi_{\boldsymbol{\theta}}(\tau) \frac{\nabla_{\boldsymbol{\theta}} \pi_{\boldsymbol{\theta}}(\tau)}{\pi_{\boldsymbol{\theta}}(\tau)} R(\tau) \\ &= \sum_{\tau} \pi_{\boldsymbol{\theta}}(\tau) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) R(\tau) d\tau \\ &= \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left\{ \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) R(\tau) \right\} \end{aligned} \tag{2.31}$$

Since $\pi_{\boldsymbol{\theta}}(\tau)$ is defined as:

$$\begin{aligned} \pi_{\boldsymbol{\theta}}(\tau) &= \pi_{\boldsymbol{\theta}}(s_t, a_t; s_{t+1}, a_{t+1}; \ldots; s_T, a_T) \\ &= p(s_t) \prod_{i=t}^{T-1} \pi_{\boldsymbol{\theta}}(a_t | s_t) p(s_{t+1} | s_t, a_t) \end{aligned} \tag{2.32}$$

[3]Gradient descent is the inverse of gradient descent

the gradient of $\log \pi_{\boldsymbol{\theta}}(\tau)$ is:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) &= \nabla_{\boldsymbol{\theta}} \{\log p(s_t) \prod_{i=t}^{T} \pi_{\boldsymbol{\theta}}(a_t|s_t) p(s_{t+1}|s_t, a_t)\} \\
&= \nabla_{\boldsymbol{\theta}} \{\log p(s_t) + \sum_{i=t}^{T} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) + \sum_{i=t}^{T} \log p(s_{t+1}|s_t, a_t)\} \\
&= \nabla_{\boldsymbol{\theta}} \log p(s_t) + \nabla_{\boldsymbol{\theta}} \sum_{i=t}^{T} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) + \nabla_{\boldsymbol{\theta}} \sum_{i=t}^{T} \log p(s_{t+1}|s_t, a_t) \\
&= \sum_{i=t}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t)
\end{aligned} \tag{2.33}$$

where the first and last term in the third line can be removed because they are independent of $\boldsymbol{\theta}$. Applying Eq.2.33 to Eq.2.31, the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ then becomes:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \{\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\tau) R(\tau)\} \\
&= \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left\{ \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \right) R(\tau) \right\} \\
&= \mathbb{E}_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left\{ \left(\sum_{t=0}^{T} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(a_t|s_t) \right) \left(\sum_{t=0}^{T} \gamma^t r(s_t, a_t) \right) \right\}
\end{aligned} \tag{2.34}$$

3

Learning Physical-layer Communication with Quantized Feedback

3.1 Introduction

Previous work has shown that end-to-end learning of transmitter and receiver can outperform conventional model-based transmitter and receiver design. However, joint optimization of transmitter and receiver in practice is problematic since it requires a known differentiable channel model. More recent work shows that reinforcement learning enables sufficient learning of transmitter without knowledge about channel model. However, this approach requires a feedback signal from receiver to transmitter. In this research, we applied reinforcement learning to transmitter design for fiber-optic communication and we investigate the impact of quantized feedback signal on system performance of fiber-optic communication.

3.2 System Model

The proposed system model is illustrated in Fig.3.1. The goal is to transmit messages $m \in \{1, \cdots, M\} \triangleq [M]$ from transmitter to receiver over an priori unknown fiber-optic channel, M is the total number of messages. The communication system is implemented by representing the transmitter and receiver as two parameterized functions: $f_{\boldsymbol{\theta}_T} : [M] \to \mathbb{C}$ and $f_{\boldsymbol{\theta}_T} : \mathbb{C} \to [0,1]^M$, where $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_R$ are sets of transmitter and receiver parameters, known as trainable weights. The transmitter maps the k^{th} message m_k to a complex symbol $x_k = f_{\boldsymbol{\theta}_T}(m_k)$, and sends x_k to the fiber-optic channel. When receiver observes channel output y_k, it maps y_k to a probability distribution vector $\boldsymbol{p}_k$ over $[M]$, and then generates an estimation of transmitted message $\hat{m}_k$ according to $\hat{m}_k = \arg\max_m [\boldsymbol{q}_k]_m$.

To enable the learning of transmitter, we assume there exists a feedback link from receiver to transmitter. Besides, in the transmitter training phase, there also exists a random perturbation in the transmitter, which will be discussed in Sec.3.3.2.

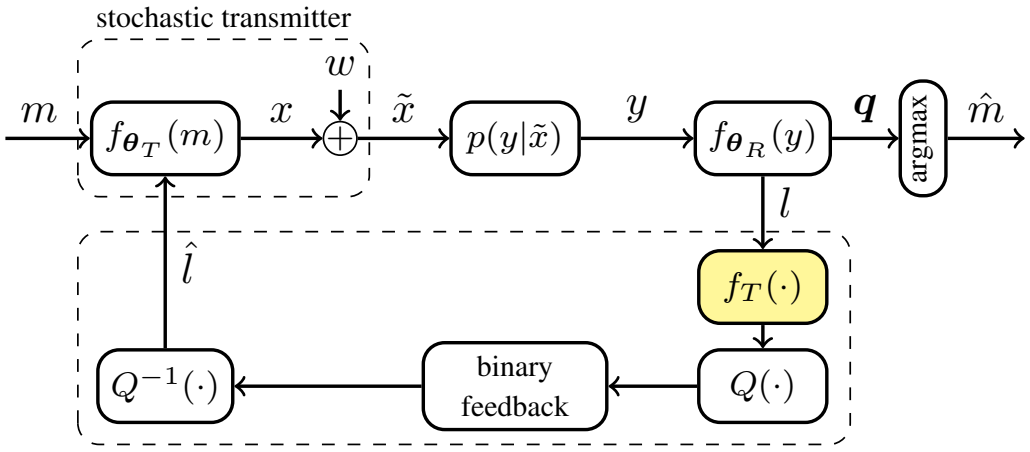


Figure 3.1: System Model

3.2.1 Transmitter structure

As mentioned in Sec. 3.2, the role that transmitter plays is to map a message m_k to a complex symbol x_k so that it's suitable for transmitting over the fiber-optic channel. The structure of transmitter is illustrated in Fig. 3.2(a), where there are an embedding layer, multi-hidden layers and a normalization layer. The embedding layer maps a message m_k to a M-dimension one-hot vector $\mathbf{u}$ in which the k^{th} element is one and all other elements equals to 0. Then, the multiple dense layers take the one-hot vector $\mathbf{u}$ as input and generates a two dimensional vector $[z_r, z_i]$ that forms the complex channel input z_k. Before z_k is transmitted over the channel, the last layer normalizes $z_k \to x_k$ such that the average power constrain $\mathbb{E}\{x\} \leq P$ is satisfied.

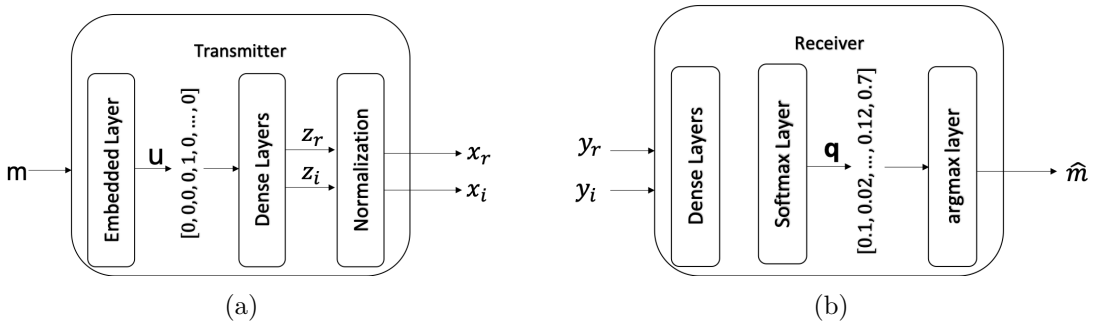


Figure 3.2: Architectures of transmitter and receiver

3.2.2 Receiver structure

The goal of receiver is to reconstruct the transmitted message m_k from the channel observation y_k, and it's structure is illustrated in Fig. 3.2(b). As is shown in the figure, receiver consists of multi-dense layers, a softmax layer and an argmax layer. The multi-dense layers process the channel observation y_k, and their output is passed to the softmax layer for generating a probability distribution vector $\boldsymbol{q}$. Here,

$\sum_{i=1}^{M} q_i = 1$ and q_i can be interpreted as the estimated posterior probabilities for each possible message. Then, the argmax layer generates an estimation of transmitted message $\hat{m}_k$ by choosing the one with highest probability.

3.2.3 Feedback link

To facilitate the learning of transmitter, we assume that there exists a binary feedback link from the receiver to the transmitter as shown in Fig.3.1. Here, $f_T(\cdot)$ denotes a pre-process an each feedback signal, $Q(\cdot)$ denotes a mapping from real number to bits sequence, and $Q^{-1}(\cdot)$ denotes the inverse process of $Q(\cdot)$ which is to map a bits sequence to a real number. Sec. 3.3.3 will discuss this feedback link in more detail.

3.3 Data-Driven Learning

Training of communication system is achieved by applying alternative training approached proposed in [34], where the transmitter and receiver are optimized alternatively. This approach is based on the assumption that the transmitter and receiver share the common knowledge about the data set used for training. Each training iteration consists of two phase, one for training receiver and one for training receiver. After a certain number of iterations, the overall system performance should improve.

3.3.1 Receiver Learning

Optimization of receiver is a supervised learning task since we assume receiver and transmitter shares common knowledge about what is transmitted over the channel. The training process is illustrated in Algorithm 1.

Algorithm 1 Receiver Learning

function ReceiverLearning(B_R, Seed)
  repeat
    ▷`Transmitter`
    $\boldsymbol{m}_R \leftarrow$`PseudoRandom(`$B_R$`, Seed)`
    $\mathbf{X} = f_{\boldsymbol{\theta}_T}(\boldsymbol{m}_R)$
    `Send(`$\boldsymbol{X}$`)`
    ▷`Receiver`
    $\boldsymbol{Y} \leftarrow$`Receive()`
    $\boldsymbol{q} = f_{\boldsymbol{\theta}_R}(\boldsymbol{Y})$
    $\boldsymbol{m}_R \leftarrow$`PseudoRandom(`$B_R$`, Seed)`
    $L(\boldsymbol{\theta}_R) \leftarrow$`ComputeCrossEntropy(`$\boldsymbol{m}_R$`,`$\boldsymbol{p}$`)`
    `Receiver parameters update`
  until `transmitter parameters updated for` N_R `times`
end function

The transmitter generates a mini-batch of B_R uniformly distributed training messages $\mathbf{m}_R$, then these messages are mapped to a vector of complex symbol $\mathbf{X}$ and

sent to the channel. The receiver makes observation of channel outputs $\boldsymbol{Y}$ and maps each of them to a probability vector $\mathbf{q}_k$ over M. Then the receiver generates a mini-batch of B_R training examples with the same seed and computes the cross entropy loss based on these probability vectors and training example. After that, the receiver parameters $\boldsymbol{\theta}_R$ are updated according to

$$\boldsymbol{\theta}_R^{j+1} = \boldsymbol{\theta}_R^j - \alpha_R \nabla_{\boldsymbol{\theta}_R} L(\boldsymbol{\theta}_R^j) \tag{3.1}$$

where α_R is the receiver learning rate, and $L(\boldsymbol{\theta}_R)$ is the cross entropy loss defined by

$$L(\boldsymbol{\theta}_R) = -\frac{1}{B_R} \sum_{i=1}^{B_R} log([f_{\boldsymbol{\theta}_R}(y_k)]_{m_k}) \tag{3.2}$$

where B_R is the mini-batch size for receiver training. In each training iteration, $\boldsymbol{\theta}_R$ is iteratively updated for N_R iterations.

3.3.2 Transmitter Learning

Transmitter learning is considered as a reinforcement task, where we regard the transmitter as an agent exploring in the physical world (the physical channel and receiver) and learning to take best actions (transmitter output) such that maximum rewards (minimum loss) can be achieved. The training process is illustrated in Algorithm 2.

Algorithm 2 Transmitter Learning

function TRANSMITTERLEARNING(B_T, Seed)
  repeat
    ▷`Transmitter`
    $\boldsymbol{m}_T$ ←`PseudoRandom(`B_T`, Seed)`
    $\mathbf{X} = f_{\boldsymbol{\theta}_T}(\boldsymbol{m}_T)$
    $\boldsymbol{X}_P = \boldsymbol{X} + \boldsymbol{W}$
    `Send(`$\boldsymbol{X}_P$`)`
    ▷`Receiver`
    $\boldsymbol{Y}$ ←`Receive()`
    $\boldsymbol{q} = f_{\boldsymbol{\theta}_T}(\boldsymbol{Y})$
    $\boldsymbol{m}_T$ ←`PseudoRandom(`B_T`, Seed)`
    $\boldsymbol{l}$ ←`ComputePerExampleLoss(`$\boldsymbol{m}_T$`,`$\boldsymbol{p}$`)`
    `Send(`$\boldsymbol{l}$`)`
    ▷`Transmitter`
    $\hat{\boldsymbol{l}}_k$ ←`ReceivePerExampleLoss()`
    `Transmitter parameters update`
  until `transmitter parameters updated for` N_T `times`
end function

The transmitter generates a mini-batch of B_T uniformly distributed training messages $\boldsymbol{m}_T$, and maps them to a vector of complex symbols $\boldsymbol{X}$. To enable transmitter exploration, a small Gaussian perturbation $w_k \sim \mathcal{CN}(0, \sigma_p^2)$ is added to each of the

transmitter output, such that the transmitter follows a Gaussian policy described by the PDF

$$\pi_{\boldsymbol{\theta}_R}(\tilde{x}_k|m_k) = \frac{1}{\sqrt{\pi\sigma_p^2}} \exp\left(-\frac{|\tilde{x}_k - f_{\boldsymbol{\theta}_T(m_k)}|^2}{\sigma_p^2}\right) \tag{3.3}$$

When receiver observes the perturbed messages $\boldsymbol{Y}$, it generates a mini-batch of B_T pseudo-random uniformly distributed messages with the same seed and computes the per-example losses $\boldsymbol{l}$ according to $l_k = -log([f_{\boldsymbol{\theta}_R}(y_k)]_{m_k})$ for $k \in \{1, 2, \ldots, M\}$ and sends these back to the transmitter via the feedback link. When transmitter receives the feedback per-example loss denoted by $\hat{\boldsymbol{l}}$, it updates its parameters $\boldsymbol{\theta}_T$ according to

$$\boldsymbol{\theta}_T^{j+1} = \boldsymbol{\theta}_T^j - \alpha_T \nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T^j) \tag{3.4}$$

where α_T is the transmitter learning rate and $L(\boldsymbol{\theta}_T)$ is the reward function defined by

$$L(\boldsymbol{\theta}_T) = \frac{1}{B_T} \sum_{i=1}^{B_T} \hat{l}_k log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k) \tag{3.5}$$

where B_T is the mini-batch size for training transmitter. In each training iteration, $\boldsymbol{\theta}_T$ is iteratively updated for N_T times.

3.3.3 Loss feedback

Learning of transmitter requires the knowledge of per-example loss. Hence, as mentioned in Sec. 3.2.3, we consider there exists a binary feedback channel from receiver to transmitter. In our set up, the loss feedback consists of two phases, which are loss transformation and loss quantization.

3.3.3.1 Loss transformation

The necessity of loss transformation comes from the fact that the distribution of per-example losses varies over time as illustrated in Fig. 3.3.

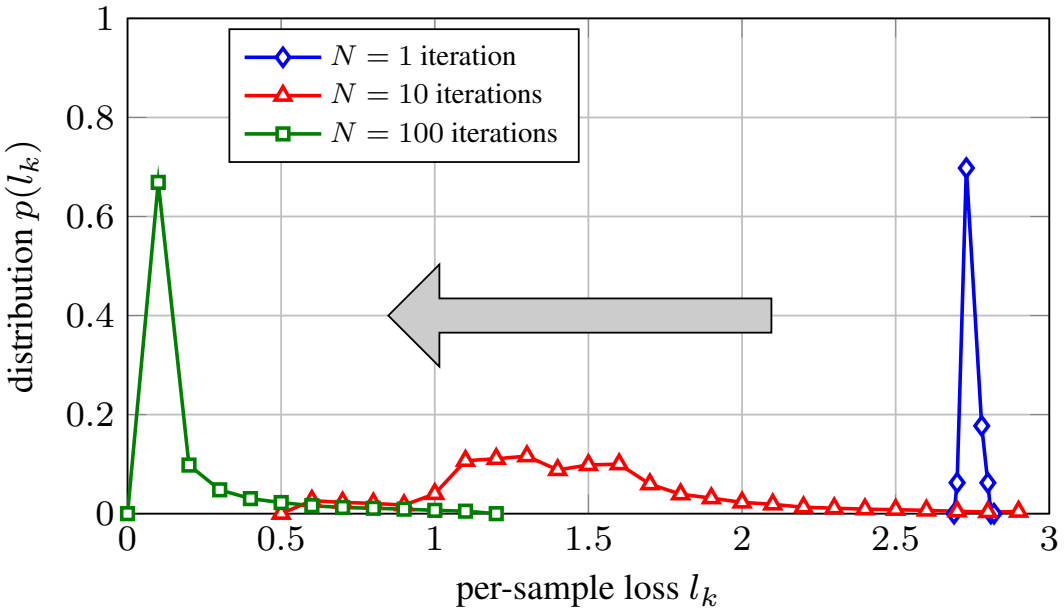


Figure 3.3: Illustration of the non-stationary loss distribution as a function of the number of training iterations in the alternating optimization.

Due to this non-stationary loss distribution, quantization becomes tricky since a large quantization range is required to cover large per-example losses in the initial training stage while large quantization range dramatically reduces the quantization resolution. To deal with this problem, we propose a pre-processing on each per-example loss such that after pre-processing all example losses lie within a fixed range. And this process is denoted by $f_T(\cdot)$, which consists of the following 3 steps:

1. Clipping: The losses are firstly clipped to lie within a range $(l_{\min}, l_{\max})$, where $l_{\min}$ is the minimum example loss in the current mini-batch B_T, and $l_{\max}$ is chosen such that 5 percent of largest losses in the current mini-batch are clipped. These operation excludes very large per-example losses which maybe be regarded outliers, thus stabilizing the training process. This operation is denoted by $f_{clip}(\cdot)$.
2. shifting: After the clipping operation, the per-example losses are then shifted by a factor of $l_{\min}$ such that all losses lie within range $(0, l_{\max} - l_{\min})$. This operation is referred to as baseline in reinforcement learning, which can help to reduce monte-carlo variance in some cases. We denote this operation as $f_{shift} \cdot ()$.
3. Scaling: Before the shifted per-example losses are quantized for transmission, all losses are scaled by a factor of $1/(l_{\max} - l_{\min})$ such that all per-sample losses lie within range $(0, 1)$. We denote this operation by $f_{sc}(\cdot)$.

In summary, the proposed loss transformation scheme consists of a loss cliping, a loss shifting and a loss scaling. We can write the entire loss transformation process as $f_T(\cdot) = f_{sc}(f_{shift}(f_clip)(\cdot))$

3.3.3.2 Loss quantization

After the pre-process denoted by $f_T(\cdot)$, all example losses lies within $[0, 1]$). To facilitate the successful transmission of each example loss over a binary feedback channel, we further perform quantization on each example loss. Denoted by $f_Q(\cdot)$, our quantization process applies the following steps:

1. Mapping from real number to N bits: This is done by uniform quantization. We divide the range into 2^N equal-size sub-ranges, and each sub-range has size $\Delta = \frac{1}{2^N}$. Then we represent each transformed loss $f_T(l_k)$ with an integer m_k according to $(m_k - 1)\Delta \leq f(l_k) < m_k\Delta$. Finally, we convert m_k to its N-bit binary representation. We denote this process as $Q(\cdot)$. The N-bit represented per-example loss then can be send back over the binary feedback channel.

2. Mapping from N bits to real number: When transmitter receives the N-bit represented per-example loss from the binary feedback channel, it first converts each N bits to an integer $\hat{m}_k$, and then maps $\hat{m}_k$ back to real number according to $\hat{l}_k = (\hat{m}_k + \frac{1}{2})\Delta$. We denote this process as $Q^{-1}(\cdot)$.

In summary, our quantization strategy is to map a each example loss to N bits, so that it can be restored at the transmitter side by sending that N bits back to the transmitter over a binary feedback channel. Our full quantization process can be written as $f_Q(\cdot) = Q^{-1}(Q(\cdot))$.

4

Performance Analysis

In this chapter, we provide the numerical results that we achieved from the proposed training approach. We start with showing results that illustrates the impact of feedback quantization on the trained system performance. Then, we compare the performances of communication systems trained with different quantization bits. After that, we further investigate the impact of qunantization on learning speed. In the end, we show the system performance of trained system when noise exists in the feedback link.

4.1 Setup and Parameters

4.1.1 Channel model

The communication system is trained over a simplified fiber-optic channel model described by (2.2). Numerical results are achieved by setting channel parameters are set to $L = 2000$km, $K = 20$, $\gamma = 1.27$, and $\sigma^2 = -21.3\,$dBm. The signal-to-noise ration (SNR) is defined as $SNR = P/\sigma^2$.

4.1.2 Transmitter and receiver neural networks.

The network parameters for transmitter and receiver are shown in Table 4.1

Table 4.1: Neural network parameters, where $M = 16$

	transmitter $f_{\boldsymbol{\theta}_T}$			receiver $f_{\boldsymbol{\theta}_R}$		
layer	1	2-3	4	1	2-3	4
number of neurons	M	30	2	2	50	M
activation function	-	ReLU	linear	-	ReLU	softmax

The neural network parameters are chosen by try-and-fail, for example, we tried several kinds of activation functions (e.g. TanH, Sigmoid, ReLU) and it turned out ReLU function enbles the best performance in our scenario. As for number of hidden layers that should be chose, we simply follow the trace that a too small neural network would lead to low accuracy and a too large network might take a long time to converge. In our setup when $M = 16$, a neural network with two hidden layers is shown to be sufficient for modelling the transmitter or receiver. The input layer of transmitter consists of $M = 16$ neurons, this is because we wish the input to the

neural network is one-hot encoded, which can helps the learning of neural network. The output layer of receiver is chosen to be a softmax layer, the intuition behind is that softmax layer outputs a probability distribution vector, and then the cross-entropy loss can be computed for trainining of neural network.

4.1.3 Training procedure

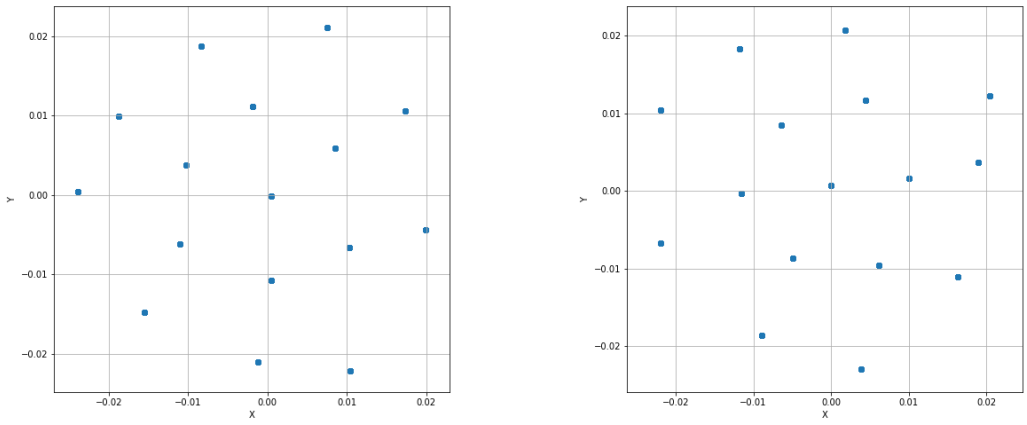
The alternative optimization is repeated for $N = 4000$ iterations, and each iteration consists of a receiver training phase and a transmitter training phase. In receiver training phase, transmitter parameters are fixed, and receiver parameters are updated for $N_R = 30$ iterations. In each receiver training iteration, a mini-batch of $N_R = 64$ samples are used for training. In transmitter training phase, receiver parameters are fixed, and transmitter parameters are updated for $N_T = 20$ iterations. In each transmitter training iteration, a mini-batch size $N_T = 64$ samples are used for training. Adam optimizer is chosen to perform gradient descent based parameters update, and the initial learning rate are set to $\alpha_R = 0.008$ and $\alpha_T = 0.001$ respectively.

4.1.4 Transmitter exploration variance

Transmitter exploration variance σ_p^2 has to be chosen carefully so that transmitter can be successfully learned. In particular, choosing σ_p^2 too small will result in insufficient exploration and slow down the training process. On the other hand, if σ_p^2 is chosen too large, the resulting noise may in fact be larger than the actual channel noise, resulting in many falsely detected messages and unstable training. In our simulations, we use $\sigma_p^2 = P \cdot 10^{-3}$.

4.2 Perfect vs quantized feedback

To visualize the impact of quantization on learning of the communication system, we start at looking at the learned constellations and their corresponding learned decision region. Fig. 4.1 shows the learned constellation for the quantized (left) and unquantized (right) schemes assuming the optical channel with $P = -5$dBm. Fig. 4.2 shows the corresponding learned decision regions. As one can see from the figures, both constellations have very similar constellation format and only slight difference can be observed between their corresponding decision regions.



(a) (b)

Figure 4.1: Learned constellations for the nonlinear optical channel, $M = 16$, and $P = -5\,\text{dBm}$ (a) without quantizing per-sample losses and (b) using the proposed quantization scheme and 1-bit quantization.

(a) (b)

Figure 4.2: Learned decision regions for the nonlinear optical channel, $M = 16$, and $P = -5\,\text{dBm}$ (a) without quantizing per-sample losses and (b) using the proposed quantization scheme and 1-bit quantization.

We further evaluate the impact of quantization on system performance. Here, the system performance is measured in terms of symbol error rate (SER). Fig. 4.3 shows the achieved SER assuming both perfect feedback without quantization and 1-bit quantization based on proposed quantization scheme. Note that for each input power P, a separate transmitter-receiver pair is trained due to the fact that optimal signal constellations and receivers for fiber-optic communication systems are highly

dependent on the transmission power [35]. From the figure we can see that system trained with 1-bit quantized feedback achieves very similar performance to the one trained with perfect feedback. As a reference, the performance of standard 16-QAM and Maximum-likelihood detector is also shown in the figure, we can see that both learning approach outperform this baseline.

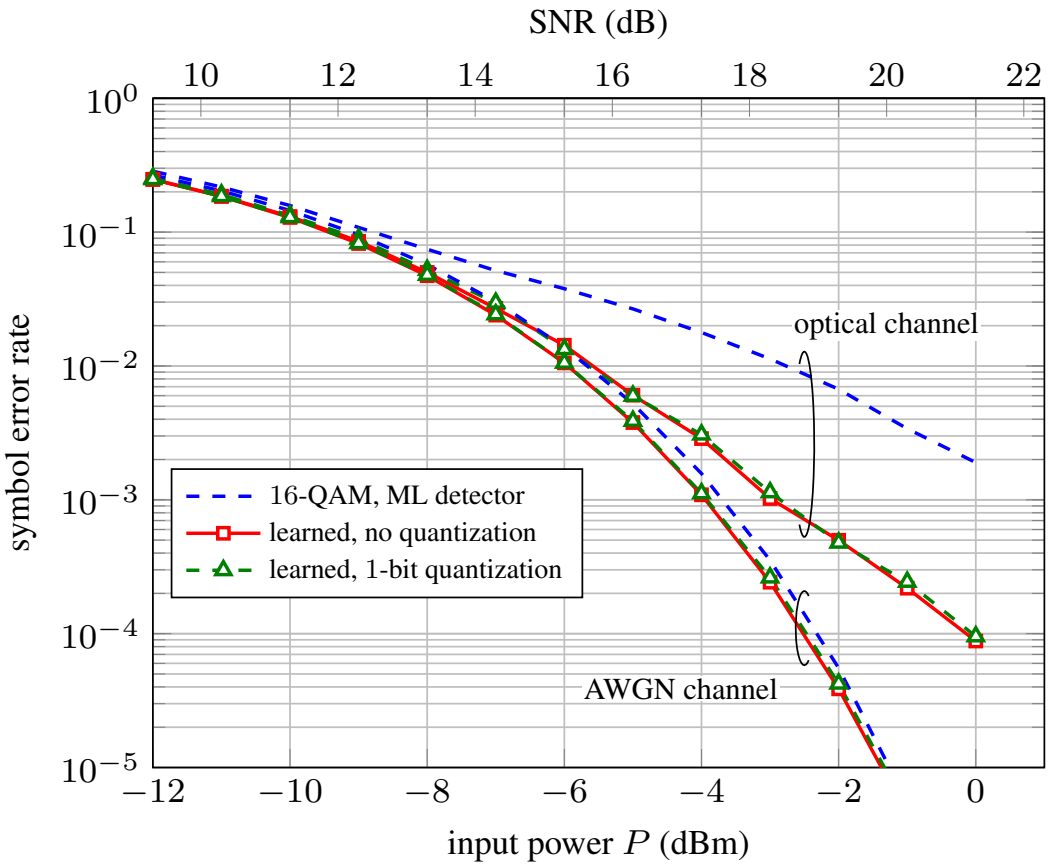


Figure 4.3: Symbol error rate achieved for $M = 16$. The training SNR is 15 dB for the AWGN channel, whereas training is done separately for each input power (i.e., SNR) for the optical channel.

To show that the proposed learning approach can be extended to any type of physical layer communication, we also consider a using case where the learning is performed over an AWGN channel. The parameters used for AWGN channel is consistent with optical channel except the fact that the transmitter and receiver are trained for a fixed $SNR = 15dB$. The resulting system is also shown in Fig. 4.3, and we can see that the same conclusion holds for AWGN channel.

4.3 Impact of number of quantization bits

In this section, numerical results are provided to evaluated the impact of the number of quantization bits on the performance. Fig. 4.4 shows the acheived SER when per-example losses are quantized with different quantizing schemes. The results show

that when feedback signal is uniformly quantized without loss transformation, the performance of trained system is highly dependent on the number of bits used for quantization and the assumed quantization range $(0, \bar{l})$. For example, when $\bar{l}$ is set to 10, the resulted system with 1-bit quantization has noticeable poorer and more unstable performance (as indicated by the error bar) then the case where more bits are used for quantization.When $\bar{l}$ is decreased to 3, the system performance improves essentially. As for the proposed quantization scheme, the resulted performance of trained system is independent of number of bits used for quantization and virtually indistinguishable from a system trained with unquantized feedback. Hence we can conclude that our proposed quantization scheme outperforms conventional uniform quantization and has negligible impact on system performance.

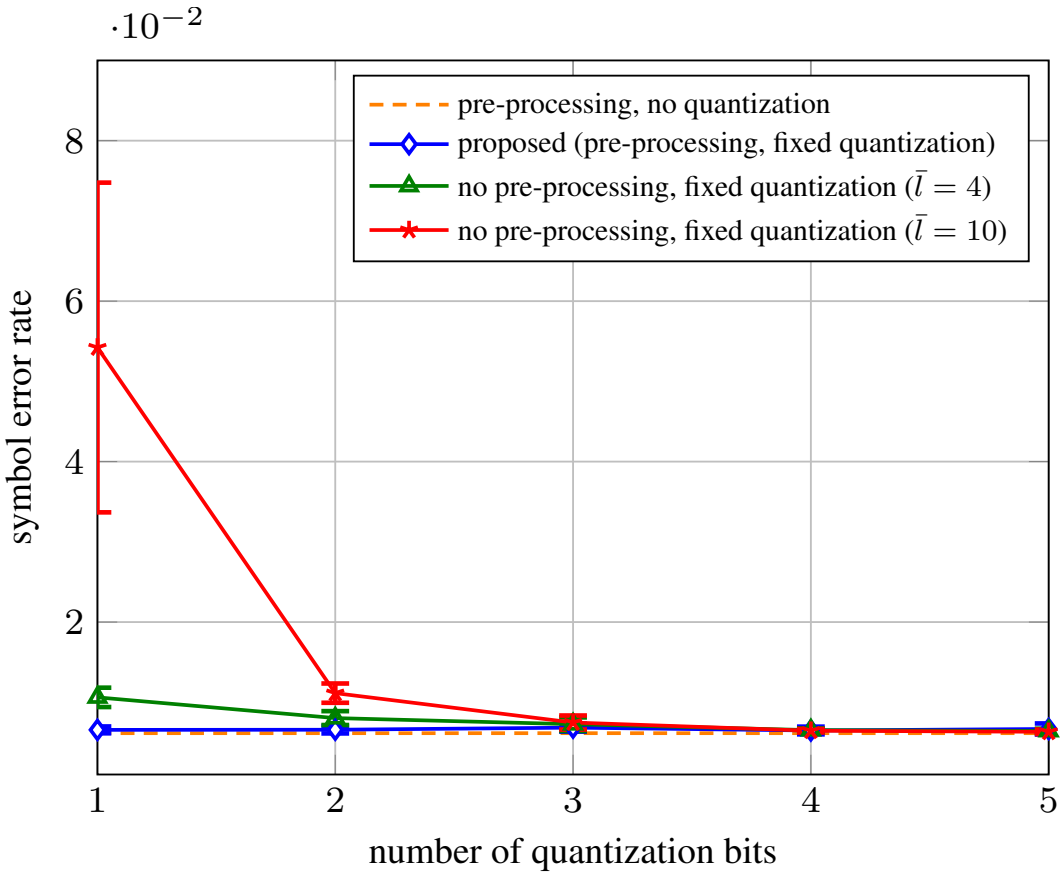


Figure 4.4: Impact of the number of quantization bits on the achieved performance for the nonlinear optical channel with $M = 16$, $P = -5$ dBm. Results are averaged over 10 different training runs where error bars indicate the standard deviation between the runs.

4.4 Impact on convergence rate

We further investigate the impact of quantization on convergence rate. Fig. 4.5 visualizes the evolution of empirical cross entropy loss $L(\boldsymbol{\theta}_R)$ during the alternative optimization for the optical channel with $P = -5dBm$. It can be seen that quan-

tization results in a slightly decreased convergence rate during the training. When per-example losses are quantized with 5 bits, the empirical loss takes roughly 80 iterations to converge, which is very similar to the case where per-example losses are not quantized. When only 1 bit is used for quantizing per-example losses, the training convergences roughly after 100 iterations, which is slightly slower then the unquantized scheme.

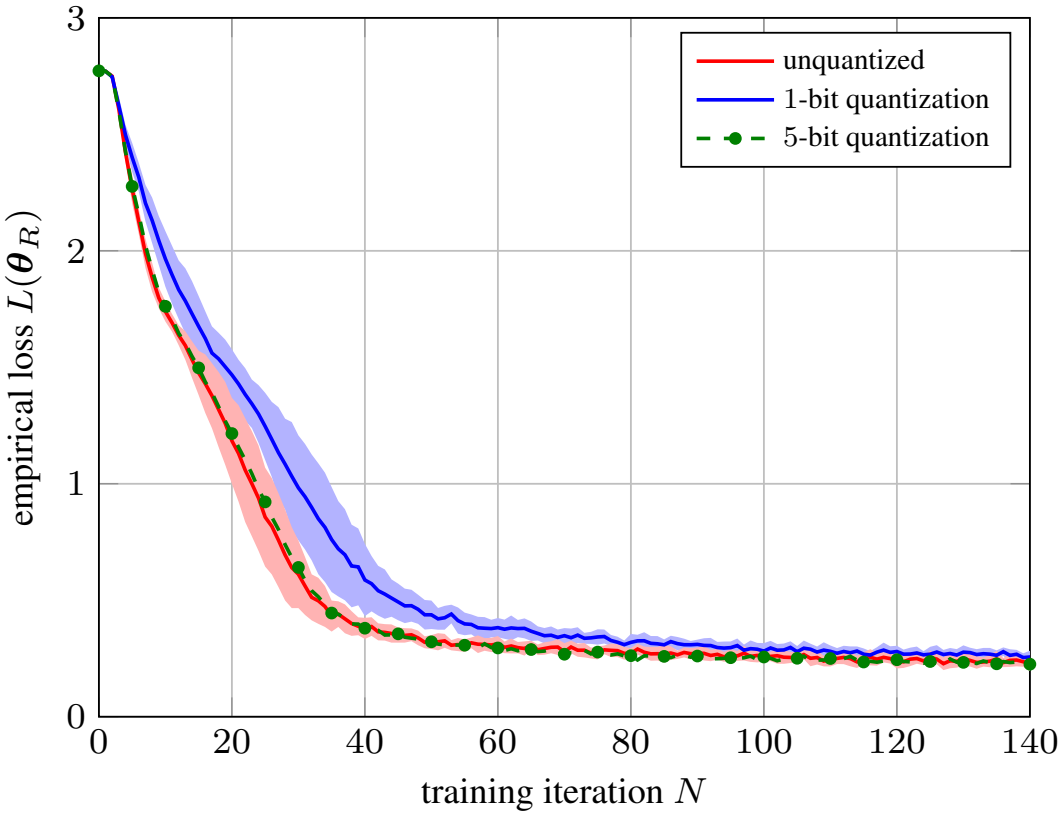


Figure 4.5: Evolution of $L(\boldsymbol{\theta}_R)$ during the alternating optimization for the fiber optical channel with $M = 16$, $P = -5$ dBm. Results are averaged over 15 different training runs where the shaded area indicates one standard deviation between the runs.

4.5 Impact of Noisy feedback

Due to the fact that noise always exists in communication systems, we further evaluate the performance of trained system when per-example losses are transmitted over a feedback link where noise exists. For numerical results, a fiber-optical channel with a fixed input power $P = -5$dBm is considered. Fig. 4.6 shows the resulted SER when feedback signal is transmitted over a binary symmetric channel with flipping rate $p \in [0, 0.5]$.

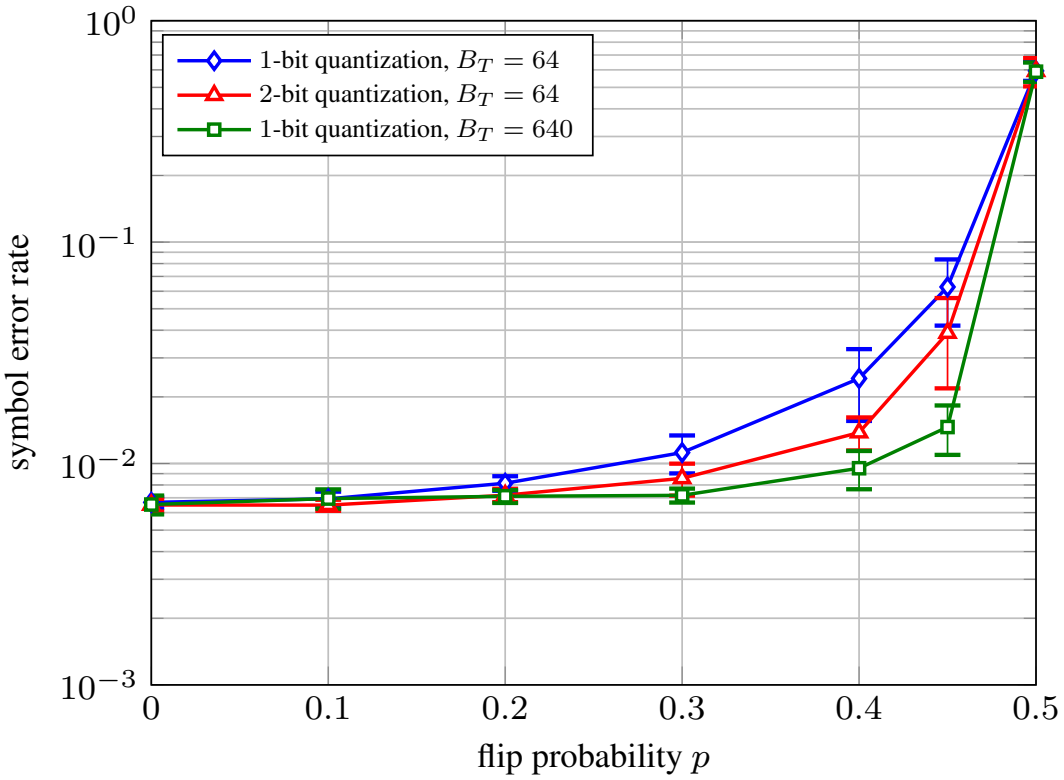


Figure 4.6: Performance on fiber optical channel with $M = 16$, $P = -5$dBm when transmitting quantized losses over a noisy feedback channel modeled as a binary symmetric channel with flip probability p. Results are average over 10 runs where the error bars indicate one standard deviation between runs.

Our simulation results indicate that the proposed quantization scheme is highly robust to the channel noise. In Fig. 4.6, one may observe that when the assumed mini-batch size is $B_T = 64$, the performance of resulted system starts to decrease only for high flipping rate and remains essentially unchanged when $p < 0.1$ with 1-bit quantization and $p < 0.2$ with 2-bit quantization. We also find that increasing the mini-batch size can essentially help training when there is high flipping probability. Fig. 4.6 shows that when the mini-batch size is increased from $B_T = 64$ to $B_T = 640$, the performance for 1-bit quantization is significantly improved and the performance remains unchanged when the flipping rate is as high as $p = 0.3$.

One may also notice that for $p = 0.5$, the achieved SER is slightly better than $(M - 1)/M \approx 0.938$ corresponding to random guessing. This is because receiver is still learning even though transmitter just performs random exploration.

5

Discussion

In Chapter 4, we show that quantiztion of feedback signal has negligible impact on the performance of trained system, and our proposed loss quantization approach is robust to noise in the feedback link. In this chapter, we discuss why the proposed approach enables sufficient learning of transmitter and receiver.

5.1 Impact of Quantized Feedback Signal

The effect of quantization can be assessed via the Bussgang Theorem [36], which is a generalization of MMSE decomposition. If we assume $l_k \sim p(l)$ with mean μ_l and variance σ_l^2, then

$$f_Q(l_k) = gl_k + w_k, \tag{5.1}$$

in which $g \in \mathbb{R}$ is the Bussgang gain and w_k is a random variable, uncorrelated with l_k provided, we set

$$g = \frac{\mathbb{E}\{l_k f_Q(l_k)\} - \mu_l \mathbb{E}\{f_Q(l_k)\}}{\sigma_l^2}. \tag{5.2}$$

When the number of quantization bits N increases, $f(l_k) \to l_k$ and thus $g \to 1$.

Let $\boldsymbol{\gamma}_k = l_k \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)$, $l_k \in [0,1]$, with $\nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T) = \mathbb{E}\{\boldsymbol{\gamma}_k\}$, and $\boldsymbol{\gamma}_k^{\mathrm{q}} = f_Q(l_k)\nabla_\tau \log \pi_\tau(\tilde{x}_k|m_k)$, if we replace l_k with $f_Q(l_k)$ in (3.4), denote the corresponding gradient function by $\nabla_{\boldsymbol{\theta}_T} L^{\mathrm{q}}(\boldsymbol{\theta}_T)$, and substitute (5.1), then the quantized version of the gradient function, which is the expectation of $\boldsymbol{\gamma}_k^q$ can be described by:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_T} L^{\mathrm{q}}(\boldsymbol{\theta}_T) &= \mathbb{E}\{\boldsymbol{\gamma}_k^q\} \\
&= \mathbb{E}\{f_Q(l_k)\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \\
&= g\mathbb{E}\{l_k \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} + \mathbb{E}\{w_k \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\}
\end{aligned} \tag{5.3}$$

Due to the fact that w_k is unrelated to l_k, we can further rewrite $\nabla_{\boldsymbol{\theta}_T} L^{\mathrm{q}}(\boldsymbol{\theta}_T)$ as:

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}_T} L^{\mathrm{q}}(\boldsymbol{\theta}_T) &= g\mathbb{E}\{l_k \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} + \mathbb{E}\{w_k\}\mathbb{E}\{\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \\
&= g\mathbb{E}\{l_k \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \\
&= g\nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T)
\end{aligned} \tag{5.4}$$

where the penultimate step follows the fact that $\mathbb{E}\{\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} = 0$.

The variance of the loss gradient $\mathbb{V}\{\boldsymbol{\gamma}_k^q\}$ is

$$\begin{aligned}\mathbb{V}\{\boldsymbol{\gamma}_k^q\} &= \mathbb{V}\{f_Q(l_k)\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\}\\ &= \mathbb{E}\{\|f_Q(l_k)\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\|^2\} - \|\mathbb{E}\{f_Q(l_k)\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\}\|^2\\ &= \mathbb{E}\{(f_Q(l_k))^2\|\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\|^2\} - g^2\|\nabla_{\boldsymbol{\theta}_T}L(\boldsymbol{\theta}_T)\|^2\\ &= g^2\mathbb{E}\{l_k^2\|\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\|^2\} - g^2\|\nabla_{\boldsymbol{\theta}_T}L(\boldsymbol{\theta}_T)\|^2\\ &+ \mathbb{E}\{w_k^2\|\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\|^2\}\\ &+ 2g\mathbb{E}\{l_k w_k\|\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\|^2\}\end{aligned} \tag{5.5}$$

since $-w_k l_k = l_k(gl_k - f_Q(l_k)) \leq \max_{l_k}|gl_k - f_Q(l_k)| = \bar{w}$, that $l_k \leq 1$, and that $\mathrm{tr}\{\mathbf{J}(\boldsymbol{\theta}_T)\} = \mathbb{E}\{\|\nabla\log\pi_\tau(x_k|m_k)\|^2\}$, the variance of $\boldsymbol{\gamma}_k$ can be bounded by:

$$\begin{aligned}\mathbb{V}\{\boldsymbol{\gamma}_k\} &\leq g^2\mathbb{V}\{\boldsymbol{\gamma}_k\} + \bar{w}^2\mathrm{tr}\{\mathbf{J}(\boldsymbol{\theta}_T)\} - 2g\mathbb{E}\{w_k l_k\|\nabla\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\|^2\}\\ &\leq g^2\mathbb{V}\{\boldsymbol{\gamma}_k\} + \bar{w}^2\mathrm{tr}\{\mathbf{J}(\boldsymbol{\theta}_T)\} + 2g\bar{w}\mathrm{tr}\{\mathbf{J}(\boldsymbol{\theta}_T)\}\end{aligned} \tag{5.6}$$

where $\mathbf{J}(\boldsymbol{\theta}_T) = \mathbb{E}\{\nabla_{\boldsymbol{\theta}_T}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\nabla_{\boldsymbol{\theta}_T}^{\mathsf{T}}\log\pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \succeq 0$ is the fisher information matrix of the transmitter parameters $\boldsymbol{\theta}_T$ and $\bar{w} = \max_l|gl - f_Q(l)| = |1 - 1/2^{N-1} - g|$ is a measure of the maximum quantization error. From (5.6) one can see that the variance is affected in two ways: a scaling with g^2 and an additive term that depends on the maximum quantization error and the Fisher information at $\boldsymbol{\theta}_T$. When number of quantization bits N increases, $g \to 1$ and $\bar{w} \to 0$, so that $\mathbb{V}\{\boldsymbol{\gamma}_k^{\mathrm{q}}\} \to \mathbb{V}\{\boldsymbol{\gamma}_k\}$, as expected. Hence, the impact of quantization is simply a scaling of the expected gradient when the mini-batch size is sufficiently large.

In general, the value of g is hard to compute in closed form, but for 1-bit quantization and a Gaussian loss distribution, (5.2) admits a closed-form solution.[1] In particular:

$$g = \begin{cases}1/\sqrt{8\pi\sigma_l^2} & \mu_l = 1/2\\ e^{-1/(8\sigma_l^2)}/\sqrt{8\pi\sigma_l^2} & \mu_l \in \{0,1\}.\end{cases} \tag{5.7}$$

By looking into the distributions of per-exampl losses from Fig. 3.3, we observe that (after loss transformation) for most iterations, $\mu_l \approx 1/2$ and σ_l^2 will be moderate (around $1/(8\pi)$), leading to $g \approx 1$. Only after many iterations $\mu_l < 1/2$ and σ_l^2 will be small, leading to $g \ll 1$. Hence, for sufficiently large batch sizes, 1-bit quantization should not significantly affect the learning convergence rate.

5.2 Impact of Noisy Feedback Channel

The impact of noise in the feedback channel on system performance can be evaluated by looking into the impact of flipped bits on the gradient function. We start with the case when only one bit is used for quantization. When per-example losses

[1] For Gaussian losses, $\bar{w}$ in (5.6) is not defined. The proposition can be modified to deal with unbounded losses.

are all quantized with one bit, $\Delta = 1/2$ and the quantized losses $\hat{l}_k$ are either $f(l_k) = \frac{\Delta}{2} = 1/4$ with probability $(1-p)$ or $1 - f(l_k) = 3/4$ with probability p. Then the gradient of reward function in Eq. 3.4 can be written as:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T) &= \mathbb{E}\{\hat{l}_k \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \\ &= \mathbb{E}\{f_Q(l_k)^{1-n_k}(1 - f_Q(l_k))^{n_k} \nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \end{aligned} \tag{5.8}$$

where n_k are independent and identically distributed Bernoulli random variables with parameters p. Since n_k is independent of all other random variables, we can compute:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T) &= \mathbb{E}\{((1-2p)f_Q(l_k) + p)\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \\ &= (1-2p)\mathbb{E}\{f_Q(l_k)\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} + p\mathbb{E}\{\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \\ &= (1-2p)\mathbb{E}\{f_Q(l_k)\nabla_{\boldsymbol{\theta}_T} \log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \end{aligned} \tag{5.9}$$

where the last step follows the property of policy gradient learning that $\mathbb{E}\{\nabla_{\theta_T} log \pi_{\theta_T}(\hat{x}_k|m_k)\} = 0$.

When each example loss is quantized with two bits, $\Delta = 1/4$, and the received losses have four possible values which are 1/8, 3/8, 5/8, 7/8. Then for a transmitted loss $f(l_k)$, the received loss follows

$$\begin{aligned} f_Q(l_k) &\text{ with probability } (1-p)^2 \\ 1 - f_Q(l_k) &\text{ with probability } p^2 \\ \text{other two} &\text{ with the same probability } p(1-p) \end{aligned}$$

and the expected received loss is

$$\mathbb{E}\{\hat{l}_k\} = (1-2p)f(l_k) + p \tag{5.10}$$

then, the gradient in Eq. 3.4 becomes

$$\nabla_{\boldsymbol{\theta}_T} L(\boldsymbol{\theta}_T) = (1-2p)\mathbb{E}\{f_Q(l_k)\nabla_{\boldsymbol{\theta}_T} log \pi_{\boldsymbol{\theta}_T}(\tilde{x}_k|m_k)\} \tag{5.11}$$

which is the same as the one for 1-bit quantization.

Hence, we can see that given a sufficiently large mini-batch, the gradient in Eq. 3.4 is simply scaled by a factor $1-2p$. This means that even under very noisy feedback, learning should still be possible. This is consistent to our simulation results that system performance improves when the mini-batch size is increased from $N_T = 64$ to $N_T = 640$.

6
Conclusion

We have proposed a novel method for data-driven learning of fiber-optic communication in the presence of a binary feedback channel. The training of communication system is achieved through alternative training approach, and the loss feedback relies on an adaptive clipping, shifting and scaling of losses followed by a fixed quantization at the receiver, and a fixed reconstruction method at transmitter. We have shown that the proposed method (i) can lead to good performance even under 1-bit feedback; (ii) does not significantly affect the convergence speed of learning; and (iii) is highly robust to noise in the feedback channel.

However, these still exist limitations on this approach. Firstly, training a separate transmitter and receiver pair for each input power is very inefficient, and more efficient training scheme needs to be developed. Secondly, each example loss requires a feedback signal from receiver to transmitter, this slows the training down when the feedback link is very long. Thirdly, constellation size M is limited to 16, higher constellation orders may require a big fully-connected neural network, which may take a long time to train. For the last limitation, possible solution can be using convolutional neural network (CNN)instead of fully-connected neural network, since has been shown to have better performance in pattern recognition problems, and each input message can be regard as one pattern.

Bibliography

[1] C. E. Shannon, "A mathematical theory of communication, bell systems tech," *J*, vol. 27, pp. 379–423, 1948.

[2] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[3] P. Weinberger, "John kerr and his effects found in 1877 and 1878," *Philosophical Magazine Letters*, vol. 88, no. 12, pp. 897–907, 2008.

[4] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2016, pp. 223–228.

[5] T. O'Shea 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, 2017.

[6] R. Fritschek, R. F. Schaefer, and G. Wunder, "Deep learning for channel coding via neural mutual information estimation," *arXiv preprint arXiv:1903.02865*, 2019.

[7] N. Farsad, M. Rao, and A. Goldsmith, "Deep learning for joint source-channel coding of text," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2326–2330.

[8] H. Ye, G. Y. Li, and B.-H. Juang, "Power of deep learning for channel estimation and signal detection in ofdm systems," *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, 2018.

[9] T. J. O'Shea, T. Erpek, and T. C. Clancy, "Deep learning based mimo communications," *arXiv preprint arXiv:1707.07980*, 2017.

[10] T. Erpek, T. J. O'Shea, and T. C. Clancy, "Learning a physical layer scheme for the mimo interference channel," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–5.

[11] T. J. O'Shea, T. Roy, and N. West, "Approximating the void: Learning stochastic channel models from observation with variational generative adversarial networks," in *2019 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2019, pp. 681–686.

[12] F. A. Aoudia and J. Hoydis, "End-to-End Learning of Communications Systems Without a Channel Model," *arXiv:1804.02276*, 2018.

[13] ——, "Model-free Training of End-to-end Communication Systems," *arXiv:1812.05929*, 2018.

[14] C. de Vrieze, S. Barratt, D. Tsai, and A. Sahai, "Cooperative Multi-Agent Reinforcement Learning for Low-Level Wireless Communication," *arXiv:1801.04541*, 2018.

[15] V. Raj and S. Kalyani, "Backpropagating Through the Air: Deep Learning at Physical Layer Without Channel Models," *IEEE Commun. Lett.*, vol. 22, no. 11, pp. 2278–2281, Nov. 2018.

[16] M. Goutay, F. A. Aoudia, and J. Hoydis, "Deep reinforcement learning autoencoder with noisy feedback," *arXiv preprint arXiv:1810.05419*, 2018.

[17] W. House, "Big data: A report on algorithmic systems, opportunity, and civil rights. executive office of the president," 2016.

[18] F. Idachaba, D. U. Ike, and H. Orovwode, "Future trends in fiber optics communication," 2014.

[19] H. F. Taylor, *Fiber optics communications*. Artech House Dedham, Mass, 1983.

[20] K. Keykhosravi, G. Durisi, and E. Agrell, "A tighter upper bound on the capacity of the nondispersive optical fiber channel," in *2017 European Conference on Optical Communication (ECOC)*. IEEE, 2017, pp. 1–3.

[21] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE transactions on information theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[22] E. Bullmore and O. Sporns, "Complex brain networks: graph theoretical analysis of structural and functional systems," *Nature reviews neuroscience*, vol. 10, no. 3, p. 186, 2009.

[23] A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, *Knowledge-Based and Intelligent Information and Engineering Systems, Part II: 15th International Conference, KES 2011, Kaiserslautern, Germany, September 12-14, 2011, Proceedings*. Springer, 2011, vol. 6882.

[24] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[25] K.-l. Hsu, H. V. Gupta, and S. Sorooshian, "Artificial neural network modeling of the rainfall-runoff process," *Water resources research*, vol. 31, no. 10, pp. 2517–2530, 1995.

[26] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[27] L. Bottou, "Stochastic gradient descent tricks," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 421–436.

[28] A. C. Wilson, B. Recht, and M. I. Jordan, "A lyapunov analysis of momentum methods in optimization," *arXiv preprint arXiv:1611.02635*, 2016.

[29] M. C. Mukkamala and M. Hein, "Variants of rmsprop and adagrad with logarithmic regret bounds," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2545–2553.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[31] P. J. Huber, "Robust estimation of a location parameter: Annals mathematics statistics, 35," 1964.

[32] R. S. Sutton, A. G. Barto *et al.*, *Introduction to reinforcement learning*. MIT press Cambridge, 1998, vol. 2, no. 4.

[33] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 2219–2225.

[34] F. A. Aoudia and J. Hoydis, "End-to-end learning of communications systems without a channel model," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2018, pp. 298–303.

[35] S. Li, C. Häger, N. Garcia, and H. Wymeersch, "Achievable information rates for nonlinear fiber communication via end-to-end autoencoder learning," in *2018 European Conference on Optical Communication (ECOC)*. IEEE, 2018, pp. 1–3.

[36] H. Rowe, "Memoryless nonlinearities with Gaussian inputs: Elementary results," *The BELL system technical Journal*, vol. 61, no. 7, pp. 1519–1525, 1982.