IE 56100 Final Project Proposal

Qinbo Bai

1 Introduction of the topic

As the deep learning technology becomes popular, it has been successfully applied to many fields. In recent years, it has been considered in the physical layer of communication system. Different from the traditional communication system, which consists of several blocks to finish the communication task, end-to-end communication system has been fulfilled by deep learning and achieve similar performance. Fig. 1 gives the concept of such a system.

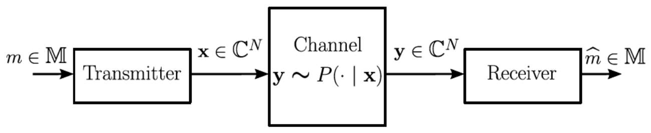
$m \in \mathbb{M}$
Transmitter
$\mathbf{x} \in \mathbb{C}^N$
Channel
$\mathbf{y} \sim P(\cdot \mid \mathbf{x})$
$\mathbf{y} \in \mathbb{C}^N$
Receiver
$\widehat{m} \in \mathbb{M}$

Figure 1: The architecture of end-to-end communication system

Using deep learning technology, we implement both the transmitter and receiver as the neural network and interpret the whole system as an autoencoder. In order to optimize the whole system, we need to do the back-propitiation with respect to transmitter and receiver. However, without the exact formulation of channel function, the optimization for transmitter cannot be implemented directly, and the project aims to optimize the whole system with zero-order information.

2 Problem formulation

An end-to-end communication system is designed to reliably exchange message at two nodes. The transmitter maps a message $\boldsymbol{s}$ from the set $\mathbf{S} = 1, 2, ..., s$ to a N dimensional complex number $\boldsymbol{x}$. At the receiver, a distorted symbol $\hat{\boldsymbol{x}}$ is received due to the impairments of the channel. Finally, the receiver maps the distorted symbol $\hat{\boldsymbol{x}}$ to the estimated $\hat{\boldsymbol{s}}$. The goal of a successful system is to establish the faithful reconstruction.

To make the following statement clear, we denote the transmitter and receiver as function f_T and f_R with the parameter $\boldsymbol{\theta_T}$ and $\boldsymbol{\theta_R}$ in the neural network. Furthermore, we donate the channel as function h with some parameter $\boldsymbol{\theta_H}$. Thus, the restored symbol can be express as:

$$\hat{\boldsymbol{s}} = f_R(h(f_T(\boldsymbol{s}, \boldsymbol{\theta_T}), \boldsymbol{\theta_H}), \boldsymbol{\theta_R}) \tag{1}$$

In order to optimize the parameter in the transmitter and receiver, a loss function between $\boldsymbol{s}$ and $\hat{\boldsymbol{s}}$ needs to be established. A common choice is the cross-entropy function over M symbols and the per-example loss is as below: as below:

$$l^i = \sum_{s_j \in \boldsymbol{s}_i} -s_j \log(\hat{s}_j) \tag{2}$$

And the total loss is the sum of M symbols as below

$$\mathcal{L}(\boldsymbol{s}_{1:M}, \hat{\boldsymbol{s}_{1:M}}) = \sum_{i=1}^{M} l^i = \sum_{i=1}^{M} \sum_{s_j \in \boldsymbol{s_i}} -s_j \log(\hat{s_j}) \tag{3}$$

Using the gradient descent algorithm to optimize, we do back-propagation on $\hat{\boldsymbol{s}}$ with respect to $\boldsymbol{\theta_T}$ and $\boldsymbol{\theta_R}$. Receiver optimization is trivial. However, based on the chain rule,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta_T}} = \frac{\partial \mathcal{L}}{\partial f_R} \cdot \frac{\partial f_R}{\partial h} \cdot \frac{\partial h}{\partial f_T} \cdot \frac{\partial f_T}{\partial \boldsymbol{\theta_T}} \tag{4}$$

However, without the knowledge of function h, this cannot be calculated directly.

3 Training Transmitter with Reinforcement Learning

(Aoudia and Hoydis, 2018) uses the reinforcement learning to train the transmitter without using the knowledge of channel function. The implement is shown in Fig. 3.

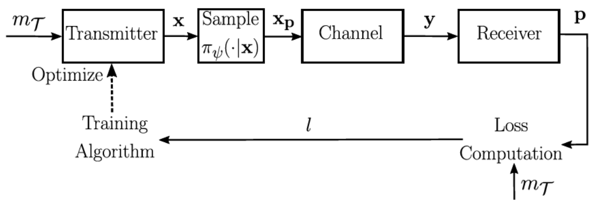

Figure 2: RL approach to train the transmitter

In order to enable the exploration, the author maps the output of the transmitter $\boldsymbol{x}$ to the random variable $\boldsymbol{x}_p$ with Gaussian distribution. It means,

$$\boldsymbol{x}_p = \sqrt{1-\sigma^2}\boldsymbol{x} + \boldsymbol{\omega} \tag{5}$$

where ω is a complex gaussian random vector with $\boldsymbol{\omega} \sim \mathcal{CN}(\mathbf{0}, \sigma^2\mathbf{I})$. Obviously, the pdf of $\boldsymbol{x}_p$ can be expressed as:

$$\pi(\boldsymbol{x}_p | f_{\theta_T}(s)) = \frac{1}{(\pi\sigma^2)^N} \exp(-\frac{\|\boldsymbol{x}_p - \sqrt{1-\sigma^2} f_{\theta_T}(s)\|^2}{\sigma^2}) \tag{6}$$

To see how to modify this problem to a RL problem, we regard the message s as the state, the input of channel $\boldsymbol{x}_p$ as the action, the per-example loss l as the cost. The policy is the action distribution given the state, which is the Gaussian policy shown exactly by equation Eq. (6). It is obvious that the next state(next message) is independent of the current state(current message) and current action(current input of the channel). Thus, we can consider each episode of the reinforcement learning as 1.

Using the Monte-Carlo simulation and famous policy gradient algorithm (Sutton and Barto, 2018), the parameter in transmitter can be updated by

$$\boldsymbol{\theta_{T+1}} = \boldsymbol{\theta_T} - \eta \frac{1}{B_T} \sum_{i=1}^{B_T} l^i \nabla_{\boldsymbol{\theta}_T} \log(\pi(\boldsymbol{x}_p^i | f_{\boldsymbol{\theta}_T}(s^i))) \tag{7}$$

where the gradient in the above equation can be computed as below,

$$\nabla_{\boldsymbol{\theta_T}} \log(\pi(\boldsymbol{x}_p | f_{\boldsymbol{\theta_T}}(s^i))) = \frac{2\sqrt{1-\sigma^2}}{\sigma^2} \left(\nabla_{\boldsymbol{\theta}_T} f_{\boldsymbol{\theta}_T}(s^i)\right)^T \left(\boldsymbol{x}_p - \sqrt{1-\sigma^2} f_{\boldsymbol{\theta}_T}(s^i)\right) \tag{8}$$

In order to implement RL algorithm, it is convenient to use popular deep learning platform such as **Tensor-flow** to run auto-gradient.

4 Training Transmitter with direct non-convex zeroth-order optimization

Based on Jin et al. (2017) work, I purpose a zeroth-order algorithm (Bai et al., 2019) to direct optimization the function with only the access to the function value, which means the exact form of the channel function isn't needed anymore. The basic idea of this algorithm is to split the optimization into two parts. Firstly, we need to use the function evaluation to estimate the gradient at a given point. Then, follow the framework of Jin et al. (2017) algorithm to finish the gradient descent. Next, we show the algorithm to estimate the gradient.

Algorithm 1 Gradient Estimation $GE(d, l, B, c', \hat{\epsilon}, \boldsymbol{x})$

```
1: $v \leftarrow \frac{\hat{\epsilon}}{c'l(d+3)^{1.5}}$, $\sigma^2 \leftarrow 2c'^2(d+4)B^2$, $m \leftarrow \frac{32\sigma^2}{\hat{\epsilon}^2}(\log(\frac{1}{\hat{\epsilon}}) + \frac{1}{4})$
2: Generate $\boldsymbol{u}_1, ...\boldsymbol{u}_m$, where $\boldsymbol{u}_i \sim \mathcal{N}(0, \mathbf{I}_d)$
3: $\hat{\nabla} f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} \frac{f(\boldsymbol{x}+v\boldsymbol{u}_i)-f(\boldsymbol{x})}{v}\boldsymbol{u}_i$
4: return $\hat{\nabla} f(\boldsymbol{x})$
```

Consider the whole cost function as f with variable $\boldsymbol{\theta}_T$. Algorithm 4 gives the approach to estimate the gradient with the method of Gaussian smoothing. d is the dimension of $\boldsymbol{\theta}_T$, l is l-smooth parameter of function f, B is the bound for gradient norm $\|\nabla f\|$, $c' > 1$ is a constant. $\boldsymbol{x}$ is the point we make an estimation and $\hat{\epsilon}$ is the intended gap between the estimated gradient and true gradient. The line 1 in the algorithm gives the parameter used in the following lines. v is the Gaussian smoothening parameter. σ^2 is the bound for the variance of gradient estimator in one sample. m gives the total number of samples we need to get error less than $\hat{\epsilon}$. Line 2 generates m Gaussian random vector with zero mean and variance $\mathbf{I}_d$ which are used to calculate the estimate of the gradient. The estimation algorithm (Line 3) takes an average of estimated gradient using m samples.

Next algorithm utilizes the estimated gradient and optimize the transmitter.

Algorithm 2 Estimated Gradient Descent Algorithm $EGD(\boldsymbol{x}_0, d, l, B, \chi_1, \theta, \rho, \epsilon, \hat{\epsilon}, c, c', \delta, \Delta_f)$

```
1: $\chi \leftarrow \max\{(1+\theta)\log(\frac{2d\ell\Delta_f}{c\epsilon^2\delta}), \chi_1\}$, $\eta \leftarrow \frac{c}{l}$, $g_{thres} \leftarrow \frac{\sqrt{c}}{\chi^2}\cdot\epsilon$, $f_{thres} \leftarrow \frac{c}{\chi^3}\cdot\sqrt{\frac{\epsilon^3}{\rho}}$, $t_{thres} \leftarrow \frac{\chi}{c^2}\cdot\frac{l}{\sqrt{\rho\epsilon}}$,
   $t_{temp} \leftarrow -t_{thres} - 1$, $r \leftarrow \frac{g_{thres}}{l}$
2: for $t = 0, 1, ...$ do
3:     $\hat{\nabla} f(\boldsymbol{x}_t) = GE(d, l, B, c', \hat{\epsilon}, \boldsymbol{x}_t)$
4:     if $\|\hat{\nabla} f(\boldsymbol{x}_t)\| \leq g_{thres}$, $t - t_{temp} > t_{thres}$ then
5:         $\boldsymbol{x}_t \leftarrow \boldsymbol{x}_t + \boldsymbol{\xi}_t$, $\boldsymbol{\xi}_t \sim \mathbb{B}_{\boldsymbol{x}_t}(r)$
6:         $t_{temp} \leftarrow t$
7:     end if
8:     if $t - t_{temp} = t_{thres}$ and $f(\boldsymbol{x}_t) - f(\boldsymbol{x}_{t-t_{thres}}) > -f_{thres}$ then
9:         return $\boldsymbol{x}_{t-t_{thres}}$
10:    end if
11:    $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t - \eta\hat{\nabla} f(\boldsymbol{x}_t)$
12: end for
```

It can be proved that the final output of the algorithm is an second-order ϵ stationary point. Thus, it can be the local minimal of the function f (escape all strict saddle point).

5 Final project plan

The plan of this project is to implement the two methods given in section 3 and 4 and compare the performance and convergence rate of these two ideas. Besides, the project will include the formal problem formulation, the analysis of problem properties, theoretical analyze of the proposed algorithm.

References

Aoudia, F. A. and Hoydis, J. (2018). End-to-end learning of communications systems without a channel model. In *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, pages 298–303.

Bai, Q., Agarwal, M., and Aggarwal, V. (2019). Escaping Saddle Points for Zeroth-order Nonconvex Optimization using Estimated Gradient Descent. *arXiv e-prints*, page arXiv:1910.01277.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. (2017). How to escape saddle points efficiently.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* The MIT Press, second edition.