MMSE Based Channel Estimator

Zhao-Jie, Luo

janny00kevin@gmail.com

Advisor: Professor Carrson C. Fung

National Yang Ming Chiao Tung University

May. 9 2024

Reference

1. C. C. Fung, and D. Ivakhnenkov, "Model-Driven Neural Network Based MIMO Channel Estimator"
2. M. Eisen, C. Zhang, L.F.O. Chamon, D.D. Lee and A. Ribeiro, "Learning optimal power allocations in wireless systems," *IEEE Trans. on Signal Processing*, vol. 67(10), pp. 2775-2790, May 2019.
3. OpenAI Spinning Up introduction to RL Part 3: Intro to Policy Optimization

Problem Formulation

Our objective is to minimize the expected mean square error:

$$\min_{\hat{\mathbf{h}}} \mathbb{E}_{\mathbf{y},\mathbf{h}} \left[\left\| \mathbf{h} - \hat{\mathbf{h}} \right\|_2^2 \right]$$

and it can be written in epigraph form as:

$$\begin{aligned} &\min_{t,\mathbf{h}} t \\ &\quad s.t. \ \mathbb{E}_{\mathbf{y},\mathbf{h}} \left[\left\| \mathbf{h} - \hat{\mathbf{h}} \right\|_2^2 \right] \leq t \end{aligned}$$

We use parameterize channel estimator so that $\hat{\mathbf{h}} = \phi(\mathbf{y}; \boldsymbol{\theta})$, with $\boldsymbol{\theta}$ denoting the parameters of the neural network.

Then the Lagrangian function of (15) can be written as

$$\begin{aligned}\mathcal{L}\left(\hat{\mathbf{h}}, t, \lambda\right) &= t + \lambda\left(\mathbb{E}_{\mathbf{y},\mathbf{h}}\left[\left\|\mathbf{h} - \hat{\mathbf{h}}\right\|_2^2\right] - t\right) \\ &= t + \lambda\left(\mathbb{E}_{\mathbf{y},\mathbf{h}}\left[\|\mathbf{h} - \phi(\mathbf{y}; \boldsymbol{\theta})\|_2^2\right] - t\right)\end{aligned}$$

It is uncertain whether or not the duality gap equals zero.
However, the stationary point of $\mathcal{L}\left(\hat{\mathbf{h}}, t, \lambda\right)$ can be found via the KKT conditions by solving for the primal and dual variables alternately using gradient descent and ascent, respectively:

$$\begin{aligned}
\boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k - \alpha_{\boldsymbol{\theta},k}\lambda_k \nabla_{\boldsymbol{\theta}_k} \mathbb{E}\left[\|\mathbf{h} - \phi(\mathbf{y}; \boldsymbol{\theta}_k)\|_2^2\right] \\
t_{k+1} &= t_k - \alpha_{t,k}(1 - \lambda_k) \\
\lambda_{k+1} &= \left[\lambda_k + \alpha_{\lambda,k}\left(\mathbb{E}\left[\|\mathbf{h} - \phi(\mathbf{y}; \boldsymbol{\theta}_{k+1})\|_2^2\right] - t_{k+1}\right)\right]_+
\end{aligned}$$

Policy Gradient (1)

We have the policy gradient theorem:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[G(\tau)] = \mathbb{E}_\tau \left[\sum_{t=0}^{T-1} G(\tau) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t | S_t) \right]$$

At each time step, $t = 1, ..., T-1$:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_\tau[G(\tau)] = \mathbb{E}_\tau \left[G(\tau) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t | S_t) \right]$$

And we can estimate the policy gradient with sample mean:

$$\widehat{\nabla}_{\boldsymbol{\theta}} \mathbb{E}_\tau[G(\tau)] = \frac{1}{|\mathcal{D}|} \sum_{\mathcal{D}} G(\tau) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(A_t | S_t)$$

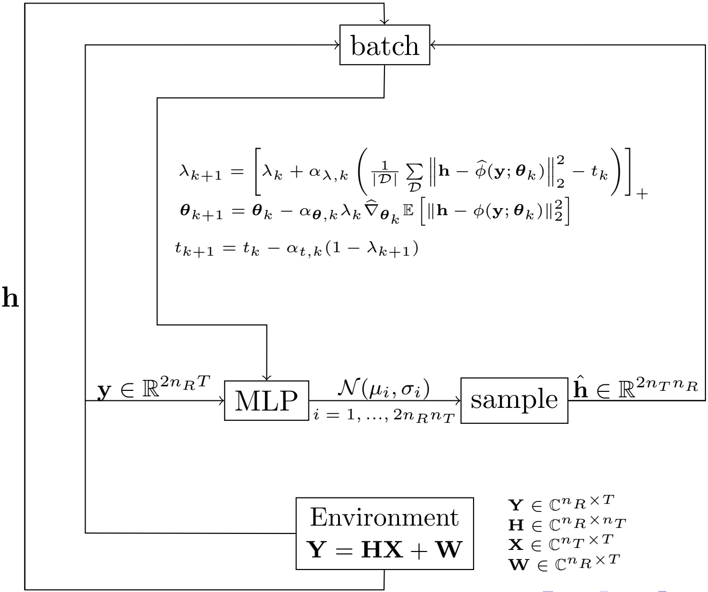
Policy Gradient (2)

Our goal is to minimize the mean square error, by substituting $\mathbb{E}_\tau[G(\tau)]$, $\pi_{\boldsymbol{\theta}}(A_t|S_t)$ with $\mathbb{E}_{\mathbf{y},\mathbf{h}}\left[\|\mathbf{h}-\phi(\mathbf{y};\boldsymbol{\theta})\|_2^2\right]$, and $\pi_{\boldsymbol{\theta}}(\hat{\mathbf{h}}|\mathbf{y})$.
Thus, we obtain the estimated policy gradient for our problem:

$$\widehat{\nabla_{\boldsymbol{\theta}}}\mathbb{E}_{\mathbf{y},\mathbf{h}}\left[\|\mathbf{h}-\phi(\mathbf{y};\boldsymbol{\theta})\|_2^2\right] = \frac{1}{|\mathcal{D}|}\sum_{\mathcal{D}}\left\|\mathbf{h}-\widehat{\phi}(\mathbf{y};\boldsymbol{\theta})\right\|_2^2 \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}\left(\hat{\mathbf{h}}|\mathbf{y}\right) \quad (1)$$

where $\hat{\phi}(\mathbf{y};\boldsymbol{\theta}) = \hat{\mathbf{h}}$ is the sampled output of the policy.

Experiment Diagram



$\mathbf{Y} \in \mathbb{C}^{n_R \times T}$
$\mathbf{H} \in \mathbb{C}^{n_R \times n_T}$
$\mathbf{X} \in \mathbb{C}^{n_T \times T}$
$\mathbf{W} \in \mathbb{C}^{n_R \times T}$

Simulation Result (1)

"Epoch" and "iteration" in the following 3 pages:

1. The process consisting of data collection followed by model updating is called one "iteration".
2. Performing the processes above for a number of iterations, which means that updating the model for a number of iterations, is called one "epoch" here.
3. After Performing the whole process number of epoch times, taking the average of these loss curve and then plot the figures in the following 3 pages.

Simulation Result (2)

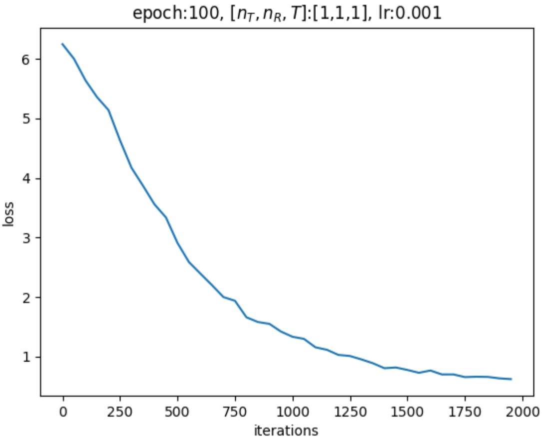


Figure: hidden layer sizes: [64,32], $\mu_H = 5, \sigma_H = 0.2, \mu_W = 0, \sigma_W = 0.1$

Simulation Result (3)

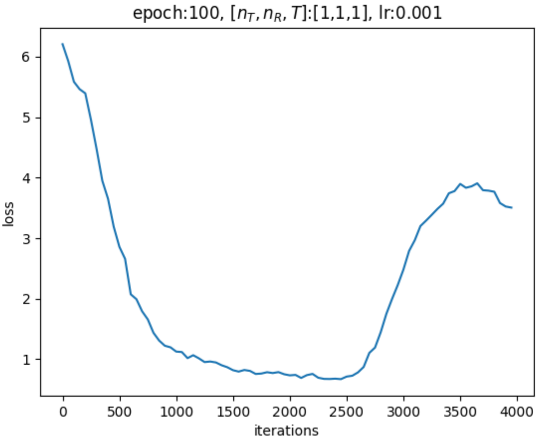


Figure: increase the iterations from 2000 to 4000

Simulation Result (4)

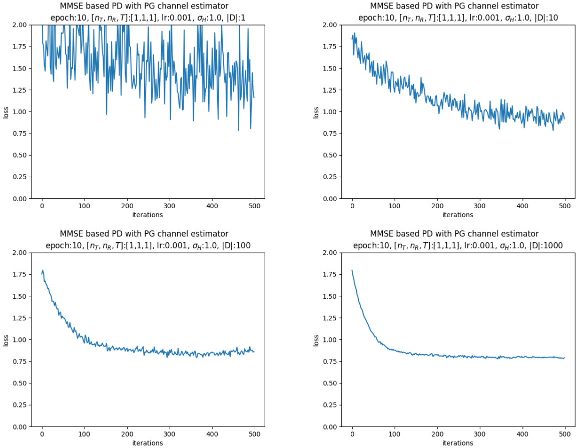


Figure: Number of trajectories from 1 to 1000

Simulation Configuration (1)

parameters	values
$[n_R, n_T]$	[1,1] or [4,36]
T	$n_T \times 1$
μ_H, σ_H	0, 1
Hidden layer size	[64,32] or [512,512]
Length of each trajectory	1
Batch size $= \|\mathcal{D}\|$	10,000
Number of batches	100
Trainning dataset	1,000,000
Validation dataset	2,000
Epoch	100 $\sim$ 2,000
Learning rate	1e-3 $\sim$ 1e-5

Simulation Configuration (2)

- Let the loss be Normalized Mean Squared Error (NMSE):

$$NMSE = \frac{1}{N}\sum_{n=1}^{N}\frac{\|\mathbf{h}_n - \phi(\mathbf{y}_n;\boldsymbol{\theta}_k)\|_2^2}{\|\mathbf{h}_n\|_2^2} \tag{2}$$

  Here, N represents the size of the training or validation dataset, implying that the Normalized Mean Squared Error (NMSE) is calculated for each epoch.

- Let step size $\alpha_{t,k}$ and $\alpha_{\lambda,k}$ to be $\alpha_{t,0}/k$ and $\alpha_{\lambda,0}/k$, where k is the iteration index.

Simulation Result: $n_R, n_T = 1, 1$

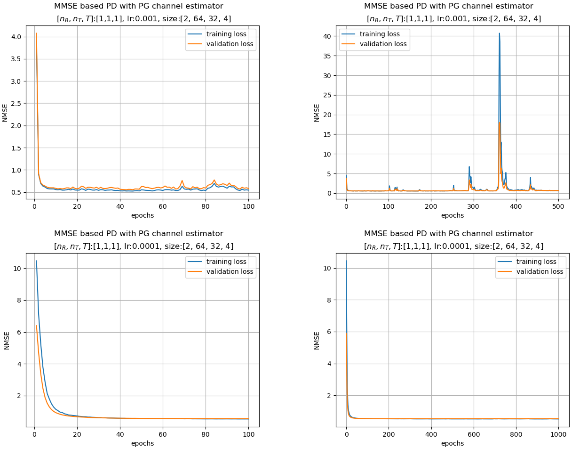


Figure: n_R, n_T = [1,1], hidden layer size = [64,32] (a)(b) lr: 1e-3, epoch from 100 to 500 (c)(d) lr: 1e-4, epoch from 100 to 1000

Simulation Result: $n_R, n_T = 4, 36$

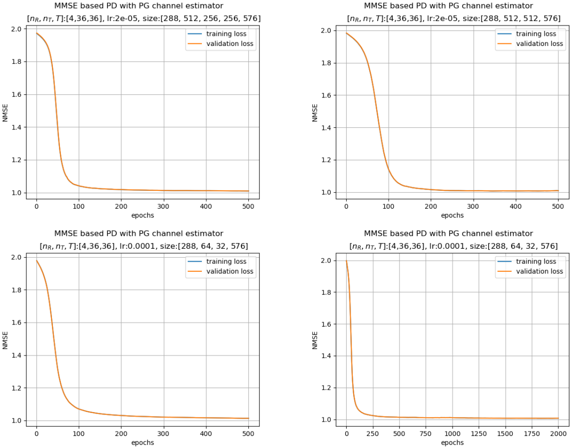


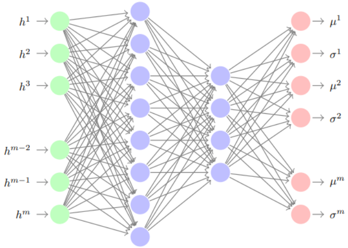
Figure: n_R, n_T = [4,36] (a)(b) hidden layer size = [512,256,256], [512,512] lr: 2e-5, epoch: 500 (c)(d) hidden layer size = [64,32] lr: 1e-4, epoch from 500 to 1000

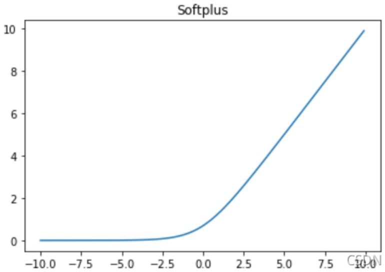
Conclution & Problems

- In Rayleigh fading channel, if we choose an appropriate learning rate, the loss will converge without raising up or spike and there's no overfitting problem in at least 2000 epochs.
- However, Professor noticed that the NMSE in linear scale approaches 1 but not 0. This could be attributed to the variance generated by the MLP converging to 0. Consequently, $\hat{\mathbf{h}}$ sampled from this distribution also converge to $\mathbf{0}$.
- Professor suggested first checking $\hat{\mathbf{h}}$ to identify any potential issues. Alternatively, removing the sampling process to see if there is any improvment.

Discussion (1)

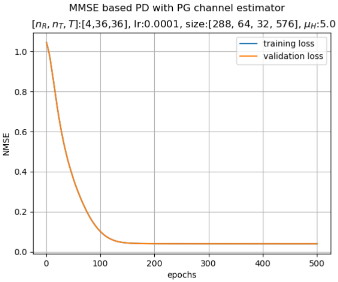
- Regarding the Rayleigh fading channel case with parameters $\mu_H = 0$ and $\sigma_H = 1$, the mean component of the output from the MLP converges to 0, while the standard deviation component converges to less than 10^{-2}. Consequently, the sampled output tends to converge to 0.
- The output of MLP need to represent a distribution, otherwise policy gradient(1) can't be implemented.

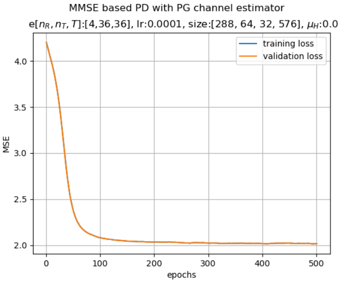




Discussion (2)

- The denominator in the NMSE(2) is close to 0. Try $\mu_H \neq 0$.
- The activation function of the final layer of the MLP is Softplus $(\log 1 + e^x)$ to make the standard deviation stay positive. In order to deal with $\mu_H \leq 0$, changing the activation to identity function, then take the exponetial of the standard deviation part of the MLP outputs.





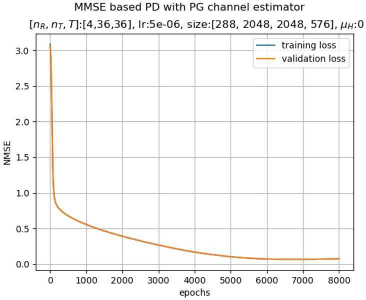
Discussion (3)

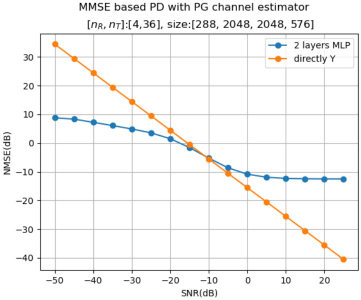
- For the case $\mu_H = 5$ the mean part of the MLP outputs are close to 5, the deviation part are less than -2. Then it just simply converge to 5.
- MSE here is defined as $MSE = \frac{1}{N}\sum_{n=1}^{N}\frac{\|\mathbf{h}_n - \phi(\mathbf{y}_n;\boldsymbol{\theta}_k)\|_2^2}{n_T n_R}$
  MSE converges to 2 when $n_T, n_R = 4, 36$ no matter μ_H is.
- It seems like the estimator just let the output approach μ_H, then let $\hat{\mathbf{h}} = y$, then the MSE is much less than 2.

Updates on 5/16 (1)

- I increased the size of the MLP model's hidden layers to 2048 and 2048. It converges better than the previous one with hidden layers of 64 and 32 (converges to 1).
- The "directly $\mathbf{Y}$" curve is the LS solution
- I plotted the NMSE of this model and the LSE with respect to SNR. The model's curve appears unusual. Additionally, I made a mistake in calculating the signal power: it should be the power of $\mathbf{HX}$ and not just $\mathbf{X}$.

Updates on 5/16 (2)





The figure on the right is incorrect because I mistakenly used the power of $\mathbf{X}$ instead of the power of $\mathbf{HX}$.

Updates on 5/23 (1)

$$\mathbf{Y} = \mathbf{H}\mathbf{X} + \mathbf{W} \xrightarrow{\text{vectorize}} \mathbf{y} = (\mathbf{X}^T \otimes \mathbf{I}_{n_T})\mathbf{h} + \mathbf{w} \rightarrow \mathbf{y} = \mathbf{h} + \mathbf{w}$$

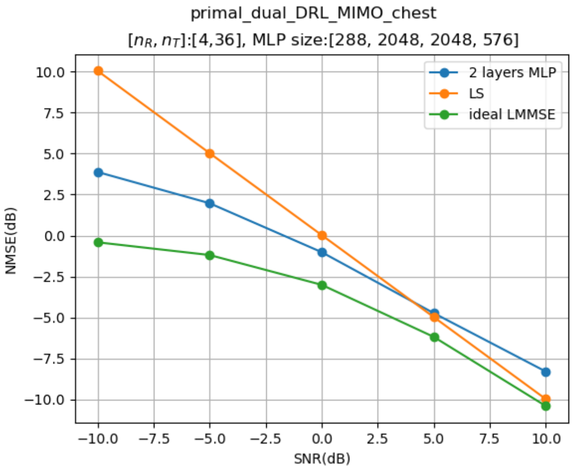
LMMSE estimator: $\hat{\mathbf{h}} = \mathbb{E}[\mathbf{h}] + \mathbf{C}_{\mathbf{hy}}\mathbf{C}_{\mathbf{yy}}^{-1}(\mathbf{y} - \mathbb{E}[\mathbf{y}]) \xrightarrow{\text{zero mean}} \mathbf{C}_{\mathbf{hy}}\mathbf{C}_{\mathbf{yy}}^{-1}\mathbf{y}$,

where $\mathbf{C}_{\mathbf{hy}} = \mathbb{E}[\mathbf{h}\mathbf{y}^T] = \mathbb{E}[\mathbf{h}(\mathbf{h} + \mathbf{w})^T] = \mathbb{E}[\mathbf{h}\mathbf{h}^T]$,

$\mathbf{C}_{\mathbf{yy}} = \mathbb{E}[(\mathbf{h} + \mathbf{w})(\mathbf{h} + \mathbf{w})^T] = \mathbb{E}[\mathbf{h}\mathbf{h}^T] + \mathbb{E}[\mathbf{w}\mathbf{w}^T]$

- ideal case:
  $\mathbf{C}_{\mathbf{hh}}(\mathbf{C}_{\mathbf{hh}} + \mathbf{C}_{\mathbf{ww}})^{-1}\mathbf{y} = \sigma_{\mathbf{H}}^2\mathbf{I}(\sigma_{\mathbf{H}}^2\mathbf{I} + \sigma_{\mathbf{W}}^2\mathbf{I})^{-1}\mathbf{y} = \frac{\sigma_{\mathbf{H}}^2}{\sigma_{\mathbf{H}}^2 + \sigma_{\mathbf{W}}^2}\mathbf{y}$
- sample case: let $\mathbf{C}_{\mathbf{hh}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{h}_n\mathbf{h}_n^T$

Updates on 5/23 (2)



This model is trained with $\sigma_{\mathbf{W}} = 0.1$ (SNR = 20). When trained with a group of different SNR values, it seems that a larger model is required.