CSE910 Final Project: Beam-Selection ML

Jeremy Arsenault

May 2023

[1]

1 Introduction

mmWave communication tasks often use beamforming to achieve directional or spatially selective communication. Long-range communication tasks use beamforming to boost SNR and in MIMO tasks base stations require spatial selectivity to communicate simultaneously with spatially distinct users. A common method is analog beamforming, where transmitters / receivers choose from a discrete set of predetermined beams on the antenna before transmitting / receiving. On the transmission side different beam choices apply specific amplitude/phase variation to the analog signal before transmission, and on the receiver side different beam choices apply specific amplitude / phase variation then sum the received analog signals.

In practice antenna arrays require high-resolution beamforming, so exhaustively searching the space of possible transmitter-receiver beam pairs to establish a high SNR link is not practical. Further, transmission mediums have complex structure that is difficult to exploit using hand-designed search algorithms. For example one might expect that hill-climbing algorithms work well to search for optimal beam pairs, but in practice the beam pair SNR landscape can be non-convex. What then is the best policy for selecting beam-pairs during transmission? We propose formulating beam-selection as an optimization problem on a Markov reward process (MRP), and explore how we can use machine learning to design beam-selection policies which effectively leverage the complex relationship between beam-pair and SNR.

2 System Model

We model the process of repeatedly choosing a beam pair, transmitting data, and observing its SNR as a Markov reward process (MRP)[5].

We seek to send C bits of data as efficiently as possible by using the least possible total number of transmissions of fixed duration. With perfect knowledge of the environment, an optimal policy would repeatedly send messages

[1]Code is available on Github

on the highest SNR beam pair until all C bits had been transmitted. We assume each transmission is coded optimally and sends $\propto \log(1+\mathrm{SNR})$ bits. Let $R_e^\pi(T) = \sum_{t=0}^{T} \log(1+\mathrm{SNR}_{a_t})$ be the number of bits sent by taking actions $\{a_t \sim \pi(s_t)\}_{0\le t\le T}$ under policy π in environment e. Formally, we seek to solve the following optimization problem:

$$\min_{\pi} \mathop{\mathbb{E}}_{e\sim E\ a\sim\pi} \left[\sum_{t=0}^{\infty} 1_{[R_e^\pi(t)<C]}\right]$$

Where 1_x is a function that takes value 1 then x is true, and otherwise takes value 0. The more efficiently the C bits have been transmitted, the lower the expected value.

An environment e in this MRP, as shown in figure 1, is a stationary map from all possible beam pairs to SNR. The environment is represented as a $N\times M$ array where N is the number of receiver beam indices and M is the number of transmitter beam indices.

An observation or state (r, s) in a partial observation of the environment and contains all information of past actions. It is represented as a tuple of how much data has already been transmitted $r \in \mathbb{R}$, and which beam pairs have been transmitted on with their respective SNR $s \in \mathbb{R}^{2\times N\times M}$.

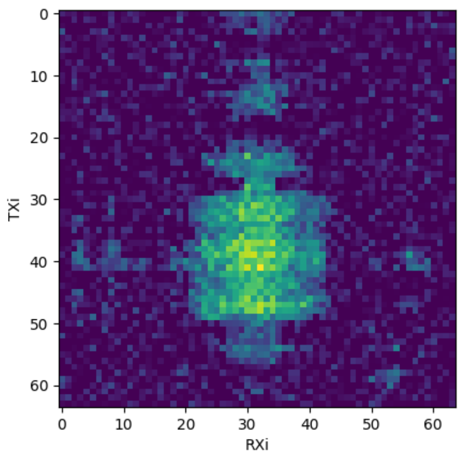

Figure 1: Representation of example environment as heatmap. Pixel color is determined by the SNR of the corresponding TXi, RXi beam pair.

3 Data

The dataset used to create different environments was recorded from a real transmitter and receiver on a test bench[3]. 20% of the environments were set aside for testing. The dataset was augmented by adding gaussian noise, and rotating and translating the SNR maps.

4 Model

We approximate functions on $\{(r,s)\} \to \mathbb{R}^d$ (e.g. value functions, constrained policy functions) using a typical architecture of several CNN layers followed by two linear layers, before which the scalar state information is concatenated with the CNN output. We approximate functions on $\{(r,s)\} \to \mathbb{R}^{N\times M}$ (e.g. unconstrained policy functions) using a U-Net like architecture [2], where the above above network architecture is used as the encoder and the decoder is constructed from CNN layers with residual connections.

In reinforcement learning tasks it is often desirable to simplify the task by freezing most of your network and only updating the parameters of the last few layers. As such, the encoder and decoder are pretrained to learn features of the environment on a denoising task. 'Noisy' inputs are generated by simulating $t \sim \mathcal{U}_{[1,N]}$ timesteps under a random policy and observing the state s_t. The denoising pretraining objective is then $\min_{\pi_b} ||\pi_b(s_t) - e||_{l_2}$.

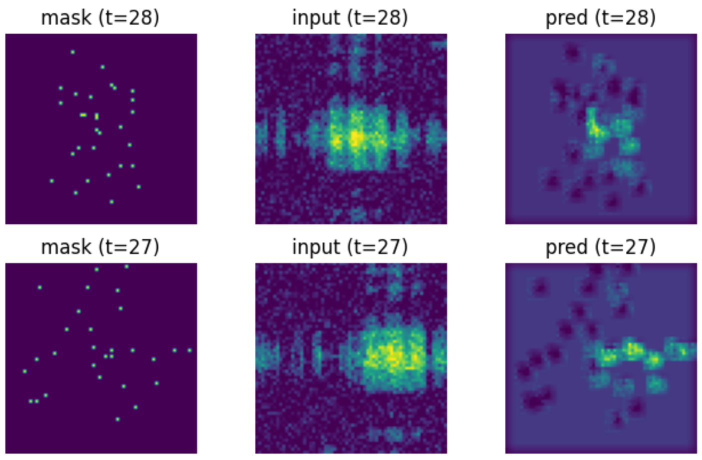

Figure 2: (left) Channel 1 of the input state. Yellow pixels are beam pairs with previously observed SNR. Purple pixels have not been observed and are effectively 'masked'. (center) Environment used to generate input and used as target. (right) Prediction of environment made using state as input.

5 Baseline Policy

A natural search policy we can create using the pretrained base model is to explore by following a random policy on $t < k$ then exploit using the pretrained base model. We can choose a good k empirically by running simulating on the training environment for different values of k.

Algorithm 1 Baseline policy π_k with pretrained model π_b

```
1: if $R_e^\pi(t) < k$ then                                        ▷ Explore
2:     $a \sim \mathcal{U}_A$
3: else if $R_e^\pi(t) \geq k$ then                                ▷ Exploit
4:     $a \sim \delta^{a \in A}_{\arg\max_a \pi_b(r,s)}$           ▷ $\delta$ is delta kronecker distribution
5: end if
   return $a$
```

This is a surprisingly good heuristic, transmitting the data in an average of 58 messages ($n = 100$) on the validation set, but is not directly related to our optimization problem.

6 Experiments with Reinforcement Learning

Reinforcement learning (RL) is the subject dealing with optimization on Markov reward processes. Policy gradient methods are a family of RL algorithms that are well suited for problems with large action spaces. Of these, we choose an implementation of PPO [4] to optimize our policy.

6.1 Unconstrained Policy

In the unconstrained problem, we search for a policy $\pi : \{(r, s)\} \to \mathbb{R}^{N \times M}$ approximates the 'best next action' distribution over the entire action space. The problem is unconstrained because we can learn any action distribution.

Using this approach we were unable to find policies which outperformed the baseline. We suspect this is primarily due to high variance policy updates due to many good exploratory and exploitative actions in any given state. This high variance problem is well known in policy gradient methods. We attempted to mitigate this using the clipping method described in PPO [4] but were still unable to learn a good policy.

6.2 Constrained Policy: adaptive k

In the constrained problem, we search for a policy $\pi : \{(r, s)\} \to \{\theta\}$ that approximates the 'best next action' parameters $\{\theta\}$ of some parameterized distribution over the action space. The problem is constrained because we can only learn a subset of all policies. Learning a less complicated

We attempt to improve the baseline policy by learning a policy which uses the state to dynamically choose a value of k deciding when to explore or exploit.

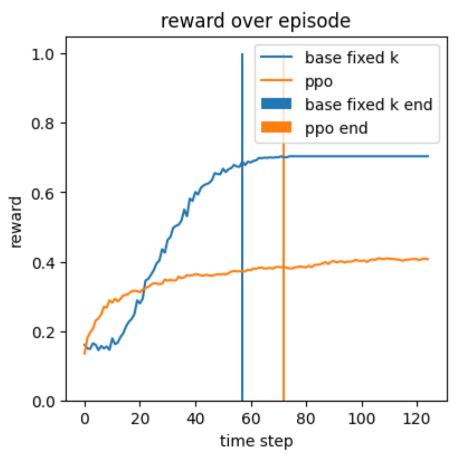

Figure 3: Performance of unconstrained policy learned under PPO (orange). This method learned to transmit the data in an average of 72 messages ($n = 100$) on the validation set. The line plots show the number of bits sent at each timestep, and the vertical bar shows the time there C bits were transmitted.

Algorithm 2 Constrained RL policy π_c for adaptive explore / exploit

```
1:  $p_1 \leftarrow \mathcal{U}_A$
2:  $p_2 \leftarrow \delta^{a \in A}_{\arg\max_a \pi_b(r,s)}$
3:  **if** $R^\pi_e(t) < k_{\min}$ **then**                          ▷ Explore
4:      $a \sim p_1$
5:  **else if** $k_{\min} \leq R^\pi_e(t) < k_{\max}$ **then**       ▷ Dynamic explore / exploit
6:      $k \sim \pi^*_c(r, s)$                   ▷ $\pi^*_c$ gives the scalar policy parameter
7:      $a \sim (1-k)p_1 + kp_2$
8:  **else if** $k_{\min} \leq R^\pi_e(t) < k_{\max}$ **then**       ▷ Exploit
9:      $a \sim p_2$
10: **end if**
    **return** $a$
```

Using this method we were able to outperform the baseline, transmitting the data in an average of 48 messages ($n = 100$) on the validation set.

7 Conclusion

Formulating beam search as a transmission time optimization problem may be preferable to treating it as a classic search problem. We show that reinforcement learning techniques can be used to find good beam search policies using this objective. Although we were unable to learn a good policy without imposing some additional constraints, we do not believe these results suggest any inherent problems with our approach. These algorithms are very sensitive to hyperparameter

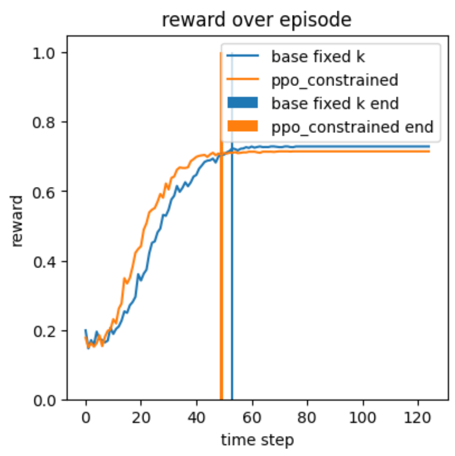

Figure 4: Performance of constrained policy learned under PPO (orange). This method learned to transmit the data in an average of 58 messages ($n = 100$) on the validation set. The line plots show the number of bits sent at each timestep, and the vertical bar shows the time there C bits were transmitted.

choice and implementation details [1] which make the optimization problem challenging. We show that constraining the reinforcement learning problem can mitigate some of these challenges and that there is a trade-off between optimally of the learnt policy and the difficulty of the optimization problem.

We hope that this work will motivate others to explore the use of reinforcement learning in classic beam search problems.

References

[1] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on PPO and TRPO. *CoRR*, abs/2005.12729, 2020.

[2] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[3] Alphan Sahin. Iq data for a given tx-rx beam index pair for link-level analysis in the 60 ghz band, 2023.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[5] Wikipedia contributors. Markov decision process — Wikipedia, the free encyclopedia, 2023. [Online; accessed 3-May-2023].