

# Programming Assignment - 1

Team: 3

B. SHREEYAN: SE21UARI024

K. HEMANTH: SE21UARI049

M. SHASHANK: SE21UARI087

TP SRAVYA: SE21UARI147

## Section: 5.1

The sources of noise in mmWave vehicular communication are as follows:

1. **Position and Velocity Measurement Noise:** Vehicles measure their positions (x, y) and velocities using GPS, and these measurements are subjected to random noise. This noise affects the accuracy of the context that the base stations receive, leading to variations in the car's position and velocity data. As a result, the base station might select a beam direction based on imprecise information, introducing randomness into the reward.
2. **Blockages:**
  - **Mobile Blockage:** Communication links can be blocked by other moving objects, such as pedestrians or other vehicles. The environment has a mobile blockage probability, which introduces randomness in whether a car's communication will succeed. Even if a beam is directed accurately, the link may fail due to unforeseen blockages.
  - **Fixed Blockages:** Stationary objects like buildings or trees can block the communication path between a base station and the vehicles. These fixed blockages, combined with the random placement of cars, add variability to the reward since a car's position relative to a blockage can change across time slots.
3. **Number of Cars:** The number of cars that want to communicate at each time slot is a random variable (between 1 and 4). This directly affects the reward, as fewer cars might result in lower total received power, even if the chosen beam direction is optimal.
4. **Beam Direction and Transmission:** Each base station can transmit only in specific beam directions, and the reward is dependent on whether the chosen beam aligns with the position of the vehicle. Since vehicle movement introduces variability, there's randomness in whether the beam will successfully cover the vehicle at the right time.

# Programming Assignment - 1

## Section: 5.2

- **Padding and Masking:**
  - One approach is to pad all context vectors to a fixed size by adding zeros (or other placeholder values). Then, use a masking layer in your machine learning (ML) or deep learning (DL) model to ignore these padded values during training and inference. For example, if the maximum number of vehicles is 4, you could pad smaller context vectors (e.g., for 2 cars) to the length required for 4 cars.
- **Recurrent Neural Networks (RNNs):**
  - RNN-based models like LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units) are well-suited for handling variable-length sequences. These models can process each car's context sequentially and maintain a memory of previous inputs, allowing them to handle a varying number of cars effectively without the need for padding.
- **Attention Mechanism:**
  - An attention mechanism allows the model to focus on relevant parts of the input (i.e., the context vectors of individual vehicles) regardless of the input size. This mechanism dynamically computes a weighted sum of the input vectors based on their importance, eliminating the need for fixed-size inputs.
- **Pooling Techniques:**
  - You can apply global pooling operations like max-pooling or average-pooling over the context vectors. These operations reduce the input size to a fixed-size summary of the variable-length input, which can then be processed by a standard feedforward neural network.

## Section: 5.6



# Programming Assignment - 1

## Section: 5.7

- a. In designing a neural network for a value function-based policy in the context of mmWave vehicular communication, the inputs and outputs that we need to consider are:

### ***inputs to Neural Network:***

1. **Context Features:** These would represent the current state of the environment and can include various parameters that affect communication performance such as:

- Vehicle speed.
- Distance to the nearest base station.
- Signal to noise ratio (**SNR**).
- Number of nearby vehicles or users.
- Current network load (how many other vehicles are communicating)

2. **Action Features:** This represents the action taken by the agent in response to the context. In mmWave communication, possible actions might include:

- Selecting a specific base station to connect.
- Choosing a beam direction for communication.
- Deciding on a frequency band or communication protocol.

***Total Number of Inputs:*** If we assume there are:

- $n_c$  context features (i.e, Vehicle speed, distance, SNR etc)
- $n_a$  possible actions (i.e, number of base stations or beam options)

**Total Inputs:**  $(n_c + n_a)$

**Outputs:** the output of the neural network would be a *single value* which is the estimated mean reward associated with the context-action pair. This predicted reward will help guide the agent's decision-making process. Therefore, the number of outputs is 1 (the predicted reward).

- b. Training this neural network is a regression problem.

**Reason:** In regression tasks, we predict a continuous output based on input features. In this context, the neural network is designed to predict the expected reward (a continuous value) associated with a specific context-action pair. Since the output is not discrete (like class labels) but rather a real-valued score, it aligns with the definition of regression.

## Programming Assignment - 1

### c. Gradient of $L(\theta)$ with respect to and update equation:

The loss function for a single time step  $t$  is defined as:

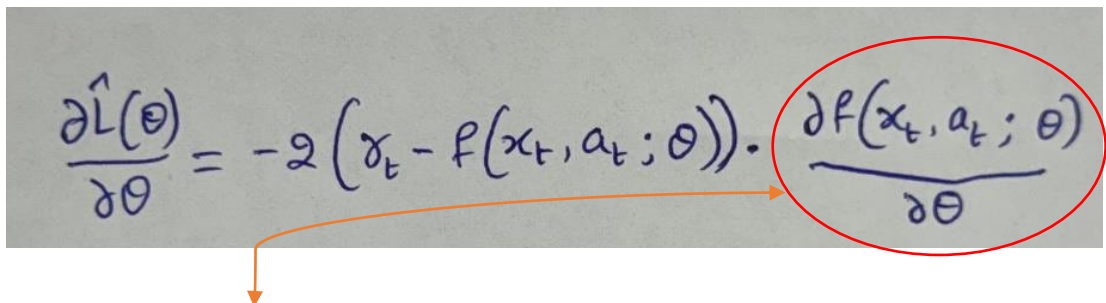
$$L(\theta) = (r_t - f(x_t, a_t; \theta))^2$$

where:

→  $r_t$  is the observed reward at time  $t$ ,

→  $f(x_t, a_t; \theta)$  is the predicted reward from the neural network, which depends on the context-action pair  $(x_t, a_t)$  and the network parameters  $\theta$ .

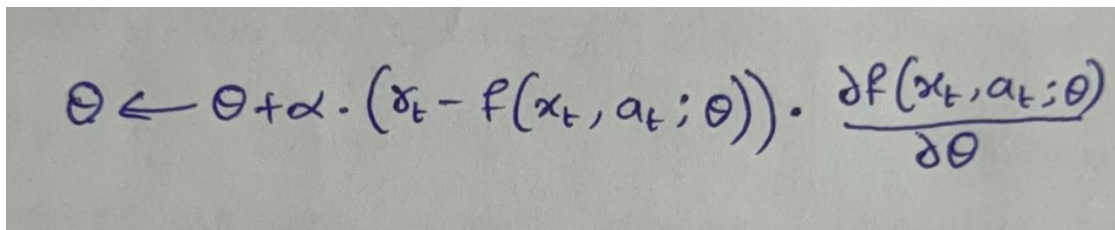
The gradient  $L(\theta)$  with respect to  $\theta$  is given by:



A handwritten equation on a grey background:  $\frac{\partial L(\theta)}{\partial \theta} = -2(x_t - f(x_t, a_t; \theta)) \cdot \frac{\partial f(x_t, a_t; \theta)}{\partial \theta}$ . The second term,  $\frac{\partial f(x_t, a_t; \theta)}{\partial \theta}$ , is circled in red. An orange arrow points from this circled term down to the text below.

where the 2<sup>nd</sup> expression is the gradient of the neural network output with respect to its parameters  $\theta$ .

the **gradient descent updates equation** for the parameter  $\theta$  is:



A handwritten equation on a grey background:  $\theta \leftarrow \theta + \alpha \cdot (x_t - f(x_t, a_t; \theta)) \cdot \frac{\partial f(x_t, a_t; \theta)}{\partial \theta}$ .

where:

→  $\alpha$  is the learning rate,

→  $r_t - f(x_t, a_t; \theta)$  is the error between the actual and predicted reward.

## Programming Assignment - 1

d.

- The pseudocode describes a reinforcement learning approach using the epsilon-greedy algorithm and a neural network to balance exploration and exploitation.
- A neural network is created with two dense layers:
  - The first layer has 64 neurons with ReLU activation.
  - The second layer predicts rewards for possible actions.
  - It uses Adam optimizer and MSE loss to minimize errors and improve predictions.
- The epsilon-greedy policy allows:
  - Random exploration with a probability of epsilon.
  - Exploitation by selecting the action with the highest predicted reward when a random number exceeds epsilon.
  - Epsilon decays over time, shifting focus from exploration to exploitation.
- During training:
  - The agent takes actions, observes rewards, and trains the network by minimizing the error between predicted and actual rewards.
  - Over episodes, the agent becomes better at predicting optimal actions based on the context.
- Training loop runs for a fixed number of episodes:
  - Each episode resets the environment, updates actions, and trains the model.
  - Epsilon decreases, shifting from random exploration to more informed decisions.
- This approach enables the agent to learn optimal actions over time, balancing exploration and exploitation with neural network predictions improving with training.