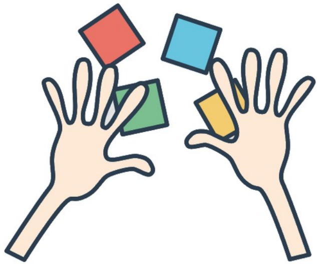


Hands-on: #1

1. Understanding and Execution of ANN; Hebbian Learning; MLPs

Study the code from class:

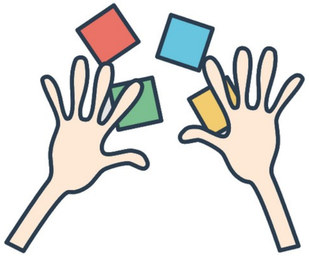
- MLP without Backprop
- Hebbian Learning with random weights
- MLP with Backprop
- Using Torch.

a) Análisis del Código

- Read and analyze the implementation of the mentioned algorithms ANNs.
- Identify the key components of each method

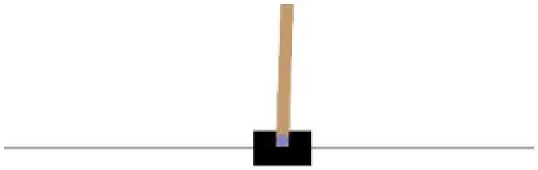
b) Experiment with Simulated Data

- Use a synthetic data to test each algorihtm: binaryclassification on a 2D dataset, Gaussian blobs, etc). Split the data into training and testing.
- Implement the methods and observe their behavior on this dataset.



Hands-on: #2

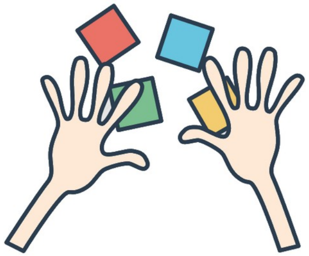
DQN for Cart-pole



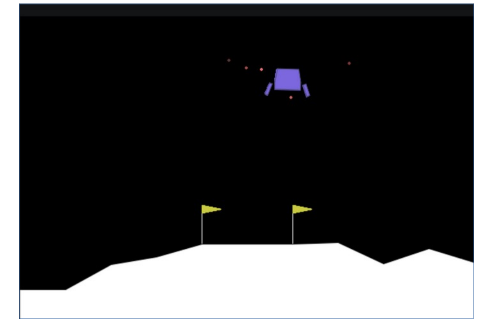
1. Run the Provided DQN Code
    - Objective: Verify that you can reproduce results similar to those shown in class.
    - Tasks:
        - Execute the CartPole DQN training code as is.
        - Track and log the average episode return (reward) over training.
        - Record how many episodes are typically needed to reach the CartPole “solved” threshold (e.g., averaging 195 points over 100 episodes).

2. Observe Convergence
    - Objective: Understand how hyperparameters affect training speed and stability.
    - Tasks:
        - Generate a plot of average reward vs. training episode.
        - Note how quickly (or slowly) the DQN stabilizes.
        - Discuss potential reasons if training is unstable or if the agent never “solves” the task.

3. Small Exploration & Tweaks (Optional Mini-Tasks)
    - Objective: Gain insight into how DQN hyperparameters affect performance.
    - Possible Modifications:
        - Learning Rate: Increase or decrease it (e.g., 1e-3, 1e-4) and compare training curves.
        - Discount Factor (γ): Try different values (0.95, 0.99) to see how it changes learning.
        - Experience Replay Buffer Size: Increase or decrease and track performance changes.



Hands-on: #3



DQN for Lunar Lander

1. Load the Lunar Lander Environment
    - Objective: Familiarize yourself with the environment's observation space, action space, and reward structure.
    - Tasks:
        - Replace any references to "CartPole-v1" with "LunarLander-v2".
        - Observe the new state vector (8-dimensional for Lunar Lander) and how many actions are available (4 discrete actions in the default environment).
2. Adjust the Network Architecture
    - Objective: Ensure the network can handle higher-dimensional state input.
    - Tasks:
        - Check your DQN's input layer size—now it should match 8 (the dimension of the observation in Lunar Lander).
        - Keep or modify hidden layer sizes as needed (e.g., 64–128 units).
        - The output layer should have 4 neurons (one for each discrete action).
3. Tune Hyperparameters
    - Objective: Because Lunar Lander is more complex, you may need different training parameters.
    - Tasks:
        - Increase the number of episodes (Lunar Lander often needs more training time).
        - Consider adjusting learning rate, epsilon decay schedule, batch size, or buffer size.
        - Track the average reward over time to see if it eventually stabilizes around successful landings.
4. Observe the Agent's Performance
    - Objective: Determine whether the agent successfully lands (score ≥200 is typically considered solved).
    - Tasks:
        - Log your returns over training episodes.
        - Plot your results (e.g., rolling average of episode rewards).
        - Optionally, record a short video or run a few test episodes after training to visualize the landings.