**DSE 3260 ARTIFICIAL INTELLIGENCE LABORATORY**

**WEEK 1 Representational Learning using Autoencoders.**

1. **Use the Fashion MNIST data set and train a basic autoencoder to reconstruct images.**
   Train the model using x_train as both the input and the target. The encoder should learn to compress the dataset from 784 dimensions to the latent space, and the decoder should learn to reconstruct the original images.
   Let the autoencoder have two Dense layers:
   a. an encoder, which compresses the images into a 64 dimensional latent vector, use reLU as activation function.
   b. and a decoder, that reconstructs the original image from the latent space, use sigmoid as activation function.

   Once the model is trained for at least 10 epochs , with loss function as mean square error, test it by encoding and decoding images from the test set. Display original and reconstructed images.

2. **Train an autoencoder to detect anomalies using the ECG5000 dataset.**
   Plot and observe a normal ECG vs an anomalous ECG.
   Design and Train an autoencoder on the normal rhythms only, then use it to reconstruct all the data.
   The autoencoder is trained using only the normal ECGs, but is evaluated using the full test set. Classify an ECG as anomalous if the reconstruction error is greater than one standard deviation from the normal training examples.

**WEEK 2: Simple Agents**

1. Take the TF-Agents Environments Tutorial:
   https://colab.research.google.com/github/tensorflow/agents/blob/master/docs/tutorials/2_environments_tutorial.ipynb

2. Use the **CartPole-v0** environment and write a program to
a. Implement the CartPole environment for a certain number of steps
b. Implement the CartPole environment for a certain number of episodes
c. Compare and comment on the rewards earned for both approaches.

3. plot the cumulative reward of each game and note down the

**WEEK 3: Problem Solving Agents & SEARCH**

**The Game :**

According to the "Six Degrees of Kevin Bacon" game, anyone in the Hollywood film industry can be connected to Kevin Bacon within six steps, where each step consists of finding a film that two actors both starred in. To solve the problem, find the shortest path between any two actors by choosing a sequence of movies that connects them. For example, the shortest path between Jennifer Lawrence and Tom Hanks is **2**:

Jennifer Lawrence is connected to Kevin Bacon by both starring in "X-Men: First Class," and Kevin Bacon is connected to Tom Hanks by both starring in "Apollo 13."

**Problem Solving Agent:**

Given two actors nodes in the graph we need to find the distance (shortest path) between the nodes.

Write a python program to determine how many "degrees of separation" apart two actors are. Find the distance or the degree of separation., using

a. Breadth first search
b. Depth first search

**Distribution Code:**

Data & Download the distribution code from:
https://cdn.cs50.net/ai/2020/x/projects/0/degrees.zip

The distribution code contains two sets of CSV data files: one set in the large directory and one set in the small directory. Use the small dataset for ease of testing and experimentation. Each dataset consists of three CSV files.

1. **small/people.csv:** each person has a unique id, corresponding with their id in IMDb's database. They also have a name, and a birth year.
2. **small/movies.csv:** You'll see here that each movie also has a unique id, in addition to a title and the year in which the movie was released.
3. **small/stars.csv:** This file establishes a relationship between the people in people.csv and the movies in movies.csv. Each row is a pair of a person_id value and movie_id value.

For example: The first row (ignoring the header), for example, states that the person with id 102 starred in the movie with id 104257. Checking that against people.csv and movies.csv, you'll find that this line is saying that Kevin Bacon starred in the movie "A Few Good Men."

4. **degrees.py**:

At the top, several data structures are defined to store information from the CSV files. The names dictionary is a way to look up a person by their name: it maps names to a set of corresponding ids (because it's possible that multiple actors have the same name). The people dictionary maps each person's id to another dictionary with values for the person's name, birth year, and the set of all the movies they have starred in.

And the movies dictionary maps each movie's id to another dictionary with values for that movie's title, release year, and the set of all the movie's stars. The load_data function loads data from the CSV files into these data structures.

The main function in this program first loads data into memory (the directory from which the data is loaded can be specified by a command-line argument). Then, the function prompts the user to type in two names. The person_id_for_name function retrieves the id for any person (and handles prompting the user to clarify, in the event that multiple people have the same name). The function then calls the shortest_path function to compute the shortest path between the two people, and prints out the path.

**The shortest_path function has to be coded using**

**a. Breadth First Search**
**b. Depth First Search**

**WEEK 4: Gaming Agent & Negamax search**

EasyAI is an artificial intelligence framework for two-players abstract games.
**http://zulko.github.io/easyAI/index.html**

Write a python program to define and implement a tic—tac-toe game with one human player. Solve the game using the built in algorithms and compare the solutions.

a. Iterative deepening
b. Depth first search

**WEEK 5: MULTI ARMED BANDITS IN TF-AGENTS**

1. Do the tutorial for MAB in TF-Agents

https://www.tensorflow.org/agents/tutorials/bandits_tutorial

2. Exercise 1 - Create a environment a. for which the observation is a random integer between -5 and 5, there are 3 possible actions (0, 1, 2), and the reward is the product of the action and the observation.

b. Define an optimal policy manually. The action only depends on the sign of the observation, 0 when is negative and 2 when is positive.

c. Request for 50 observations from the environment, compute and print the total reward.

3. Exercise 2 – Create an environment a. Define an environment will either always give reward = observation * action or reward = -observation * action. This will be decided when the environment is initialized.

b. Define a policy that detects the behavior of the underlying environment. There are three situations that the policy needs to handle:

- i. The agent has not detected know yet which version of the environment is running.
- ii. The agent detected that the original version of the environment is running.
- iii. The agent detected that the flipped version of the environment is running.
-

- c. Define the agent that detects the sign of the environment and sets the policy appropriately.
-

**WEEK 6 – MULTI-ARMED BANDITS – AD OPTIMIZATION**

Consider the given dataset "Ads_clicks" containing data about which add was clicked in each time step. Suppose an advertising company is running 10 different ads targeted towards a similar set of the population on a webpage. We have results for which ads were clicked by a user Each column index represents a different ad. We have a 1 if the ad was clicked by a user, and 0 if it was not.

A. Write down the MAB agent problem formulation in your own words.

B. Compute the total rewards after 2000-time steps using the ε-greedy action. a. for ε=0.01, ε= 0.3

C. Compute the total rewards after 2000-time steps using the Upper-Confidence-Bound action method for c= 1.5

D. For all approaches, explain how the action value estimated compares to the optimal action.

**WEEK 7 – MULTI-ARMED BANDITS- MOVIE RECOMMENDATION**

1. Write down the "Movie Recommendation" as a Reinforcement Learning problem formulation. Use comment lines for proper documentation.

2. For the environment, use the built-in TF-agent agent, MovieLensPyEnvironment (non-per-arm)

https://github.com/tensorflow/agents/blob/master/tf_agents/bandits/environments/movielens_py_environment.py

3. Compute the regret using the built-in metric in TF-agents

https://github.com/tensorflow/agents/blob/master/tf_agents/bandits/metrics/tf_metrics.py

4. Plot the regret against 20,000-time steps using the built-in agents: a. LinUCB b. LinTS c. NeuralEpsilonGreedy

And identify the best agent for the movie recommendation

5. Write the Recommendation policy, given a new observation request (i.e. a user vector), the policy will produce actions, which are the recommended movies

**WEEK 8 MDP & DYNAMIC PROGRAMMING**

Use the Frozen Lake environment.
https://www.gymlibrary.dev/environments/toy_text/frozen_lake/

Learn the optimal policy for the frozen lake environment using the Policy Iteration vs the Value Iteration technique.

1. Create a Policy Iteration function with the following parameters
    - policy: 2D array of a size n(S) x n(A), each cell represents a probability of taking action a in state s.
    - environment: Initialized OpenAI gym environment object
    - discount_factor: MDP discount factor
    - theta: A threshold of a value function change. Once the update to value function is below this number
    - max_iterations: Maximum number of iterations
2. Create a Value Iteration function with the following parameters
    a. environment: Initialized OpenAI gym environment object
    b. discount_factor: MDP discount factor
    c. theta: A threshold of a value function change. Once the update to value function is below this number
    d. max_iterations: Maximum number of iterations
3. Compare the number of wins, average return after 1000 episodes and comment on which method performed better.

**WEEK 9 MDP & MONTE CARLO METHODS**

Use the Cliff Walking Environment:
https://www.gymlibrary.dev/environments/toy_text/cliff_walking/

Learn the optimal policy using 500 episodes :

1. Monte Carlo ES (Exploring Starts)
2. On-policy first-visit MC control (for Ɛ-soft policies), for Ɛ = 0.1

Compare and comment on the performance of both methods in terms of number of steps needed to learn optimal policy and the number of episodes .