Artificial Intelligence
COMP 5600/ COMP 6600/ COMP 6600 - D01

Instructor: Dr. Shubhra ("Santu") Karmaker
TA 1: Souvika Sarkar
TA 2: Sanjoy Kundu
Department of Computer Science and Software Engineering
Auburn University
Spring, 2024

April 5, 2024

# Assignment #5
# Neural Networks and Markov Decision Process

## Submission Instructions

This assignment is due on **April 16, 2024, at 11:59 pm**. Please submit your solutions via Canvas (https://auburn.instructure.com/). You should submit your assignment as a PDF file. Please do not include blurry scanned/photographed equations, as they are difficult for us to grade.

## Late Submission Policy

The late submission policy for assignments will be as follows unless otherwise specified:

1. 75% credit within 0-48 hours after the submission deadline.
2. 50% credit within 48-96 hours after the submission deadline.
3. 0% credit after 96 hours after the submission deadline.

## Question 1 [50 points]: Neural Networks

- **[25 points]** For the softmax function defined below,

$$P(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_i$ is the dot product of the last layer, $\phi$ is the last layer activations, and (w,b) are the last layer weights and biases, i.e., $a_i = w_i^T \phi + b_i$, mathematically show that the derivative of $y_k$ will take the following form:

$$\frac{\partial y_k}{\partial a_j} = y_k(I_{kj} - y_j)$$

- **[25 points]** Given the following negative log-likelihood function defined as the loss function:

$$E(w_1, ..., w_k) = -\ln P(T|w_1, ..., w_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

  Where $T$ is the target matrix of the output class in a multi-class classification problem and $y_{nk}$ is the softmax function $y_{nk} = y_k(\phi_n) = \frac{\exp(w_k^T \phi_n)}{\sum_j \exp(w_j^T \phi_n)}$, mathematically show that the derivative of $E$ will take the following form:

$$\nabla_{w_j} E(w_1, ..., w_k) = -\sum_{n=1}^{N} (y_{nj} - t_{nj})\phi_n$$

## Question 2 [50 points]: Markov Decision Process

**Overview:** In this assignment, you will apply dynamic programming techniques to solve a simple Markov Decision Process (MDP) problem. MDPs are commonly used to model sequential decision-making problems, and dynamic programming is a powerful approach to finding the optimal policy and value function.

**Problem Statement:** Imagine a simple grid world with the following characteristics:

- The grid world is represented as a 25x25 grid.
- Each cell in the grid can be in one of three states: empty, blocked, or goal.
- The goal state is always in the leftmost point, i.e., the cell in the 1st row and 1st column.
- Empty cells can be traversed by an agent.
- Blocked cells are obstacles and cannot be traversed.
- The goal cell is where the agent should reach.
- The agent can move in four directions: up, down, left, and right. It cannot move diagonally.
- At each time step, the agent can choose to move in one of the available directions or stay in the same cell.
- The agent receives a reward of -1 for each time step it takes to reach the goal cell.
- The agent cannot leave the grid or move into blocked cells.

**Your Tasks:**

1. Define the MDP for the given grid world. Specify the state space, action space, transition probabilities, reward function, and discount factor.

2. Implement the policy iteration using dynamic programming algorithms to solve the MDP. Here is a simple pseudocode as a refresher:
    a. Start with an initial random policy.
    b. Implement policy evaluation to calculate the value function for the policy.
    c. Implement policy improvement to update the policy based on the value function.
    d. Repeat policy evaluation and improvement until the policy converges.
3. Visualize the results:
    a. Show the optimal policy found using both policy iteration and value iteration.
    b. Calculate and display the optimal value function.
4. Write a report describing your approach to the problem. It must contain the following:
    a. The choice of data structure to implement the MDP and a justification.
    b. The performance of a random policy. Compute it by greedily evaluating it and reporting the median (out of three trials) number of steps it took to reach the goal state from 3 different start states. Randomly choose these states for your experiments.
    c. The performance of an intermediate policy obtained during policy iteration. Compute it by greedily evaluating it and reporting the median (out of three trials) number of steps it took to reach the goal state from 3 different start states. Use the same start states as defined in the previous section for your experiments.
    d. The performance of the optimal policy found using your approach. Compute it by greedily evaluating it and reporting the median (out of three trials) number of steps it took to reach the goal state from 3 different start states. Use the same start states as defined in the previous section for your experiments.

Your code must be written in Python as an IPython Notebook and must be executable on Google Colab. You should write the report as text cells in the Colab notebook and submit a single IPython notebook as your solution for this assignment.