CS 240: Lab 6
Neural Networks &
8 Queen Problem

TAs: Soham Dahane, Dion Reji

Instructions

- This lab will be **ungraded** but marks for attendance may count.
- Please read the problem statement and submission guidelines carefully.
- For any doubts or questions, please contact either the TA assigned to your lab group or one of the two TAs involved in making the lab.
- The deadline for this lab is **Thursday, 6 March, 5 PM** but solutions till 5:30 PM will be accepted. No submissions will be accepted after 5:30 PM.
- The submissions will be checked for plagiarism, and any form of cheating will be penalized.

Part I: Neural Network Training

In this part you will continue working with the Neural Network training from Lab 5.

The objective of this part is to continue to build and train a neural network using sigmoid activation functions on all neurons except the input neurons on the MNIST data set as well as the XOR dataset. Use only **one** and **two** hidden layer for the XOR part.

1. **Preprocessed MNIST dataset:**
   MNIST is a widely used dataset of handwritten digits, containing 60,000 training and 10,000 testing grayscale images of size 28×28 pixels. Each image represents a single digit (0–9). The preprocessed data (flattening and normalization) is given in the template file.

   Input: Each sample (image) has 28×28=784 float values between 0 and 1 (after normalization)
   Output: 10 float values, each corresponding to the probability of the input being in a class (used to find the predicted class from 0-9)

2. **XOR (Odd Parity) function:**
   XOR takes two binary inputs and outputs 1 if odd number of inputs are 1s, and 0 otherwise.

   Input: Two binary inputs
   Output: A single binary value (1 if the number of 1s is odd, otherwise 0).
   Number of hidden layers: 1,2
   You also need to to interpret what these one and two hidden layers are doing and compare the two cases.

Part II: 8 Queen Problem

In this part, you are required to implement the 8 Queen problem as you did in your Midsem using the A* search algorithm (strictly A* and no other method). The objective is to position 8 queens on an 8×8 chessboard such that no two queens threaten each other. Recall that a queen in chess can move any number of squares horizontally, vertically, or diagonally across the board. To do this, start with an empty chessboard and keep adding queens one after another until all eight queens can be placed in a *non-threatening* fashion.

The following are the specific tasks you must address in your solution. Each requirement must be evident from your code and the accompanying comments.

1. **State Definition**
   Provide a precise definition of the state in the context of the 8-Queens problem. For example, a state could represent the positions of queens placed so far on the board, such as a list $[r_1, r_2, \ldots, r_k]$ where r_i is the tuple $\langle p, q \rangle$ representing the row (p) and cloumn of (q) of the ith queen. You are free to choose any other representation of the state as well.

2. **Operators Definition**
   Fully and accurately define the operators (actions) that transition between states. An operator can only involve placing a queen in a specific row of the next unoccupied column, ensuring that there are no conflicts with existing queens. Recall that removing a queen or moving a queen from one position to another are **NOT** valid operators.

3. **Cost and $g(\cdot)$ Functions**
   Define the cost function $\text{cost}(s, s')$, representing the cost of moving from state s to state s', and the path cost function, $g(s)$, which is the cumulative cost from the initial state to state s.

4. **Performance with $h(s) = 0$**
   Analyze and report the performance of your A* implementation when the heuristic function is $h(s) = 0$ (the baseline case). Report performance in terms of the number of nodes expanded.

5. **Performance with a Monotonic Heuristic**
   Propose a heuristic function $h(s)$ for the A* algorithm and evaluate its performance. Ensure that the heuristic satisfies the *monotone restriction* (i.e., $h(s) \leq h(s') + \text{cost}(s, s')$ for all states s and s'). Compare its performance against the baseline $(h(s) = 0)$.

6. **Comparison with an Alternative Monotonic Heuristic**
   Develop a second heuristic function that is either inferior or superior to the previous one and also satisfies the monotone restriction. Compare its performance with both the baseline $(h(s) = 0)$ and the previous one.

Submission

- Submissions should be made on Moodle. Submit a jupyter notebook renamed as `rollnumber1_rollnumber2.ipynb` (the "b" in roll number should be in small case). Add both part 1 and part 2 (and subsections of part 1) using proper markdowns and commenting in a single notebook.
- Penalty will be imposed on wrong file naming.
- The hard deadline for submission is 5:30 pm. No submission after that will be evaluated.
- Only one person per team should submit their solution.