CS601-Intro to AI

Assignment 2

**Your solution for this assignment should have two parts: a pdf document and coding files.**

* Have a single pdf document that shows your solution for different questions (show either numerical values if the question asks for it, and/or theoretical justification if required). Include in this pdf, the code you wrote for the solution.
* Upload your real coding files that you used to solve the question. Make sure your code is neatly organized per question, runs correctly, and has comments that highlight the part you implemented so that your TA and Instructor can easily understand it.
* Combine your solution pdf and code files in a single zip folder, rename it using your email id (e.g., the email id of account [qianrusun2024@mitb.smu.edu.sg](mailto:qianrusun2024@mitb.smu.edu.sg) is qianrusun2024) and upload it on the eLearn assignment folder. The deadline of submission is **2 Jul 11:59 pm**.
* Solution should be typeset using a professional software (word, keynote, latex, etc.) and saved in a pdf file. Figures should be made using drawing softwares. No handwritten solutions are allowed and will not be graded.

**Question 1 [20 points] ANN**

In this question, you will create an artificial neural-based classifier to classify the k\_mnist dataset which is available at https://[www.tensorflow.org/datasets/catalog/kmnist.](http://www.tensorflow.org/datasets/catalog/kmnist) More details about this dataset are here: https://github.com/rois-codh/kmnist.

You can build your solution on top of the python notebook covered in class to classify the standard (fashion\_)MNIST dataset. The solution you provide should perform the following tasks:

a) Download and load the k\_mnist dataset. [2 points]

b) Create an ANN with 1 input layer, at least one hidden layer with at least 2 nodes per layer. You can use relu or any other activation function for hidden layers. Feel free to create additional hidden layers or increase the number of nodes to maximize the final accuracy (it would require some trial and error). Design your neural network so that you get an overall accuracy of 80% or more on the testing dataset after training. Additionally, use advanced techniques like dropout or batch normalization (optional) to enhance model performance and prevent overfitting. Do your own research on these techniques and if you use some of them, please explain your choices. [4 points]

c) Create 1 output layer. What is the size of the output layer? What should be the activation function for the output layer? Also, explain the rationale behind choosing this activation function. [3 points]

d) Compile and train the neural network with the appropriate loss function (what should be the loss function type?) and optimizer (explain your choice of optimizer). Experiment with at least two different optimizers and compare their performance. Please note that we did not list all optimizers for you in the class but have introduced their idea of doing adaptive learning rate updates. Please do some research on your own for more details when working on this question. [4 points]

e) Plot the average training error (sum of the error over the training dataset divided by the total number of training examples) on the y-axis vs. the epoch number. Also, plot the validation error vs. epoch number on the same graph to compare training and validation performance. [2 points]

f) What is the final accuracy for different classes and overall accuracy on the testing data? Try to get an overall accuracy of 80% or more. Additionally, provide a confusion matrix for the classification results and analyze the types of errors your model makes (e.g., which classes are most frequently confused). [5 points]

For each of the above parts, also show the relevant code snippets in your solution pdf. Ensure that your solution is well-documented and includes comments explaining each part of the code.

For this question, please limit yourselves to using dense ANNs (rather than CNNs).

**Question 2 [20 points] RL**

In this question, we explore the application of Q-Learning for solving the classic Gridworld problem. The goal is to implement and understand Q-Learning and its application to navigating and finding the optimal path in a grid environment.

A. Environment Setup [5 marks]

A.1 Implement the following Gridworld environment using a suitable framework or custom code. Ensure the environment provides a clear representation of the grid. [5 marks]

Define a 5x5 Gridworld environment with the start state at (0, 0), the goal state at (4, 4), and obstacles at (2, 2) and (3, 3); Use the following defined rewards for reaching the goal state, penalties for hitting obstacles, and standard penalties for each step taken:

Reward = 1 if the goal state is reached.

Penalty = -0.1 for each step taken.

Penalty = 0 if hitting an obstacle or boundary.

B. Q-Learning Algorithm Implementation [10 marks]

B.1 Initialize the Q-table with appropriate dimensions based on the state and action spaces. Explain the initialization process. [2 marks]

B.2 Implement the Q-Learning algorithm. Ensure your implementation includes the following [5 marks]:

1. Exploration strategy (e.g., ε-greedy policy)
2. Learning rate (α) and discount factor (γ)
3. Update rule for the Q-values

B.3 Train the Q-Learning agent on the Gridworld environment. Monitor and record the learning process, including the number of episodes and steps taken to reach the goal state. Plot the cumulative reward per episode to visualize the learning progress. [3 marks]

C. Parameter Tuning and Analysis [4 marks]

C.1 Experiment with different values for the learning rate (α), discount factor (γ), and exploration rate (ε). Discuss how each parameter affects the learning process and the final policy learned by the agent. [3 marks]

C.2 Visualize and compare the policies learned under different parameter settings or exploration strategies. Provide a clear explanation of the differences observed. [1 mark]

D. Policy and Evaluation [1 mark]

D.1 Evaluate the optimal policy learned by the Q-Learning agent. Provide a clear visual representation of the optimal path from the start state to the goal state. [1 mark]

***Useful pointers for running on Colab:***

*1. Note that resizing the training and testing images will consume a relatively large amount of RAM. Please make sure the usage of memory is low before running your program otherwise your program may crash. What you can do is to restart runtime (Runtime- >Restart runtime) before running your program.*

*2. To use GPU for training, you will need to enable GPUs for the notebook (Runtime- >Change runtime type->Hardware accelerator->GPU). Please change the runtime type to CPU when you are not using GPU resources as GPUs will be prioritized for users who have recently used less resources.*

*3. If the Colab still runs out of memory, you can decrease the size of the training dataset. However, it might reduce your accuracy on the test dataset also.*