COMP3211:
Fundamentals of Artificial Intelligence

Homework Assignment 2

Release Date: Oct. 24, 2023

The HKUST Academic Honor Code

Honesty and integrity are central to the academic work of HKUST. Students of the University must observe and uphold the highest standards of academic integrity and honesty in all the work they do throughout their program of study. As members of the University community, students have the responsibility to help maintain the academic reputation of HKUST in its academic endeavors.

1 Introduction

In this homework, there are two problems comprising six questions in total. The focus is on k-means clustering as well as the A* algorithm, with an emphasis on their applications in image segmentation and maze searching.

- K-means based Unsupervised Image Segmentation (Problem I):
  - Develop your own k-means algorithm.
  - Apply the algorithm to different features.
  - Compare and analyze the results.
- A* Algorithm for Maze Searching (Problem II):
  - Develop your own A* algorithm based on provided heuristic functions.
  - Design your own heuristic function.

The submission deadline for this homework is **23:59:59, Nov 10, 2023**. Please start as early as possible, and feel free to discuss with us if you have any questions about your design and implementation. The late penalty is 5% per day.

This assignment should be solved individually. No collaboration, sharing of solutions, or exchange of models is allowed. Please, do not directly copy existing code from anywhere other than your previous solutions, or the previous master solution. We will check assignments for duplicates. See below for more details about the homework.

2 Preliminaries

2.1 Environment Setup

Students are required to do the following programming assignments with the anaconda environment, as introduced in tutorial1.ipynb on Canvas, or students can choose C++ to implement the code. Before running the code, dependencies are required to install. You should run

```
pip3 install matplotlib
```

or

```
conda install matplotlib
```

The code is written by the jupyter notebook, you can open the assignment using

```
jupyter notebook problem1.ipynb
```

2.2 Import Packages

We run the first code block to import the packages.

```
import torch
import numpy as np
import torch.nn as nn
import matplotlib.pyplot as plt
import csv
import random
```

NOTE: the anaconda environment should contain the above packages once you follow tutorial1.pptx. If you encounter a reported error like "matplotlib not found", just refer to Sec. 2.1 and install the "matplotlib" package.

Further, we use some advanced features from Torchvision. Please also import the following functions.

```
from torchvision.io.image import read_image
from torchvision.models.segmentation import fcn_resnet50
from torchvision.models.segmentation import FCN_ResNet50_Weights
from torchvision.transforms.functional import to_pil_image
```

NOTE: generally speaking, the utilities of the above functions are related to image processing, segmentation, and model loading. If you are curious about advanced usage, you can refer to the following:
https://pytorch.org/vision/stable/models.html#semantic-segmentation

2.3 Support Code Snippets (Python)

We provide the code pipeline and support code snippets. These snippets are provided with explanations. For example:

```
im = cv2.imread('./elephant.jpg') # Reads an image into BGR Format

im = cv2.cvtColor(im,cv2.COLOR_BGR2RGB)
original_shape = im.shape

print(im.shape)
plt.imshow(im) # as RGB Format
plt.show()

# Flatten Each channel of the Image
all_pixels = im.reshape((-1,3))
print(all_pixels.shape)
```

You **CANNOT USE** any advanced packages (e.g., ski-learn) for the implementation of K-means and A* algorithm.

2.4 Support Code Snippets (C++)

Similar to Python version, we also provide useful code snippets and explanations for you. For example:

```
// Read the image in BGR format
    cv::Mat im = cv::imread("./elephant.jpg");

    if (im.empty()) {
        std::cout << "Could not open or find the image" << std::endl;
        return -1;
    }

    // Convert the image from BGR to RGB format
    cv::cvtColor(im, im, cv::COLOR_BGR2RGB);

    // Print the shape/size of the image
    std::cout << im.rows << " x " << im.cols << " x " << im.channels() << std::endl;

    // Display the image in a window
    cv::namedWindow("Image", cv::WINDOW_AUTOSIZE);
    cv::imshow("Image", im);
    cv::waitKey(0);  // Wait for a keystroke in the window

    // Flatten each channel of the image
    cv::Mat all_pixels = im.reshape(1, im.rows * im.cols);
    std::cout << all_pixels.rows << " x " << all_pixels.cols << std::endl;
```

Similarly, you **CANNOT USE** any advanced packages for the implementation of K-means and A* algorithms.

2.5 Other Tips

Please note that for problem 1, we provide images and a feature file for you. For problem 2, we provide a TXT file for you. You need to show the required results in your report and provide the source code, as well as a markdown document to illustrate how to run your code. Some tips for C++:

- For the replacement of Numpy, you can refer to NumCpp (https://github.com/dpilger26/NumCpp).
- For the replacement of Matploit, you may use OpenCV for visualization.

3 Problem 1: K-means based Unsupervised Image Segmentation (Total: 55 points)

3.1 Implementation of K-Means Clustering (25 points)

Implement the K-Means clustering algorithm tailored for image segmentation based on attributes. Requirements:

- The number of clusters should be an argument of your algorithm, and by default, it should be 4 for Section 3.2.
- Initialize cluster centroids by randomly choosing from data.
- Assign each pixel to the nearest cluster centroid based on Euclidean distance.
- Recalculate the cluster centroids based on the mean value of the pixels assigned to each cluster.
- Repeat the assignment and update steps until the centroids do not change significantly (the total Euclidean distance between new and old centroids is smaller than 1e-3).
- Return the final cluster centroids and the labels for each pixel.

3.2 Toy Image Segmentation using Color Features (15 points)

For this problem, we provide a sample image titled "elephant.jpg". Your task is to perform k-means clustering on this image using the implementation from 3.1. Test with cluster values of K=2,4,6,8,10. Each pixel's RGB value should be taken as the input feature for the clustering.

3.3 Real-world Image Segmentation using FCN Features (15 points)

We provide a real-world image "dog.jpg" for you. First, use a Fully Convolutional Network (FCN)[1] to extract features from the image. Subsequently, run the K-means clustering algorithm on these extracted features. Again, try K=2,4,6,8,10.

Note: We implement the feature extraction in ipynb. If you want to choose other languages, you can also use the extracted features provided.

For Python Users: https://hkustconnect-my.sharepoint.com/:u:/g/personal/hche_connect_ust_hk/ERvyqiRMNv5DmfyWnxArf6cBAZrFnepaIe9ntG4M70VWDQ?e=PHQJZf

For C++ Users: https://hkustconnect-my.sharepoint.com/:u:/g/personal/hche_connect_ust_hk/ERp_AU5eUWxJr2WFMdcfDyABrLGjji9NO1EmnUmiSPQGaQ?e=u1AQ1W

[1] Long, Jonathan, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

4 Problem 2: A* for Maze Searching (Total: 45 points)

You are given a maze represented as a binary matrix. Your task is to write a program to navigate through this maze from the top-left corner to the bottom-right corner. In this question, you are required to implement the A* algorithm to find a route in a maze. The arguments of the program are:

- `-m maze`
- `-p path file`
- `-n nodes file`
- **Input file maze:** The first line contains the size (height and width) of the maze, which is then followed by a binary *height* × *width* matrix. A matrix element with the value '1' indicates that it is a wall, while a value of '0' indicates that you can move to that grid. An example is shown below:

```
0 0 1 1 1
1 0 0 0 1
1 0 1 0 1
1 0 0 0 1
1 1 1 0 0
```

- Your initial position is at the upper-left corner (cell (1; 1)), and the exit is at the lower-right corner, and the exit is at the lower-right corner (cell (height, width)).
- Outputs: Your program has to output two matrices of the same size as the maze in files. The first one contains the solution path, labeled with the value '2'. The second matrix contains all the expanded nodes, labeled with the value '3'. In both files, the first line should be your student ID. The following is an example with h = 0.

```
(path file)
10123456
22111
12221
10121
10021
11122
(nodes file)
10123456
33111
13331
13131
13331
11133
```

4.1 Implement A* Algorithm (25 points)

Implement your A* algorithm with two heuristic functions:

$$1.h_1((x,y)) = 0 \quad and,$$
$$2.h_2((x,y)) = height - x + width - y$$

4.2 Heuristic Function Design (20 points)

Design another heuristic h3((x, y)), and prove that it is admissible. Run the A* algorithm with your h3((x, y)). For this part, you will (i) get a full mark if your solution is correct and better than h1 in part a) (ii) get zero mark if your heuristic is not admissible; the output is incorrect; is worse than h1 in part a), or; is identical to h1 or h2.

4.3 Machine Execution for Grading

In this question, we will use programs to grade the functions you provide. Please organize your final submission of the program in the extra .py/.cpp file. The program's name should be problem2.py/.cpp, and it can accept the argument and return the required results, as mentioned in the previous section.

5 Submission

Upon completion, students are required to consolidate their code and a comprehensive report detailing the results for each problem. The code file and the report should be packaged together and uploaded for evaluation. Required results should be shown clearly. Name any files in the format **YourStudentID_name.*** and upload them to Canvas. Note that any code error would lead to 0 points for the respective question.

Submission Examples: A student with student ID 20000000 used Python for the first problem and C++ for the second problem should submit the following:

- 20000000_report.pdf
- 20000000_problem1.ipynb
- 20000000_problem2.cpp
- 20000000_README.md

A student with student ID 20000000 used C++ for the first problem and Python for the second problem should submit the following:

- 20000000_report.pdf
- 20000000_problem1.cpp
- 20000000_problem2.py
- 20000000_README.md