Geometric Machine Learning Algorithms

Problem Set 1

The purpose of this homework is to make sure you have installed python and run the right packages successfully. In particular, you will need to review lecture 0 and see the packages and the links there that I told you to install. The mains ones are going to be

1- ScikitLearn
2- Networkx
3- Numpy
4- Matplotlib

All those packages come with Anaconda. If you have already installed Anaconda except networkx you will not need to do a lot of work. To install networkx you go here : Networkx

We have two questions for the homework :

Problem 1

In this problem you will need to load the file "color_quantization.py" quantization that is given to you. This is mainly the code that of the color quantization that we learned in lecture 1. At this point we do not need to understand the details of the code but you are welcome to try. What you will need to do is to change the following command :

```
loaded_image =mpimg.imread('C:/Users/steve/Dropbox/damas.jpg')
```

Which basically loads the image from the hard drive. In the case above I am loading an image from my local folder "C:/Users/steve/Dropbox/damas.jpg'". You will need to change that folder to your local folder where you have the image. Please choose a non-personal image that you are conformable sharing Do not choose your face photo and any body's face photo or any copy-righted photo—choose something that is appropriate and suitable. Maybe take a picture with your phone to a tree or a car. Try to choose a relatively low resolution so that the code runs without any trouble. Run the code on that image after putting it on your harddrive. The program will generate three images as output : the original image, the Quantized image with K-means and the Quantized image with random color selection. Save those image on the homework folder. So for this problem, you are required to give the three images output of the image you select—that's all!

Problem 2

In this problem you will need to implement Kruskal's algorithm to find the minimal spanning tree of a graph. When you finish your implementation you will need to run in on an input graph I am giving to you in file " "graph_example.py".

I am giving you for this homework two files :

1-"graph_example.py" has a graph example using networkx. This will get you started using the networkx library. We do not need much beyond knowing how to create a graph, how to add a node and edge to that graph. In this file I am loading a graph example that comes with the package and then list its nodes and edges. I then assigned a weight to the edges of that graph. If you run that file you will see that it prints to you the list of nodes, list of edges and their weights.

2- The second file "unionfind.py" is a simple implementation for union-find data structure. You will need this file in order to easily implement the Kruskal's algorithm

After you make sure that the first file is running correctly (it requires networkx) you will need to go ahead and implement the Kruskal's algorithm. As I explained in class the algorithm is a minimal spanning tree algorithm so you will need to create a graph object using networkx and then start adding the vertices and the edges from an input graph with the help with union find data structure in order to create the final tree.

A tutorial on how to create a graph and add nodes and edges for networkx can be found here.

I am also giving you a file named "kruskalsalgorithm.py" in case you want to a template to do an object-oriented implementation for the algorithm (which is highly recommended).

After you finish your implementation make sure you run the algorithm you implemented on the graph example provided in "graph_example.py". Record the list of final minimal spanning tree you obtain in the same file and save it.