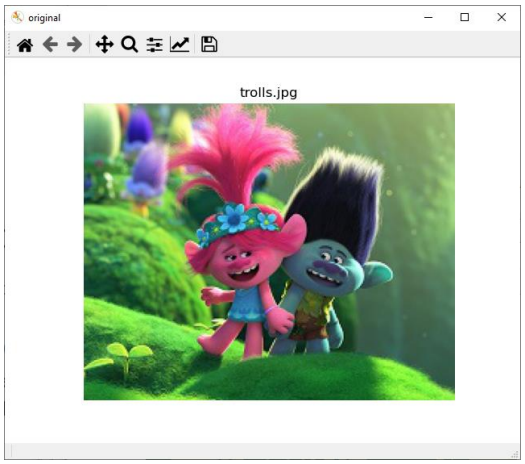
COMP10200: Assignment 5 – Part 1

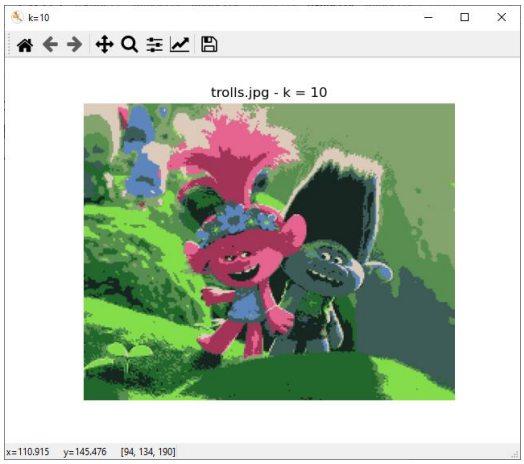
Overview

This assignment is about machine learning techniques beyond supervised classification. You will explore unsupervised clustering in part 1, and regression in part 2.

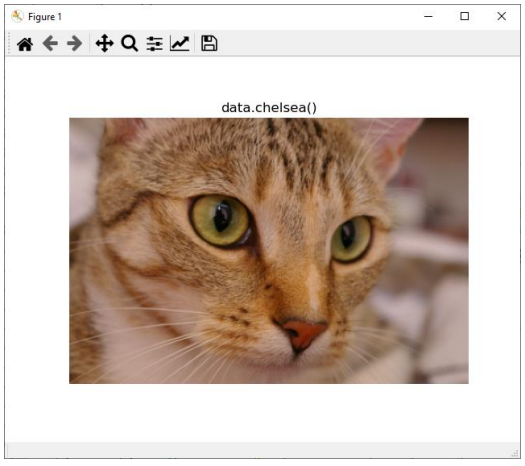
Part 1

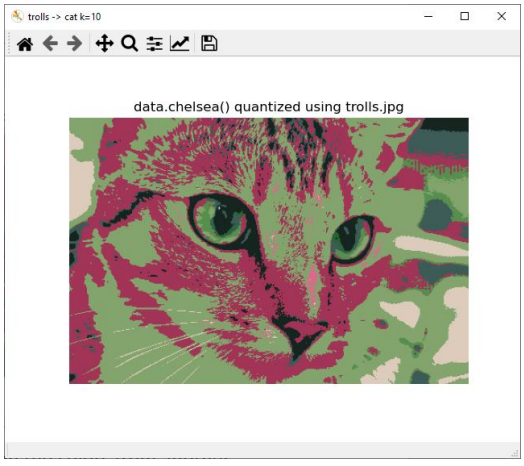
For this part of the assignment, you will use the k-means clustering algorithm to perform a color quantization of an image. In color quantization, you reduce the number of colors in an image to a fixed number. In the example below, the original image is on the left, and the image on the right has been quantized to 10 colors.





You can also do some cool special effects by quantizing one image based on the colors from another image. In the example below, the original image is on the left. The image on the right was quantized using the 10 color values identified in the trolls.jpg example above.





Quantizing with k-Means

To quantize with the k-Means algorithm, you reduce each image to a list of pixels. Each pixel has 3 values (red, green, and blue). So if your image is 100x200 pixels, your data will have 20000 rows and 3 columns. That's your data.

Then you can use k-Means clustering to cluster the colors in the image. If you use k=10, you will end up with 10 centroids, each of which represents the rgb values of a color. You can create a new image map by replacing each pixel with its closest centroid.

You can also use the kMeans predict method in sklearn to quantize a completely different image (like the cat example above).

The Data

Choose any two images, but don't use the skimage built in samples. Keep in mind that very large images might lead to long processing times. You can use paint or a similar program to resize the images. Reducing the image to half size will cut the number of pixels by 4.

The Code

The code you use for this part of the assignment should be written by you in Python using sklearn, matplotlib, numpy, and the skimage library.

SKImage

The SKImage library makes it easy to load and manipulate images using numpy. See the **skimagetest.py** code on Canvas for the basics of loading and manipulating images. Each image will be a 3D array. You can think of it as a 2D array of pixels, but each pixel is an array of 3 values for red, green, and blue (each with a value ranging from 0 to 255).

You will need to reshape the array. For example, suppose you have an image variable holding a 100x200 pixel image.

```
image.shape → (100, 200, 3)
```

What you want for clustering is an array of pixels that is 100 x 200 = 20 000 rows and 3 columns. Fortunately you can do this easily:

```
image = image.reshape((100*200, 3))
```

Then you can do the clustering and quantization. When you're ready to display the image, reshape again.

```
Image = image.reshape( (100, 200, 3) )
```

Believe it or not, that will work just fine. Numpy is smart enough to figure out the best way to do the reshaping for you, and this best way matches what you need for this application.

The First Task

Your first task is to look for a "natural" quantization of the image. Quantize it using k-Means for a variety of k values, show each quantized image, report the inertia for each run, then graph the result and try to identify an "elbow".

An example output and graph is shown in the appendix. For each value of k, make sure you configure sklearn to try at least 3 different runs of k-Means before selecting the best one. This is to make sure that you are not reporting the result of the algorithm getting stuck in a bad configuration.

When you identify an elbow, note it in the comment header at the top of your code, and then use the k value you identified as the elbow for the second task. In some cases there might be no obvious elbow. In that case, use a k value that is most appealing to you.

The Second Task

Once you have identified a good k value from the first task, run k-Means again for that k value, and this time use the colors to quantize a different image. Show the before and after version of the new image as well.

Handing In

The code file you hand in should automatically perform Task 1 followed by Task 2 when it is run. The header comment should contain your name and the information required from each task. An example is shown below.

Zip up your code and your data files for both parts of the assignment and hand them in together. It should be possible for me to unzip your folder and run your code easily (i.e. without having to move files around or make changes to the code). Simply running your code file with Ctrl-Shift-E should be enough for me to see all text and graphical results.

See the drop box for the exact due date.

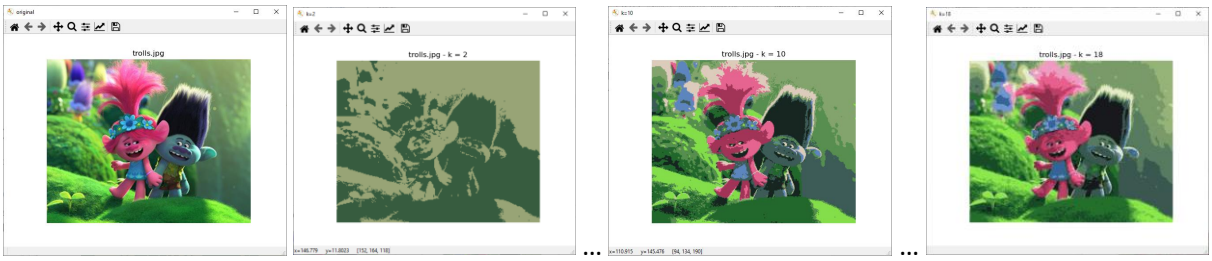
Evaluation

This assignment will be evaluated based on the coding rubric shown in the drop box. To get full marks, you must follow the documentation standards for the course, in the Student Resources section of Canvas.

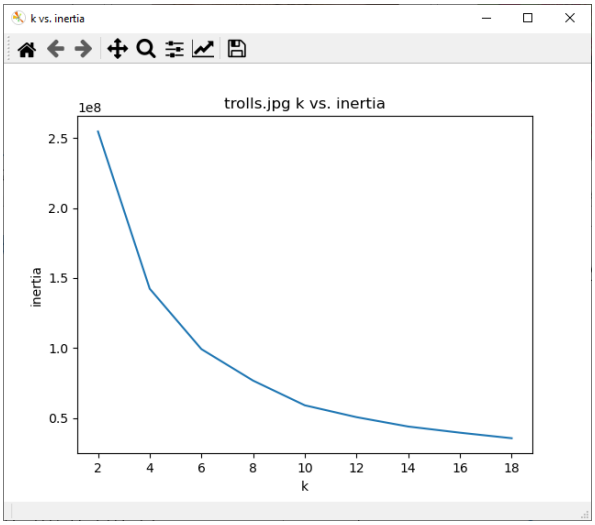
Appendix: Example Run

Original and Quantized Images

Note: Not all images in the output are shown below.



Sample Output and Interpretation, Task 1




```
k = 2    SSE = 254746304.31172678
k = 4    SSE = 142364456.07686552
k = 6    SSE = 99175680.8759278
k = 8    SSE = 76649084.8013816
k = 10    SSE = 59004006.87606526
k = 12    SSE = 50535158.30232017
k = 14    SSE = 43829888.31420271
k = 16    SSE = 39432438.40932504
k = 18    SSE = 35483407.70821022
```

For the above data, there is no obvious “elbow”. Maybe there’s a little bit of an elbow at k=4 or k=6, but it’s a bit of a stretch. Other runs of the code turned up slightly different looking graphs. Because there was no obvious elbow, I used k=10 for task 2, which is the quantization I liked best.

Sample Output, Task 2