1. Image color reduction with $k$-means

Explain how $k$-means could be used to reduce a $w \times h$ RGB image to the “closest” matching 5-color image, as in the figures below.

The left image is the full-color RGB image. The right image has only 5 colors.

a. How many vectors are input to $k$-means?

b. What is the dimension of each vector?

c. What is the value of $k$ here?

d. Describe mathematically how the output image could be generated from the $k$ means.
2. Running $k$-means on single clusters

Consider running $k$-means with $k=2$ on a set of points uniformly distributed within an ellipsoidal shape. You can actually perform this experiment by downloading the `kmeans.zip` code from the web page and running it on the `ellipse.txt` dataset.

a. Why does $k$-means reveal a similar first principal component to what PCA would discover? What is the mathematical relationship between the two means and the first principal component?

b. Would running with $k = 4$ recover the second component as well? Why or why not?

3. Texture classification with textons

Download the `textons.zip` distribution from the course website, and reproduce the results that we saw in class by running the three commands

```
python textons.py 4 lm4_filters.js synth_easy_dataset.js
python textons.py 4 lm4_filters.js synth_hard_dataset.js
python textons.py 4 s_filters.js synth_hard_dataset.js
```

They should get 100%, 50% and 100% accuracy, respectively.

a. Run the command

```
python textons.py -4 lm4_filters.js synth_hard_dataset.js
```

What does setting $k$ to a negative number tell the program to do? (See `readme.html` for details.) How does this affect the appearance of the textons (versus positive $k$), and why?

b. Why do you think running

```
python textons.py -4 s_filters.js synth_hard_dataset.js
```

fails to achieve 100% accuracy on the test set, even though running with a positive number $k$ works fine?