Homework 8

No submission

Instructions: This homework consists of a reading assignment and two coding exercises. It is **optional**; there is no submission. We strongly encourage students to experiment with these questions and refer to the solutions, which will be published before the final lecture.

Reading Assignment

- Read Sections 13.2 and 14.5 in Elements of Statistical Learning.
- Review Lecture 8.
- Review Lab 8 (available from the Course Materials page on Canvas).

Exercise 1

In this exercise we will explore the k-means algorithm in the context of *color quantization* for images. Color quantization is a lossy compression method by which the pixel-wise color values of an image are constrained to lie in a small set. Then, rather than store a full array of 3-dimensional color values, we need only to store the reduced set of colors plus an integer array indicating the color taken by each pixel.

The main problem in color quantization lies in selecting the reduced set of colors. We can frame this as a clustering problem and solve it with the k-means algorithm. In particular, we use the cluster centers as the reduced set of colors, and assign each observation (i.e. pixel) to take the color of the cluster to which it is assigned. We return an array of the same size as the original image, but for which each pixel takes one of just k colors.

(a) Load and visualize the five images provided for this homework. You can load a `.jpg` image in Python by running

```
from matplotlib import image
img = image.imread(path)
```

where `path` is the path to the image file. You can visualize the image with

```
import matplotlib.pyplot as plt
plt.imshow(img)
```

(b) Each image is a 3-dimensional array. The first two dimensions represent the height H and width W of the image, respectively, while the third dimension contains the RGB color values of the image at a given pixel location. Reshape the images to 2-dimensional arrays of size $HW \times 3$. Report the number of unique colors in each image.

(c) Implement `random`, `k-means++`, and `LBG` initialization schemes for k-means clustering. The `random` initialization simply draws three observations from the dataset by selecting uniformly at random - you may use, for example, `np.random.choice()`. For `k-means++`, you may use the `sklearn.clustering` function `kmeans_plusplus()`. For `LBG` you may re-use code from the lab. In each case, your initialization scheme should return a $k \times 3$ array, with the initial cluster centers given by the rows of the array. This can then be passed to the algorithm in part (d).

(d) Implement k-means color quantization. The inputs to the algorithm are a reshaped image array, a positive integer k, and an initialization method for k-means. The algorithm performs the following steps:

- Instantiate an `sklearn` k-means model with the given initialization and choice of k. Also set `n_init = 1` and `algorithm = 'full'`.
- Fit the k-means model to the data.
- Retrieve the cluster centers and labels.
- Generate an output array of the same dimension as the input data. In place of each observation, substitute the value corresponding to the cluster center of the cluster to which the observation has been assigned.
- Return the output array.

(e) Choose one of the images above, and run k-means color quantization using `random`, `k-means++`, and `LBG` initialization schemes. Compare the results to each other and to the original image.

(f) Compare the accuracy and time of k-means quantization across these choices of initialization and for $k = 2^j$, with $j = 1, 2, 3, 4, 5$. Accuracy can be reported as the norm of the difference between the quantized and original image. Repeat each trial $n = 20$ times to assess the variation in the results. Plot the mean accuracy and time for each choice of initialization as a function of k. Plot error bars reporting the standard error of each estimate.

Exercise 2

Prepare for the deep learning lecture by reading Section 19.5 of Dive Into Deep Learning. On your Azure portal, find a VM image with hardware specifications similar to the AWS instances listed in the section introduction.