# Assignment 6

Author

Torin White - 657467127

## Q1 [70 pt, Ex 7.1 of Alpaydin]

In image compression, $k$-means can be used as follows. Suppose all the $N$ images are sized $n$ pixels by $n$ pixels, and each pixel is represented by $d = 24$ bits. The image is divided into nonoverlapping $c \times c$ windows and these $c^2$-dimensional vectors make up the sample. Suppose $c$ wholly divides $n$. For a given $k$, which is generally a power of two, we do k-means clustering. The reference vectors and the indices for each window are sent over the communication line. At the receiving end, the image is then reconstructed by reading from the table of reference vectors using the indices.

### (a) [50 pt] Write the pseudo-code that does this for different values of $k$ and $c$. Your code only needs to cover the construction of the training set for clustering, followed by the k-means clustering algorithm. No need of sending and receiving/reconstructing the images.

```
# set initial variables
X = [] # empty list
k =    # prototypes, power of 2
N =    # images
n =    # number of pixels height and width (square), divisible by c
c =    # divisions of n to create windows/patches

# Step 1: Construct dataset X
FOR i = 1 ... N-1             # iterate over N images
    FOR r = 1 ... n/c - 1     # iterate over rows of image N
        FOR t = 1 ... n/c -1  # iterate over cols of image N
            ADD coordinate r, t to X

# Step 2: Construct prototype
v = random.sample(X, k)       # randomly select coordinates of X of k # of values for
b = []                        # initialize empty list to hold labels of closest vi

# loop will run until values of vi...k converge
REPEAT
    # loop over elements of X and compute closest value in vector vi ... k-1
    FOR i = 1 ... len(X)-1        # iterate over dataset X
        COMPUTE distance between Xi and centroid values in v
        COMPUTE argmin of distances between Xi and vi...k-1
        ADD result of argmin value to list b as label for Xi

    # find new centroid for vi...k-1 by taking mean of labels in b
    FOR j = 1 .... len(k)-1
        COMPUTE mean of values of X of jth element of b and reassign to jth value of v

UNTIL vi ...k converge

#Step 3: Sending algorithm
    NOT PART of Q1.a

# Step 4: Reconstruction on receiving end
    NOT PART of Q1.a
```

### (b) [20 pt] Calculate the compression ratio in terms of $n, c, k, N, d$

$n$ = integer divisible by c
$c$ = # of divisions of n to form n/c x n/c windows
$k$ = # of clusters, power of two
$N$ - # of images
$d$ = bit depth = 24

orginal image file size = $pixels * d$

for N images of $nxn = N * n^2 * d$

compressed file size = $(\frac{n}{c})^2 log_2 k + kd$

$compressionratio = \frac{uncompressedimagesize}{compressedimagesize}$

$compressionratio = \frac{Nn^2d}{(N\frac{n}{c})^2 log_2 k + kd}$ *where k*d is the space needed to save the prototype/decoder values

# Q2. Given a set of three points, –2, 0, and 10, we want to use k-means clustering with $k == 2$ to cluster them into two clusters.

### (a) [20 pt] If the initial cluster centers are C1 = –4.0 and C2 = 1.0, show two iterations of k-means clustering,indicating at each iteration which points belong to each cluster and the coordinates of the two new cluster

centers. In other words, fill in the tables below.

**Iteration 1:**

| Point | Cluster (C1 or C2) |
|---|---|
| -2 | C1 |
| 0 | C2 |
| 10 | C2 |

New cluster centers:

C1 = -2 C2 = 5

**Iteration 2:**

| Point | Cluster (C1 or C2) |
|---|---|
| -2 | C1 |
| 0 | C1 |
| 10 | C2 |

New cluster centers:

C1 = -1 C2 = 10

**(b) [10] True or False: In general, a good way to pick the best number of clusters, k, used for k-means clustering is to try multiple values of k and select the value of k at the "elbow" of the monotonically decreasing curve of reconstruction error as a function of k. Briefly explain why.**

**TRUE** The choice of k can be subjective and greatly affect the accuracy of the clustering algorithm. When plotting reconstruction error with the elbow method, choose the K where teh curve starts to level out. After it levels out, larger values of K (more clusters) do not greatly improve results/accuracy to justify the increasing computational needs. Also, there may be cases where increasing k decreases reconstruction error at first but than larger values of k increase reconstruction error again. Hence, the "elbow" visualization of choosing k at the crook of the elbow aka where reconstruction error is the least.

## Q3. [30 pt] Exercise 7.7 of Alpaydin. Just briefly explain in 3-5 lines. How can we do hierarchical clustering with binary input vectors—for example, for text clustering using the bag of words representation?

In bag of words representation we start with a vocabulary vector $v$. Then, examining N documents we create a binary vector representing each document of length $v$ representing the presence (1) or abscence (0) of that word in $v$. For hierarchical clustering with binary vectors such as bag of words, we can compare each index between binary vectors of any two documents, subtracting one vector from another. Where the indices match, the result will be 0, where they do not, the result will be 1. Summing the results of that subtraction we find that lower sum results (aka shorter distances) share more words in common from vocab v, whereas larger sum results (aka larger distances) are more dissimilar. The we can apply hierarchical clustering to accomplish text clustering of documents that are similar to one another.

## Q4. [30 pt] True or False: in Q2, k-means clustering is guaranteed to find the same final clusters for the above three points, no matter what the initial cluster center values are. Here we assume the initial cluster centers ensure that neither of the clusters becomes empty in the process of running k-means. Briefly explain why.

**TRUE** for this specific example in Q2 k-means clustering with the same 3 data points and k=2 assumming no clusters become empty will always converge to one cluster with {0, -2} and another with {10} because the distance between 10 and 0 and 10 and -2 is significantly larger than compared to 0 and -2. **However,** k means clustering is sensitive to the initial random values of k so in general for other applications this may result in different clusters across runs with different initial cluster center values. With more data, it is more likely to find a fairly stable solution, but there are also chances depending on the data that the algorithm never converges.