CS 376A
Digital Image Processing

03 / 01 / 2023

Instructor: Michael Eckmann

Today's Topics

- Questions? Comments?
- K means clustering examples based on 3 dimensions: R, G, B
- More clarity on L1, L2, and Earth Mover's Distance and the drawbacks of L1 and L2 on some kinds of data
- Revisit the K-means clustering algorithm and class discussion on designing code for the algorithm

Clustering

- Reminder … we could have n dimensions of features to cluster based on similarity among all, each dimension could be one measure on the pixel. e.g.
  - texture
  - Red color
  - Green color
  - Blue color
  - intensity
  - etc.

Clustering

- First we decide on the n features to segment on and K
- The algorithm randomly chooses K means (of those n features)
- repeat
  - For each pixel, compute its n features and compare the distance between those n features and each of the K means. Assign that pixel to the group whose distance is smallest.
  - If no pixel changes groups (the means will have stayed the same from last iteration) then stop. Otherwise
  - Update the K means based on the pixels currently in each group and repeat

Clustering

- Can think of each mean as a centroid in n-dimensions
- K-means is a greedy algorithm
- Results clearly depend on chosen K
- Results depend on initially chosen means
  - standard solution (according to some) is try various different initial means and then pick your favorite result

Let’s see some results

- Let's see some results of doing k means clustering on images using 3 features (R, G and B)
- The resulting labels are shown as the mean RGB value of the cluster
- Let’s run my code on more images and different values of k

Distance

- Distance between 2 n-dimensional vectors
- L1 aka Manahattan distance aka city block distance
  - Sum of absolute differences
- L2 aka Euclidean distance
  - Square root of the Sum of the differences squared
  - Square root may not be computed in practice (why?)
    - Typical use is to find a pair of n-dimensional vectors that are closest (least distance)
- Earth mover's distance
  - Useful when dimensions are ordered by similarity (e.g. a histogram of grayvalues)

Distance

- Suppose comparing colors of pixels in R, G, B (but say they are in the range 0-1.0 rather than 0-255

- An orange pixel might have: R,G,B = (0.88, 0.32, 0.0)
- A red pixel might have: R,G,B = (0.91, 0.05, 0.05)
- A blue pixel might have: R,G,B = (0.07, 0.07, 0.78)

- L1(orange, red) = |0.88 - 0.07| + |0.32 – 0.05| + |0.0 – 0.05| = 0.35
- L1(orange, blue) = |0.88 - 0.07| + |0.32 – 0.07| + |0.0 – 0.78| = 1.84
- That says orange is further away from blue than it is from red

- L2 would produce similar results but uses sqrt of the sum of the differences squared, whereas L1 uses sum of absolute value of differences

Distance

- Suppose we have histograms of grayvalues (256 bins)

- A black image: (**1.0**, 0.0, 0.0, …, 0.0, 0.0)
- An almost black image: (0.0, **1.0**, 0.0, …, 0.0, 0.0)
- A white image: (0.0, 0.0, 0.0, …, 0.0, **1.0**)

- L1(black, almost black) = ?
- L1(black, white) = ?
- L1(almost black, white) = ?
- L2(black, almost black) = ?
- L2(black, white) = ?
- L2(almost black, white) = ?

Distance

- Suppose we have histograms of grayvalues (256 bins)

- A black image: (**1.0**, 0.0, 0.0, …, 0.0, 0.0)
- An almost black image: (0.0, **1.0**, 0.0, …, 0.0, 0.0)
- A white image: (0.0, 0.0, 0.0, …, 0.0, **1.0**)

- L1(black, almost black) = 1
- L1(black, white) = 1
- L1(almost black, white) = 1
- L2(black, almost black) = 1
- L2(black, white) = 1
- L2(almost black, white) = 1

Distance

- Suppose we have histograms of grayvalues (256 bins)

- A black image: (**1.0**, 0.0, 0.0, …, 0.0, 0.0)
- An almost black image: (0.0, **1.0**, 0.0, …, 0.0, 0.0)
- A white image: (0.0, 0.0, 0.0, …, 0.0, **1.0**)

- In an L1 or L2 sense all three of those are equally far apart from each other w/ distance = 1
- However, EMD will compute a distance that shows the black and almost black images to be much closer (less distance) than the almost black and the white image

Distance

- Suppose we have histograms of grayvalues (256 bins)

- A black image: (**1.0**, 0.0, 0.0, …, 0.0, 0.0)
- An almost black image: (0.0, **1.0**, 0.0, …, 0.0, 0.0)
- A white image: (0.0, 0.0, 0.0, …, 0.0, **1.0**)

- EMD(black, almost black) = 1.0 * 1 = 1 because we move 1.0 amount of "dirt" over by 1 space
- EMD(white, almost black) = 1.0 * 254 = 254 because we move 1.0 amount of "dirt" over by 254 spaces
- EMD says that black and almost black are closer than white and almost black

Distance

- Earth Mover's Distance wouldn't be appropriate for certain data.
- Example, say we have data with the following 3 feature values describing a pixel:
  - Intensity
  - Texture (maybe measured as Standard Deviation of Intensity in a neighborhood)
  - Hue
- The ordering of the 3 features doesn't matter as they are all independent
- Whereas ordering 0, 1, 2, …, 255 for gray values makes sense --- nearby bins are more similar than bins further away
  - This fact makes EMD appropriate for this type of data

Clustering

- Do a worksheet on K means

- First we decide on the n features to segment on and K
- The algorithm randomly chooses K means (of those n features)
- repeat
  - For each pixel, compute its n features and compare the distance between those n features and each of the K means. Assign each pixel to the group whose distance is smallest.
  - If no pixel changes groups (the means will have stayed the same from last iteration) then stop. Otherwise
  - Update the K means based on the pixels currently in each group and repeat

Coding K-means

- Let's design code to handle doing k-means
  - should allow any kind of and any number of features
  - each pixel will have features and we wish to group them into k classes
  - how to represent a pixel belonging to a certain class?
  - I already wrote code to do k means clustering (you saw that code running), but instead of me telling you how I designed it, I'd like to hear your design ideas before we start coding them up
  - thoughts?