Homework 2

36-350: Data Mining

Due at the start of class, Friday, 19 September 2008

1. Quantize the (R,G,B) cube with 8 prototypical colors:

Color	Red	Green	Blue	Color	Red	Green	Blue
Black	0.25	0.25	0.25	Red	0.75	0.25	0.25
Green	0.25	0.75	0.25	Yellow	0.75	0.75	0.25
Blue	0.25	0.25	0.75	Magenta	0.75	0.25	0.75
Cyan	0.25	0.75	0.75	White	0.75	0.75	0.75

   That is, each pixel's RGB vector gets replaced by that of the closest prototypical color.

   Take a six-pixel image, with RGB values (0.22,0.37,0.8), (0.19,0.8,0.19), (0.6,0.1,0.05), (0.8,0.3,0.22), (0.7,0.32,0.8), and (1,0.4,0.34). What is the bag-of-colors representation of the image? What is the representation after norming by Euclidean length?

2. Suppose we search for the image in question 1 via the color (0.7,0.2,0.2). What is the bag-of-colors representation for this query, and what is the query's Euclidean distance from the image, after both have been divided by Euclidean length?

3. Describe a potential problem in measuring distance if we use too many prototypical colors in the representation (besides increased computation and storage). Describe a potential problem if we use too few prototypical colors in the representation.

4. Suppose we have a collection of 50 flower and 50 ocean images, and use just three prototypical colors, red, green and blue. The flower images have red and green but no blue, and the ocean images have green and blue but no red. What are the inverse-picture-frequency (IPF) weights for the three colors?

5. Suppose the query vector is $(1, 0, 0, 0)$ and there are two items in the database, with vectors $(3, 1, 1, 1)$ and $(3, 2, 0, 0)$ (these might be documents or images or something else). Which vector is closer to the query when we normalize by the sums? Which one is closer when we normalize by the Euclidean length? Does it matter whether we are talking about images or documents? Why or why not?

6. In a typical data mining application, a news agency wants to search through video archives and detect all frames depicting an event, e.g. fireworks at night, given some examples. You can regard a video as a sequence of images. How could this be implemented using similarity search? (Don't give code, just a brief description of the method.)

7. Recall that in **nearest-neighbor** classification, we guess that a new vector belongs to the same class as the closest perviously-seen vector whose class is known. In **prototype** classification, we represent each class by the average of the vectors belonging to that class, and assign new vectors to the class whose prototype is closests.

   (a) Write an R function to do nearest-neighbor classification. Your function should tkae as inputs the vector to be classified, and a data frame of labeled example vectors, and should give as its output the guessed label of the new vector. You can re-use code from last time and from the solutions. *Remember to comment your code, and explain the reasoning behind it.*

   (b) Test the accuracy of your nearest-neighbor classifier on the newsgroups data from the last assignment. That is, what fraction of documents does it mis-classify? (You should use IDF weighting and Euclidean-length normalization.) Include, with comments and explanations, the code you used to calculate the error rate.

   (c) A simple mistake in the previous part would lead you to conclude that nearest neighbor classification is *always* 100% accurate on *any* data set. Describe the mistake, and how to avoid it.

   (d) Write a function to do prototype classification. It should have the same inputs and outputs as your nearest neighbor classifier. (*Hint*: one way to do this is to use that function!)

   (e) Calculate the error rate of your prototype classifier.

   Note: there are a number of functions already in various R packages (e.g., `class` and `knnflex`) to do nearest-neighbor classification. Writing a function which just calls one is not acceptable[1]; however, you can use them to check whether your code is working properly.

[1] Yes, someone tried that before.