Assignment 4 - Clustering*

Due: Monday, March 5
Late assignments accepted (with full credit) until Wednesday, March 7
Turn in a hard copy at the start of class

Overview

In this assignment you will explore clustering: hierarchical and point-assignment. You will also experiment with high dimensional data.

You will use two data sets for this assignment:

- `http://www.cs.utah.edu/~jeffp/teaching/cs5955/A4/C1.txt`
- `http://www.cs.utah.edu/~jeffp/teaching/cs5955/A4/C2.txt`

These data sets each describe the location of 26 points, each on one line of the file. The first character is a label (from the lower case letters a,b,c,d,e,...). Then separated by white space are two numbers, the x and the y coordinate. We'll use L_2 distance to say which are close

$$d(a,b) = \sqrt{(a.x - b.x)^2 + (a.y - b.y)^2}.$$

The data sets are small enough that it may be possible to run the algorithms below *by hand* if you have less programming experience. However, it may also be useful (or faster) to implement the algorithms.

You are also asked to prove things in this assignment (even outside the BONUS question). The questions should be simple, and by playing with algebra you should be able to solve them. I suggest you work in groups to discuss the problems, but recall, that you must write up your solutions by yourself. And remember to make your proofs clear, they need to be verifiable to be graded as correct.

As usual, it is highly recommended that you use LaTeX for this assignment. If you do not, you may lose points if your assignment is difficult to read or hard to follow. Find a sample form in this directory: `http://www.cs.utah.edu/~jeffp/teaching/latex/`

1 Hierarchical Clustering

There are several variants of hierarchical clustering; here we explore 4. The key difference is how you measure the distance $d(S_1, S_2)$ between two clusters S_1 and S_2.

Single-Link: measures the shortest link $d(S_1, S_2) = \min_{(s_1,s_2)\in S_1\times S_2} \|s_1 - s_2\|_2$.

Complete-Link: measures the longest link $d(S_1, S_2) = \max_{(s_1,s_2)\in S_1\times S_2} \|s_1 - s_2\|_2$.

Average-Link: measures the average link $d(S_1, S_2) = \frac{1}{|S_1||S_2|} \sum_{(s_1,s_2)\in S_1\times S_2} \|s_1 - s_2\|_2$.

Mean-Link: measures the distances to the means. First compute $a_1 = \frac{1}{|S_1|}\sum_{s\in S_1} s$ and $a_2 = \frac{1}{|S_2|}\sum_{s\in S_2} s$ then $d(S_1, S_2) = \|a_1 - a_2\|_2$.

A (4 points): Run all hierarchical clustering variants on data set `C1.txt` until there are $k = 3$ clusters, and report the results as sets.

Which variant did the best job, and which was the easiest to compute (think if the data was much larger)? Explain your answers.

B (2 points): Prove that for points in $\mathbb{R}^1$ that Average-Link and Mean-Link are equivalent.

2 Point Assignment Clustering

Point assignment clustering works by assigning every point $p \in P$ to the closest cluster centers C. Let $\mu : P \to C$ be this assignment map so that $\mu(p) = \arg\min_{c \in C} d(p, c)$. All points that map to the same cluster center are in the same cluster.

Two good heuristics for these types of cluster are the MinMax (Algorithm 2.1) and k-Means++ (Algorithm 2.2) algorithms. They both incrementally build up sets of center points, where we let C_i be the set of first i centers we have picked. Note that $C_k = C$ is our output.

Algorithm 2.1 MinMax(P, k)

1: Choose some point $p \in P$ as c_1.
2: **for** $i = 1$ **to** $k - 1$ **do**
3: Let $c_{i+1} = \arg\max_{p \in P} \min_{c \in C_i} d(p, c)$

Algorithm 2.2 k-Means++(P, k)

1: Choose some point $p \in P$ as c_1.
2: **for** $i = 1$ **to** $k - 1$ **do**
3: **for all** $p \in P$ **do**
4: Calculate $w(p) = \min_{c \in C_i} d(p, c)^2$
5: Choose c_{i+1} from P proportional to $w(p)$ *(using $u_i \in [0, 1]$)*

A: (4 points) Run MinMax and k-Means++ on data set `C2.txt` for $k = 3$. Report the centers and the subsets.

To avoid variation in the class, choose c_1 as the point a. To compute the weighted sample, use of the techniques described in class, either the efficient tree approach or just lining up the elements up. In either case, keep the elements P lined up in the same order as they are presented. Then choose a random value $u \in [0, 1]$, multiply it by $W = \sum_{p \in P} w(p)$, getting $v = u \cdot W$, and select the point p_i such that $\sum_{j=1}^{i-1} w(p_i) < v \leq \sum_{j=1}^{i} w(p_i)$.

Also to avoid variation in the class, run k-Means++ twice with different random values $u_1, u_2 \in [0, 1]$. First use random values $\{u_1 = .35, u_2 = .6\}$. Second use random values $\{u_1 = .3, u_2 = .99\}$.

Which run of k-Means++ is more typical? Explain in which situations you would use each algorithm.

B: (2 points) Recall that Lloyd's algorithm for k-means clustering starts with a set of k centers C and runs as described in Algorithm 2.3.

Prove for $P \in \mathbb{R}^1$ that Lloyd's algorithm will eventually terminate. *Hint: Use similar techniques from problem 1.B and the part of the convergence proof covered in class.*

Algorithm 2.3 k-Means(P, C)

repeat
  for all $p \in P$ **do**
    Calculate $\mu(p) = \arg\min_{c \in C} d(p, c)$
  for all $c \in C$ **do**
    Calculate $P_c = \{p \in P \mid \mu(p) = c\}$
    Set $c = \frac{1}{|P_c|} \sum_{p \in P_c} p$
until (no assignment maps μ change)

3 High Dimensions

We will explore a couple potentially unintuitive properties of high dimensional data.

A: (3 points) We will explore what happens to the distribution of a Gaussian distributions in high-dimensions. A d-dimensional uniform Gaussian distribution is defined:

$$G(x) = \frac{1}{(2\pi)^{d/2}} e^{-\|x\|_2^2/2}.$$

If we have two uniform random numbers $u_1, u_2 \in [0, 1]$ then we can generate two independent 1-dimensional Gaussian random variables as

$$y_1 = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$$
$$y_2 = \sqrt{-2\ln(u_1)}\sin(2\pi u_2).$$

A uniform Gaussian has the property that all coordinates (in any orthogonal basis) are independent of each other. Thus to generated a point $x \in \mathbb{R}^d$ from a d-dimensional Gaussian, for each coordinate i we assign it the value of an independent 1-dimensional Gaussian random variable.

Your task is to generated d-dimensional Gaussian random variables, and plot their L_2 and L_1 norms. Plot however many you need until you feel you capture the distribution accurately. Specifically: present 12 plots (5 L_2 plots and 5 L_1 plots) for $d = \{1, 2, 3, 5, 10, 50\}$.

B: (3 points) We will again explore the difference between different L_p distances in high dimensions. Generate uniform random variables y in $[-1, 1]^d$ (each coordinate is an independent random variable in the range $[-1, 1]$. (i.e. a random $u \in [0, 1]$ is transformed to $x = 2u - 1$.)

For dimensions $d = \{1, 2, 3, 5, 10, 50\}$ estimate the probability that the random variable y has an L_p norm less than 1 for $p = \{0.5, 1, 2, 3, \infty\}$. Thus you should report 30 numbers. Recall that the L_p-norm of a vector is defined

$$\|x\|_p = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p},$$

and when $p = \infty$ this converges to

$$\|x\|_\infty = \max_{i=1}^{d} |x_i|.$$

(Random sampling is a good way to estimate these values empirically, but make sure you have used a large enough random sample. If you are unsure, justify the size of your sample choice.)

• Also, reason about what you expect to happen as p approaches 0.

4 BONUS)

A: (2 points) Extend the proofs for both 1.B and 2.B to $\mathbb{R}^d$ for arbitrary fixed d.

B: (2 points) Recall that the k-center problem is to find a set of k centers C to minimize

$$E_0(P, C) = \max_{p \in P} \min_{c \in C} d(p, c).$$

Let C^* be the optimal choice of k centers for the k-center problem, and let $E^* = E_0(P, C^*)$. Prove that the MinMax algorithm always finds a set of k centers C such that

$$E_0(P, C) \leq 2E^*.$$