Final_Solution 2012

Question 3. **Decision trees and Hierarchical clustering**

Assume we are trying to learn a decision tree. Our input data consists of N samples, each with k attributes ($N \gg k$). We define the depth of a tree as the maximum number of nodes between the root and any of the leaf nodes (including the leaf, not the root).

(a) [2 points] If all attributes are binary, what is the maximal number of leaf (decision) nodes that we can have in a decision tree for this data? What is the maximal possible depth of a decision tree for this data?

> **Solution:**
> $2^{(k-1)}$. Each feature can only be used once in each path from root to leaf. The maximum depth is O(k).

(b) [2 points] If all attributes are continuous, what is the maximum number of leaf nodes that we can have in a decision tree for this data? What is the maximal possible depth for a decision tree for this data?

> **Solution:**
> Continuous values can be used multiple times, so the maximum number of leaf nodes can be the same as the number of samples, N and the maximal depth can also be N.

(c) [2 points] When using **single link** what is the maximal possible depth of a hierarchical clustering tree for the data in 1? What is the maximal possible depth of such a hierarchical clustering tree for the data in 2?

> **Solution:**
> When using single link with binary data, we can obtain cases where we are always growing the cluster by 1 node at a time leading to a tree of depth N. This is also clearly the case for continuous values.

(d) [2 points] Would your answers to (3) change if we were using **complete link** instead of **single link**? If so, would it change for both types of data? Briefly explain.

> **Solution:**
> While the answer for continuous values remain the same (its easy to design a dataset where each new sample is farther from any of the previous samples) for binary data, if k is small compared to N we will not be able to continue to add one node at a time to the initial cluster and so the depth will change to be lower than N.

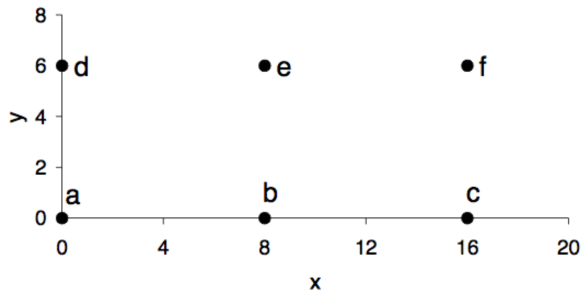
FINAL2003_solution

9 K-means Clustering (9 points)

There is a set S consisting of 6 points in the plane shown as below, $a = (0, 0)$, $b = (8, 0)$, $c = (16, 0)$, $d = (0, 6)$, $e = (8, 6)$, $f = (16, 6)$. Now we run the k-means algorithm on those points with $k = 3$. The algorithm uses the Euclidean distance metric (i.e. the straight line distance between two points) to assign each point to its nearest centroid. Ties are broken in favor of the centroid to the left/down. Two definitions:

- A **k-starting configuration** is a subset of k starting points from S that form the initial centroids, e.g. $\{a, b, c\}$.
- A **k-partition** is a partition of S into k non-empty subsets, e.g. $\{a, b, e\}, \{c, d\}, \{f\}$ is a 3-partition.

Clearly any k-partition induces a set of k centroids in the natural manner. A k-partition is called *stable* if a repetition of the k-means iteration with the induced centroids leaves it unchanged.



(a) How many 3-starting configurations are there? (Remember, a 3-starting configuration is just a subset, of size 3, of the six datapoints).

$C_6^3 = 20$

(b) Fill in the following table:

3-partition	Stable?	An example 3-starting configuration that can arrive at the 3-partition after 0 or more iterations of k-means (or write "none" if no such 3-starting configuration exists)	# of unique 3-starting configurations that arrive at the 3-partition
$\{a, b, e\}, \{c, d\}, \{f\}$	N	none	0
$\{a, b\}, \{d, e\}, \{c, f\}$	Y	{b, c, e}	4
$\{a, d\}, \{b, e\}, \{c, f\}$	Y	{a, b, c}	8
$\{a\}, \{d\}, \{b, c, e, f\}$	Y	{a, b, d}	2
$\{a, b\}, \{d\}, \{c, e, f\}$	Y	none	0
$\{a, b, d\}, \{c\}, \{e, f\}$	Y	{a, c, f}	1

1 Clustering [35 points, Martin]

1.1 [25 points] K-means

In K-means clustering, we are given points $x_1, ..., x_n \in \mathbb{R}^d$ and an integer $K > 1$, and our goal is to minimize the within-cluster sum of squares (also known as the k-means objective)

$$J(C, L) = \sum_{i=1}^{n} \|x_i - C_{\ell_i}\|^2$$

where $C = (C_1, ..., C_K)$ are the cluster centers ($C_j \in \mathbb{R}^d$), and $L = (\ell_1, ..., \ell_n)$ are the cluster assignments ($\ell_i \in \{1, ..., K\}$).

Finding the exact minimum of this function is computationally difficult. The most common algorithm for finding an approximate solution is Lloyd's algorithm, which takes as input the set of points and some initial cluster centers C, and proceeds as follows:

i. Keeping C fixed, find cluster assignments L to minimize $J(C, L)$. This step only involves finding nearest neighbors. Ties can be broken using arbitrary (but consistent) rules.

ii. Keeping L fixed, find C to minimize $J(C, L)$. This is a simple step that only involves averaging points within a cluster.

iii. If any of the values in L changed from the previous iteration (or if this was the first iteration), repeat from step i.

iv. Return C and L.

The initial cluster centers C given as input to the algorithm are often picked randomly from $x_1, ..., x_n$. In practice, we often repeat multiple runs of Lloyd's algorithm with different initializations, and pick the best resulting clustering in terms of the k-means objective. You're about to see why.

(a) [**3 points**] Briefly explain why Lloyd's algorithm is always guaranteed to converge (i.e. stop) in a finite number of steps.

Answer: The cluster assignments L can take finitely many values (K^n, to be precise). The cluster centers C are uniquely determined by the assignments L, so after executing step ii the algorithm can be in finitely many possible states. Thus either the algorithm stops in finitely many steps, or at least one value of L is repeated more than once in non-consecutive iterations. However, the latter case is not possible, since after every iteration we have $J(C^{(t)}, L^{(t)}) \geq$

$J(C^{(t+1)}, L^{(t+1)})$, with equality only when $L^{(t)} = L^{(t+1)}$, which coincides with the termination condition. (Note that this statement depends on the assumption that the tie-breaking rule used in step i is consistent, otherwise infinite loops are possible.)

1.2 Hierarchical clustering [10 points]

Agglomerative hierarchical clustering is a family of hierarchical clustering algorithms that, equipped with a notion of distance between clusters, form a binary tree with leaves for each original data point as follows:

i. Initialize by placing each data point in its own cluster (i.e. singleton trees).

ii. Find the two closest clusters, join them in a single cluster (by creating a new node and making it the parent of the roots of those two clusters).

iii. If there are more than one clusters (trees) left, repeat from step i.

iv. Return the final tree.

Some of the most common metrics of distance between two clusters $\{x_1, ..., x_m\}$ and $\{y_1, ..., y_p\}$ are:

- *Single linkage*: Distance between clusters is the *minimum* distance between any pair of points from the two clusters, i.e.

$$\min_{\substack{i=1,\ldots,m \\ j=1,\ldots,p}} \|x_i - y_j\|;$$

- *Complete linkage*: Distance between clusters is the *maximum* distance between any pair of points from the two clusters, i.e.

$$\max_{\substack{i=1,\ldots,m \\ j=1,\ldots,p}} \|x_i - y_j\|;$$

- *Average linkage*: Distance between clusters is the *average* distance between all pair of points from the two clusters, i.e.

$$\frac{1}{m \cdot p} \sum_{i=1}^{m} \sum_{j=1}^{p} \|x_i - y_j\|.$$

Also, given a clustering tree, we can define a partitioning of the data into K clusters by "cutting" the tree some number of levels below the root. For example, if $K = 2$ we could define two clusters based on the left and right subtrees of the root of the clustering tree. If $K = 4$, we could use the subtrees of the children of the root, etc. (Note that if K is not a power of 2 we would need to come up with some way of deciding which subtree gets preference.)

(a) **[3 points]** Using this procedure for turning a hierarchical clustering into a partition, which of the three cluster similarity metrics described above would most likely result in clusters most similar to those given by k-means? (Assume K is a power of 2).

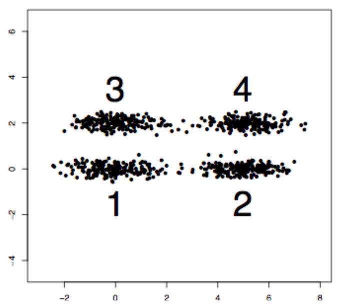
Answer: Average linkage.

(b) **[4 points]** Consider the data in Figure 8a. What would be the result if we extracted $K = 2$ clusters from the tree given by hierarchical clustering on this data set using single linkage? (Describe your answer in terms of the labels $1 - 4$ given to the four "clumps" in the data.) Do the same for complete and average linkage.

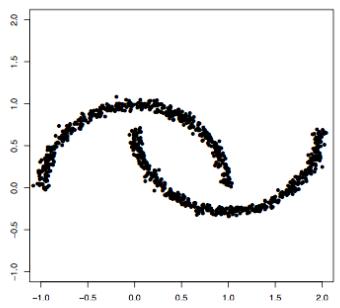
Answer: Average and complete linkage would assign "clumps" 1 and 3 to the first cluster, and 2 and 4 to the second. Single linkage would assign 1 and 2 to one cluster, 3 and 4 to the other.

(c) **[3 points]** Which of those three distance metrics (if any) would successfully separate the two "moons" in Figure 8b? What about Figure 8c? Briefly explain your answer.

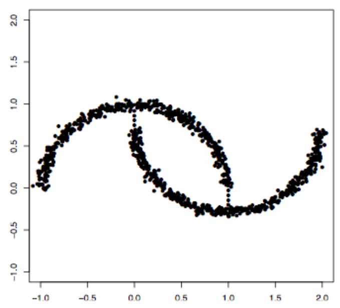
Answer: Single linkage would successfully separate the two moons in Figure 8b, average and complete linkage would not. None of the methods would work in Figure 8c.



(a)



(b)



(c)

K-Means (20 points)

In this problem we will look at the K-means clustering algorithm. Let $X = \{x_1, x_2, \dots, x_n\}$ be our data and γ be an indicator matrix such that $\gamma_{ij} = 1$ if x_i belongs to the j^{th} cluster and 0 otherwise. Let $\mu_1, \dots, \mu_k$ be the means of the clusters.
We can define the distortion J as follows

$$J(\gamma, \mu_1, \dots, \mu_k) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \mu_j||^2$$

Finally, we define $C = 1, \dots, k$ be the set of clusters.
The most common form of the K-means algorithm proceeds as follows

- Initialize $\mu_1, \dots, \mu_k$.
- While J is decreasing, repeat the following
  1. Determine γ breaking ties arbitrarily.
  $$\gamma_{ij} = \begin{cases} 1, & ||x_i - \mu_j||^2 \leq ||x_i - \mu_{j'}||^2, \forall j' \\ 0, & \text{otherwise} \end{cases}$$
  2. Recompute μ_j using the updates γ Remove j from C if $\sum_{i=1}^{n} \gamma_{ij} = 0$. Otherwise,
  $$\mu_j = \frac{\sum_{i=1}^{n} \gamma_{ij} x_i}{\sum_{i=1}^{n} \gamma_{ij}}$$

1. Show that this algorithm will always terminate in a finite number of steps. (How many different values can γ take?) (4 points)

   From class we know that J is a monotonic decreasing with every iteration of Kmeans. This means that we cannot revisit the same state of γ twice. Since γ has n^2 entries, each of which is either 0 or 1, it has a finitely many number of possible values. This implies Kmeans must eventually terminate
   Award 4 points to any answer that correctly details why it terminates in finitely many steps.

2. Let $\hat{x}$ be the sample mean. Consider the following quantities,

$$T(X) = \frac{\sum_{i=1}^{n} ||x_i - \hat{x}||^2}{n}$$
$$W_j(X) = \frac{\sum_{i=1}^{n} \gamma_{ij} ||x_i - \mu_j||^2}{\sum_{i=1}^{n} \gamma_{ij}}$$
$$B(X) = \sum_{j=1}^{k} \frac{\sum_{i=1}^{n} \gamma_{ij}}{n} ||\mu_j - \hat{x}||^2.$$

   Here, $T(X)$ is the total deviation, $W_j(X)$ is the intra-cluster deviation and $B(X)$ is the inter-cluster deviation. What is the relation between these quantities? Based on this, show that K-means can be views as minimizing a weighted average of intra-cluster deviation while approximately maximizing the inter-cluster deviation. Your relation may contain a term that was not mentioned above. (5 points)

We start from

$$\begin{aligned}
\sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}W_j(X)+nB(X) &= \sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}||x_i-\mu_j||^2+\gamma_{ij}||\mu_j-\hat{x}||^2 \\
&= \sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}(||x_i-\mu_j||^2+||\mu_j-\hat{x}||^2) \\
&= \sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}(x_i^2+\hat{x}^2-2x_i\hat{x}+2x_i\hat{x}+\mu_j^2-2x_i\mu_j-2\hat{x}\mu_j \\
&= \sum_{j=1}^{k}(\sum_{i=1}^{n}\gamma_{ij}(||x_i-\hat{x}||^2))+K
\end{aligned}$$

Here K is the remainder of the summation clumped up into one term.

$$\begin{aligned}
\sum_{j=1}^{k}(\sum_{i=1}^{n}\gamma_{ij}(||x_i-\hat{x}||^2))+K &= n\sum_{i=1}^{n}(||x_i-\hat{x}||^2)+K \\
&= n^2T(X)+K
\end{aligned}$$

Therefore, $\sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}W_j(X)+nB(X)=n^2T(X)+K$.
Note that $\sum_{j=1}^{k}\sum_{i=1}^{n}\gamma_{ij}W_j(X)$ is the same as J and is therefore minimized during K-means. Since $n^2T(X)$ is constant, this means that $nB(X)$ is approximately maximized (approximately because of the other term.
Award 3 points to any justified equation relating the terms even if its not the one presented here. Subtract unto two points for an equation provided that is not justified.
Award 2 points to any solution that shows the terms contained W_j are monotonic decreasing and since T is a constant, the $B_j's$ must be approximately maximized (approximately because of the extra term). Subtract 1 point if it is not justified that the $W_j's$ are minimized.

3. Show that the minimum of J is a non increasing function of k the number of clusters. Argue that this means it is meaningless to choose the number of clusters by minimizing J. (4 points)

This can be seen by a simple induction argument. Assume that till some k, J has been non decreasing in k. Let us now add another cluster center to some arbitrary location. Since this new run of Kmeans has not yet converged, we know that we are not at the minimum possible J for $k+1$ clusters. If we show that J is still non-decreasing, then we have completed our induction step. Observe that at least for the point that is now a cluster center, the term in J will be 0. This means that J has decreased when we have added a new cluster.
If we were to pick the k that minimized J, we would end up picking $k=n$ since this makes J 0. since we do not want to vastly overfit, we cannot simply arg max for k.
Award 2 points to any correct reasoning for why J is non increasing.
Award 2 points to the correct reasoning for why we cannot pick k from minimizing J.

4. Assume that now we use the ℓ_1 norm in J as opposed to the squared Euclidean distance

$$J'(\gamma, \mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \sum_{j=1}^{k} \gamma_{ij} ||x_i - \mu_j||_1$$

- Derive the steps to this new K-means formulation. Note that the answers may not be unique. (5 points)
- If your data contains outliers, which version of K-means would you use - the ℓ_1 norm one or the original Euclidean norm one? Justify. (2 points)

  - Initialize $\mu_1, \ldots, \mu_k$.
  - While J is decreasing, repeat the following

    (a) Determine γ breaking ties arbitrarily.

    $$\gamma_{ij} = \begin{cases} 1, & ||x_i - \mu_j||_1 \leq ||x_i - \mu_{j'}||_1, \forall j' \\ 0, & \text{otherwise} \end{cases}$$

    (b) Recompute μ_j using the updates γ Remove j from C if $\sum_{i=1}^{n} \gamma_{ij} = 0$. Otherwise,

    $$\mu_j = median(x_j | \gamma_{ij} = 1)$$

The goal is to maximize $\sum_{i=1}^{m} ||x_i - \mu_j||_1$ for the j^{th} cluster. Picking the μ that maximizes this will give us our centroid. Let us consider the derivate of this w.r.t μ. Each of the L1 norm terms will be either 1 or -1 depending on whether or not the L1 norm of the point is greater than μ. This derivative is 0 when the sum of the 1s and the -1s is 0. This happens when there are an equal number of points on either side of the centroid μ. The choice of centroid that achieves this is the median of the L1 norms of the points in the cluster. This solution may not be unique for a given cluster.
We might want the L1 norm for noisy data because the geometric median is robust to outliers. It only cares about picking the middle point in the ordering. It does not care about the distance of the furthest points to the center, only the number of points on each side.
Award 3 points to this answer or any other answer that picks the median
Award 2 points to an answer that states that L1 loss is robust to outliers.

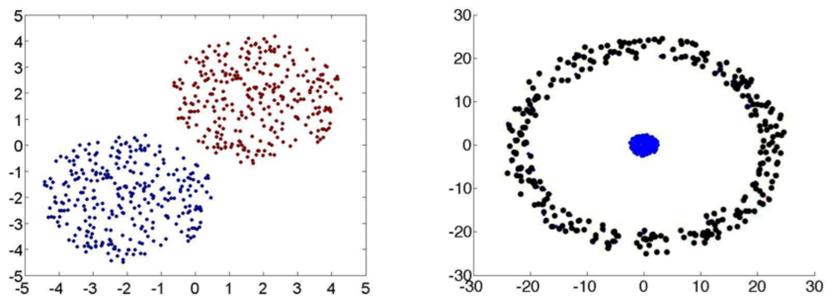


Figure 2: Sample clusters from the EM algorithm

Your outcome for dataset 1 should look very clean; two nice round clusters. For dataset 2, it should be messy in some way, and might be highly sensitive to yoru starting values. Figure 1 shows sample outputs.

1.2 K-means : evaluation [5 pt]

How well do you think k-means did for each dataset? Explain, intuitively, what (if anything) went badly and why.

Answer : K-means should do well on dataset 1 and poorly on dataset 2. This is because k-means looks for nice round clusters, and there aren't two of them in dataset 2.

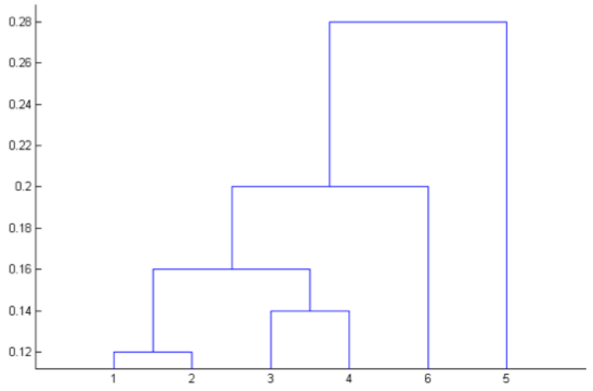
2 Clustering

The table below is a distance matrix for 6 objects.

	A	B	C	D	E	F
A	0					
B	0.12	0				
C	0.51	0.25	0			
D	0.84	0.16	0.14	0		
E	0.28	0.77	0.70	0.45	0	
F	0.34	0.61	0.93	0.20	0.67	0

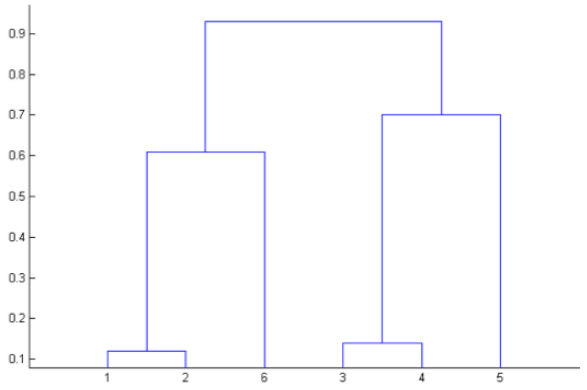
2.a Hierarchical clustering

1. Show the final result of hierarchical clustering with single link by drawing a dendrogram.



[Solution: *]*

2. Show the final result of hierarchical clustering with complete link by drawing a dendrogram.



[Solution: *]*

3. Change **two** values from the matrix so that your answer to the last two question would be same.
[Solution: There is more than one way possible, but one way would be the following:

The first step that the complete link clustering differs from the single link clustering is where AB and F are grouped together by dist(AB,F)=dist(B,F)=0.61. We'd want dist(AB, CD)=dist(A,D) to be smaller than this value, such as 0.53.

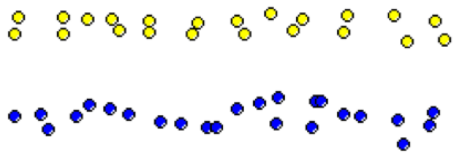
Then we want dist(ABCD,F) = dist(C, F) = 0.93 to be the smallest so that ABCD and F are grouped together. We set this value to 0.63. After these changes both dendrograms become identical. *]*

2.b Which clustering method should we use?

Which clustering method(s) is most likely to produce the following results at $k = 2$? Choose the most likely method(s) and briefly explain why it/they will work better where others will not in **at most 3 sentences**. Here are the five clustering methods you can choose from:

- Hierarchical clustering with single link
- Hierarchical clustering with complete link
- Hierarchical clustering with average link
- K-means
- GMM (with no assumption on the covariance matrices)

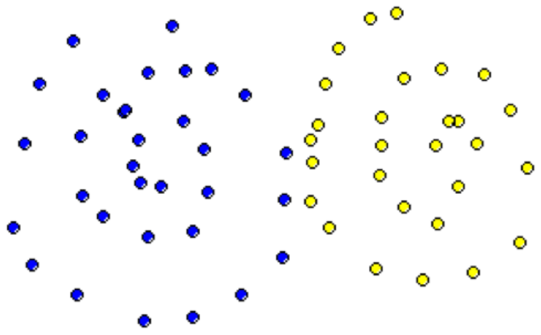
1.



[Solution: Hierarchical clustering with single link is most likely to well. GMM can also produce a decision boundary that can produce such clustering result, but depending on initialization it might converge to a different set of clusters (left half vs. right half).

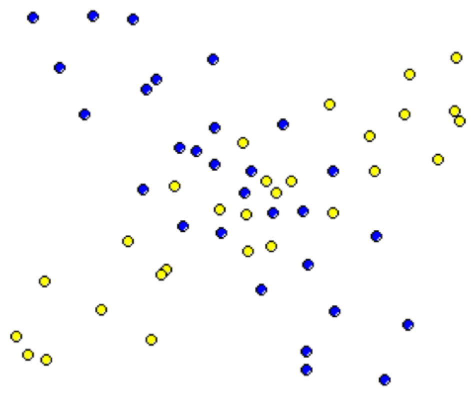
Other hierarchical clusterings won't really work well because at some point, two intermediate clusters from different true cluster will have shorter cluster distance than two from the same true cluster. *]*

2.



[Solution: K-means or GMM is most likely. Hierarchical clustering wouldn't work since the early few steps will group instances near the decision boundary (note some of them are very close). *]*

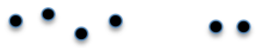
3.



[Solution: Among the five methods, only GMM has the capability of handling overlapping clusters. So GMM is the only method that would result in such clusters. *]*

midterm_sol spring 2010

(1) [3 pt] Consider the set of training data below, and two clustering algorithms: K-Means, and a Gaussian Mixture Model (GMM) trained using EM. Will these two clustering algorithms produce the same cluster centers (means) for this data set? In one sentence, explain why or why not.



Answer : Ok, almost everybody got this problem wrong, so let me explain carefully. Either algorithm will find the clusters just fine. But the difference lies in that k-means uses hard assignment of each point to a single cluster, whereas GMM uses soft assignment, where every point has non-zero (though possibly small) probability of being in each cluster. So in k-means, the means of the clusters are determined by an average of the points assigned to that cluster, but in GMM the means of each cluster are (differently) weighted averages of all points. This has the effect of skewing the center of the left cluster to the right, and the center of the right cluster to the left.

You could argue that this is a downside of the EM algorithm, that it still give some weight to points that are clearly in the other cluster. On the other hand, each point in the other cluster could just *maybe* be an outlier from the first cluster, so this skewing is not completely unreasonable. Regardless whether you like or dislike this phenomenon, you should be aware of it and understand where it comes from.

midterm_solutions fall 2012

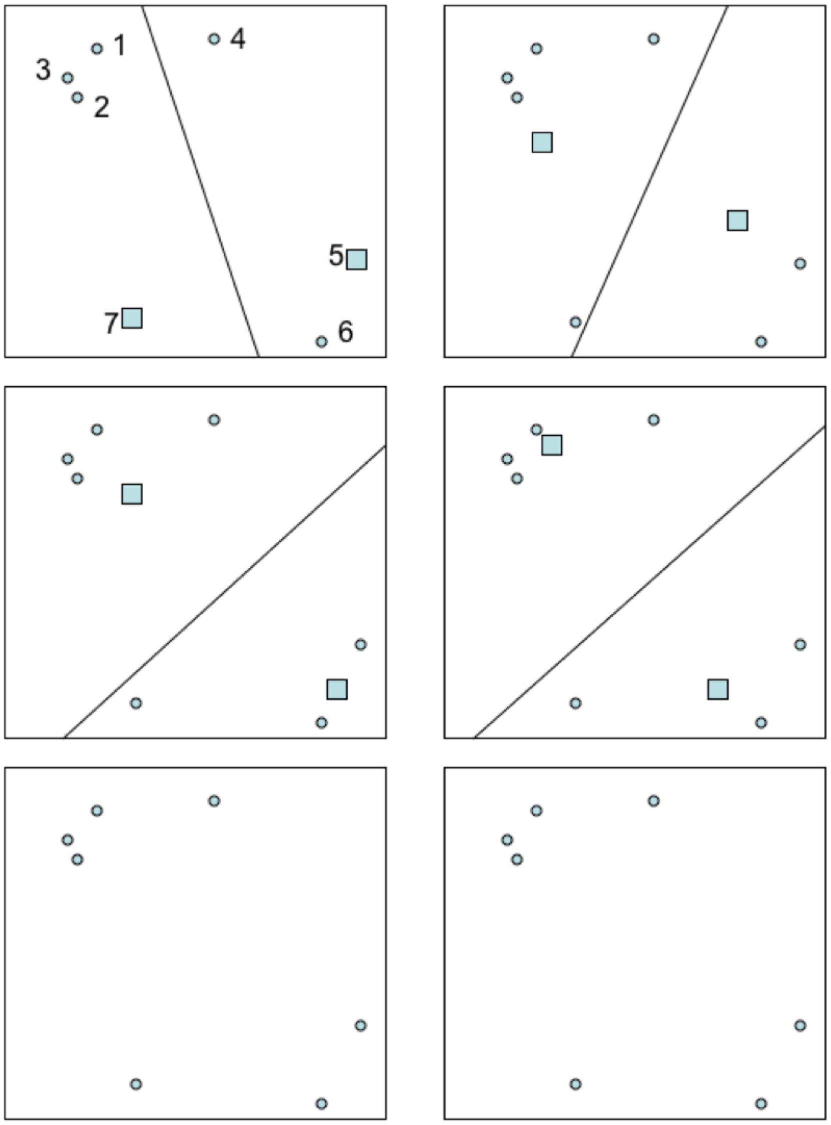
(g) [3 points] Suppose we clustered a set of N data points using two different clustering algorithms: k-means and Gaussian mixtures. In both cases we obtained 5 clusters and in both cases the centers of the clusters are exactly the same. Can 3 points that are assigned to different clusters in the k-means solution be assigned to the same cluster in the Gaussian mixture solution? If no, explain. If so, sketch an example or explain in 1-2 sentences.

Solution:
Yes, k-means assigns each data point to a unique cluster based on its distance to the cluster center. Gaussian mixture clustering gives soft (probabilistic) assignment to each data point. Therefore, even if cluster centers are identical in both methods, if Gaussian mixture components have large variances (components are spread around their center), points on the edges between clusters may be given different assignments in the Gaussian mixture solution.

4 K-means and Hierarchical Clustering (10 points)

(a) *(6 points)* Perform K-means on the dataset given below. Circles are data points and there are two initial cluster centers, at data points 5 and 7. Draw the cluster centers (as squares) and the decision boundaries that define each cluster. If no points belong to a particular cluster, assume its center does not change. Use as many of the pictures as you need for convergence.

Answer:

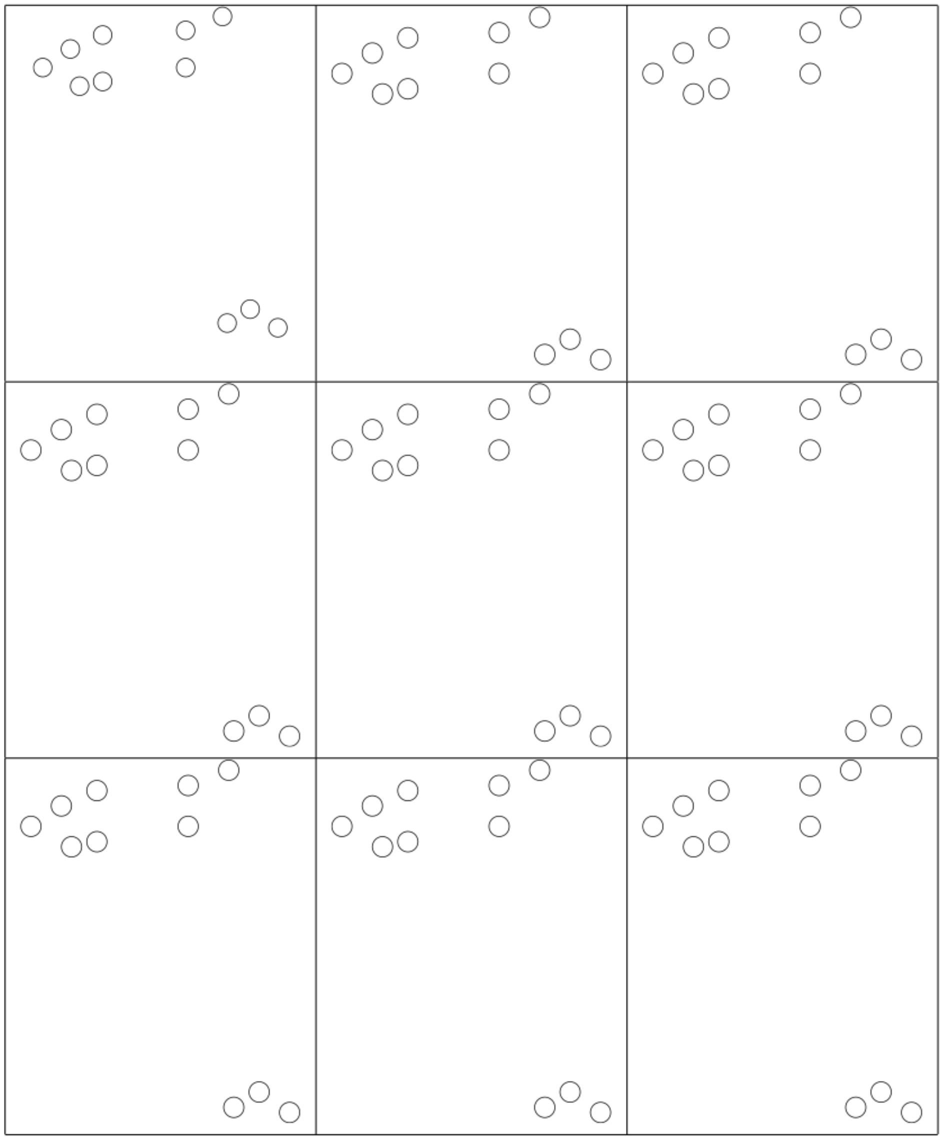


final2002

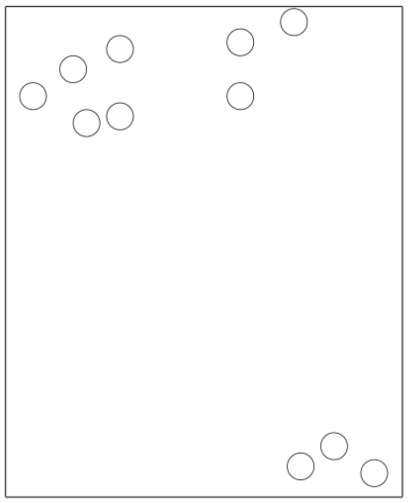
2 K-means and Gaussian Mixture Models

(a) What is the effect on the means found by k-means (as opposed to the true means) of overlapping clusters?

(b) Run k-means manually for the following dataset. Circles are data points and squares are the initial cluster centers. Draw the cluster centers and the decision boundaries that define each cluster. Use as many pictures as you need until convergence.
Note: Execute the algorithm such that if a mean has no points assigned to it, it stays where it is for that iteration.



(c) Now draw (approximately) what a Gaussian mixture model of three gaussians with the same initial centers as for the k-means problem would converge to. Assume that the model puts no restrictions on the form of the covariance matrices and that EM updates both the means and covariance matrices.



(d) Is the classification given by the mixture model the same as the classification given by k-means? Why or why not?

(b) *(4 points)* Give one advantage of hierarchical clustering over K-means clustering, and one advantage of K-means clustering over hierarchical clustering.

Answer: Many possibilities.

Some advantages of hierarchical clustering:

1. **Don't need to know how many clusters you're after**
2. **Can cut hierarchy at any level to get any number of clusters**
3. **Easy to interpret hierarchy for particular applications**
4. **Can deal with long stringy data**

Some advantages of K-means clustering:

1. **Can be much faster than hierarchical clustering, depending on data**
2. **Nice theoretical framework**
3. **Can incorporate new data and reform clusters easily**