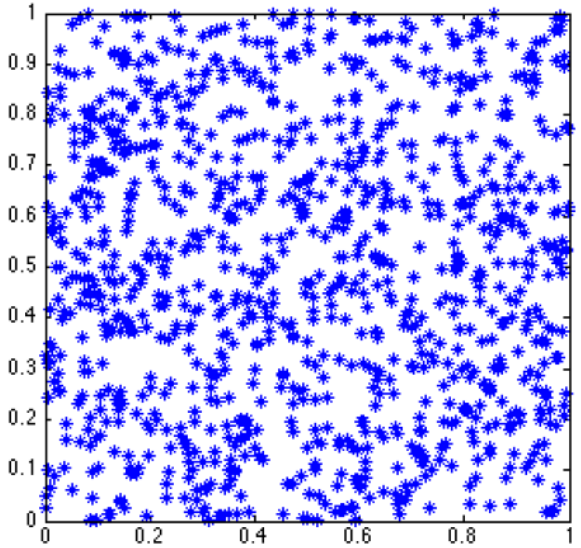
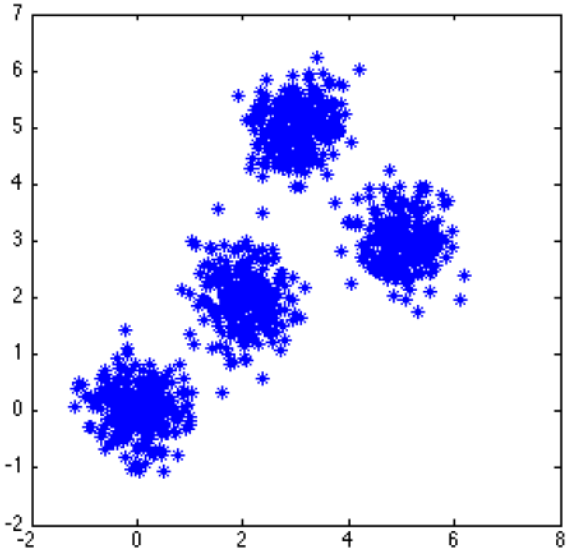
TP4 Clustering:
K-means and GMMs

0: K-means Initialization

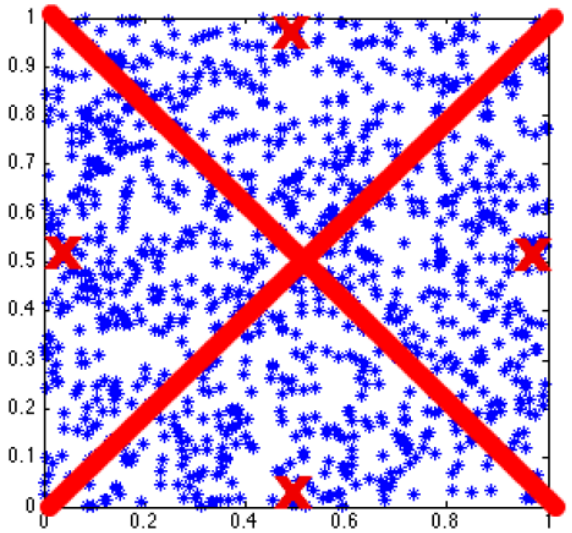


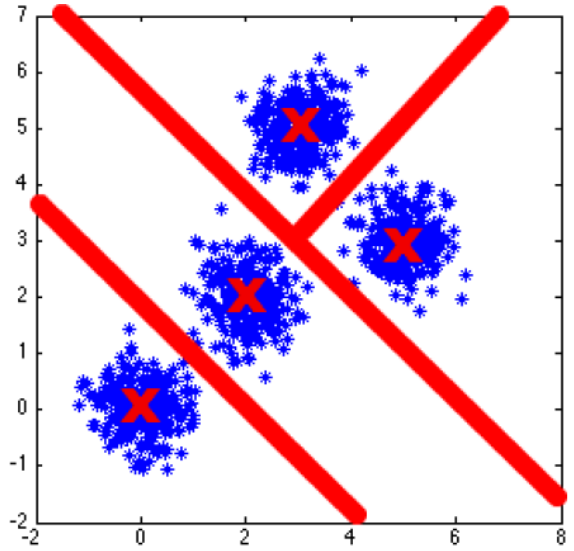


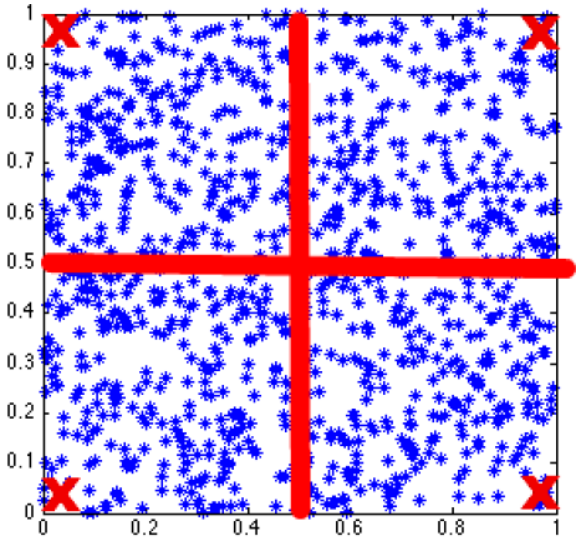
(a) You are given two datasets consisting of 1000 2-D examples each and we want to find 4 clusters in each of them

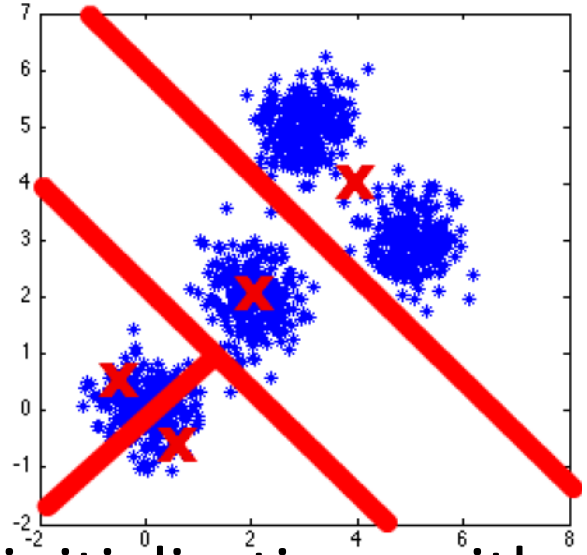
- We know that K-means is not robust to initialization
- Can you provide two different initializations for each of the datasets that would result in qualitatively different clusters?
  - Sketch the initializations and the resulting clusters

0: Solution







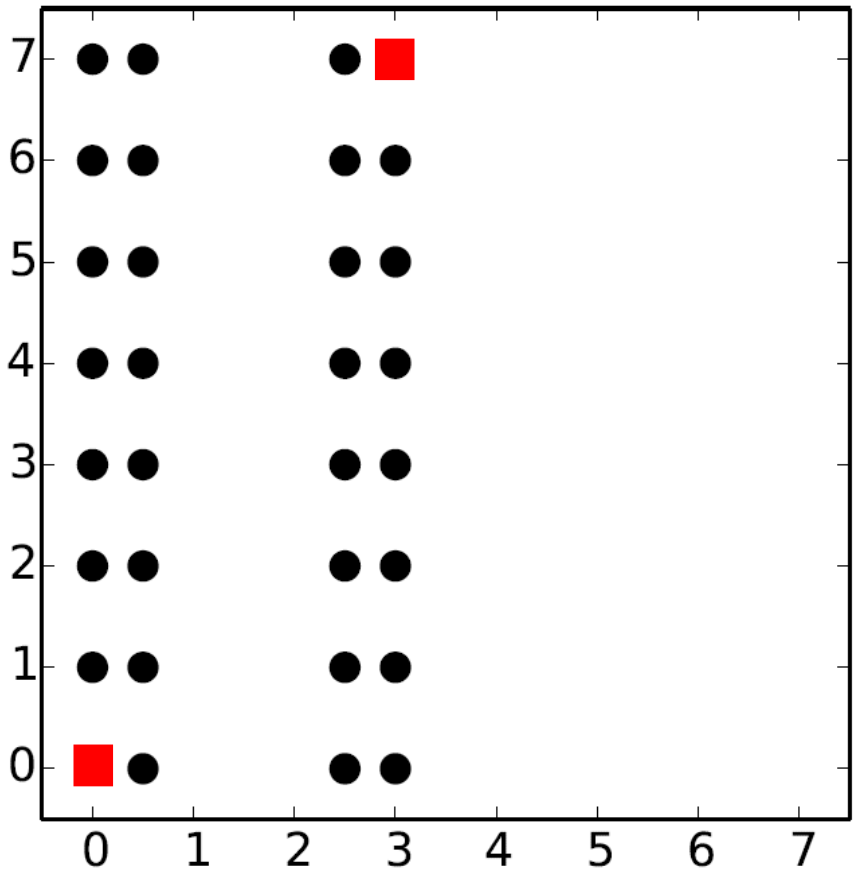


- In the above plots, you can see two possible initializations with all the resulting separating hyperplanes for each of the datasets

0: K-means Initialization

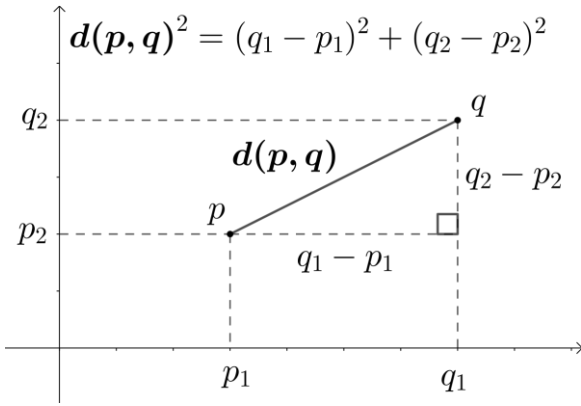
(b) You are given, and unlabeled 2D dataset represented on the next Figure

- Using the two points marked as squares as initial centroids, draw the clusters obtained after one iteration of the k-means algorithm with k = 2
- Does your solution change after another iteration of the k-means algorithm?

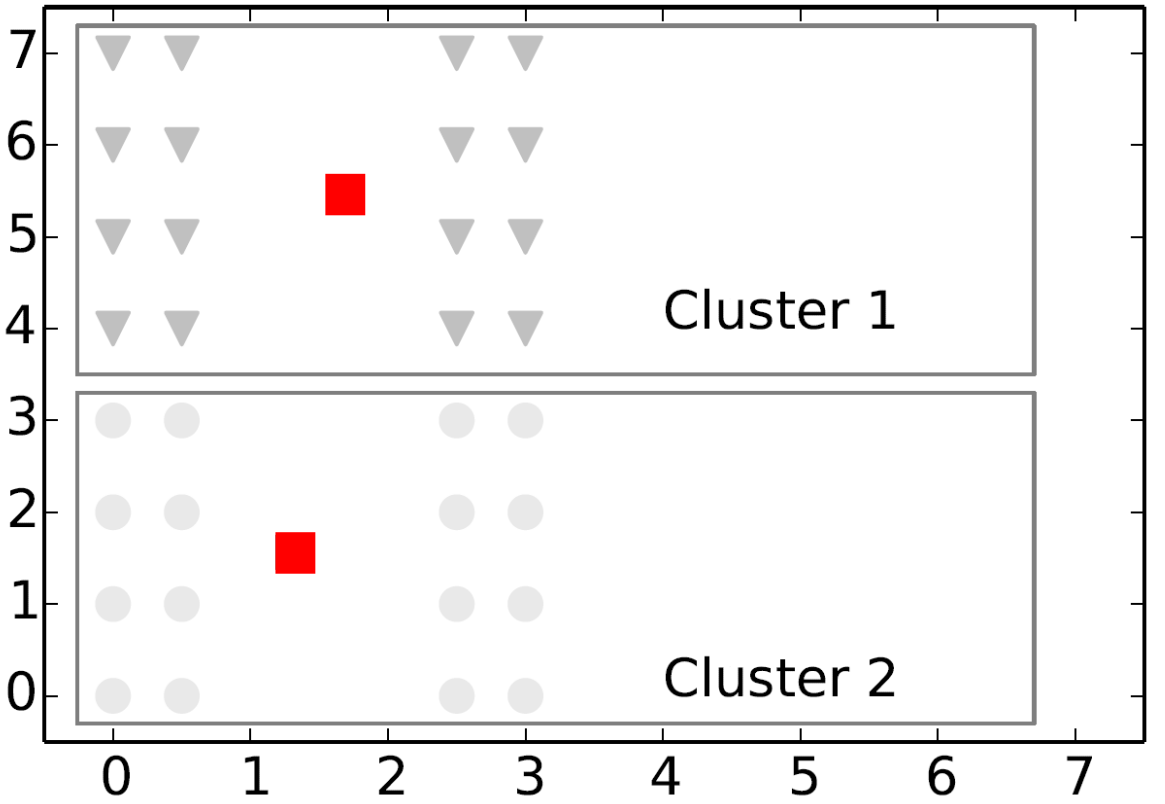


(c) What is the effect on the means found by k-means (as opposite to true means) of overlapping clusters?

0: K-means Initialization



(b) Define a square matrix of distances



- No

(c) They are pushed further apart than the true means would be

1: K-means Convergance

- In the K-means clustering algorithm, you are given a set of n points $x_i \in R^d$, $i \in \{1,2,...n\}$ and you want to find the centers of k clusters $\mu = (\mu_1,..., \mu_k)$ by minimizing the average distance from the points to the closest cluster center
  - Formally, you want to minimize the following loss function $L(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,...,k\}} \|x_i - \mu_j\|_2^2$
- To approximate the solution, we introduce for each data example x_i new assignment $z_i \in \arg\min_{j \in \{1,...,k\}} \|x_i - \mu_j\|_2^2$
- The K-means algorithm iterates between updating the variables z_i (assignment step) and updating the centers $\mu_j = \frac{1}{|\{i:z_i=j\}|} \sum_{i:z_i=j} x_i$ (refitting step)
  - The algorithm stops when no change occurs during the assignment step
- Show that K-means is guaranteed to converge (to a local optimum)
  - Hint: You need to prove that the loss function is guaranteed to decrease monotonically in each iteration until convergence
  - Prove this separately for the assignment step and the refitting step

1: Solution

- Since the loss function is non-negative, the algorithm will eventually converge when the loss function reaches its (local) minimum
  - Let **z** = (z_1, …, z_n) denote the cluster assignments for the n points

(i) Assignment step

- We can rewrite the original loss function L(μ) as $L(\mu, z) = \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|_2^2$

1: Solution

- Since the loss function is non-negative, the algorithm will eventually converge when the loss function reaches its (local) minimum
  - Let $\mathbf{z} = (z_1, \ldots, z_n)$ denote the cluster assignments for the n points

(i) Assignment step

- We can rewrite the original loss function L(μ) as $L(\mu, z) = \sum_{i=1}^{n} \|x_i - \mu_{z_i}\|_2^2$
- Let us consider an example x_i, and let z_i be the assignment from the previous iteration and z_i^* be the new assignment obtained as: $z_i^* \in \arg \min_{j \in \{1,\ldots,k\}} \|x_i - \mu_j\|_2^2$
- Let $\mathbf{z}^*$ denote the new cluster assignments for all the n points
- The change in loss function after this assignment step is then given by:

$$L(\mu, z^*) - L(\mu, z) = \sum_{i=1}^{n} \left(\|x_i - \mu_{z_i^*}\|_2^2 - \|x_i - \mu_{z_i}\|_2^2 \right) \leq 0$$

- The inequality holds by the rule z_i^* is determined, i.e., to assign x_i to the nearest cluster

1: Solution

(ii) Refitting step

- We can rewrite the original loss function L(μ) as $L(\mu, z) = \sum_{j=1}^{k} \Big(\sum_{i:z_i=j} \|x_i - \mu_j\|_2^2 \Big)$
- Let us consider the j^{th} cluster, and let μ_j be the cluster center from the previous iteration and μ_j^* be the new cluster center obtained as: $\mu_j^* = \frac{1}{|\{i : z_i = j\}|} \sum_{i:z_i=j} x_i$
- Let $\boldsymbol{\mu}^*$ denote the new cluster centers for all the k clusters
- The change in loss function after this refitting step is then given by:

$$L(\mu^*, z) - L(\mu, z) = \sum_{j=1}^{k} \Big(\big(\sum_{i:z_i=j} \|x_i - \mu_j^*\|_2^2 \big) - \big(\sum_{i:z_i=j} \|x_i - \mu_j\|_2^2 \big) \Big) \leq 0$$

- The inequality holds because the update rule of μ_j^* essentially minimizes this quantity

2 Bonus: K-medians Clustering (20 pts)

- We are intrested in deriving a new clustering algorithm based on the following loss function:

$$L(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\ldots,k\}} \|x_i - \mu_j\|_1$$

(a) (10 pts) Define the update steps both for z_i and μ_j for this algorithm

(b) (5 pts) Does your algorithm convergences?

(c) (5 pts) In which situation would you prefer to use K-medians clustering instead of K-means clustering?

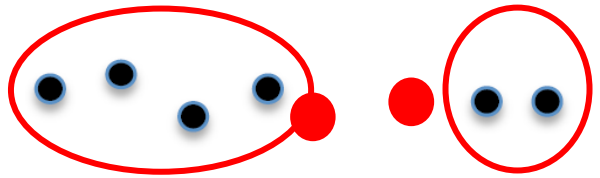
4: Hard and Soft Clustering

- We are intrested in training GMM's with two components
  - We will use μ_0, μ_1, σ_0 and σ_1 to define the means and variances of these two Gaussian components, and will use π_0 and $(1-\pi_0)$ to denote the mixture proportions of the two Gaussians i.e., $p(x) = \pi_0\, N(\mu_0, \sigma_0 \boldsymbol{I}) + (1-\pi_0)\, N(\mu_1, \sigma_1 \boldsymbol{I})$)
  - We will also use θ to refer to the entire vector of parameters (μ_0, μ_1, σ_0, σ_1 π_0) defining the mixture model $p(x)$

(a) Consider the set of training dataset 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 the above dataset?

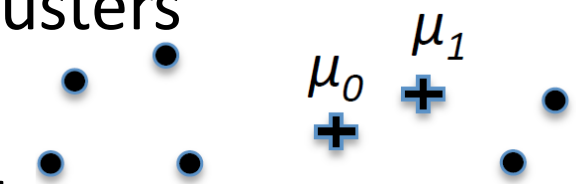
4: Solution



- Either algorithm will find the two clusters just fine
- But the difference lies in that k-means uses hard assignment of each example to a single cluster, whereas GMM uses soft assignment, where every example 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 examples
  - 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 examples that are clearly in the other cluster
  - On the other hand, each example 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

4: Hard and Soft Clustering

(b) Consider applying EM to train a Gaussian Mixture Model (GMM) to cluster the following dataset into two clusters



- The '+' points indicate the current means μ_0 and μ_1 of the two Gaussian mixture components after the k^{th} iteration of EM

(i) Draw the directions in which μ_0 and μ_1 will move during the next M-step

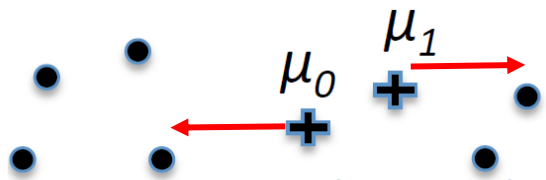
(ii)Will the marginal likelihood of the data, $\prod_j P(x^j|\theta)$ increase or decrease on the next EM iteration?

(iii) Will the estimate of π_1 increase or decrease on the next EM step?

(iv) The objective function optimized by the EM algorithm can also be optimized by a gradient descent algorithm

- Will both algorithms find locally or a globally optimal solution?

4: Solution



(b.i) μ_0 moves to the left, and μ_1 moves to the right

(b.ii) Increase

- Each iteration of the EM algorithm increases the likelihood of the data, unless you happen to be exactly at a local optimum!

(b.iii) Close

- π_0 is determined by adding the probabilities of all examples that they are in cluster 0
- In the current configuration, π_1 is close enough to π_0 that it will be stealing a lot of this probability mass, so π_1 and π_0 will be pretty close to each other

(b.iv) The gradient descent algorithm risk also to get stuck in a local optima as the EM algorithm

4: Hard and Soft Clustering

(c) Next let's consider the relationship between a Gaussian Naive Bayes (GNB) classifier and the previous Gaussian Mixture Model (GMM)

- It is easy to see that they involve the same probabilistic model: standard GNB classifier assumes p(Y | X) is of the form:

$$p(Y|X) = \frac{p(Y)\prod_i p(X_i|Y)}{p(X)}$$

  where Y is a Bernoulli random variable (i.e., P(Y = 0) = π_0)

- It also assumes each feature X_i is governed by a Gaussian distribution conditioned on Y
  - For simplicity, let's assume all features have the same variance, so $P(X_i|Y=k) \sim N(\mu_{k,i}, \sigma)$
- In essence, both models assume examples generated by choosing a Y according to π_0, then drawing an X according to a Gaussian conditioned on Y
- The difference is that we train GNB using labeled data in which the Y values are known, whereas we train GMM assuming Y values are unknown!

4: Hard and Soft Clustering

- When training this GNB, we choose the set of parameters q that maximize the data likelihood

$$\arg\max_{\theta} \prod_{j} P(x^j, y^j | \theta)$$

where again we use the superscript j to denote the j^{th} training example

(i) Write down the objective that EM seeks to maximize when it trains the same model, without known values for y^j

(ii) Write down the E and M steps in the standard EM algorithm for learning mixture of Gaussians

(iii) GNB trains using labeled examples, GMM trains using unlabeled examples

- Suppose we have a set of training examples which are partially labeled: we have known y values for x^1, x^2, …, x^m, but have additional unlabeled examples x^{m+1} , …, x^{m+n} without known values for y
- How would you propose to train the generative model in this case?
  - Write down your modified E and M steps

(iv) Write down the objective function that your modified EM is maximizing

- In your expression, distinguish between the training examples for which y is known & unknown

4: Solution

(c.i) $\Pi_j P(x^j|\theta) = \Pi_j \left(\sum_y P(x^j, y|\theta) \right)$

(c.ii)

E $\gamma_{ik} = P(y^k) N(x^i|\mu_k, \sigma_k)$

M $\pi_k = \frac{\sum_i \gamma_{ik}}{\sum_{ij} \gamma_{ij}} \qquad \mu_k = \frac{\sum_i \gamma_{ik} x^i}{\sum_i \gamma_{ik}} \qquad \sigma_k^2 = \frac{\sum_i \gamma_{ik} (x^i - \mu_k)^2}{\sum_i \gamma_{ik}}$

(c.iii)

E Same as above for x^{m+1}, …, x^{m+n}, and for $i \leq m$, $\gamma_{ij} = \delta_{j,y(i)}$

M Same as above

(c.iv) $(\Pi_{i=1}^m P(x^i, y^i|\theta)) \left(\Pi_{i=m+1}^{m+n} P(x^i|\theta) \right)$

5 Bonus: Univariate Gaussian Mixture Model (GMM) (30 pts)

- We will derive the E-M update rules for a univariate GMM with two mixture components
  - Unlike the GMMs we covered in the course, the mean will be shared between the 2 mixture components, but each component will have its own standard deviation σ_k
- The model is defined as follows:

$$z \sim \text{Bernoulli}(\theta)$$

$$x|z = k \sim \mathcal{N}(\mu, \sigma_k)$$

(a) (5 pts) Write the density defined by this model (i.e., the probability of x, with z marginalized out)

(b) (5 pts) E-Step: Compute the posterior probability $r^{(i)} = \Pr(z^{(i)} = 1 \mid x^{(i)})$

(c) (10 pts) M-Step: Update rule for μ (keeping σ_k fixed)

(d) (10 pts) M-Step: Update rule for σ_1 (keeping μ fixed)