

18-661 Introduction to Machine Learning

Clustering, Part II

Spring 2020

ECE – Carnegie Mellon University

Announcements

- **Homework 5:** deadline extended to April 3rd
- **Final exam**
 - Multiple-choice questions will be an online timed quiz during the lecture time on Apr 29th (we will take potential internet issues and timezones into account when setting the time)
 - Descriptive questions will be a take-home exam (1-2 days)
 - More details to follow
 - Please let us know asap if you have conflicting exams or need special accommodations
- Recitation this week on clustering and GMMs

1. Review: Clustering and k -means
2. Gaussian mixture models

Review: Clustering and k -means

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(x_1, y_1), \dots (x_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Supervised versus Unsupervised Learning

Supervised Learning: labeled observations $\{(x_1, y_1), \dots (x_n, y_n)\}$

- Labels 'teach' algorithm to learn mapping from observations to labels
- Examples: Classification (Logistic Reg., SVMs, Neural Nets, Nearest Neighbors, Decision Trees), Regression (Linear Reg., Neural Nets)

Unsupervised Learning: unlabeled observations $\{x_1, \dots, x_n\}$

- Learning algorithm must find latent structure from features alone
- Can be goal in itself (discover hidden patterns, exploratory analysis)
- Can be means to an end (pre-processing for supervised task)
- Examples:
 - Clustering
 - Dimensionality Reduction: Transform an initial feature representation into a more concise representation

Clustering

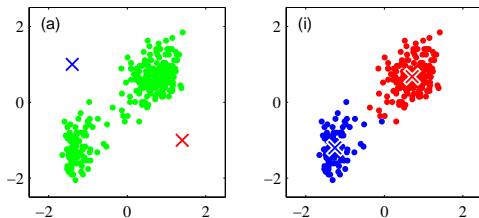
Setup Given $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ and K , we want to output:

Clustering

Setup Given $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ and K , we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters

Toy Example Cluster data into two clusters.

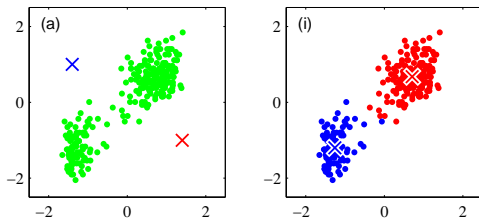


Clustering

Setup Given $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ and K , we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(\mathbf{x}_n) \in \{1, 2, \dots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

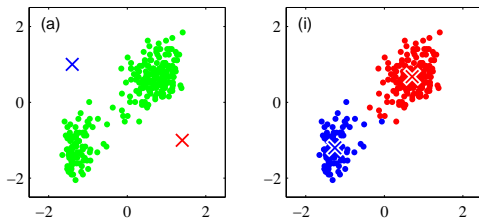


Clustering

Setup Given $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ and K , we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(\mathbf{x}_n) \in \{1, 2, \dots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.

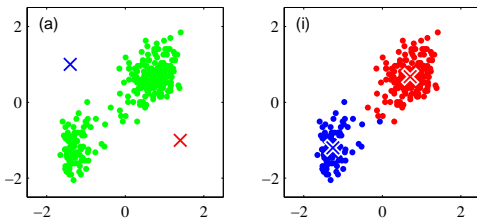


Clustering

Setup Given $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$ and K , we want to output:

- $\{\mu_k\}_{k=1}^K$: prototypes of clusters
- $A(\mathbf{x}_n) \in \{1, 2, \dots, K\}$: the cluster membership

Toy Example Cluster data into two clusters.



Example Applications

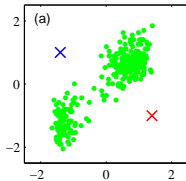
- Identify communities within social networks
- Find topic groups in news stories
- Group similar sequences into gene families

k-means: an iterative clustering method

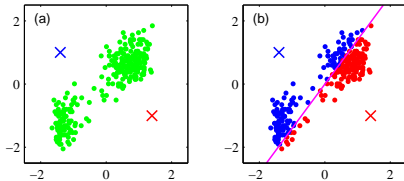
High-level idea:

- **Initialize:** Pick k random points as cluster centers, $\{\mu_1, \dots, \mu_k\}$
- **Alternate:**
 1. Assign data points to closest cluster center in $\{\mu_1, \dots, \mu_k\}$
 2. Change each cluster center to the average of its assigned points
- **Stop:** When the clusters are stable

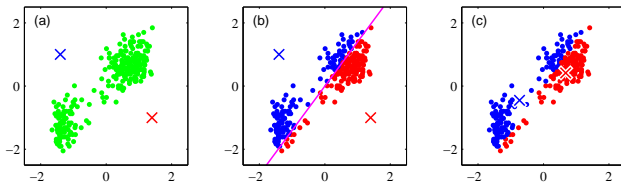
k -means example (several iterations)



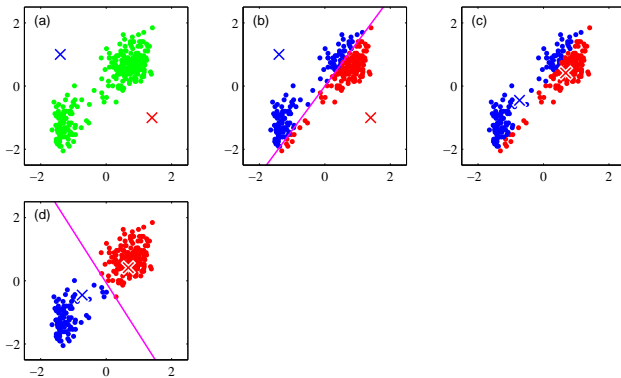
k -means example (several iterations)



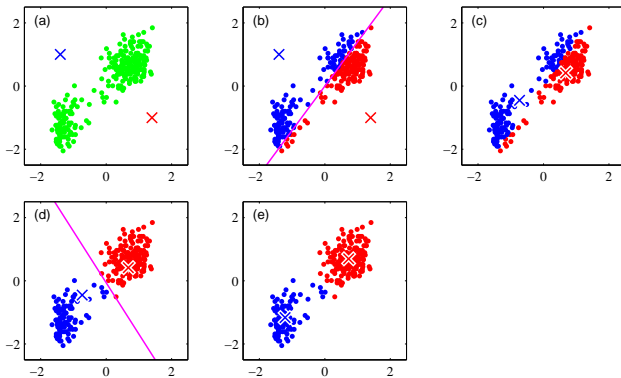
k -means example (several iterations)



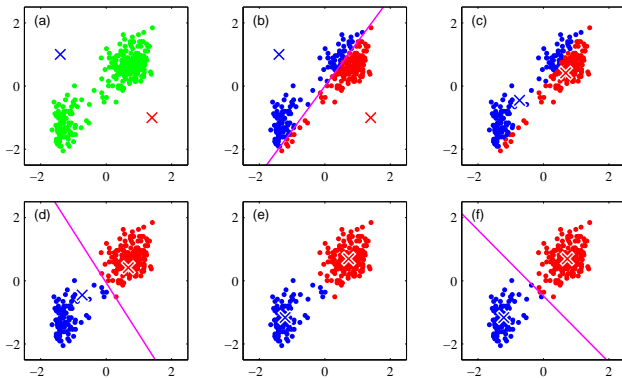
k -means example (several iterations)



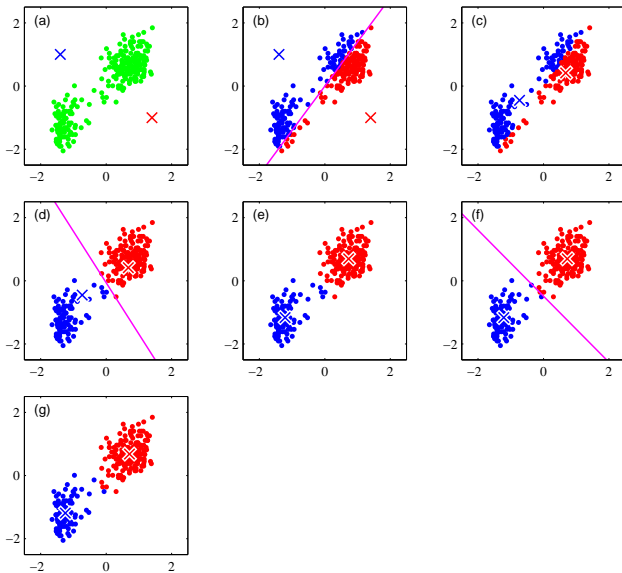
k -means example (several iterations)



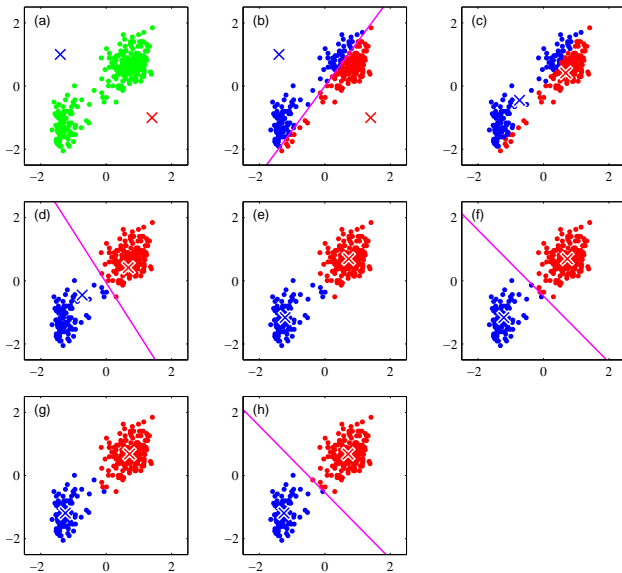
k -means example (several iterations)



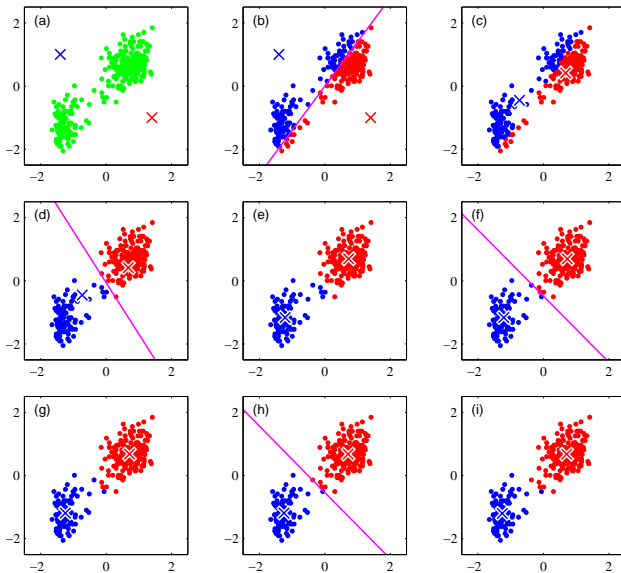
k -means example (several iterations)



k -means example (several iterations)



k -means example (several iterations)



k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype μ_k

k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype μ_k

Distortion measure: (clustering objective function, cost function)

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\mathbf{x}_n) = k$$

k-means clustering: details

Intuition: Data points assigned to cluster k should be near prototype μ_k

Distortion measure: (clustering objective function, cost function)

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\mathbf{x}_n) = k$$

Notes:

- Distance measure: $\|\mathbf{x}_n - \mu_k\|^2$ calculates how far \mathbf{x}_n is from the cluster center μ_k

k -means clustering: details

Intuition: Data points assigned to cluster k should be near prototype μ_k

Distortion measure: (clustering objective function, cost function)

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\mathbf{x}_n) = k$$

Notes:

- Distance measure: $\|\mathbf{x}_n - \mu_k\|^2$ calculates how far \mathbf{x}_n is from the cluster center μ_k
- Canonical example is the 2-norm, i.e., $\|\cdot\|_2^2$, but could be something else!

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\mu_k\}$

- **Step 0** Initialize $\{\mu_k\}$ to some values
- **Step 1** Fix $\{\mu_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j ||\mathbf{x}_n - \mu_j||^2 \\ 0 & \text{otherwise} \end{cases}$$

Why do we get this? – Try to derive it from the expression of J

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\mu_k\}$

- **Step 0** Initialize $\{\mu_k\}$ to some values
- **Step 1** Fix $\{\mu_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j ||\mathbf{x}_n - \mu_j||^2 \\ 0 & \text{otherwise} \end{cases}$$

Why do we get this? – Try to derive it from the expression of J

- **Step 2** Fix $\{r_{nk}\}$ and minimize over $\{\mu_k\}$ to get this update:

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

Why do we get this? – Try to derive it from the expression of J

Minimize distortion Alternative optimization between $\{r_{nk}\}$ and $\{\mu_k\}$

- **Step 0** Initialize $\{\mu_k\}$ to some values
- **Step 1** Fix $\{\mu_k\}$ and minimize over $\{r_{nk}\}$, to get this assignment:

$$r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j ||\mathbf{x}_n - \mu_j||^2 \\ 0 & \text{otherwise} \end{cases}$$

Why do we get this? – Try to derive it from the expression of J

- **Step 2** Fix $\{r_{nk}\}$ and minimize over $\{\mu_k\}$ to get this update:

$$\mu_k = \frac{\sum_n r_{nk} \mathbf{x}_n}{\sum_n r_{nk}}$$

Why do we get this? – Try to derive it from the expression of J

- **Step 3** Return to Step 1 unless stopping criterion is met

Properties of k -means algorithm

Does it converge?

Properties of k -means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
 - Key idea: k -means is an alternating optimization approach
 - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
 - *However*, may converge to a *local minimum* (objective is non-convex)

Properties of k -means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
 - Key idea: k -means is an alternating optimization approach
 - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
 - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

Properties of k -means algorithm

Does it converge?

- **Guaranteed to converge in a finite number of iterations**
 - Key idea: k -means is an alternating optimization approach
 - Each step is guaranteed to decrease the objective/cost function—thus guaranteed to converge
 - *However*, may converge to a *local minimum* (objective is non-convex)

What's the runtime?

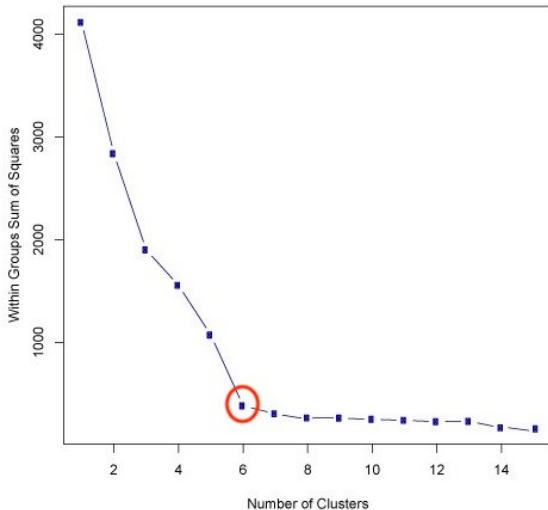
- **Running time per iteration:**
 - Assume: n data points, each with d features, and k clusters
 - Assign data points to closest cluster: $O(ndk)$
 - Re-compute cluster centers: $O(ndk)$
- **Thus, total runtime is:** $O(ndki)$, where i is the number of iterations

Practical Issues with k -means

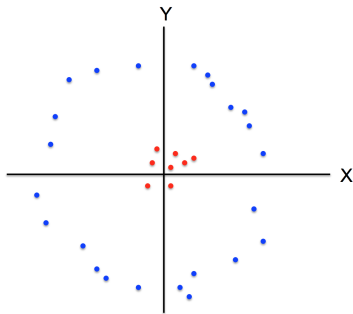
- How to select k ?
 - Prior knowledge
 - Heuristics (e.g., elbow method)
- How to select distance measure?
 - Often requires some knowledge of problem
 - Some examples: Euclidean distance (for images), Hamming distance (distance between two strings), shared key words (for websites)
- How to initialize cluster centers?
 - The final clustering can depend significantly on the initial points you pick!

Elbow method

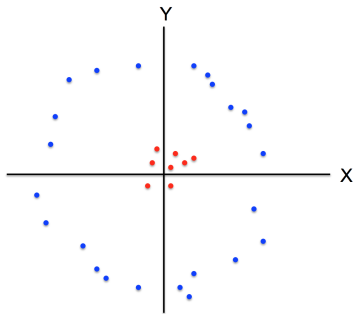
Key idea: select a small value of k that minimizes within-cluster distances



How to get k -means to work on this data?



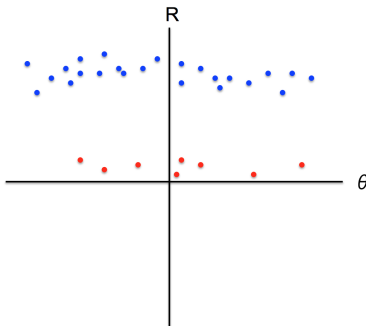
How to get k -means to work on this data?



Should look at the distance of the data points from the origin $\sqrt{x_n^2 + y_n^2}$

Distance measure

Changing features (distance measure) can help



If the cluster i mean is $(\mu_{i,x}, \mu_{i,y})$, the distance of (x_n, y_n) from it can be defined as $|\sqrt{\mu_{i,x}^2 + \mu_{i,y}^2} - \sqrt{x_n^2 + y_n^2}|$

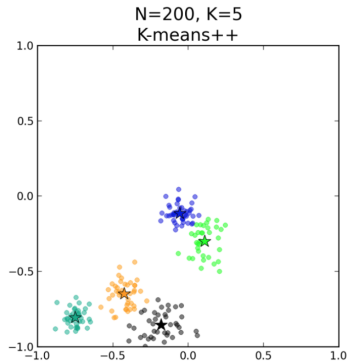
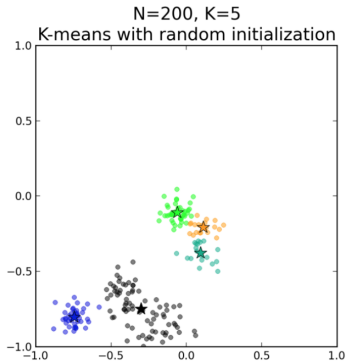
Key idea: Run k -means, but with a better initialization

- Choose center μ_1 at random
- For $j = 2, \dots, k$
 - Choose μ_j among x_1, \dots, x_n with probability:
 - $P(\mu_j = x_i) \propto \min_{j' < j} \|x_i - \mu_{j'}\|^2$

Initialization helps to get good coverage of the space

Theorem: k -means++ always obtains a $O(\log k)$ approximation to the optimal solution in expectation.

Running k -means after this initialization can only improve on the result



Connection to k -Nearest Neighbors

- Nearest Neighbors is a supervised learning method
 - Each training point \mathbf{x}_n has a corresponding given label y_n
 - Objective: Assign label to a new \mathbf{x} by looking at the labels of its k nearest points
- Clustering is an unsupervised learning method
 - We are given training points \mathbf{x}_n without labels
 - Objective: Divide them into k groups to understand patterns in the data

Clustering can make Nearest Neighbors more efficient

- A drawback of nearest neighbors is that we have to remember the training data
- Clustering can help compress the training data into a small number of representative points

Algorithm to Improve Nearest Neighbors

- For all training data points \mathbf{x}_n with label $y_n = c$, for C classes $c = 1, \dots, C$, cluster the \mathbf{x}_n into R groups.
- Store these R cluster means for each of the C classes
- For a test data point \mathbf{x} , find the k nearest neighbors among the RC cluster means and assign their majority label to \mathbf{x}

1. Review: Clustering and k -means
2. Gaussian mixture models

Gaussian mixture models

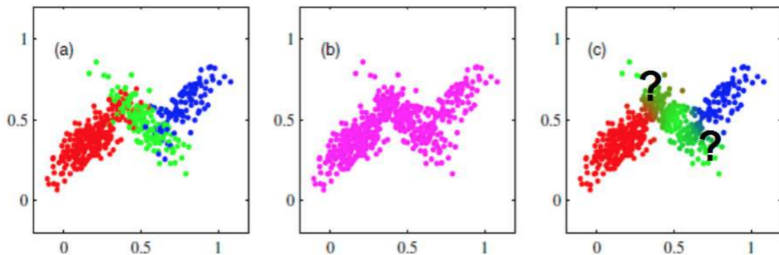
One more potential issue with k -means ...

Data points are assigned *deterministically* to one (and only one) cluster

One more potential issue with k -means ...

Data points are assigned *deterministically* to one (and only one) cluster

In reality, clusters may overlap, and it may be better to identify the *probability* that a point belongs to each cluster

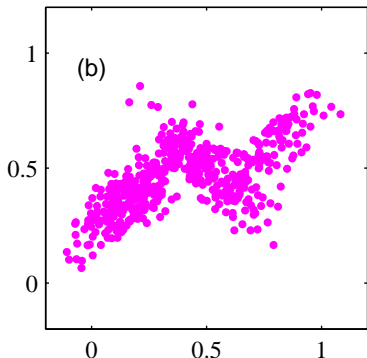


Probabilistic interpretation of clustering?

How can we model $p(\mathbf{x})$ to reflect our intuition that points stay close to their cluster centers?

Probabilistic interpretation of clustering?

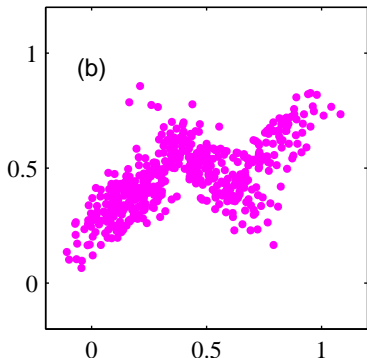
How can we model $p(\mathbf{x})$ to reflect our intuition that points stay close to their cluster centers?



- Points seem to form 3 clusters

Probabilistic interpretation of clustering?

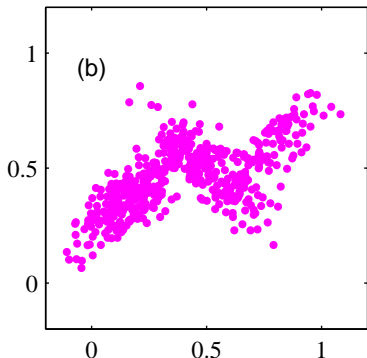
How can we model $p(\mathbf{x})$ to reflect our intuition that points stay close to their cluster centers?



- Points seem to form 3 clusters
- We cannot model $p(\mathbf{x})$ with simple and known distributions

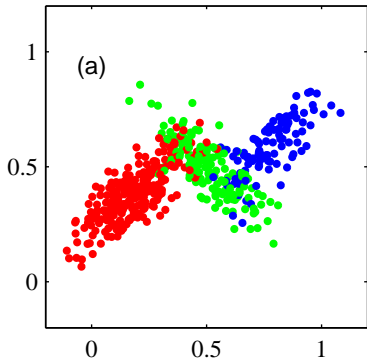
Probabilistic interpretation of clustering?

How can we model $p(\mathbf{x})$ to reflect our intuition that points stay close to their cluster centers?



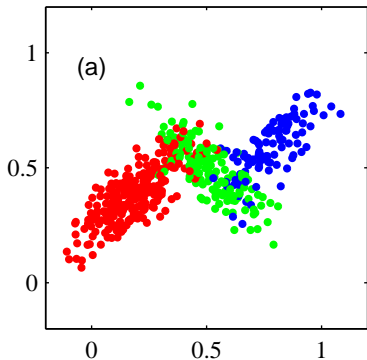
- Points seem to form 3 clusters
- We cannot model $p(\mathbf{x})$ with simple and known distributions
- E.g., the data is not a Gaussian b/c we have 3 distinct concentrated regions

Gaussian mixture models: intuition



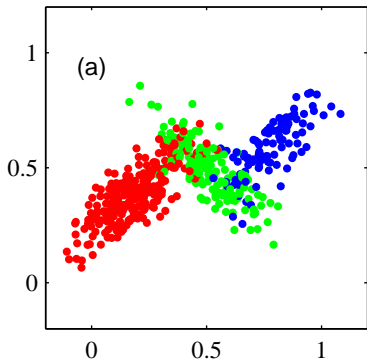
- **Key idea:** Model *each* region with a distinct distribution

Gaussian mixture models: intuition



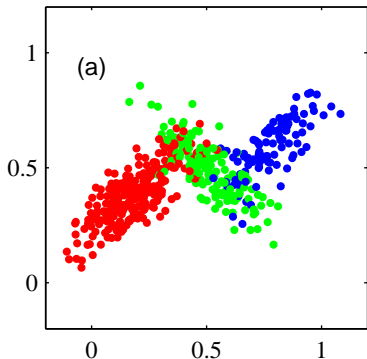
- **Key idea:** Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)

Gaussian mixture models: intuition



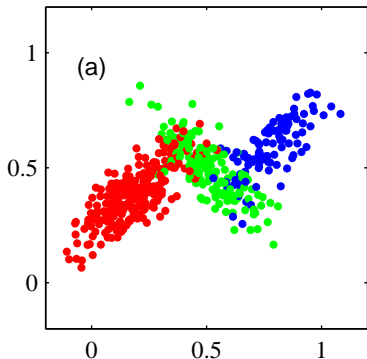
- **Key idea:** Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)

Gaussian mixture models: intuition



- **Key idea:** Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
- **However**, we don't know *cluster assignments* (label), *parameters* of Gaussians, or *mixture components*!

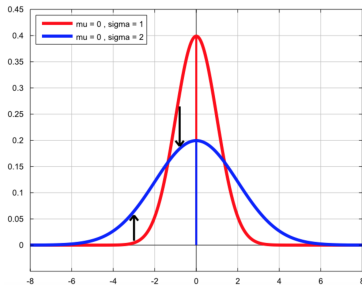
Gaussian mixture models: intuition



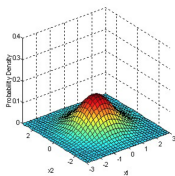
- **Key idea:** Model *each* region with a distinct distribution
- Can use Gaussians — Gaussian mixture models (GMMs)
- **However**, we don't know *cluster assignments* (label), *parameters* of Gaussians, or *mixture components*!
- Must learn from *unlabeled* data $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$

Recall: Gaussian (Normal) distributions

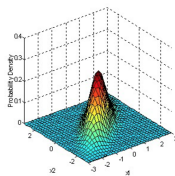
$$\mathbf{x} \sim N(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-d/2} |\boldsymbol{\Sigma}|^{-1/2} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\}$$



$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$



$$\boldsymbol{\mu} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} 1 & 0.8 \\ 0.8 & 1 \end{pmatrix}$$



Gaussian Mixture Models: Formal Definition

GMM has the following density function for \mathbf{x}

$$p(\mathbf{x}) = \sum_{k=1}^K \omega_k N(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K : number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k -th component

Gaussian Mixture Models: Formal Definition

GMM has the following density function for \mathbf{x}

$$p(\mathbf{x}) = \sum_{k=1}^K \omega_k N(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K : number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k -th component
- ω_k : mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:

Gaussian Mixture Models: Formal Definition

GMM has the following density function for \mathbf{x}

$$p(\mathbf{x}) = \sum_{k=1}^K \omega_k N(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- K : number of Gaussians — they are called mixture components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of k -th component
- ω_k : mixture weights (or priors) represent how much each component contributes to final distribution. They satisfy 2 properties:

$$\forall k, \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

These properties ensure that $p(\mathbf{x})$ is a probability density function

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\mathbf{x}, z) = p(z)p(\mathbf{x}|z)$$

where z is a discrete random variable taking values between 1 and K .

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\mathbf{x}, z) = p(z)p(\mathbf{x}|z)$$

where z is a discrete random variable taking values between 1 and K .

Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

$$p(\mathbf{x}|z = k) = N(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

$$p(\mathbf{x}, z) = p(z)p(\mathbf{x}|z)$$

where z is a discrete random variable taking values between 1 and K .

Denote

$$\omega_k = p(z = k)$$

Now, assume the conditional distributions are Gaussian distributions

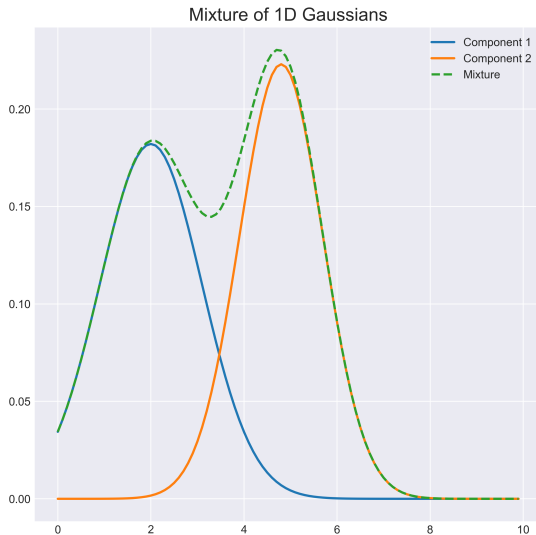
$$p(\mathbf{x}|z = k) = N(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Then, the marginal distribution of \mathbf{x} is

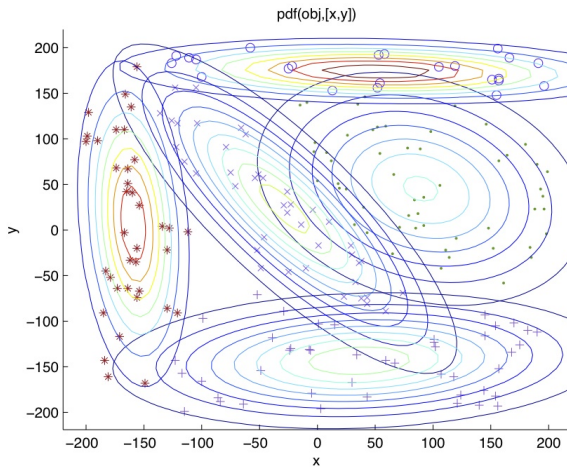
$$p(\mathbf{x}) = \sum_{k=1}^K \omega_k N(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Namely, the Gaussian mixture model

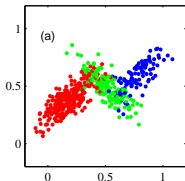
Gaussian mixtures in 1D



Gaussian mixture model for clustering



GMMs: example



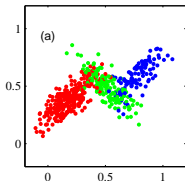
The conditional distribution between \mathbf{x} and z (representing color) are

$$p(\mathbf{x}|z = \text{red}) = N(\mathbf{x}|\mu_1, \Sigma_1)$$

$$p(\mathbf{x}|z = \text{blue}) = N(\mathbf{x}|\mu_2, \Sigma_2)$$

$$p(\mathbf{x}|z = \text{green}) = N(\mathbf{x}|\mu_3, \Sigma_3)$$

GMMs: example

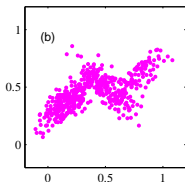


The conditional distribution between \mathbf{x} and z (representing color) are

$$p(\mathbf{x}|z = \text{red}) = N(\mathbf{x}|\mu_1, \Sigma_1)$$

$$p(\mathbf{x}|z = \text{blue}) = N(\mathbf{x}|\mu_2, \Sigma_2)$$

$$p(\mathbf{x}|z = \text{green}) = N(\mathbf{x}|\mu_3, \Sigma_3)$$



The marginal distribution is thus

$$\begin{aligned} p(\mathbf{x}) &= p(z = \text{red})N(\mathbf{x}|\mu_1, \Sigma_1) \\ &+ p(z = \text{blue})N(\mathbf{x}|\mu_2, \Sigma_2) \\ &+ p(z = \text{green})N(\mathbf{x}|\mu_3, \Sigma_3) \end{aligned}$$

Parameter estimation for Gaussian mixture models

The parameters in GMMs are

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$

Let's first consider the simple/unrealistic case where we *have labels* z

Define $\mathcal{D}' = \{\mathbf{x}_n, z_n\}_{n=1}^N$, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$

- \mathcal{D}' is the **complete** data
- \mathcal{D} the **incomplete** data

How can we learn our parameters?

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$

Let's first consider the simple/unrealistic case where we *have labels* z

Define $\mathcal{D}' = \{\mathbf{x}_n, z_n\}_{n=1}^N$, $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^N$

- \mathcal{D}' is the **complete** data
- \mathcal{D} the **incomplete** data

How can we learn our parameters?

Given \mathcal{D}' , the maximum likelihood estimation of the θ is given by

$$\theta = \arg \max \log \mathcal{D}' = \sum_n \log p(\mathbf{x}_n, z_n)$$

Parameter estimation for GMMs: complete data

The complete likelihood is decomposable

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_n \log p(z_n)p(\mathbf{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\mathbf{x}_n|z_n)$$

where we have grouped data by cluster labels z_n .

Parameter estimation for GMMs: complete data

The complete likelihood is decomposable

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_n \log p(z_n)p(\mathbf{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\mathbf{x}_n|z_n)$$

where we have grouped data by cluster labels z_n .

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

Parameter estimation for GMMs: complete data

The complete likelihood is decomposable

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_n \log p(z_n)p(\mathbf{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\mathbf{x}_n|z_n)$$

where we have grouped data by cluster labels z_n .

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_k \sum_n r_{nk} \log p(z = k)p(\mathbf{x}_n|z = k)$$

Parameter estimation for GMMs: complete data

The complete likelihood is decomposable

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_n \log p(z_n)p(\mathbf{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\mathbf{x}_n|z_n)$$

where we have grouped data by cluster labels z_n .

Let $r_{nk} \in \{0, 1\}$ be a binary variable that indicates whether $z_n = k$:

$$\begin{aligned} \sum_n \log p(\mathbf{x}_n, z_n) &= \sum_k \sum_n r_{nk} \log p(z = k)p(\mathbf{x}_n|z = k) \\ &= \sum_k \sum_n r_{nk} [\log \omega_k + \log N(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)] \end{aligned}$$

Note: in the complete setting the r_{nk} are binary, but later we will ‘relax’ these variables and allow them to take on fractional values

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_k \sum_n r_{nk} [\log \omega_k + \log N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_k \sum_n r_{nk} [\log \omega_k + \log N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

Regrouping, we have

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_k \sum_n r_{nk} \log \omega_k + \sum_k \left\{ \sum_n r_{nk} \log N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Parameter estimation for GMMs: complete data

From our previous discussion, we have

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_k \sum_n r_{nk} [\log \omega_k + \log N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]$$

Regrouping, we have

$$\sum_n \log p(\mathbf{x}_n, z_n) = \sum_k \sum_n r_{nk} \log \omega_k + \sum_k \left\{ \sum_n r_{nk} \log N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

The term inside the braces depends on k -th component's parameters. It is now easy to show that (left as an exercise) the MLE is:

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_k \sum_n r_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \mathbf{x}_n$$
$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

What's the intuition?

Since r_{nk} is binary, the previous solution is nothing but:

- ω_k : fraction of total data points whose cluster label z_n is k
 - note that $\sum_k \sum_n r_{nk} = N$
- μ_k : mean of all data points whose z_n is k
- Σ_k : co-variance of all data points whose z_n is k

Since r_{nk} is binary, the previous solution is nothing but:

- ω_k : fraction of total data points whose cluster label z_n is k
 - note that $\sum_k \sum_n r_{nk} = N$
- μ_k : mean of all data points whose z_n is k
- Σ_k : co-variance of all data points whose z_n is k

We use the knowledge of true cluster labels z_n (which imply the r_{nk}) to estimate θ .

What do we do when we *do not* know z_n (incomplete data)

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete

- Observed: $\mathcal{D} = \{\mathbf{x}_n\}$
- Unobserved (hidden): $\{\mathbf{z}_n\}$

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete

- Observed: $\mathcal{D} = \{\mathbf{x}_n\}$
- Unobserved (hidden): $\{\mathbf{z}_n\}$

Goal Obtain the maximum likelihood estimate of θ :

$$\theta = \arg \max \ell(\theta) = \arg \max \log \mathcal{D} = \arg \max \sum_n \log p(\mathbf{x}_n | \theta)$$

Parameter estimation for GMMs: Incomplete data

GMM Parameters

$$\theta = \{\omega_k, \mu_k, \Sigma_k\}_{k=1}^K$$

Incomplete Data

Our data contains observed and unobserved data, and hence is incomplete

- Observed: $\mathcal{D} = \{\mathbf{x}_n\}$
- Unobserved (hidden): $\{\mathbf{z}_n\}$

Goal Obtain the maximum likelihood estimate of θ :

$$\begin{aligned}\theta &= \arg \max \ell(\theta) = \arg \max \log \mathcal{D} = \arg \max \sum_n \log p(\mathbf{x}_n | \theta) \\ &= \arg \max \sum_n \log \sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n | \theta)\end{aligned}$$

The objective function $\ell(\theta)$ is called the *incomplete* log-likelihood.

Parameter estimation for GMMs: incomplete data

When z_n is not given, we can guess it via the **posterior probability** (recall: Bayes' rule!)

$$\begin{aligned} p(z_n = k | \mathbf{x}_n) &= \frac{p(\mathbf{x}_n | z_n = k) p(z_n = k)}{p(\mathbf{x}_n)} = \frac{p(\mathbf{x}_n | z_n = k) p(z_n = k)}{\sum_{k'=1}^K p(\mathbf{x}_n | z_n = k') p(z_n = k')} \\ &= \frac{N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \times \omega_k}{\sum_{k'=1}^K N(\mathbf{x}_n | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'}) \times \omega_{k'}} \end{aligned}$$

Parameter estimation for GMMs: incomplete data

When z_n is not given, we can guess it via the **posterior probability** (recall: Bayes' rule!)

$$\begin{aligned} p(z_n = k | \mathbf{x}_n) &= \frac{p(\mathbf{x}_n | z_n = k) p(z_n = k)}{p(\mathbf{x}_n)} = \frac{p(\mathbf{x}_n | z_n = k) p(z_n = k)}{\sum_{k'=1}^K p(\mathbf{x}_n | z_n = k') p(z_n = k')} \\ &= \frac{N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \times \omega_k}{\sum_{k'=1}^K N(\mathbf{x}_n | \boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'}) \times \omega_{k'}} \end{aligned}$$

To compute the posterior probability, we need to know the parameters $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$

Idea: Let's pretend we know these parameters so we can compute the posterior probability.

How is that going to help us?

Estimation with soft r_{nk}

We define $r_{nk} = p(z_n = k | \mathbf{x}_n)$

Estimation with soft r_{nk}

We define $r_{nk} = p(z_n = k | \mathbf{x}_n)$

- Recall that r_{nk} was previously binary
- Now it's a “soft” assignment of \mathbf{x}_n to k -th component
- Each \mathbf{x}_n is assigned to a component fractionally according to $p(z_n = k | \mathbf{x}_n)$

Estimation with soft r_{nk}

We define $r_{nk} = p(z_n = k | \mathbf{x}_n)$

- Recall that r_{nk} was previously binary
- Now it's a “soft” assignment of \mathbf{x}_n to k -th component
- Each \mathbf{x}_n is assigned to a component fractionally according to $p(z_n = k | \mathbf{x}_n)$

If we solve for the MLE of $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$ given soft r_{nk} s, we get the same expressions as before!

$$\omega_k = \frac{\sum_n r_{nk}}{\sum_k \sum_n r_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} \mathbf{x}_n$$
$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n r_{nk}} \sum_n r_{nk} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^\top$$

But remember, we're ‘cheating’ by using $\boldsymbol{\theta}$ to compute r_{nk} !

Iterative procedure

Alternate between estimating r_{nk} and computing parameters

- Step 0: initialize θ with some values (random or otherwise)
- Step 1: set $r_{nk} = p(z_n = k | \mathbf{x}_n)$ for current θ using Bayes Rule
- Step 2: update θ using these r_{nk} s using MLE
- Step 3: go back to Step 1

At the end convert r_{nk} back to binary by setting the largest r_{nk} for point \mathbf{x}_n to 1 and others to 0.

Iterative procedure

Alternate between estimating r_{nk} and computing parameters

- Step 0: initialize θ with some values (random or otherwise)
- Step 1: set $r_{nk} = p(z_n = k | \mathbf{x}_n)$ for current θ using Bayes Rule
- Step 2: update θ using these r_{nk} s using MLE
- Step 3: go back to Step 1

At the end convert r_{nk} back to binary by setting the largest r_{nk} for point \mathbf{x}_n to 1 and others to 0.

This is an example of the **EM algorithm** — a powerful procedure for model estimation with hidden/latent variables

Iterative procedure

Alternate between estimating r_{nk} and computing parameters

- Step 0: initialize θ with some values (random or otherwise)
- Step 1: set $r_{nk} = p(z_n = k | \mathbf{x}_n)$ for current θ using Bayes Rule
- Step 2: update θ using these r_{nk} s using MLE
- Step 3: go back to Step 1

At the end convert r_{nk} back to binary by setting the largest r_{nk} for point \mathbf{x}_n to 1 and others to 0.

This is an example of the **EM algorithm** — a powerful procedure for model estimation with hidden/latent variables

Connection with K -means?

Iterative procedure

Alternate between estimating r_{nk} and computing parameters

- Step 0: initialize θ with some values (random or otherwise)
- Step 1: set $r_{nk} = p(z_n = k | \mathbf{x}_n)$ for current θ using Bayes Rule
- Step 2: update θ using these r_{nk} s using MLE
- Step 3: go back to Step 1

At the end convert r_{nk} back to binary by setting the largest r_{nk} for point \mathbf{x}_n to 1 and others to 0.

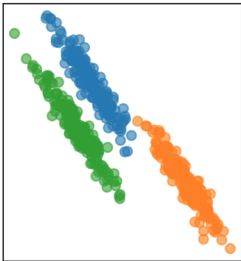
This is an example of the **EM algorithm** — a powerful procedure for model estimation with hidden/latent variables

Connection with K -means?

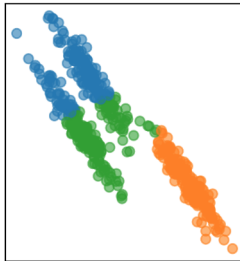
- GMMs provide probabilistic interpretation for K -means
- K -means is “hard” GMM or GMMs is “soft” K -means
- Posterior r_{nk} provides a probabilistic assignment for \mathbf{x}_n to cluster k

GMMs vs. k -means

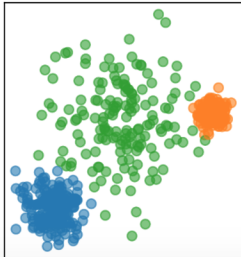
GaussianMixture



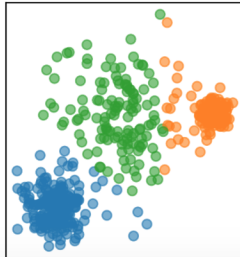
KMeans



GaussianMixture



KMeans



Pros/Cons

- k -means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering

Pros/Cons

- k -means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering
- GMMs can be more accurate, as they model more information (soft clustering, variance), but can be more expensive to compute

Pros/Cons

- k -means is a simpler, more straightforward method, but might not be as accurate because of deterministic clustering
- GMMs can be more accurate, as they model more information (soft clustering, variance), but can be more expensive to compute
- Both methods have a similar set of practical issues (having to select k , the distance, and the initialization)

What you should know . . .

- How GMMs differ from k -means (and why we care)

What you should know . . .

- How GMMs differ from k -means (and why we care)
- The difference between complete, incomplete data/likelihood

What you should know . . .

- How GMMs differ from k -means (and why we care)
- The difference between complete, incomplete data/likelihood
- How to learn the parameters in a GMM