Lecture 14: Unsupervised Learning: Clustering

Iain Styles

29 November 2019

Introduction

- Regression and Classification are *supervised*

Introduction

- Regression and Classification are *supervised*
- What if we have no labels?

Introduction

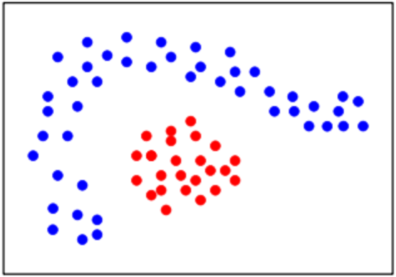
- Regression and Classification are *supervised*
- What if we have no labels?
- Can we "discover" concepts/structure in data?

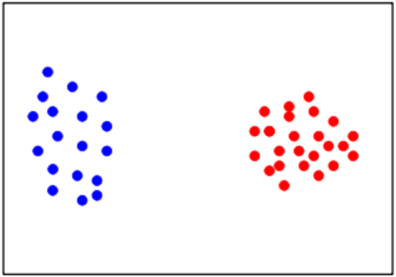
Introduction

- Regression and Classification are *supervised*
- What if we have no labels?
- Can we "discover" concepts/structure in data?
- *Unsupervised learning*

Introduction

- Regression and Classification are *supervised*
- What if we have no labels?
- Can we "discover" concepts/structure in data?
- *Unsupervised learning*
- Main approach is *clustering*





General Approaches to Clustering

- Hundreds of clustering algorithms,
- Many assume something about the data, eg clusters are normally distributed.

General Approaches to Clustering

- Hundreds of clustering algorithms,
- Many assume something about the data, eg clusters are normally distributed.
- Most are based on grouping points that are "similar"
- Outcome is sets (groups) of points that by some measure "belong together".

General Approaches to Clustering

- Hundreds of clustering algorithms,
- Many assume something about the data, eg clusters are normally distributed.
- Most are based on grouping points that are "similar"
- Outcome is sets (groups) of points that by some measure "belong together".
- Common approaches include:
  - Vector quantisation
  - Agglomerative approaches
  - Mixture modelling

The k-means Algorithm

- By far the most common clustering algorithm

The k-means Algorithm

- By far the most common clustering algorithm
- Learns a prototype vector for each class in the data

The k-means Algorithm

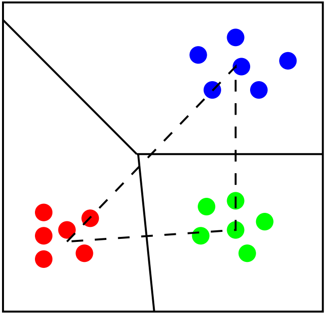
- By far the most common clustering algorithm
- Learns a prototype vector for each class in the data
- Assigns all data points to their nearest prototype

The *k*-means Algorithm

- By far the most common clustering algorithm
- Learns a prototype vector for each class in the data
- Assigns all data points to their nearest prototype
- Data space is partitioned into cells, one per prototype
- *Voronoi Tesselation* of the space.

The k-means Algorithm

- By far the most common clustering algorithm
- Learns a prototype vector for each class in the data
- Assigns all data points to their nearest prototype
- Data space is partitioned into cells, one per prototype
- *Voronoi Tesselation* of the space.
- Prototype vectors are often known as the cluster *centroids*.



The k-means Algorithm

- k-means is a parametric algorithm
- Number of clusters k has to be specified

The k-means Algorithm

- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):

The k-means Algorithm

- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):
    1) First, randommly assign every data point to one of the clusters.

The k-means Algorithm

- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):
  1) First, randommly assign every data point to one of the clusters.
  2) Compute the centroid (mean position) of each cluster

The k-means Algorithm

- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):
  1) First, randommly assign every data point to one of the clusters.
  2) Compute the centroid (mean position) of each cluster
  3) For each data point, calculate distance to centroid of each cluster

The k-means Algorithm

- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):
  1) First, randommly assign every data point to one of the clusters.
  2) Compute the centroid (mean position) of each cluster
  3) For each data point, calculate distance to centroid of each cluster
  4) Re-assign points to the cluster with the closest centroid.

The k-means Algorithm

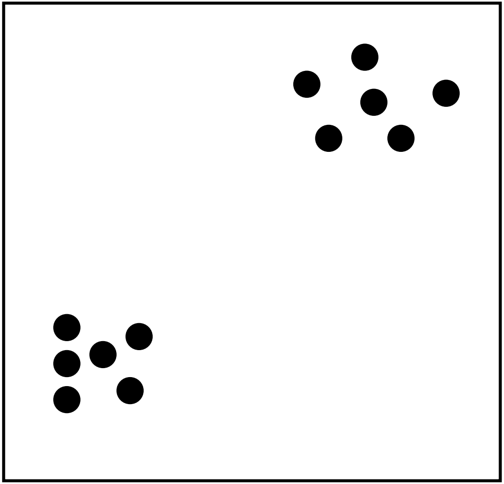
- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):
  1) First, randommly assign every data point to one of the clusters.
  2) Compute the centroid (mean position) of each cluster
  3) For each data point, calculate distance to centroid of each cluster
  4) Re-assign points to the cluster with the closest centroid.
  5) Return to step 2 and repeat until the clusters are not changing.

The k-means Algorithm

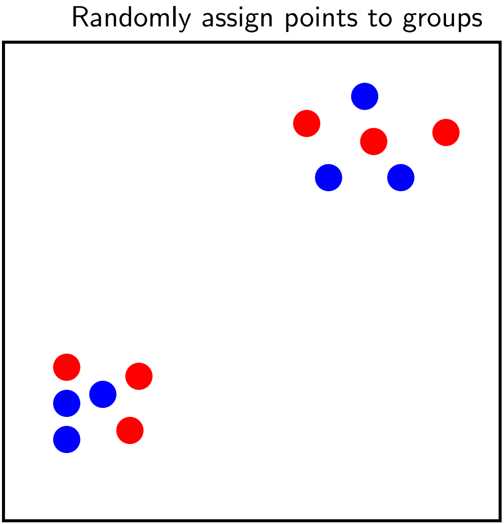
- k-means is a parametric algorithm
- Number of clusters k has to be specified
- Most common approach is *lloyd's Algorithm* (naïve k-means)
- Algorithm is (roughly):
    1) First, randommly assign every data point to one of the clusters.
    2) Compute the centroid (mean position) of each cluster
    3) For each data point, calculate distance to centroid of each cluster
    4) Re-assign points to the cluster with the closest centroid.
    5) Return to step 2 and repeat until the clusters are not changing.

The k-means Algorithm

Initial data points

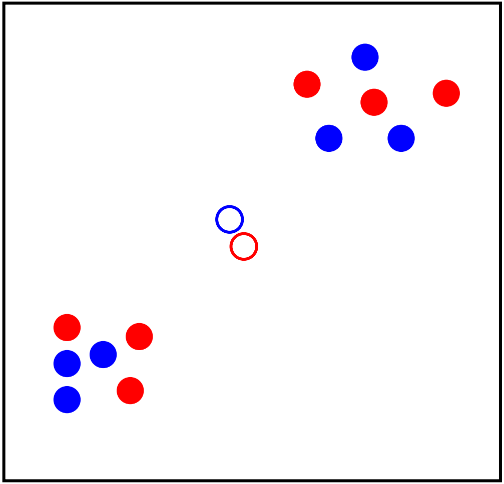


The k-means Algorithm



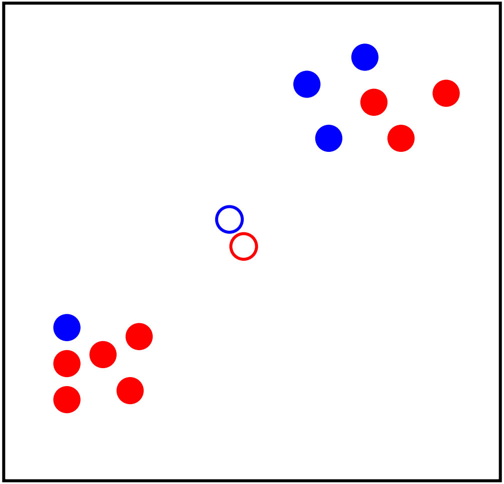
The k-means Algorithm

Compute group average

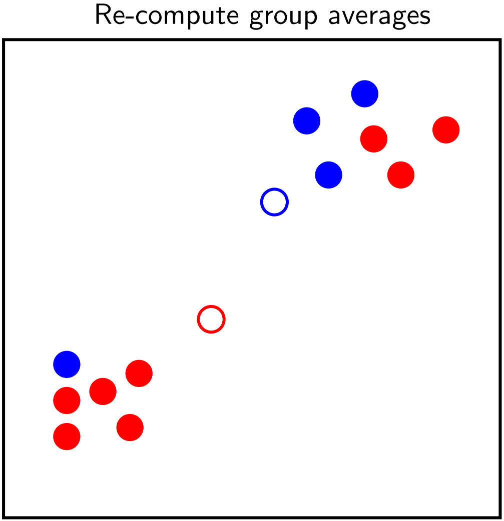


The k-means Algorithm

Re-assign points to groups

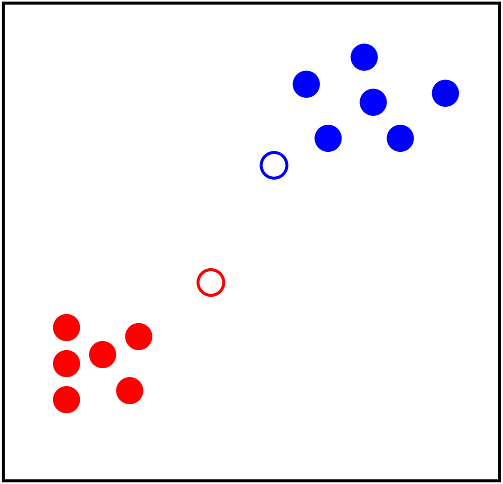


The k-means Algorithm



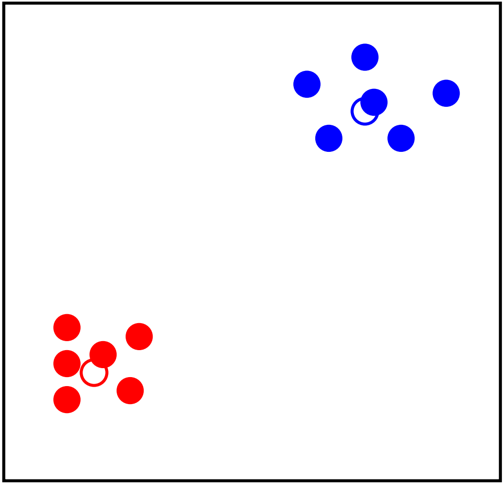
The k-means Algorithm

Re-assign points to groups



The k-means Algorithm

Re-compute group averages



Notes about k-means

There are a few points to note:

- Initial random seeding $\rightarrow$ repeats needed

Notes about k-means

There are a few points to note:

- Initial random seeding $\rightarrow$ repeats needed
- Have to "guess" the number of clusters: repeat and evaluate
  - Too many, and you will split what should be a single cluster.
  - Too few, and you will combine clusters that should be separate, or splitting clusters and merge with others.

Notes about k-means

There are a few points to note:

- Initial random seeding $\rightarrow$ repeats needed
- Have to "guess" the number of clusters: repeat and evaluate
  - Too many, and you will split what should be a single cluster.
  - Too few, and you will combine clusters that should be separate, or splitting clusters and merge with others.
- Does not work for clusters which are close together and/or are non-spherical.
- k-means is $\mathcal{O}(kN)$
  - Can be very costly for large datasets with high numbers of clusters.

Notes about k-means

There are a few points to note:

- Initial random seeding $\rightarrow$ repeats needed
- Have to "guess" the number of clusters: repeat and evaluate
  - Too many, and you will split what should be a single cluster.
  - Too few, and you will combine clusters that should be separate, or splitting clusters and merge with others.
- Does not work for clusters which are close together and/or are non-spherical.
- k-means is $\mathcal{O}(kN)$
  - Can be very costly for large datasets with high numbers of clusters.

https://colab.research.google.com/drive/1t4zdBUUBM_8I0x_ADRTSKpeR6Y2RrE0D

Agglomerative Clustering

- k-means is a top-down algorithm.
- Clusters are "flat" with no structure.
- Change $k \rightarrow$ must recluster.
- Strict convex partitioning not appropriate for many types of data.

Agglomerative Clustering

- k-means is a top-down algorithm.
- Clusters are "flat" with no structure.
- Change $k \rightarrow$ must recluster.
- Strict convex partitioning not appropriate for many types of data.
- Hierarchical clustering is an alternative "bottom-up" strategy
- Build hierarcy of relationships between data points

Agglomerative Clustering

- k-means is a top-down algorithm.
- Clusters are "flat" with no structure.
- Change $k \rightarrow$ must recluster.
- Strict convex partitioning not appropriate for many types of data.
- Hierarchical clustering is an alternative "bottom-up" strategy
- Build hierarcy of relationships between data points
- All degrees of cluster can be extracted from this

Agglomerative Clustering

1) Compute distances between all pairs of data points

Agglomerative Clustering

1) Compute distances between all pairs of data points
2) Find the closest pair
3) Remove this pair from the dataset and replace it with the grouped pair.

Agglomerative Clustering

1) Compute distances between all pairs of data points
2) Find the closest pair
3) Remove this pair from the dataset and replace it with the grouped pair.
4) Recalculate distances between all points and the new group using a linkage strategy.

Agglomerative Clustering

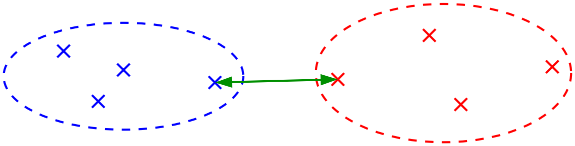
1) Compute distances between all pairs of data points
2) Find the closest pair
3) Remove this pair from the dataset and replace it with the grouped pair.
4) Recalculate distances between all points and the new group using a linkage strategy.
5) Go to 2) and continue grouping until all points are grouped

Linkage

- How do we measure similarities between groups?

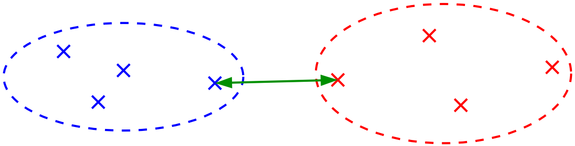
Linkage

- How do we measure similarities between groups?
  - Single Linkage

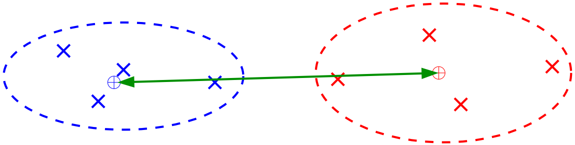


Linkage

- How do we measure similarities between groups?
  - Single Linkage

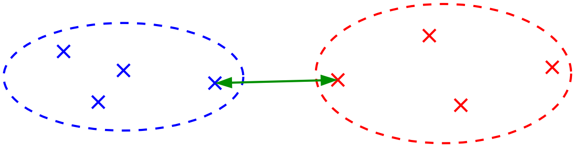


  - Average Linkage

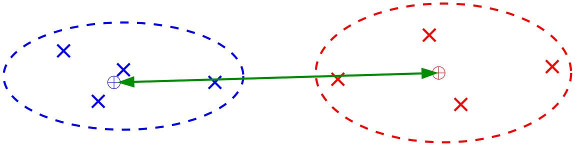


Linkage

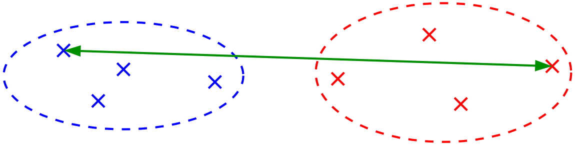
- How do we measure similarities between groups?
  - Single Linkage



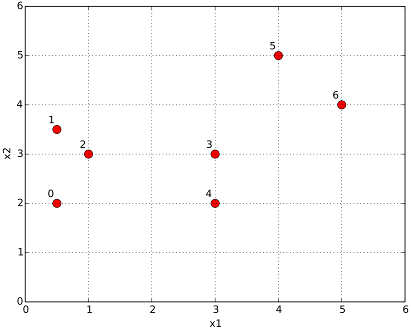
  - Average Linkage



  - Complete Linkage

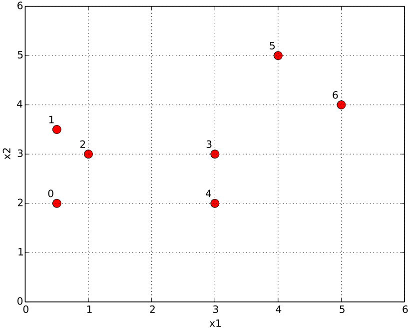


Agglomerative Clustering



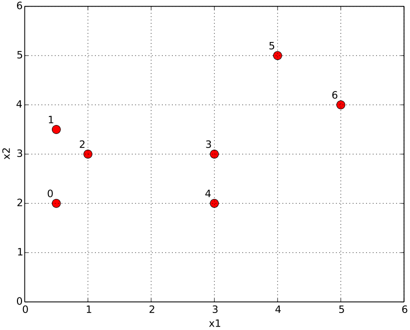
List of points 0,1,2,3,4,5,6

Agglomerative Clustering



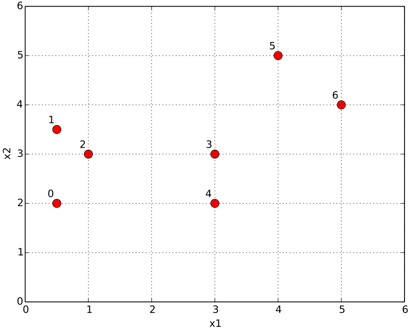
List of points		0,1,2,3,4,5,6
Group 1 with 2	$\mapsto$	0,3,4,5,6,(1,2)

Agglomerative Clustering



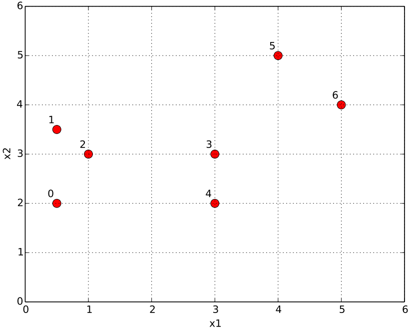
List of points		0,1,2,3,4,5,6
Group 1 with 2	$\mapsto$	0,3,4,5,6,(1,2)
Group 3 with 4	$\mapsto$	0,5,6,(1,2),(3,4)

Agglomerative Clustering



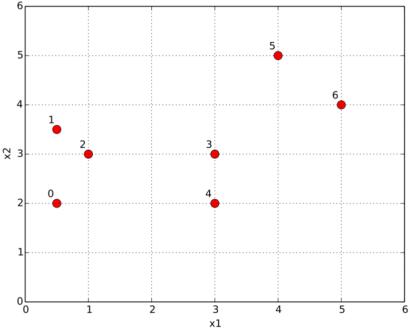
List of points		0,1,2,3,4,5,6
Group 1 with 2	$\mapsto$	0,3,4,5,6,(1,2)
Group 3 with 4	$\mapsto$	0,5,6,(1,2),(3,4)
Group 0 with (1,2)	$\mapsto$	5,6,(3,4),(0,(1,2))

Agglomerative Clustering



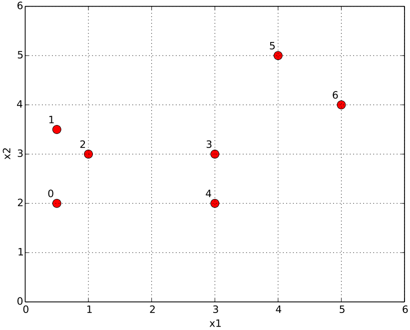
List of points		0,1,2,3,4,5,6
Group 1 with 2	$\mapsto$	0,3,4,5,6,(1,2)
Group 3 with 4	$\mapsto$	0,5,6,(1,2),(3,4)
Group 0 with (1,2)	$\mapsto$	5,6,(3,4),(0,(1,2))
Group 5 with 6	$\mapsto$	(3,4),(0,(1,2)),(5,6)

Agglomerative Clustering



List of points		0,1,2,3,4,5,6
Group 1 with 2	$\mapsto$	0,3,4,5,6,(1,2)
Group 3 with 4	$\mapsto$	0,5,6,(1,2),(3,4)
Group 0 with (1,2)	$\mapsto$	5,6,(3,4),(0,(1,2))
Group 5 with 6	$\mapsto$	(3,4),(0,(1,2)),(5,6)
Group (3,4) with (0,(1,2))	$\mapsto$	(5,6),((3,4),(0,(1,2)))

Agglomerative Clustering



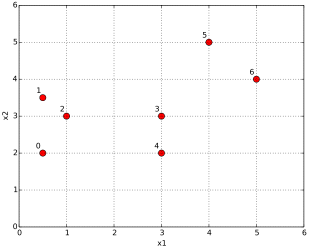
List of points		0,1,2,3,4,5,6
Group 1 with 2	$\mapsto$	0,3,4,5,6,(1,2)
Group 3 with 4	$\mapsto$	0,5,6,(1,2),(3,4)
Group 0 with (1,2)	$\mapsto$	5,6,(3,4),(0,(1,2))
Group 5 with 6	$\mapsto$	(3,4),(0,(1,2)),(5,6)
Group (3,4) with (0,(1,2))	$\mapsto$	(5,6),((3,4),(0,(1,2)))
Group (5,6) with ((3,4),(0,(1,2)))	$\mapsto$	((5,6),((3,4),(0,(1,2))))

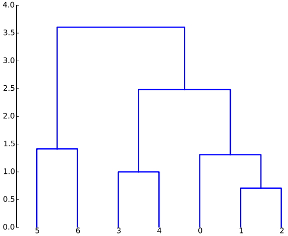
Agglomerative Clustering

- Represent as a **dendrogram** where the height of each "junction" in the dendrogram represents the distance
  - Eg. $(3, 4)$ join at height 1.

Agglomerative Clustering

- Represent as a **dendrogram** where the height of each "junction" in the dendrogram represents the distance
  - Eg. $(3, 4)$ join at height 1.





Agglomerative Clustering

- Generates all numbers of clusters.
  - Cut horizontally across the dendrogram.
- But many computations
  - $\mathcal{O}(N^2)$ calculations just for first pairing.

Agglomerative Clustering

- Generates all numbers of clusters.
  - Cut horizontally across the dendrogram.
- But many computations
  - $\mathcal{O}(N^2)$ calculations just for first pairing.
- k-means faster if you know how many clusters, but if you don't will have to run multiple times and for multiple repeats.

Agglomerative Clustering

- Generates all numbers of clusters.
  - Cut horizontally across the dendrogram.
- But many computations
  - $\mathcal{O}(N^2)$ calculations just for first pairing.
- k-means faster if you know how many clusters, but if you don't will have to run multiple times and for multiple repeats.
- They will NOT normally give identical results: choice depends on nature of data.

Agglomerative Clustering

- Generates all numbers of clusters.
  - Cut horizontally across the dendrogram.
- But many computations
  - $\mathcal{O}(N^2)$ calculations just for first pairing.
- k-means faster if you know how many clusters, but if you don't will have to run multiple times and for multiple repeats.
- They will NOT normally give identical results: choice depends on nature of data.

https://colab.research.google.com/drive/12EDJyTk7XH9_0kEHbjieKTWw0hUNjo-9

Mixture Models

- k-means and hierarchical clustering are heuristic
- "Hard" clustering assignments
- Weak statistical assumptions
- No data model makes theory difficult

Mixture Models

- k-means and hierarchical clustering are heuristic
- “Hard” clustering assignments
- Weak statistical assumptions
- No data model makes theory difficult
- A statistically principled method would be good:
- Mixture Models

Mixture Models

- k-means and hierarchical clustering are heuristic
- "Hard" clustering assignments
- Weak statistical assumptions
- No data model makes theory difficult
- A statistically principled method would be good:
- Mixture Models
- Assume data is generated by a statistical process that is a mixture of components

Mixture Models

- k-means and hierarchical clustering are heuristic
- "Hard" clustering assignments
- Weak statistical assumptions
- No data model makes theory difficult
- A statistically principled method would be good:
- Mixture Models
- Assume data is generated by a statistical process that is a mixture of components
- Clustering: which component generated the data point

Gaussian Mixture Models

- We will consider Gaussian Mixture Models

Gaussian Mixture Models

- We will consider Gaussian Mixture Models
- Assume a PDF

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

Gaussian Mixture Models

- We will consider Gaussian Mixture Models
- Assume a PDF

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

- Clustering is then formulated as
  1. Learn the parameters of the GMM which best describe the data.
  2. Determine from which component a data point is most likely to have been generated.

Learning Gaussian Mixture Models

- Some observations about $p(\mathbf{x}) = \sum_{k=1}^{K} A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$

Learning Gaussian Mixture Models

- Some observations about $p(\mathbf{x}) = \sum_{k=1}^{K} A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Since $\mathcal{N}(\cdot)$ and $p(\mathbf{x})$ are both normalised and strictly positive it can easily be shown that the *mixing coefficients* satisfy

$$\sum_{k=1}^{K} A_k = 1 \quad \text{and} \quad 0 \leq A_k \leq 1 \tag{2}$$

Learning Gaussian Mixture Models

- Some observations about $p(\mathbf{x}) = \sum_{k=1}^{K} A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$
- Since $\mathcal{N}(\cdot)$ and $p(\mathbf{x})$ are both normalised and strictly positive it can easily be shown that the *mixing coefficients* satisfy

$$\sum_{k=1}^{K} A_k = 1 \quad \text{and} \quad 0 \leq A_k \leq 1 \tag{2}$$

Learning Gaussian Mixture Models

- We also note that by sum and product rules

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k) \quad (3)$$

Learning Gaussian Mixture Models

- We also note that by sum and product rules

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k) \tag{3}$$

and by comparison with

$$p(\mathbf{x}) = \sum_{k=1}^{K} A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4}$$

Learning Gaussian Mixture Models

- We also note that by sum and product rules

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(k)p(\mathbf{x}|k) \tag{3}$$

and by comparison with

$$p(\mathbf{x}) = \sum_{k=1}^{K} A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{4}$$

we interpret

- $A_k = p(k)$ as the prior probability of choosing a point from component k
- $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = p(\mathbf{x}|k)$ as the class-conditional likelihoods.

Learning Gaussian Mixture Models

- Finally, using Bayes' theorem we compute the posterior *responsibilities*

$$
\begin{aligned}
r_k(\mathbf{x}) &= p(k|\mathbf{x}) && (5) \\
&= \frac{p(k)p(\mathbf{x}|k)}{\sum_{k'=1}^{K} p(k')p(\mathbf{x}|k')} && (6) \\
&= \frac{A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} A_{k'} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} && (7)
\end{aligned}
$$

Learning Gaussian Mixture Models

- Finally, using Bayes' theorem we compute the posterior *responsibilities*

$$
\begin{aligned}
r_k(\mathbf{x}) &= p(k|\mathbf{x}) && (5)\\
&= \frac{p(k)p(\mathbf{x}|k)}{\sum_{k'=1}^{K} p(k')p(\mathbf{x}|k')} && (6)\\
&= \frac{A_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} A_{k'} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} && (7)
\end{aligned}
$$

- Probability that component k explains $\mathbf{x}$
- GMM gives *soft* cluster assignments

https://colab.research.google.com/drive/1Gta0oTu_86UFkAMHqrhpQ7EdKrgmaPXW