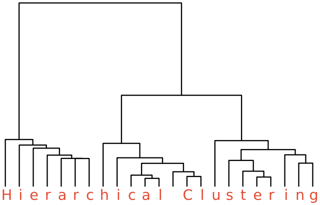


CSE 151A

Intro to Machine Learning

Lecture 16 – Part 01

Gaussian Mixtures

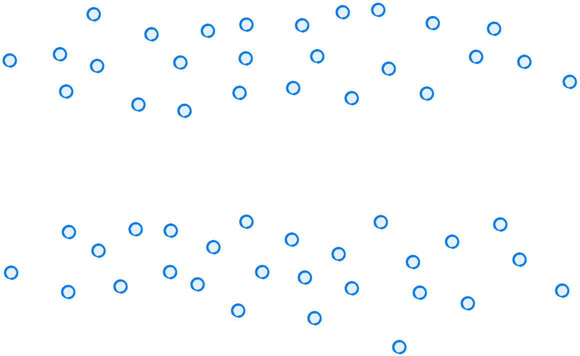
Announcements

- Please submit regrade requests for yellows!
- No class on Monday.

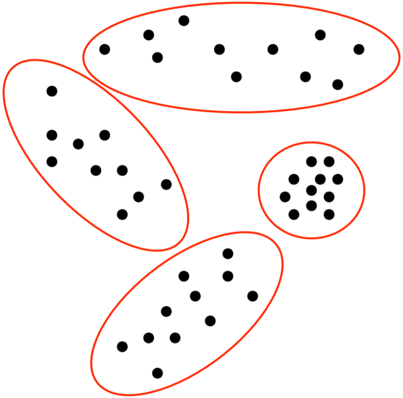
K-Means

- Perhaps the most popular clustering algorithm.
- **Fast, easy to understand**.
- **Assumes spherical clusters**.

Example



Mixtures of Gaussians



Each cluster is specified by:

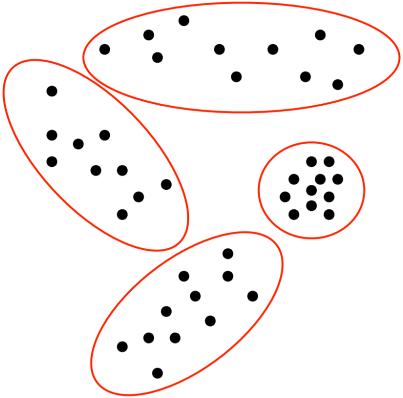
- a Gaussian $P_i = \mathcal{N}(\vec{\mu}^{(i)}, C_i)$
- a mixing weight π_i

Mixture distribution:

$$\mathbb{P}(\vec{x}) = \sum_{i=1}^{k} \pi_i P_i(\vec{x})$$

Interpretation

- **Soft-assignment**: each point belongs to multiple Gaussians



Responsibility of cluster j for point i:

$$w_{ij} = \mathbb{P}(\text{cluster } j \,|\, \vec{x}^{(i)})$$

$$= \frac{\pi_j \mathbb{P}_j(\vec{x}^{(i)})}{\sum_\ell \pi_\ell \mathbb{P}_\ell(\vec{x}^{(i)})}$$

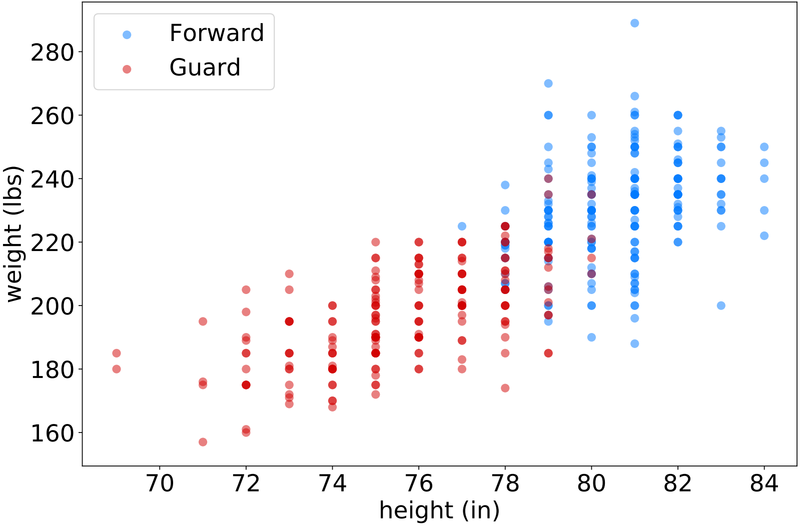
Fitting

- Recall how we fit a multivariate Gaussian.

$$\vec{\mu} = \frac{1}{n} \sum_{i=1}^{n} \vec{x}^{(i)}$$

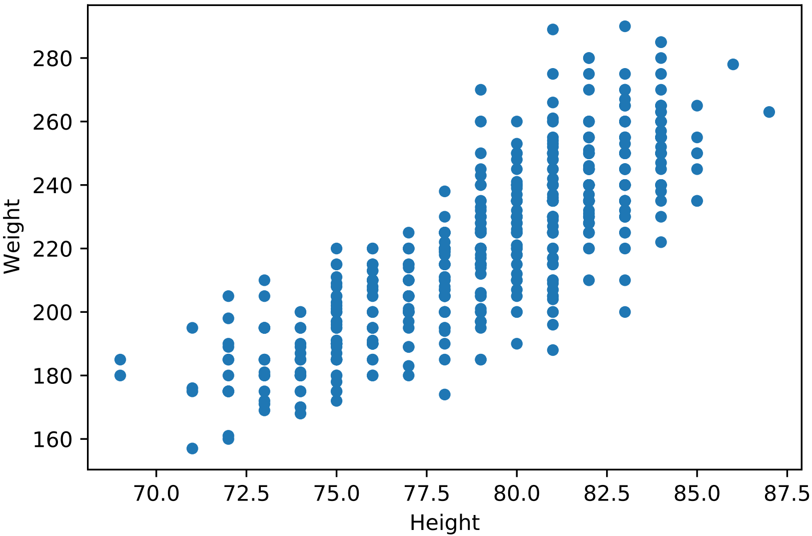
$$C = \frac{1}{n} \sum_{i=1}^{n} (\vec{x}^{(i)} - \vec{\mu})(\vec{x}^{(i)} - \vec{\mu})^T$$

Fitting



Forward
Guard
weight (lbs)
280
260
240
220
200
180
160
70
72
74
76
78
80
82
84
height (in)

Fitting



280
260
240
220
200
180
160
Weight
70.0
72.5
75.0
77.5
80.0
82.5
85.0
87.5
Height

Fitting a Mixture

- Now to fit jth Gaussian with responsibilities w_{ij}:

$$\vec{\mu}^{(j)} = \frac{1}{\sum_{i=1}^{n} w_{ij}} \sum_{i=1}^{n} w_{ij} \vec{x}^{(i)}$$

$$C_j = \frac{1}{\sum_{i=1}^{n} w_{ij}} \sum_{i=1}^{n} w_{ij} (\vec{x}^{(i)} - \vec{\mu}^{(j)})(\vec{x}^{(i)} - \vec{\mu}^{(j)})^T$$

$$\pi_j = \frac{1}{n} \sum_{i=1}^{n} w_{ij}$$

Problem

- To calculate $\vec{\mu}^{(j)}$, C_j, π_j we need responsibilities w_{ij}.
- But to calculate responsibilities, we need $\vec{\mu}^{(j)}$, π_j, C_j.

Idea

- Guess $\vec{\mu}^{(j)}$, π_j, and C_j

- Use these guesses to calculate responsibilities (i.e., make a **soft assignment**):

$$w_{ij} = \frac{\pi_j \mathbb{P}_j(\vec{x}^{(i)})}{\sum_\ell \pi_\ell \mathbb{P}_\ell(\vec{x}^{(i)})}$$

- Then update $\vec{\mu}^{(j)}$, π_j, C_j using w_{ij}. Repeat.

The EM Algorithm

- Initialize $\pi_1, \dots \pi_j$, $\vec{\mu}^{(1)}, \dots, \vec{\mu}^{(k)}$, $C_1, \dots, C_k$
- Repeat until convergence:
  - Make soft assignment (update responsibilities):

$$w_{ij} = \frac{\pi_j \mathbb{P}_j(\vec{x}^{(i)})}{\sum_\ell \pi_\ell \mathbb{P}_\ell(\vec{x}^{(i)})}$$

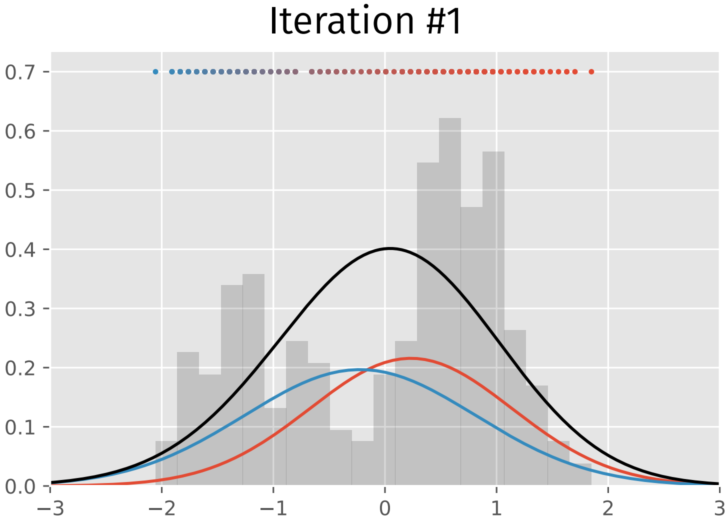
  - Update mixing weights, means, covariances:

$$\vec{\mu}^{(j)} = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} \vec{x}^{(i)}$$

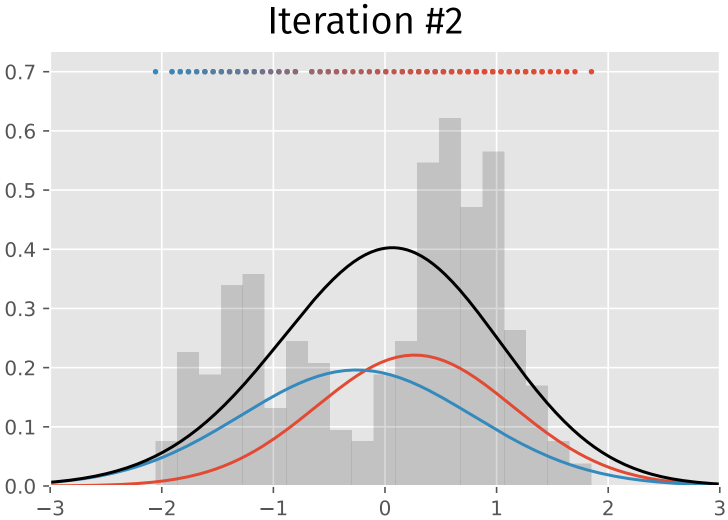
$$C_j = \frac{1}{\sum_{i=1}^n w_{ij}} \sum_{i=1}^n w_{ij} (\vec{x}^{(i)} - \vec{\mu}^{(j)})(\vec{x}^{(i)} - \vec{\mu}^{(j)})^T$$

$$\pi_j = \frac{1}{n} \sum_{i=1}^n w_{ij}$$

E-M Algorithm Demo

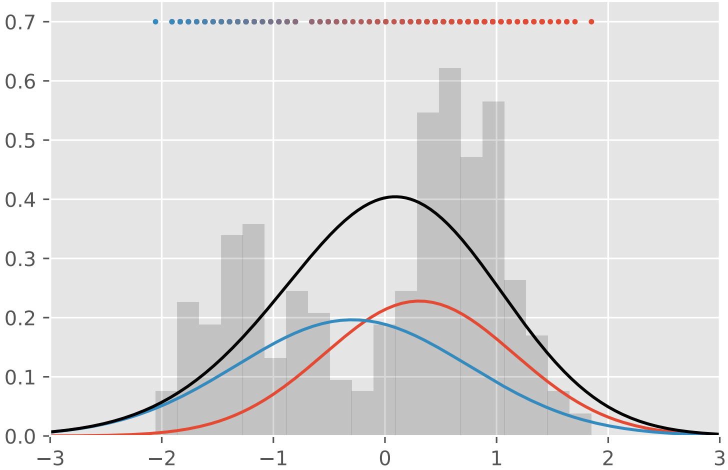


E-M Algorithm Demo

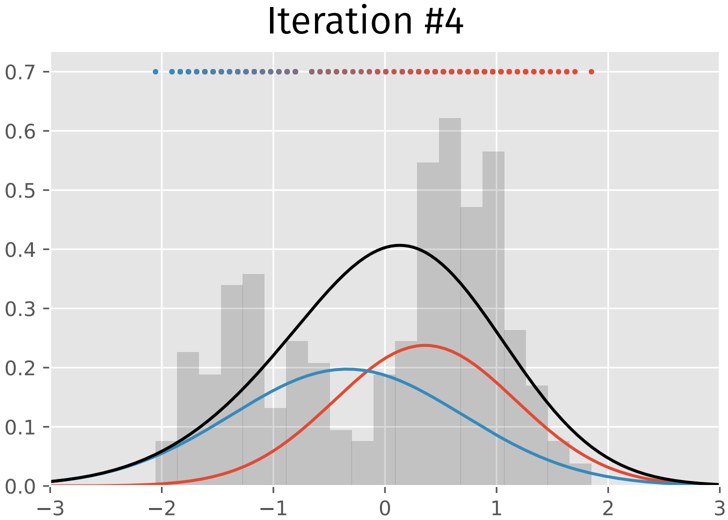


E-M Algorithm Demo

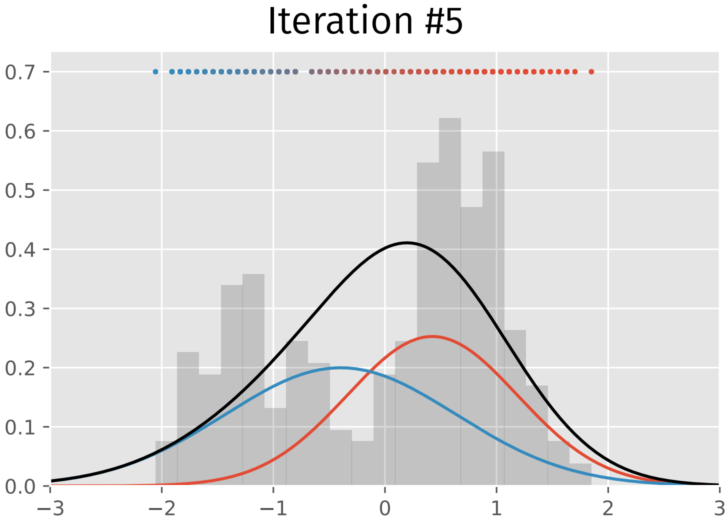
Iteration #3



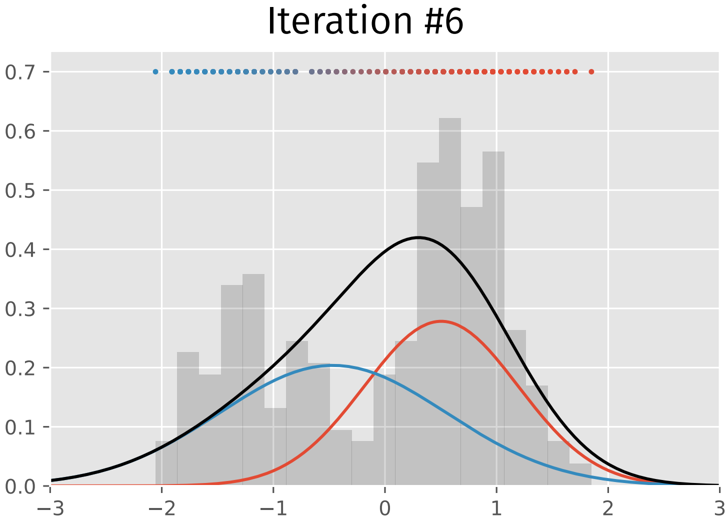
E-M Algorithm Demo



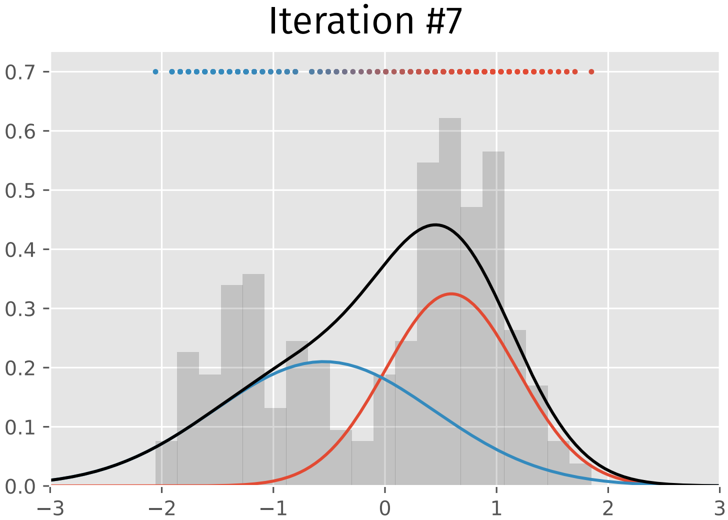
E-M Algorithm Demo



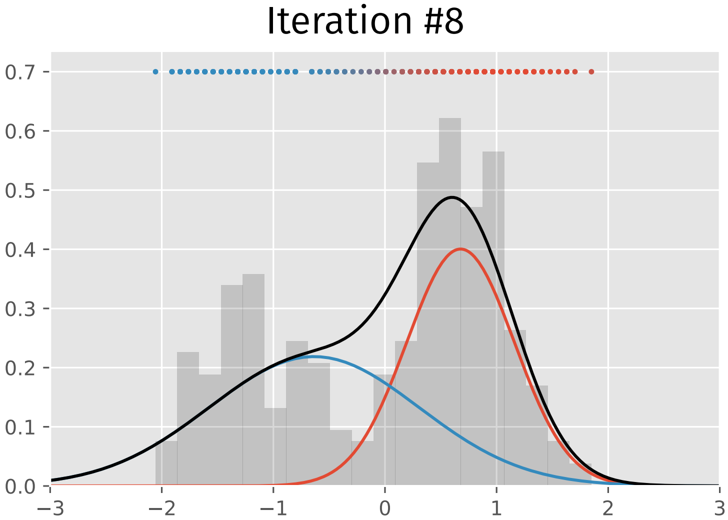
E-M Algorithm Demo



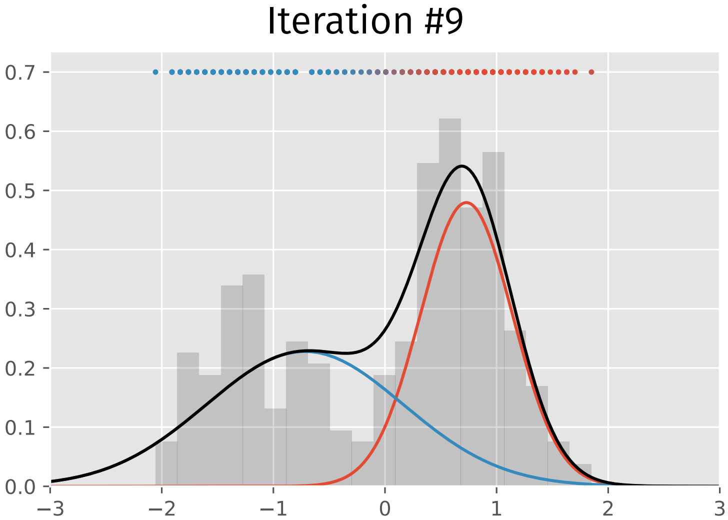
E-M Algorithm Demo



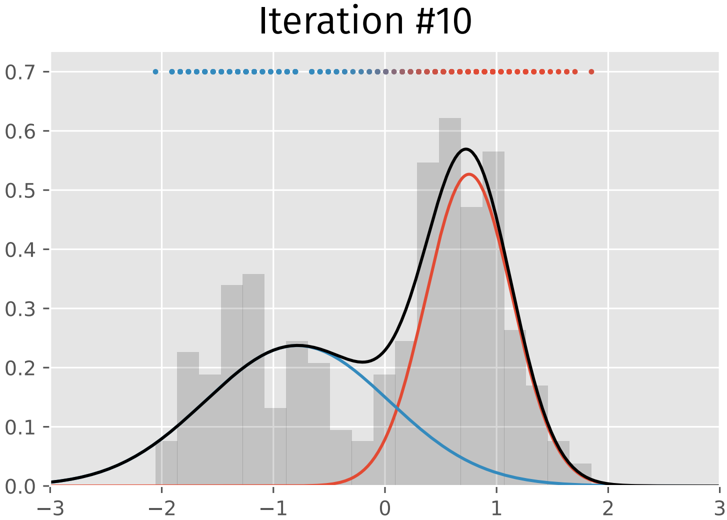
E-M Algorithm Demo



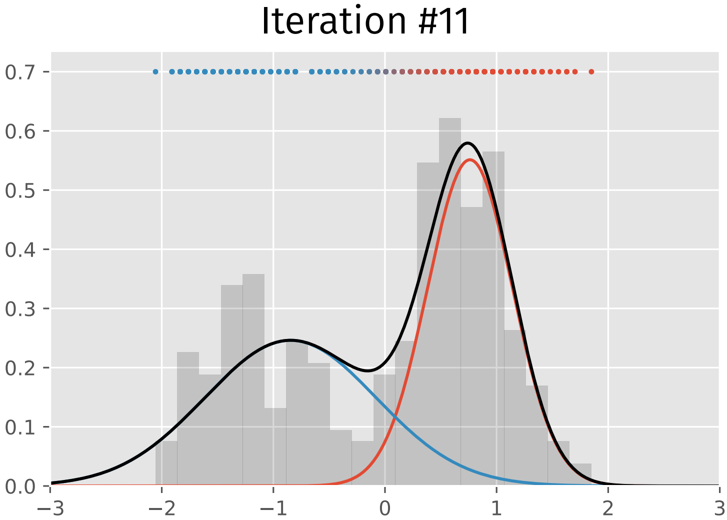
E-M Algorithm Demo



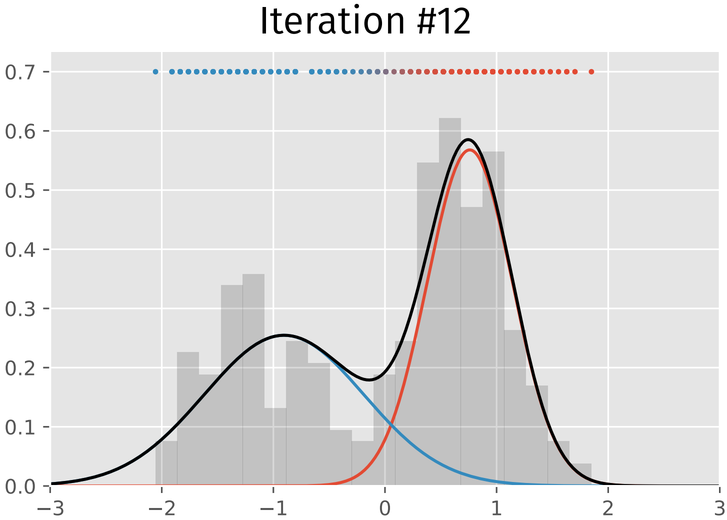
E-M Algorithm Demo



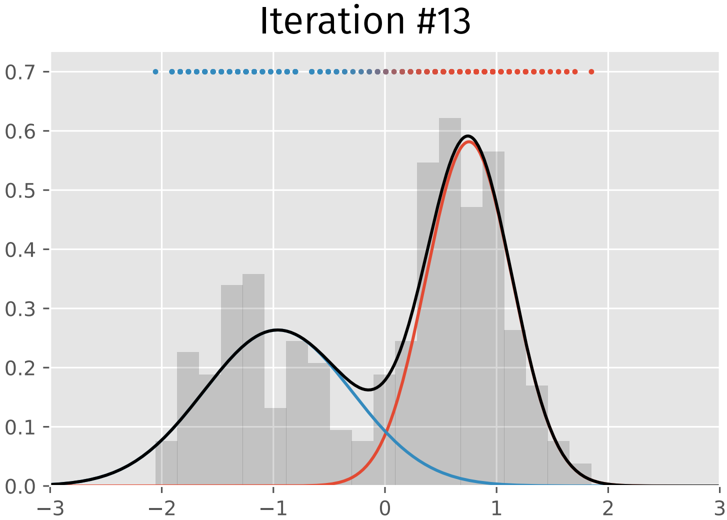
E-M Algorithm Demo



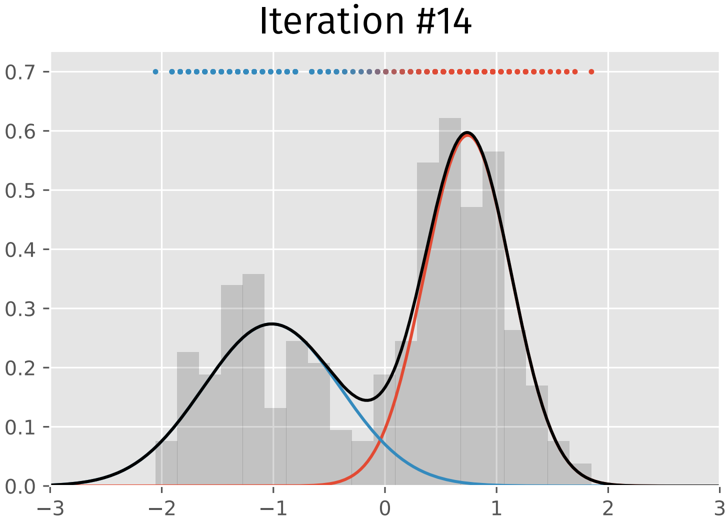
E-M Algorithm Demo



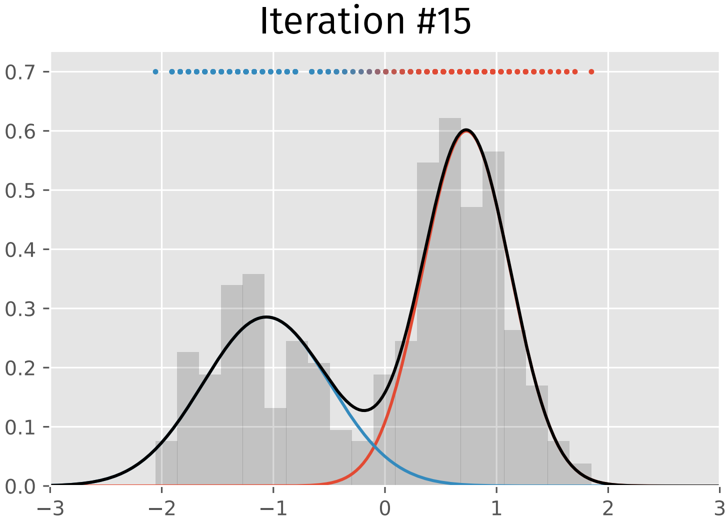
E-M Algorithm Demo



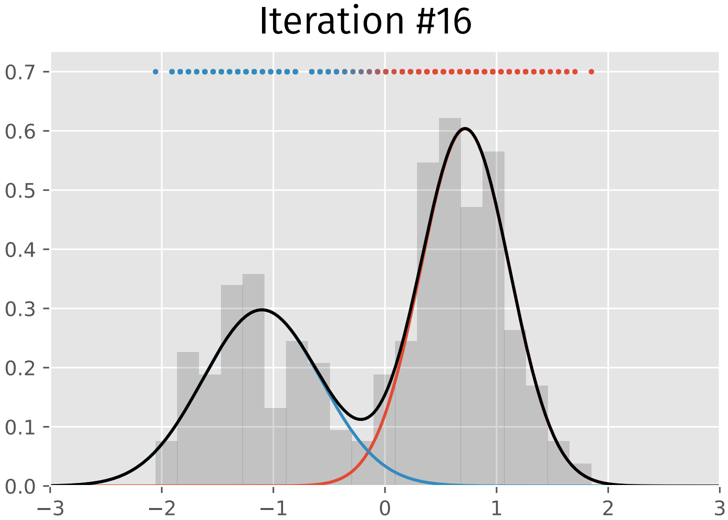
E-M Algorithm Demo



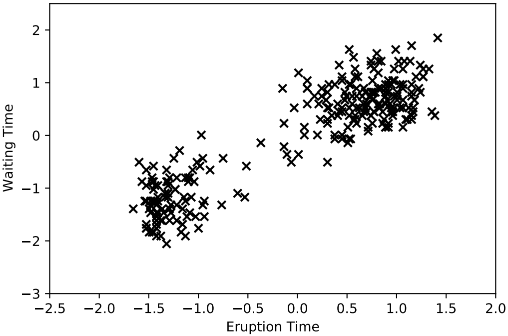
E-M Algorithm Demo



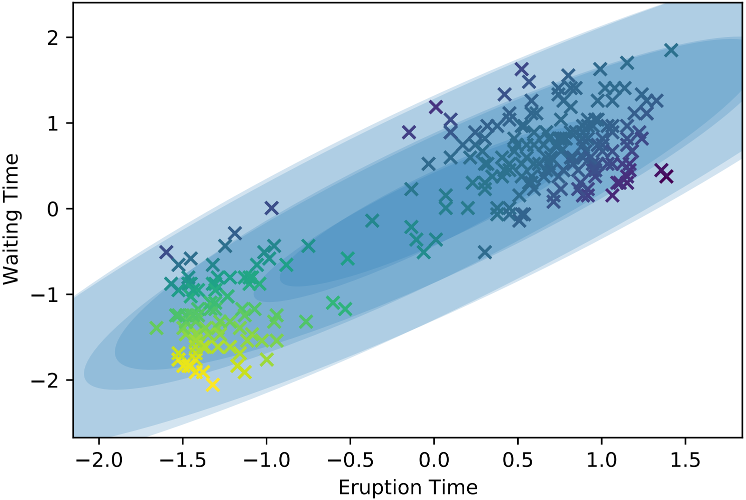
E-M Algorithm Demo



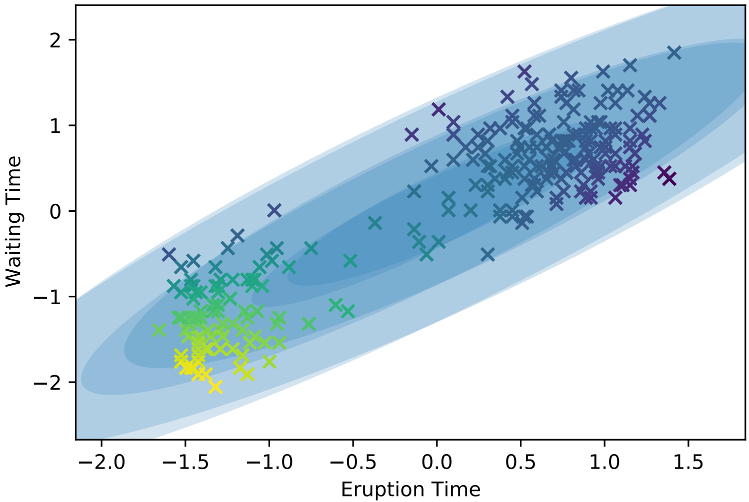
Geyser Eruptions



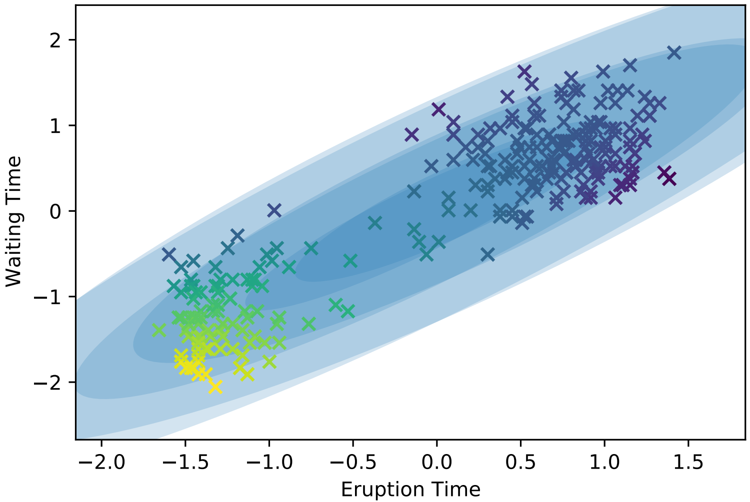
Iteration #1



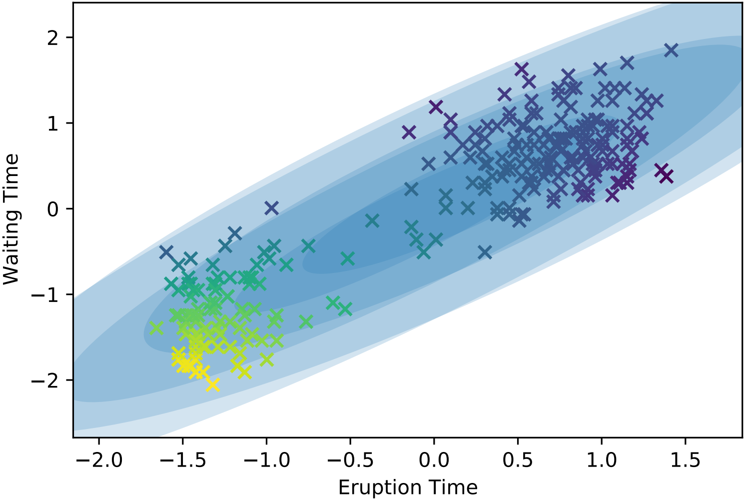
Iteration #2



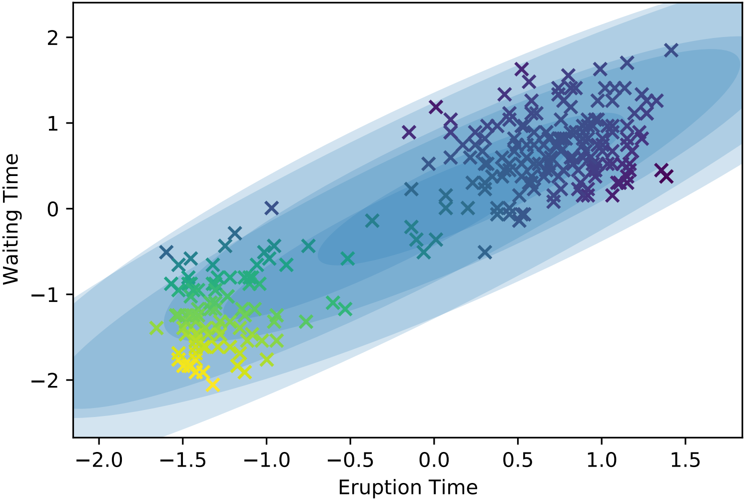
Iteration #3



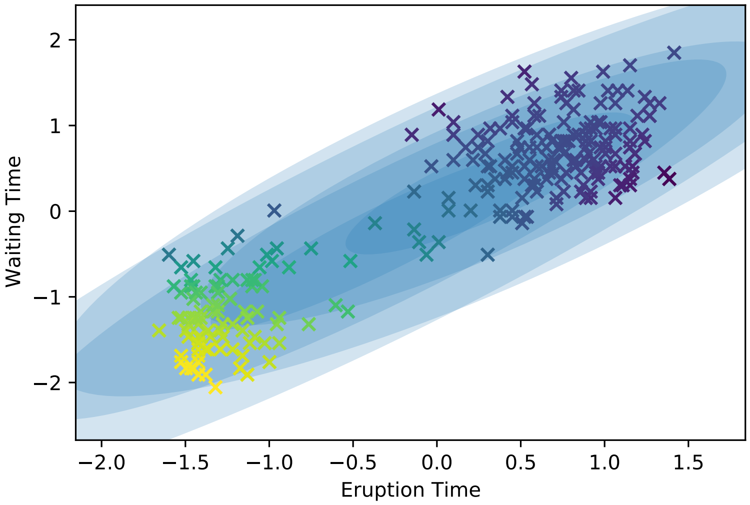
Iteration #4



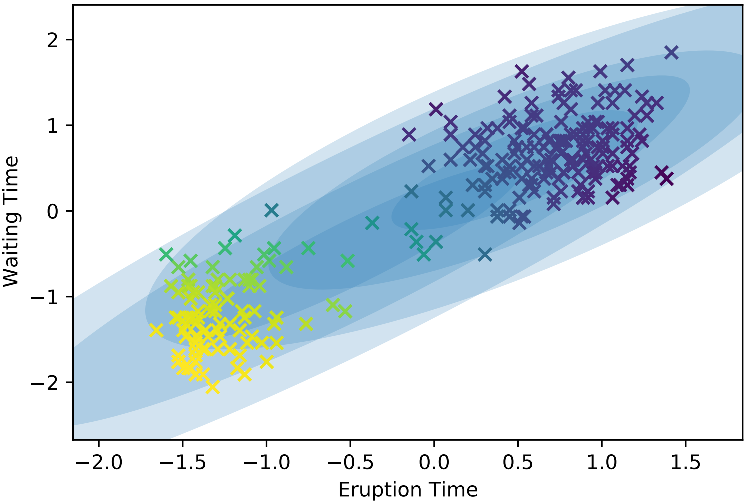
Iteration #5



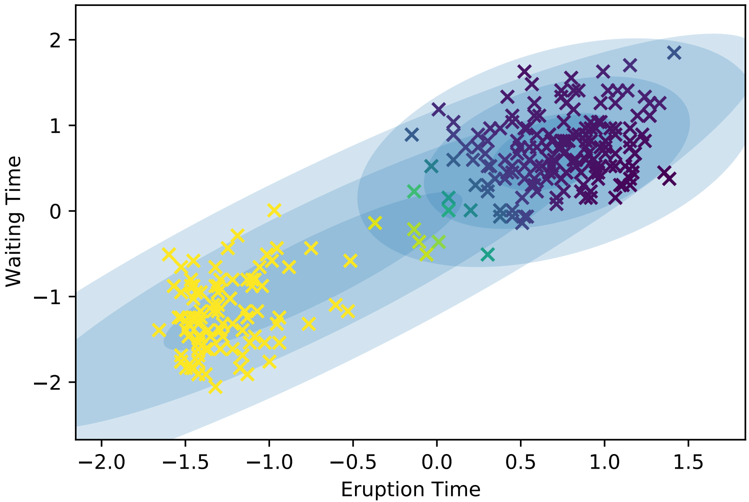
Iteration #6



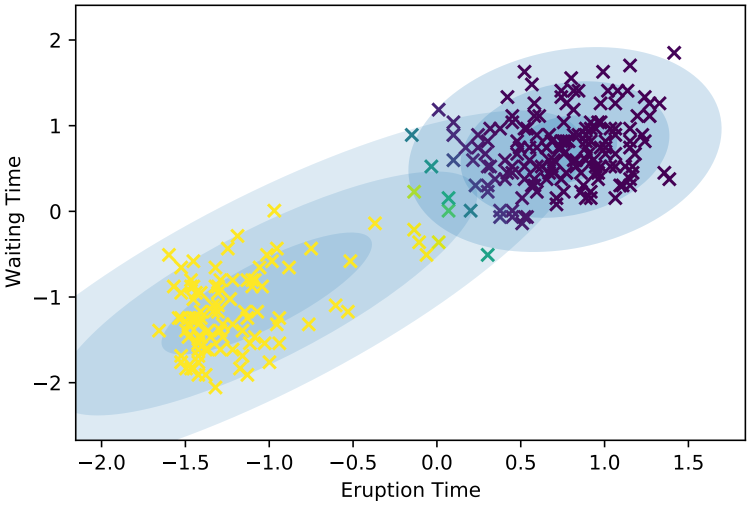
Iteration #7



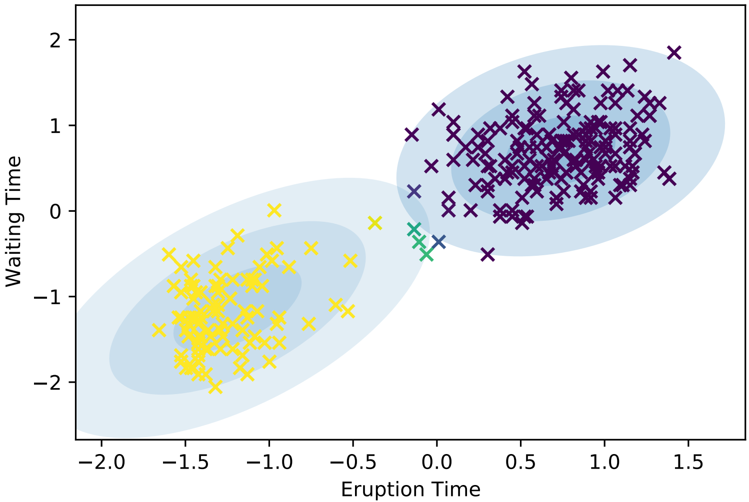
Iteration #8



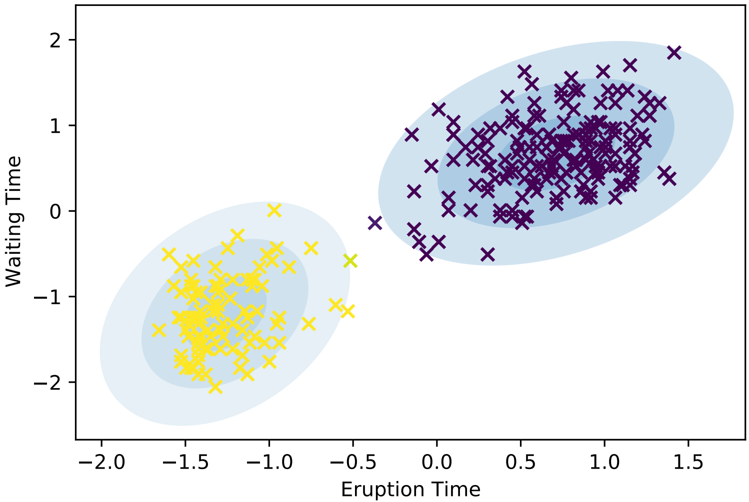
Iteration #9



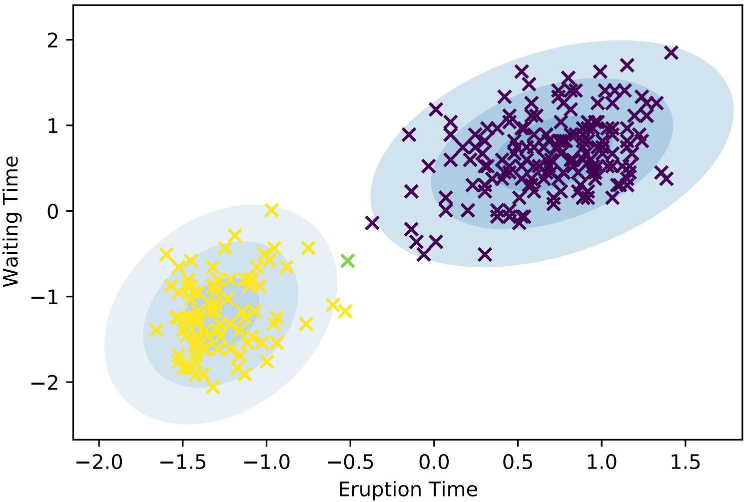
Iteration #10

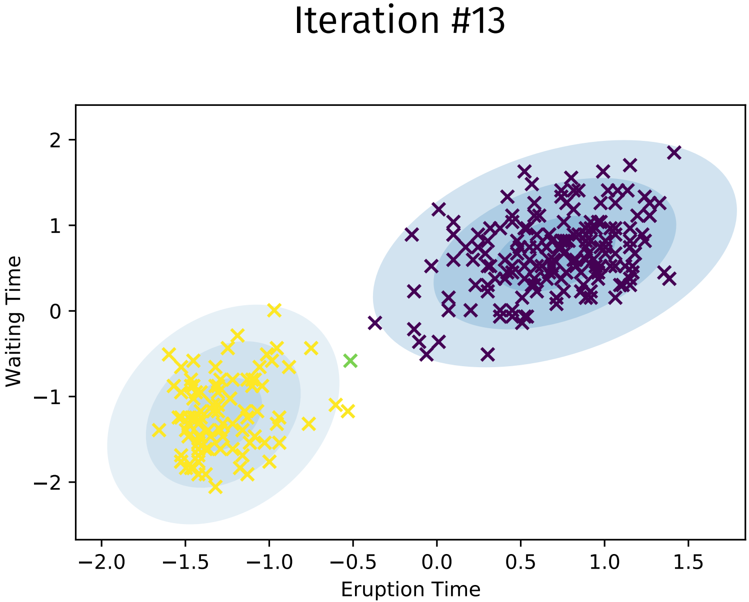


Iteration #11



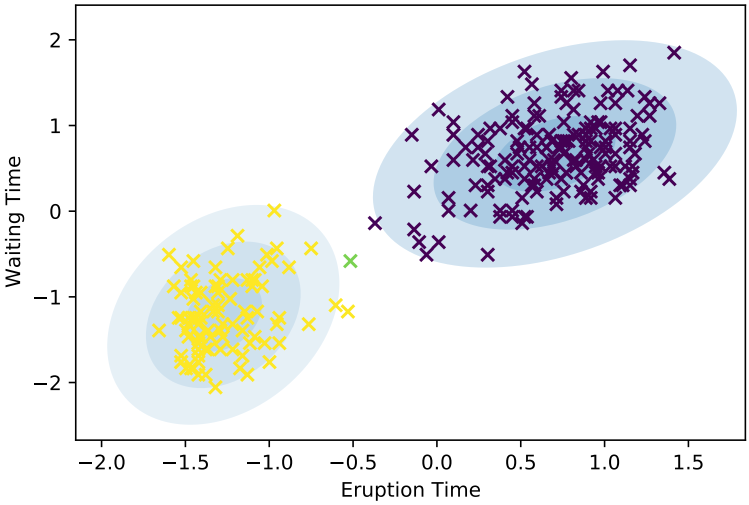
Iteration #12





Iteration #13
Waiting Time
Eruption Time
2
1
0
−1
−2
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Iteration #14

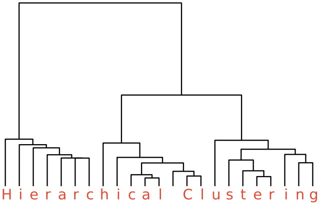


Clustering with EM

- Like with LDA/QDA, can assume spherical, diagonal, full covariance.
- May require many initializations.
- One way to initialize: k-means.

K-Means and EM

- K-Means is a limit case of EM!
- Spherical Gaussians, variance $\to 0$.



CSE 151A

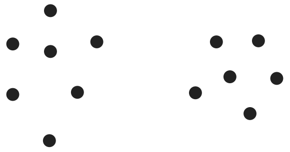
Intro to Machine Learning

Lecture 16 – Part 02

Hierarchical Clustering

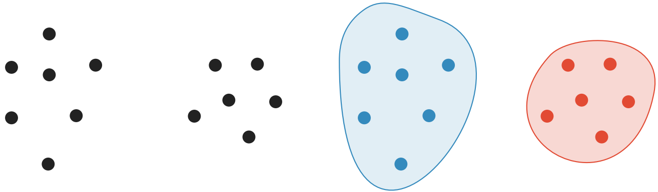
The goal of clustering:

Identify **structure** in data by grouping it into **clusters**.

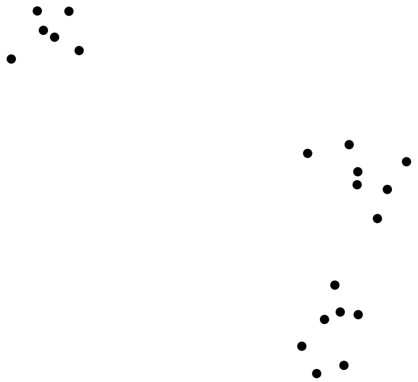


Flat Clustering

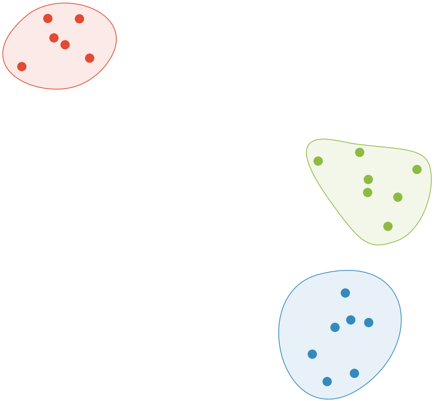
Partitioning of $\mathcal{X}$ into **disjoint** sets called **clusters** s.t. each point $x \in \mathcal{X}$ is in exactly one cluster.



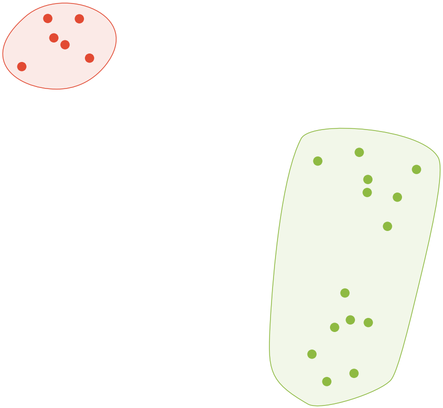
How many clusters are there?



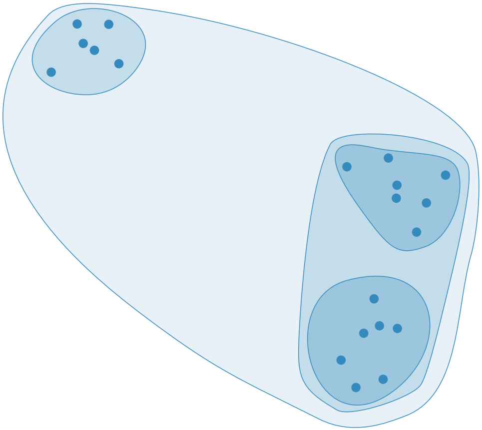
How many clusters are there?



How many clusters are there?

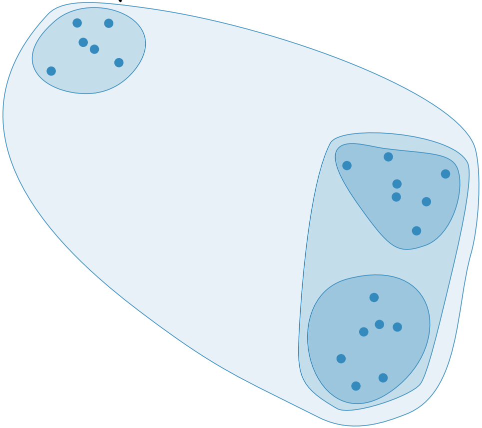


Allow clusters to nest...



A **hierarchical** clustering:

Collection $\mathbb{C}$ of clusters s.t. any two are either **disjoint**, or **nested** (one is contained in the other).



How do we build a hierarchical clustering?

- There are two general approaches...
  - **Agglomerative (bottom-up)**:
    Start with each point in own cluster, iteratively **merge** them.
  - **Divisive (top-down)**:
    Start with all points in single cluster, recursively **divide** them.

Hierarchical Clustering

Input is a set of **objects** $\mathcal{X}$ and a **dissimilarity** d:

$d(x, x') \geq 0$	**non-negativity**
$d(x, x') = d(x', x)$	**symmetry**

Linkage algorithms

- **Linkage algorithms** are a class of **agglomerative** approaches
- Idea:
  1. Start with each point in own cluster.
  2. Merge the two "**closest**" clusters.
  3. Repeat step 2 until we have a single cluster.

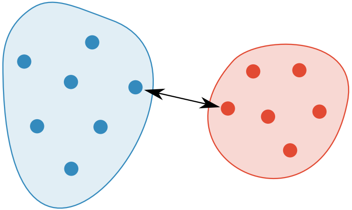
Linkage Algorithms

- How do we **measure** how **close** two clusters are?
- We use a **linkage function** $\mathcal{L}$ taking pairs of clusters to $\mathbb{R}$.
- Single-linkage, complete-linkage, average-linkage...

Single Linkage

The **smallest** distance between the clusters.

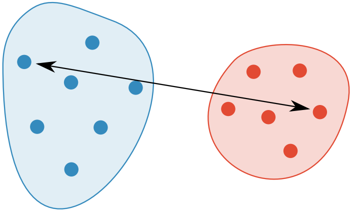
$$\mathcal{L}(C, C') = \min_{x, x' \in C \times C'} d(x, x')$$



Complete Linkage

The biggest distance between the clusters.

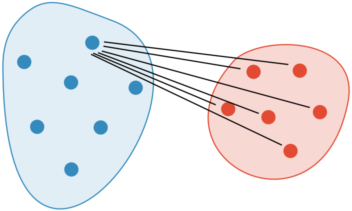
$$\mathcal{L}(C, C') = \max_{x, x' \in C \times C'} d(x, x')$$



Average-linkage (UPGMA)

The mean distance between the clusters.

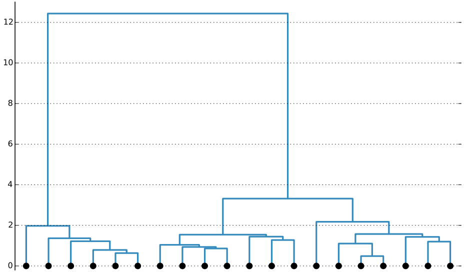
$$\mathcal{L}(C, C') = \frac{1}{|C \times C'|} \sum_{x,x' \in C \times C'} d(x, x')$$



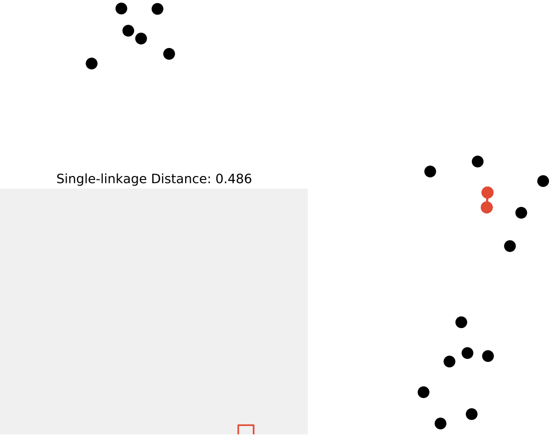
Dendrograms

Linkage clustering gives rise to a **dendrogram**.

- Rooted tree whose leaves are points in $\mathcal{X}$.
- Can read off the linkage at which any pair of points merge.
- Cutting the dendrogram at any height produces **flat** clustering.

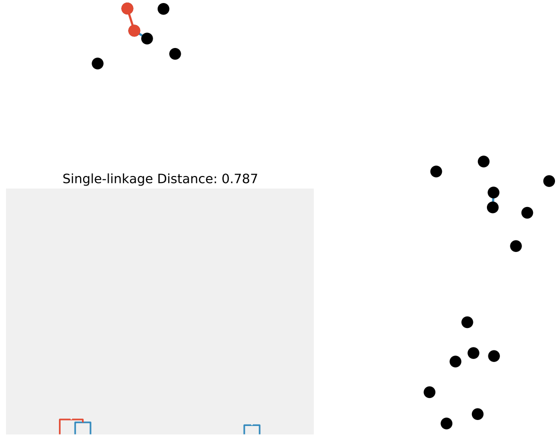


Example: Single-linkage clustering

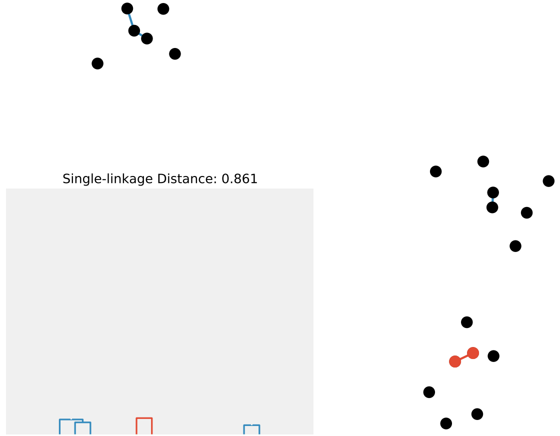


Example: Single-linkage clustering

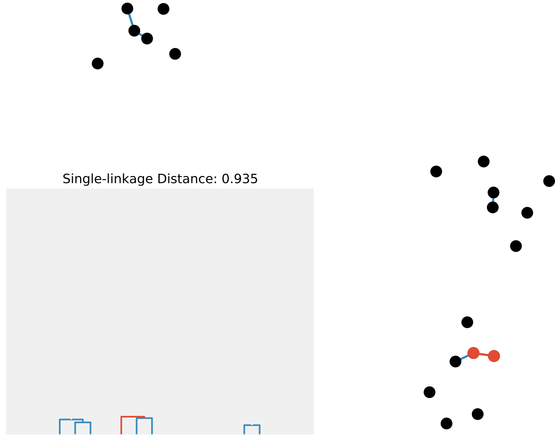
Example: Single-linkage clustering



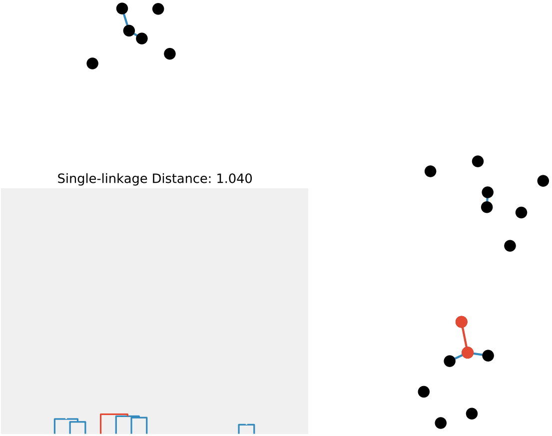
Example: Single-linkage clustering



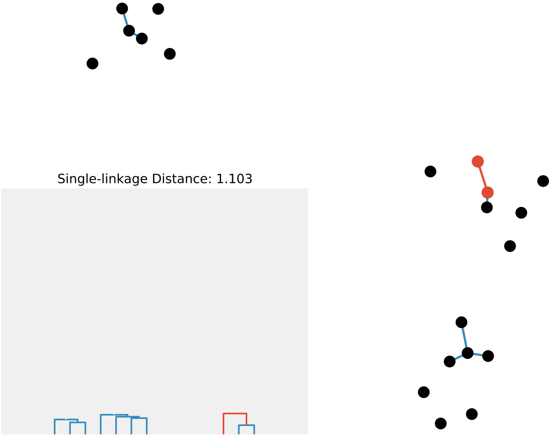
Example: Single-linkage clustering



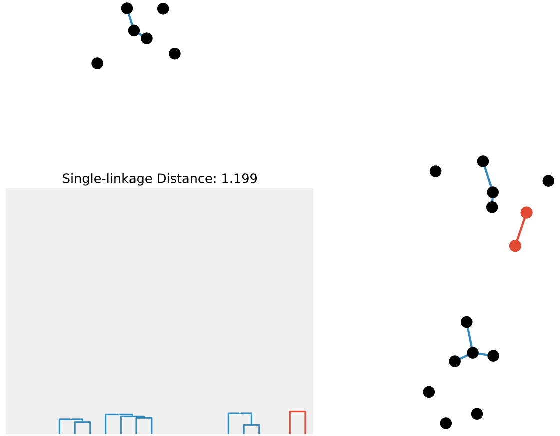
Example: Single-linkage clustering



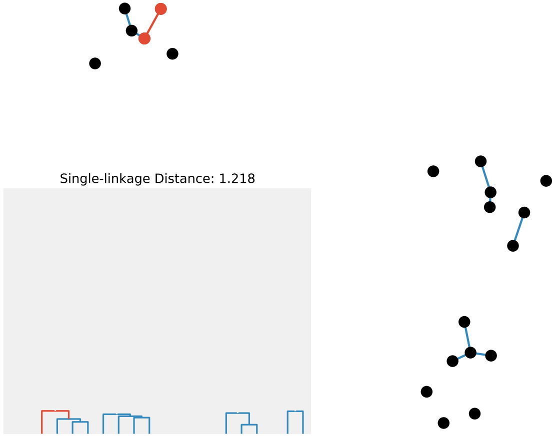
Example: Single-linkage clustering



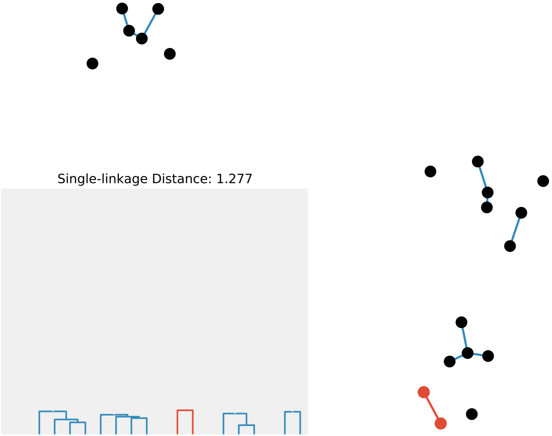
Example: Single-linkage clustering



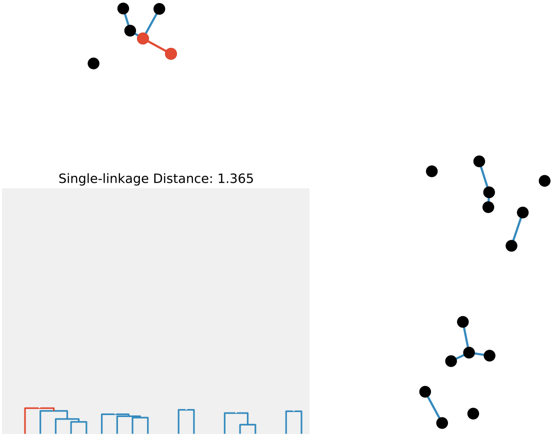
Example: Single-linkage clustering



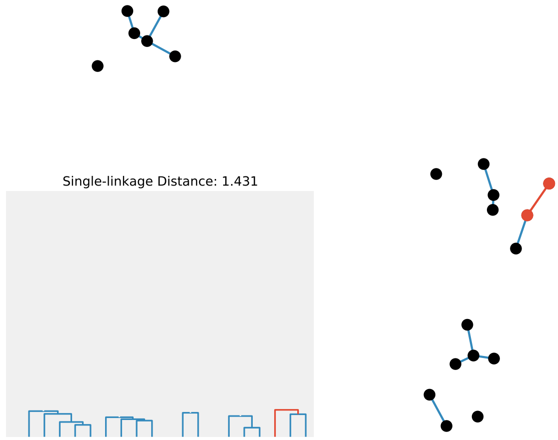
Example: Single-linkage clustering



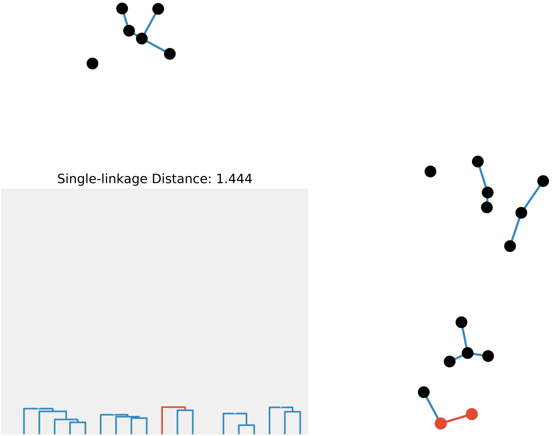
Example: Single-linkage clustering



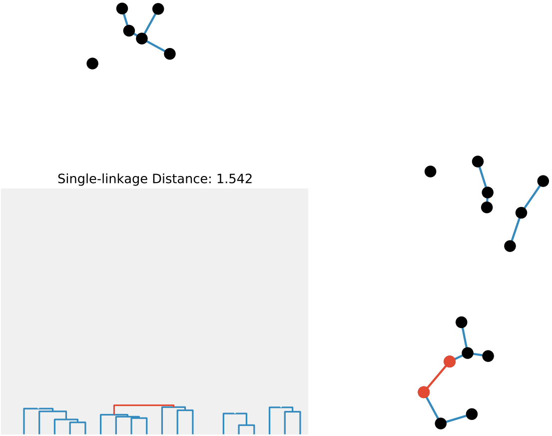
Example: Single-linkage clustering



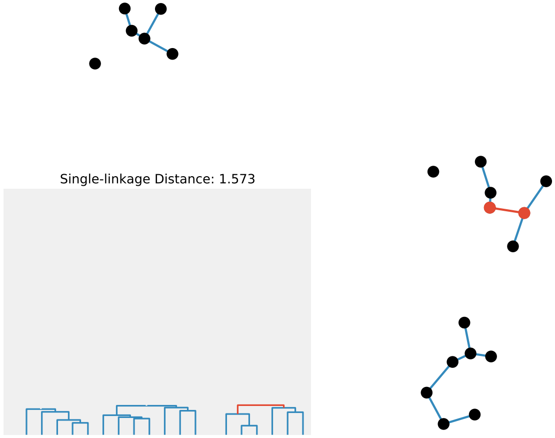
Example: Single-linkage clustering



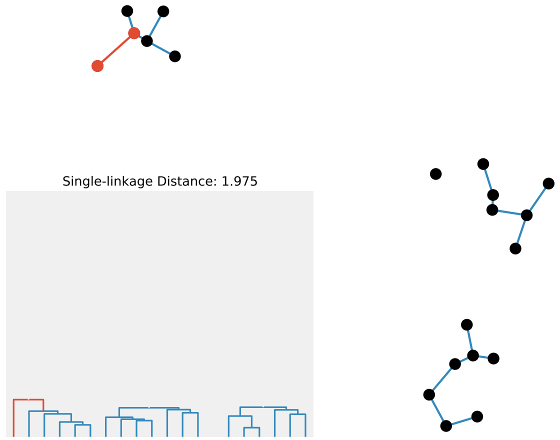
Example: Single-linkage clustering



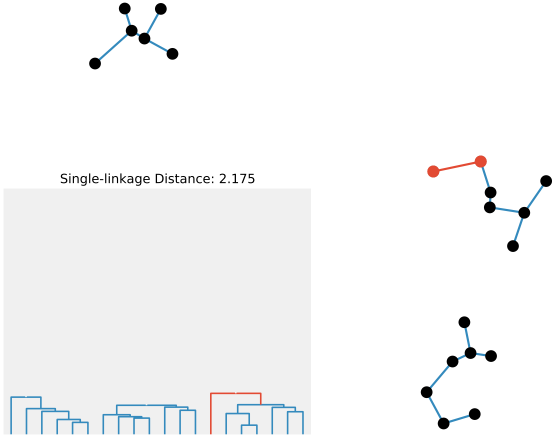
Example: Single-linkage clustering



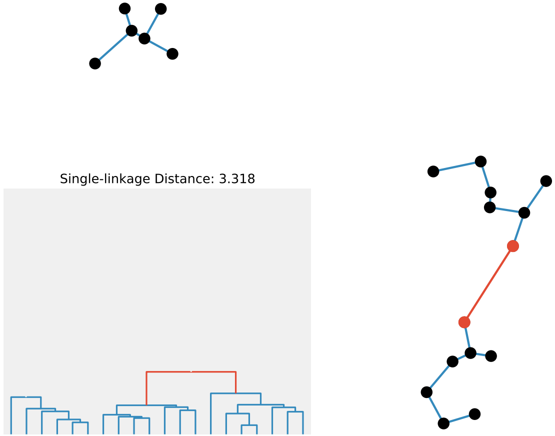
Example: Single-linkage clustering



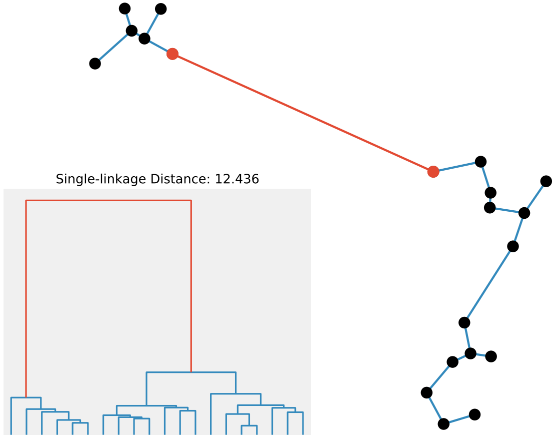
Example: Single-linkage clustering



Example: Single-linkage clustering



Example: Single-linkage clustering



Remember Kruskal's Algorithm?

- Build minimum spanning tree of weighted graph.
- Every step, add "lightest edge".

Graph-Theoretic SLC

- Define complete weighted graph
  - Nodes are data points
  - Edge weights are distances

- For any number λ, delete all edges of weight > λ.

- Connected components of resulting graph are **single-linkage clusters** at level λ.

Practical considerations

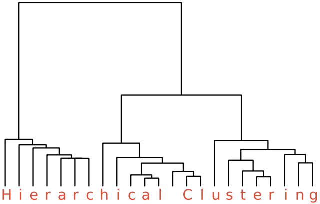
- Naïve implementations take $\Theta(n^3)$ time.

Practical considerations

- Naïve implementations take $\Theta(n^3)$ time.
- Some linkages have more efficient algorithms:
  - Single-linkage: $\Theta(n^2)$, since Prim's algorithm is $\Theta(n^2)$ on a complete graph.
  - Complete-, Average-linkage: $O(n^2 \log n)$.

Practical considerations

- Naïve implementations take $\Theta(n^3)$ time.
- Some linkages have more efficient algorithms:
  - Single-linkage: $\Theta(n^2)$, since Prim's algorithm is $\Theta(n^2)$ on a complete graph.
  - Complete-, Average-linkage: $O(n^2 \log n)$.
- Single-linkage is insensitive to density, exhibits chaining.



CSE 151A

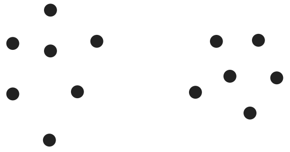
Intro to Machine Learning

Lecture 16 – Part 03

Density Cluster Trees

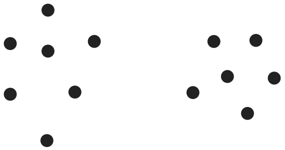
The goal of clustering:

Identify structure in data by grouping it into **clusters**

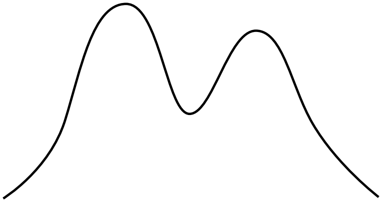


The goal of clustering:

Identify structure in data by grouping it into **clusters**

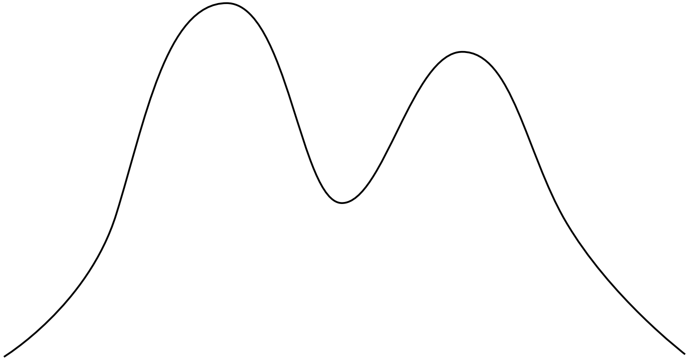


Assumption: data is drawn from some **density**.



What **structure** do we wish to recover?

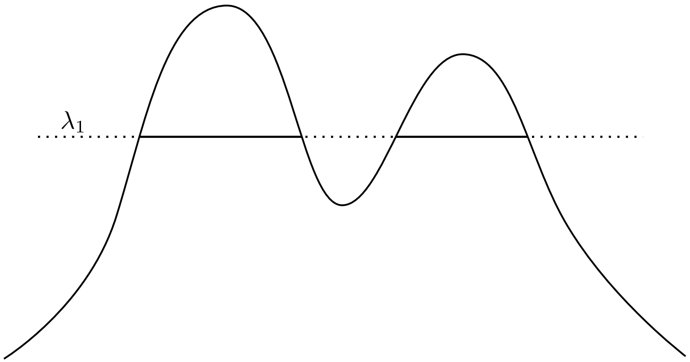
A **cluster** of a density is a ***region of high probability.***[1]



[1] Hartigan (1981), Wishart (1969)...

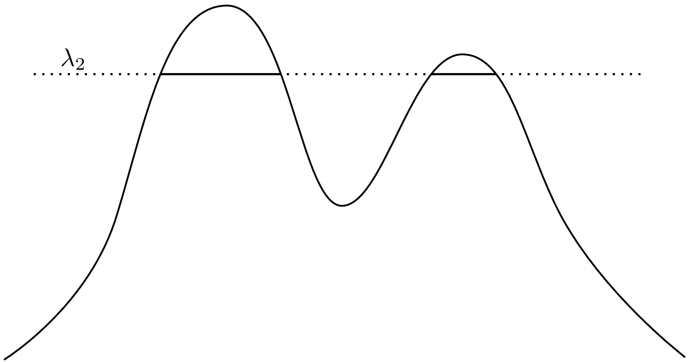
High-density clusters

Connected components of $\{f \geq \lambda_1\}$?



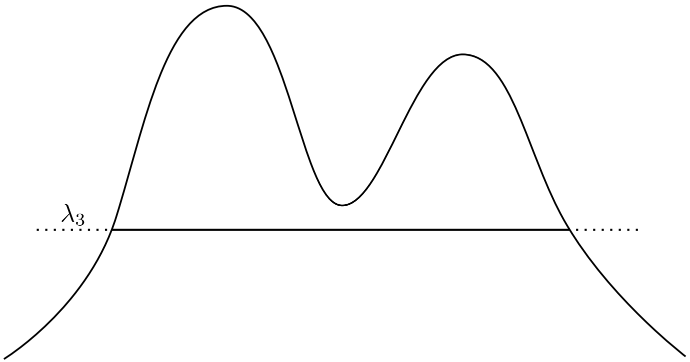
High-density clusters

Connected components of $\{f \geq \lambda_2\}$?



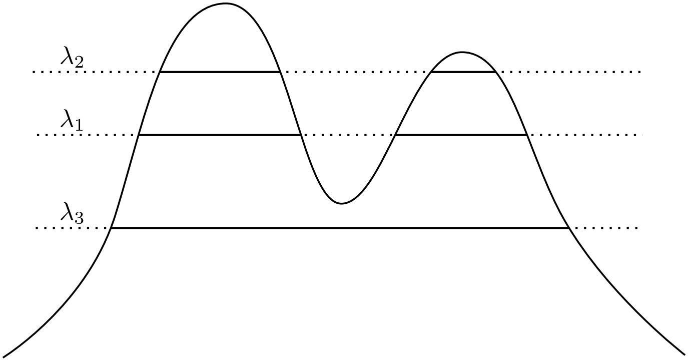
High-density clusters

Connected components of $\{f \geq \lambda_3\}$?



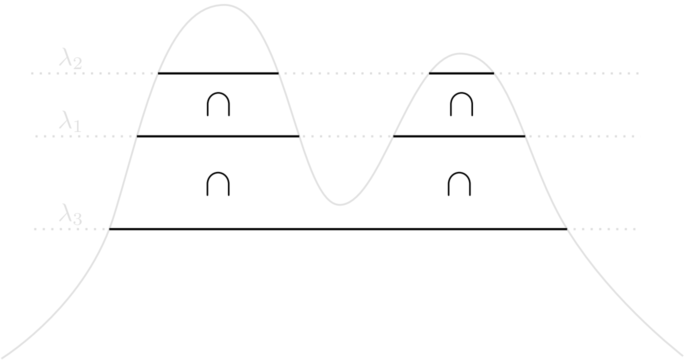
High-density clusters

A **cluster** is a connected component of $\{f \geq \lambda\}$ for any $\lambda > 0$.



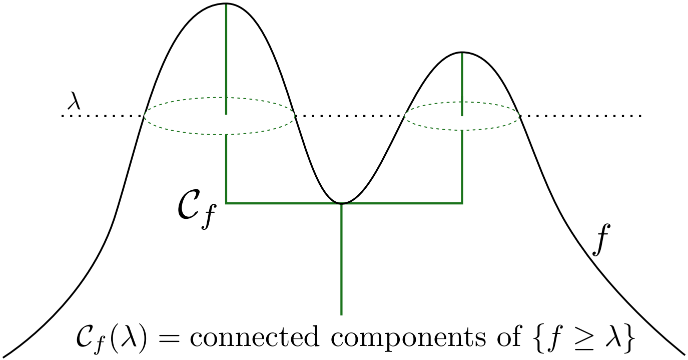
A hierarchy of clusters

Clusters from higher levels nest within clusters from lower levels.



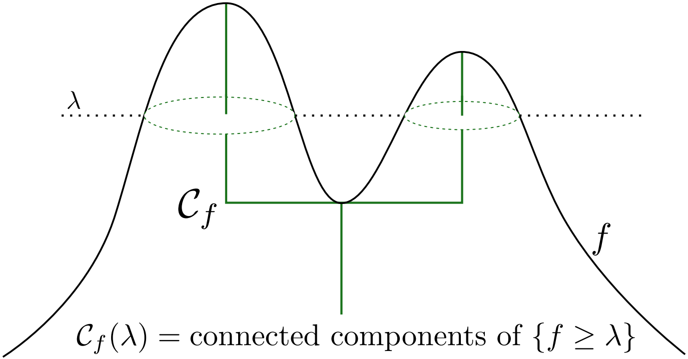
The density cluster tree

This gives rise to a tree structure called the **density cluster tree**.



What structure do we wish to recover?

This **density cluster tree** is what we hope to recover from data.

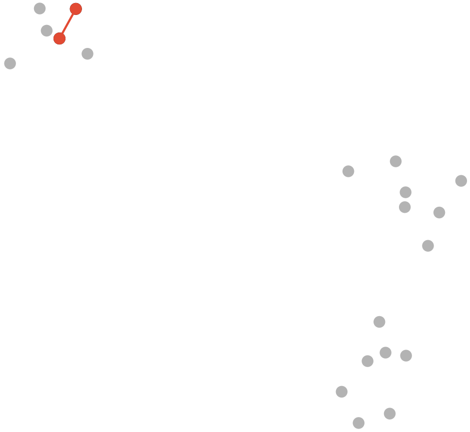


Robust single-linkage

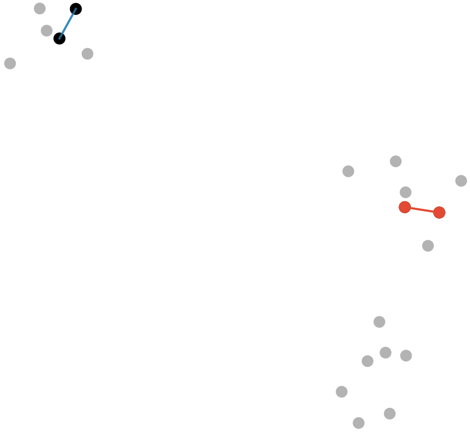
Intuition: At first, only admit **high-density** points into graph.

- Choose parameters α and k
- For each $x \in \mathcal{X}$, let $r_k(x)$ be the distance to x's k-th nearest neighbor.
- As r grows from 0 to ∞:
  - Let $V = \{x : r_k(x) \leq r\}$.
  - Let $E = \{(x, x') : d(x, x') \leq \alpha r\}$.
  - Build the graph $G_r = (V, E)$.
  - The **clusters** at time r are the **connected components** of G_r.

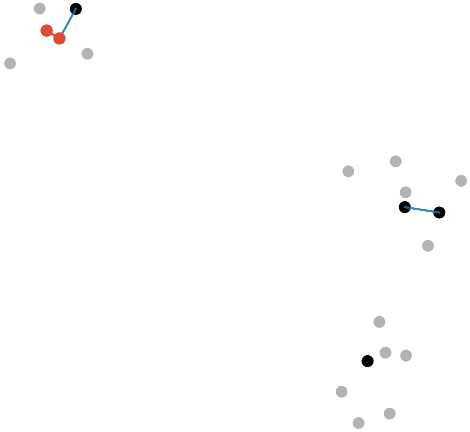
Example: Robust single-linkage



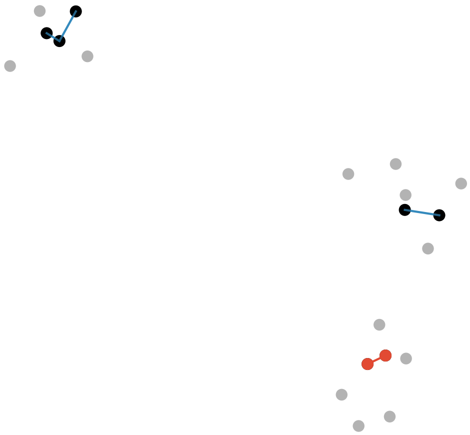
Example: Robust single-linkage



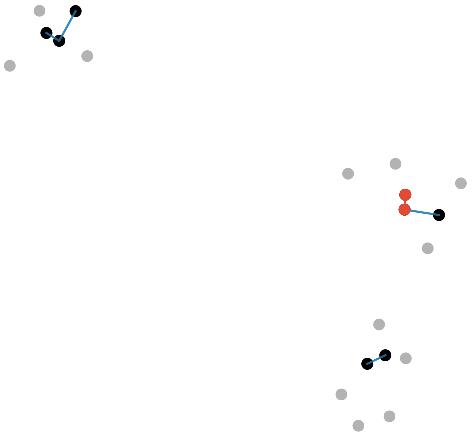
Example: Robust single-linkage



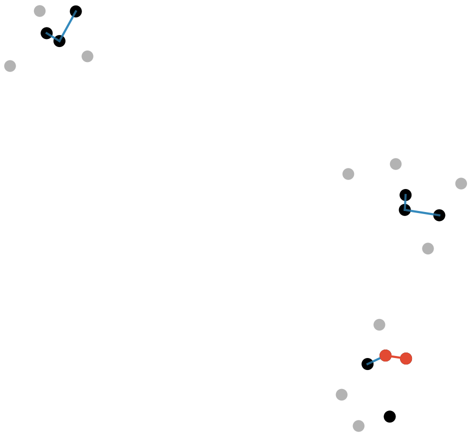
Example: Robust single-linkage



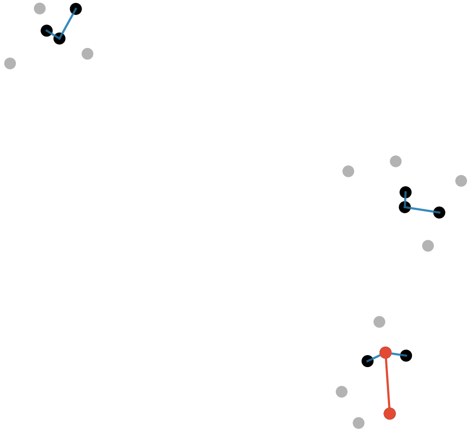
Example: Robust single-linkage



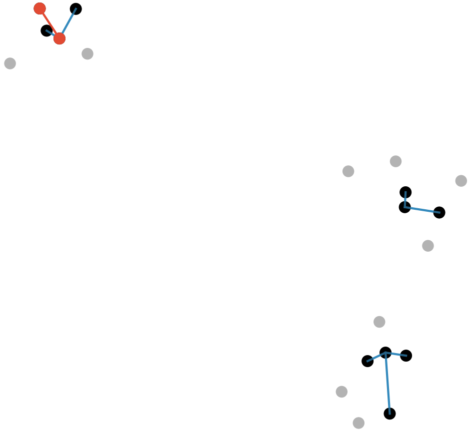
Example: Robust single-linkage



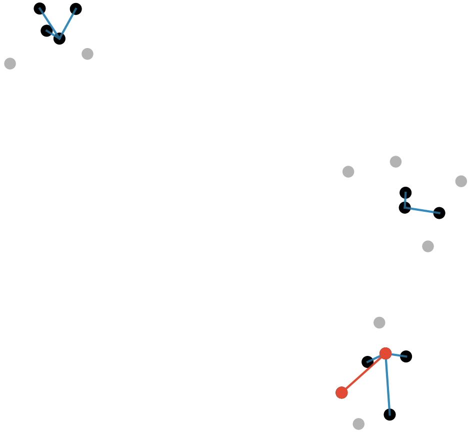
Example: Robust single-linkage



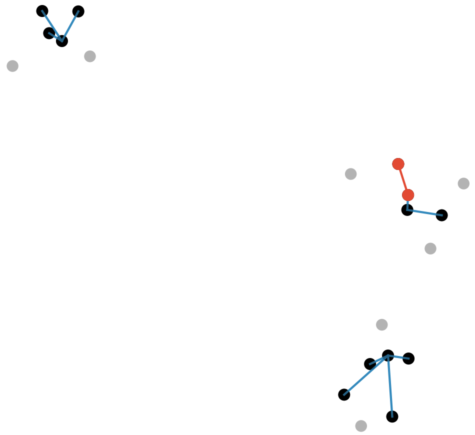
Example: Robust single-linkage



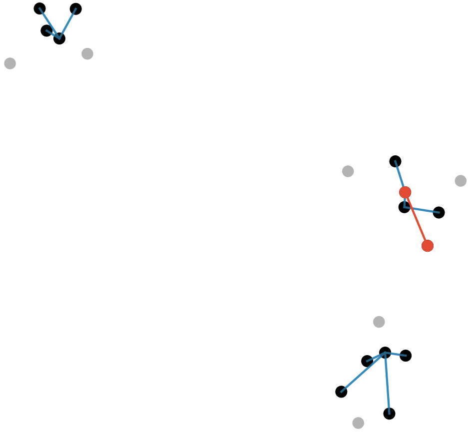
Example: Robust single-linkage



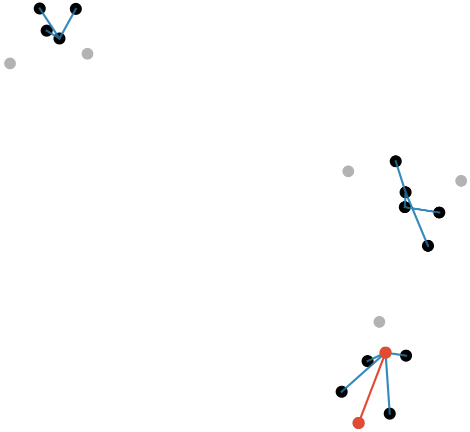
Example: Robust single-linkage



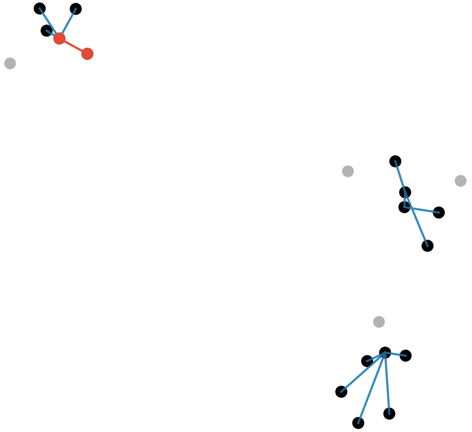
Example: Robust single-linkage



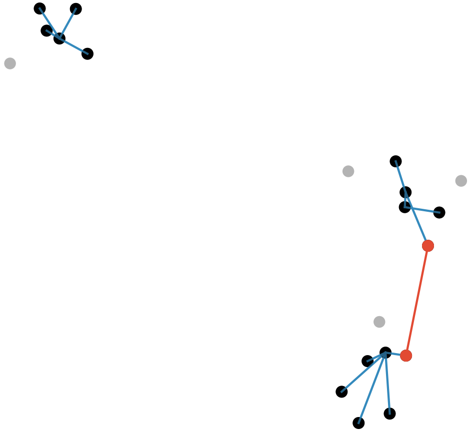
Example: Robust single-linkage



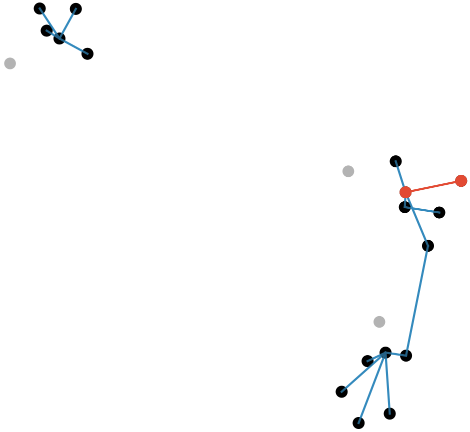
Example: Robust single-linkage



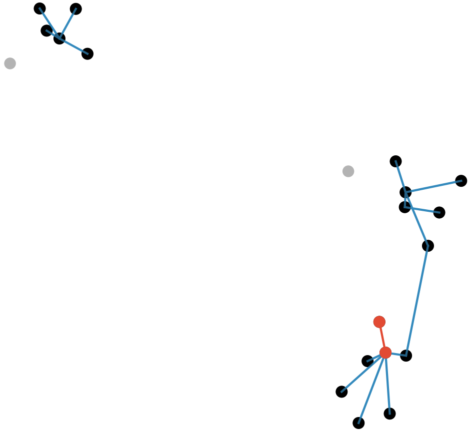
Example: Robust single-linkage



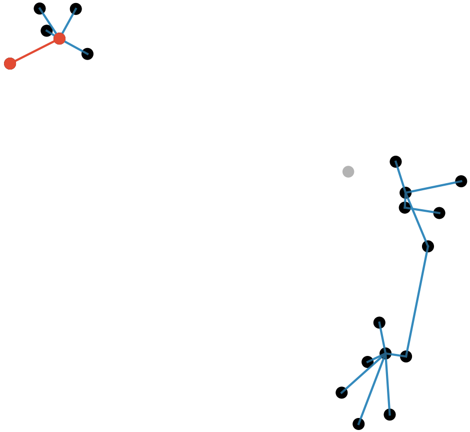
Example: Robust single-linkage



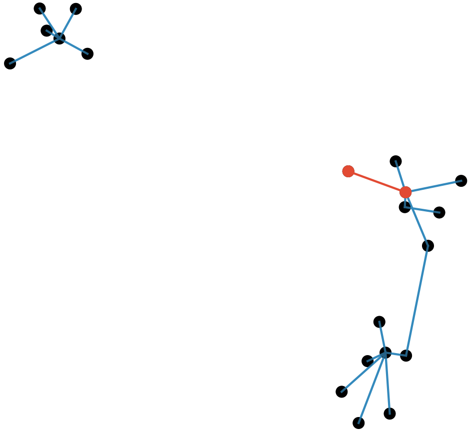
Example: Robust single-linkage



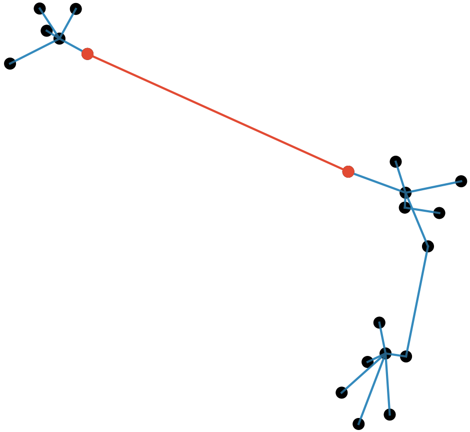
Example: Robust single-linkage



Example: Robust single-linkage



Example: Robust single-linkage



- **Robust single-linkage** recovers the density cluster tree (Chaudhuri and Dasgupta, 2010; Eldridge, Belkin, Wang 2015).

- Can be viewed as a transformation of metric, followed by single-linkage:

$$\tilde{d}(x, x') = \max\left\{r_k(x), r_k(x'), \frac{1}{\alpha} d(x, x')\right\}.$$

- And therefore can be computed in $\Theta(kn^2)$ time.