Clustering

Brooks Paige

Week 7

Clustering

How can we summarize or interpret an unknown dataset, with no labels?

$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$$

The idea behind **clustering** is that complex data can be described concisely by a small number of discrete components, each with its own clear and interpretable structure.

Clustering

How can we summarize or interpret an unknown dataset, with no labels?

$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$$

The idea behind **clustering** is that complex data can be described concisely by a small number of discrete components, each with its own clear and interpretable structure.

We'll look at a few algorithms for clustering. The key assumption we will need is that there is some reasonable notion of proximity, defined in terms of a **dissimilarity** $d(\mathbf{x}, \mathbf{x}')$, associated with our data points.

Many different domains

When might we want compact descriptions of data?

- "My clients can be considered to belong to 5 different groups, with a typical client from each group looking like this."
- Clustering web pages based on their content
- Clustering (and merging) web search results
- Clustering users based on their product preferences or purchase history
- Generally: to reduce apparent complexity in order to gain insight

Examples: David Barber

Clustering desiderata

1. Data points in the **same cluster** should have **high similarity**
2. Data points in **different clusters** should have **low similarity**

Notions of "dissimilarity" are data- and domain-specific, but squared (Euclidean) distance between points is pretty common...

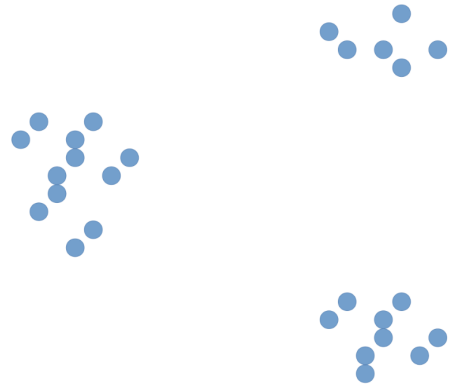


Figure: Andreas Müller

Intuitive clustering

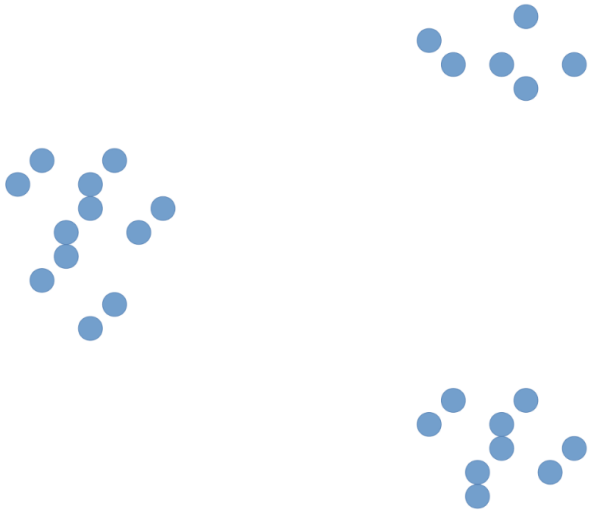


Figure: Andreas Müller

Intuitive clustering

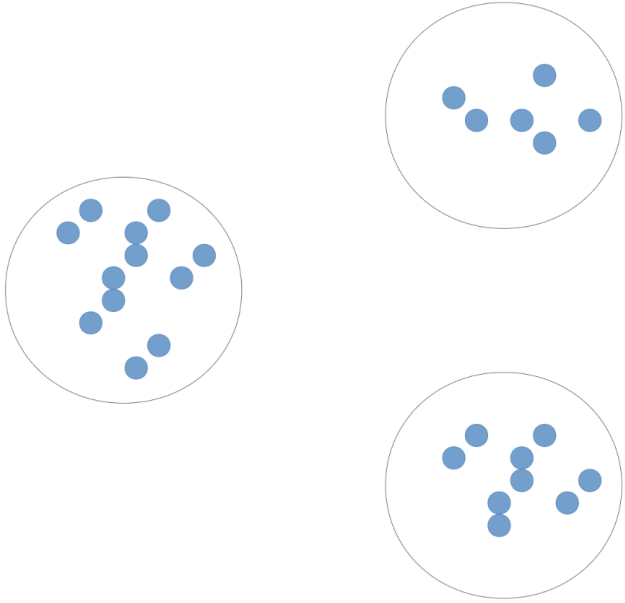


Figure: Andreas Müller

Intuitive clustering

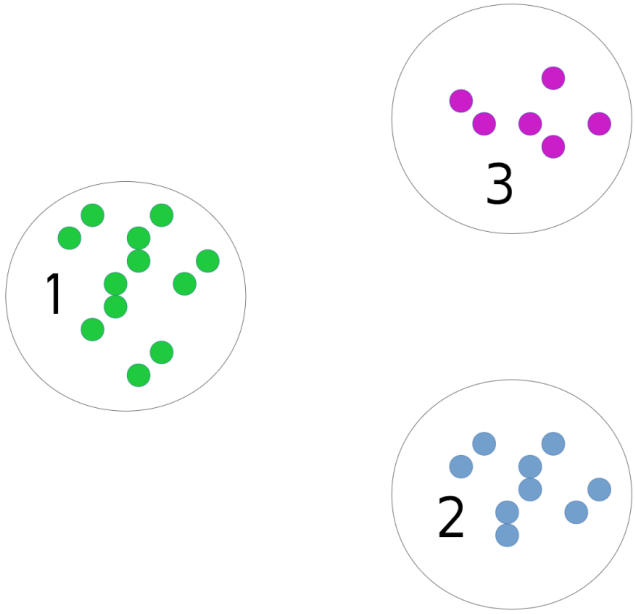


Figure: Andreas Müller

Intuitive clustering

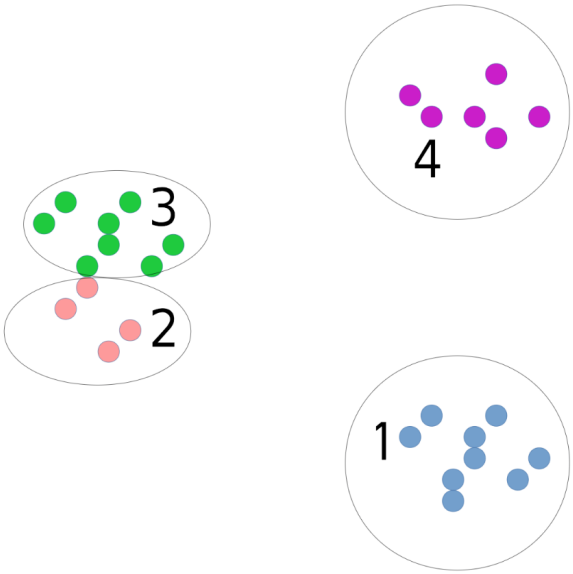
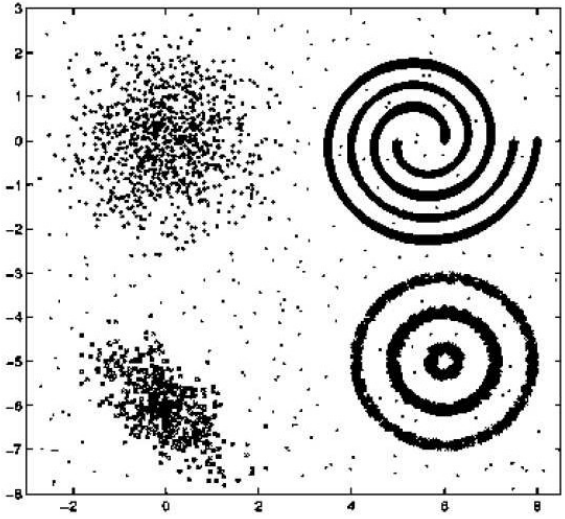
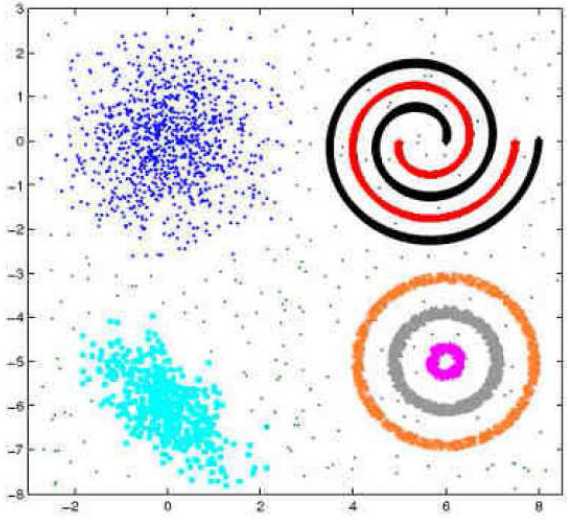


Figure: Andreas Müller

Intuitive clustering



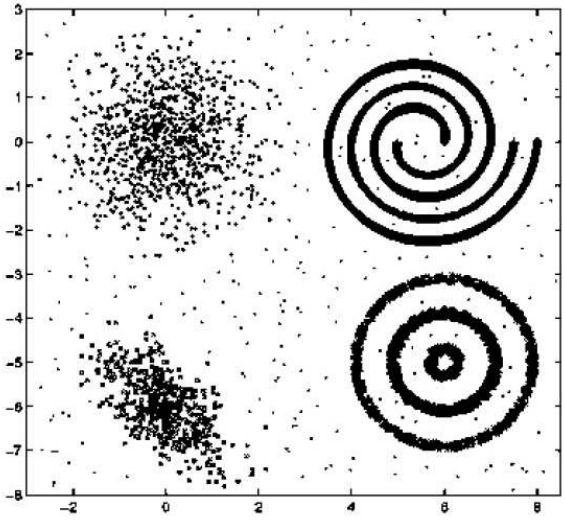
(a) Input data



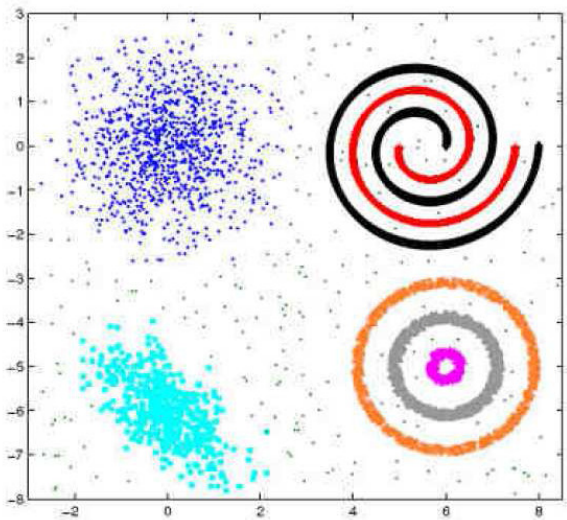
(b) Desired clustering

Figure: Jain 2009

Intuitive clustering



(a) Input data



(b) Desired clustering

Warning: None of the algorithms we'll look at can do this!

Figure: Jain 2009

Market basket analysis

- Data: a list of items in each basket
- Useful to represent this as a matrix
- Typically **very sparse**!

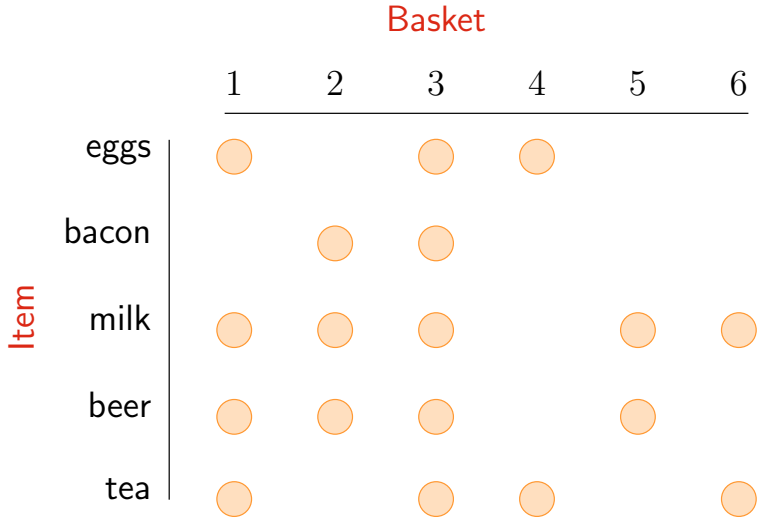


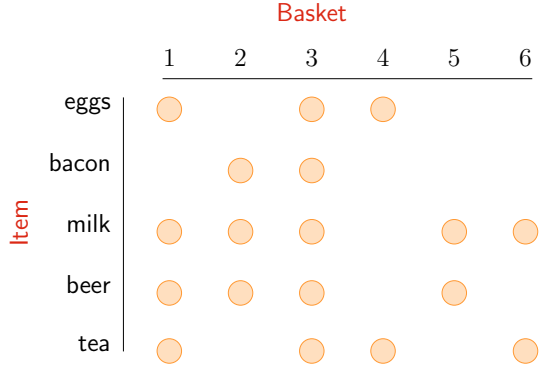
Figure: David Barber

Finding association rules

Finding frequent item sets:

- Find common single items, then common pairs, then triples, etc . . .
- For a threshold of 3: { Eggs }, { Milk }, { Beer }, {Tea }, { Eggs, Tea }, { Milk, Beer }, { Milk, Tea }

Clustering goal is slightly different: we want to find baskets that look similar and group them together into a single "cluster basket".



Clustering problem formulation

$\mathcal{D} = \{x_1, \ldots, x_N\}$ Encoder: $k = C(x_i)$

$k \in \{1, \ldots, K\}$ Dissimilarity: $d(x_i, x_j)$

Loss: $W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(x_i)=k} \sum_{C(x_{i'})=k} d(x_i, x_{i'})$

within-cluster "spread" for a "k"

$$T = \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} d(x_i, x_{i'}) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(x_i)=k} \left(\sum_{C(x_{i'})=k} d_{ii'} + \sum_{C(x_{i'})\neq k} d_{ii'} \right)$$

$d_{ii'}$

$$T = \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d(x_i, x_{i'}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(x_i)=k}\left(\sum_{C(x_{i'})=k} d_{ii'} + \sum_{C(x_{i'})\neq k} d_{ii'}\right)$$

where $d(x_i, x_{i'}) = d_{ii'}$

$$= \left(\frac{1}{2}\sum_{k}\sum_{C(x_i)=k}\sum_{C(x_{i'})=k} d_{ii'}\right) + \left(\frac{1}{2}\sum_{k}\sum_{C(x_i)=k}\sum_{C(x_{i'})\neq k} d_{ii'}\right)$$

$W(C)$ (within cluster) $\qquad B(C)$ (between cluster)

$$\min_C W(C) = \min_C T - B(C) = \max_C B(C)$$

Combinatorial opt?

$x_1, \ldots, x_N$

$c_1, \ldots, c_N \in \{1, \ldots, K\}$

# of possible assignments N data to K clusters?

$K = 4 \longrightarrow N = 10 \Rightarrow 34{,}105$ cluster assignments

$N = 19 \Rightarrow \approx 10^{10}$

K-means

K-means: $d(x_i, x_{i'}) \equiv \| x_i - x_{i'} \|_2^2$

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} \| x_i - x_{i'} \|_2^2$$

$$= \sum_{k=1}^{K} N_k \sum_{C(x_i)=k} \| x_i - \overline{x}_k \|_2^2$$

N_k: size of k

$\overline{x}_k$: mean of x_i where $C(x_i) = k$

$m_k \equiv \overline{x}_k \longleftrightarrow$ find C

(0) Initialize: K different values m_k, $1 \dots K$

Repeatedly:

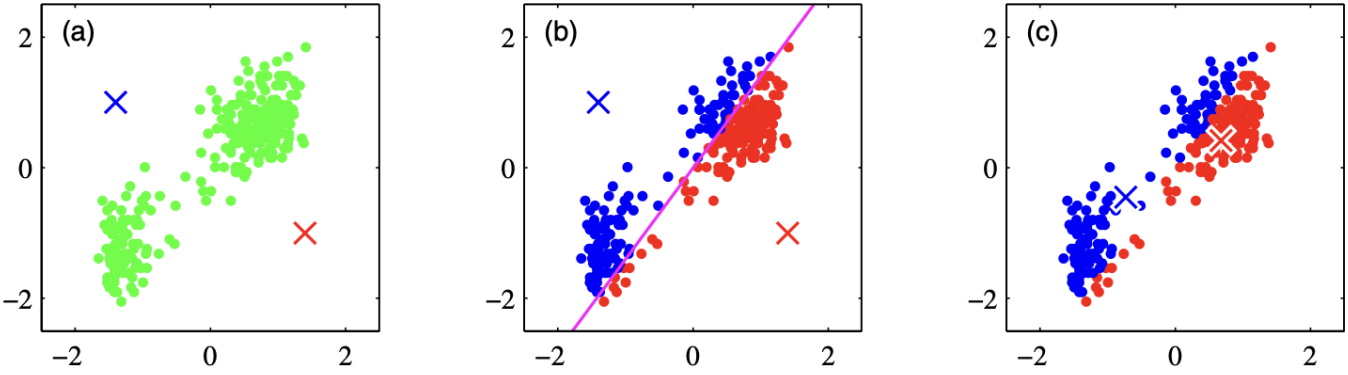
(1) Given $\{m_k\}$, assign $\underbrace{C(x_i)}_{c_i} = k$, $\forall i$

$$\Rightarrow \min_{c_i} \| x_i - m_{c_i} \|_2^2$$

(2) Given $C(x_i)$, update means:

$$m_k = \frac{1}{N_k} \sum_{c_i = k} x_i, \qquad N_k = \sum \mathbb{I}(C(x_i) = k)$$

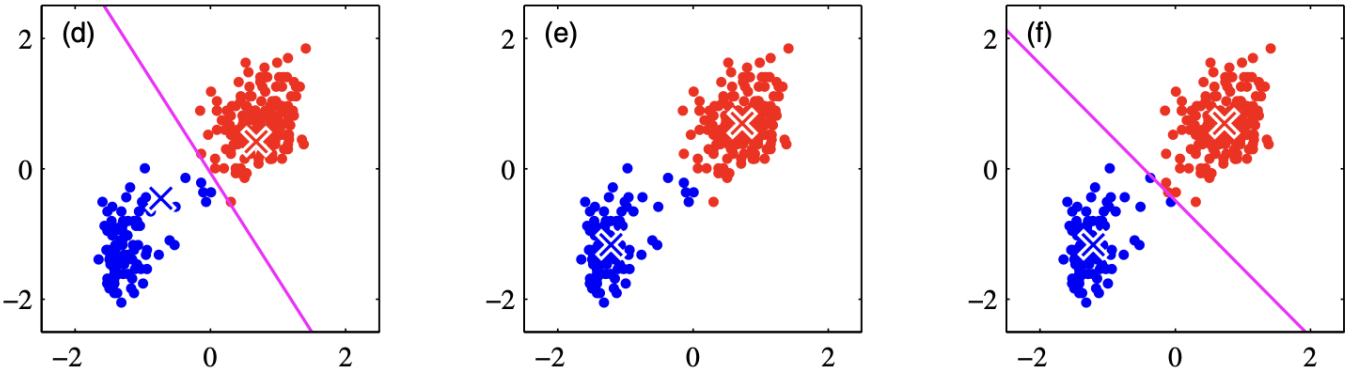
K-means estimation in action



(a) Initialization; (b) Cluster assignment; (c) Cluster center update

Figure: PRML

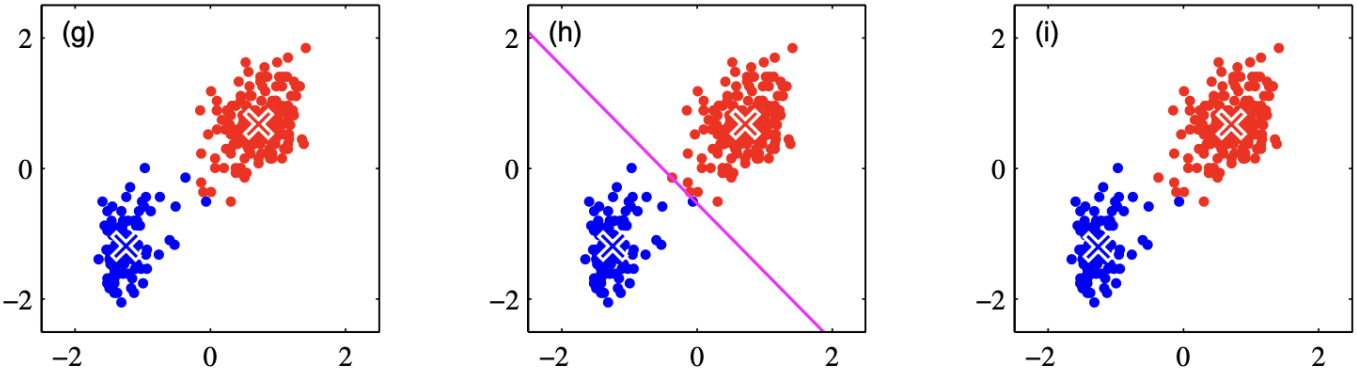
K-means estimation in action



Iteratively assign points to clusters and re-compute means

Figure: PRML

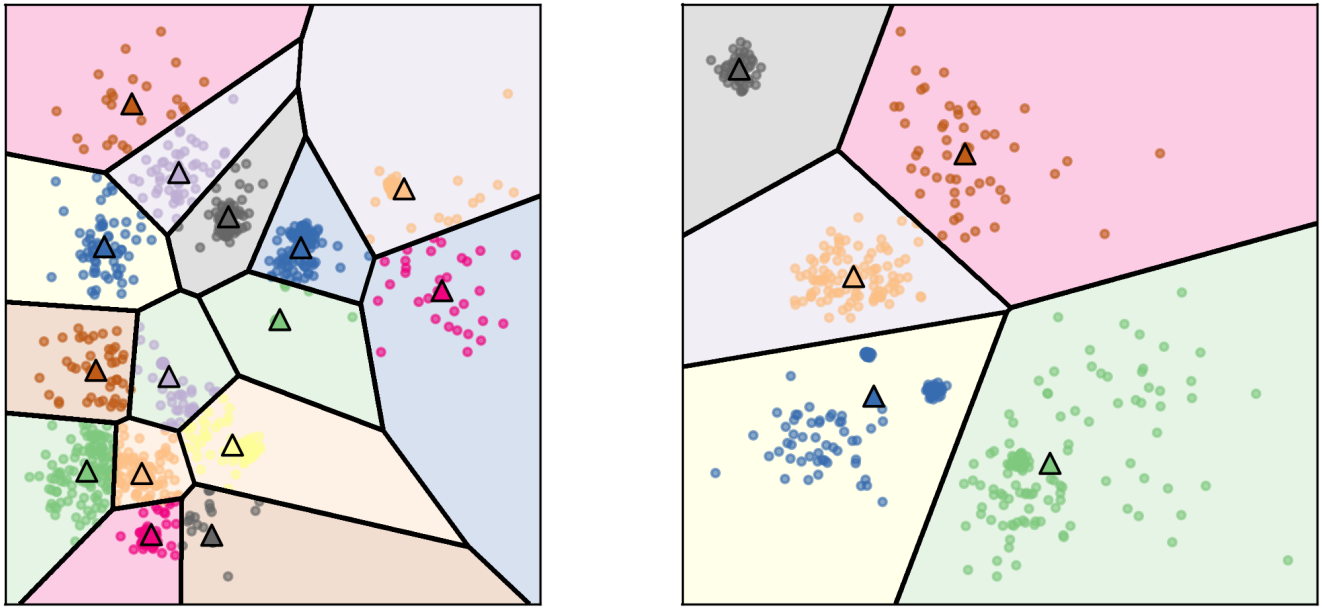
K-means estimation in action



Converges when no points change cluster given new mean

Figure: PRML

Cluster shapes (and generalization)



Cluster shapes are **Voronoi diagrams** of the centers

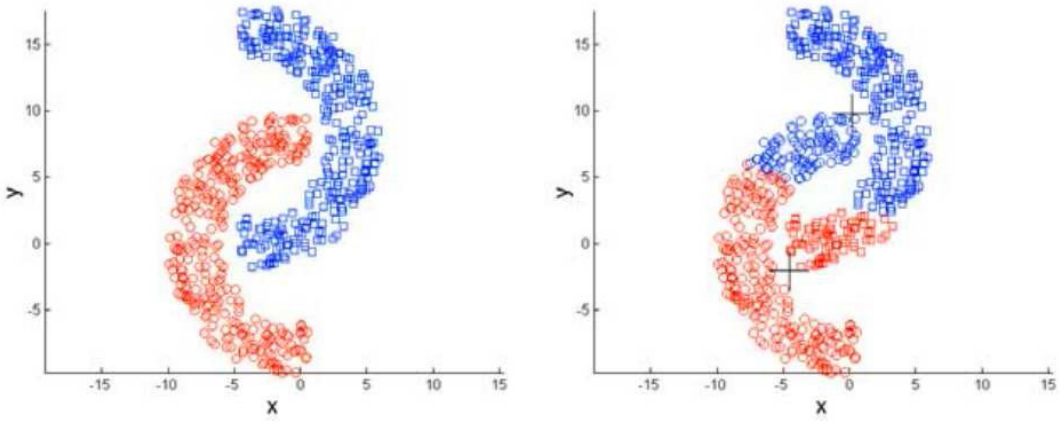
Figure: Andreas Müller

Application: Vector quantization

K-means in color space

Figure: PRML

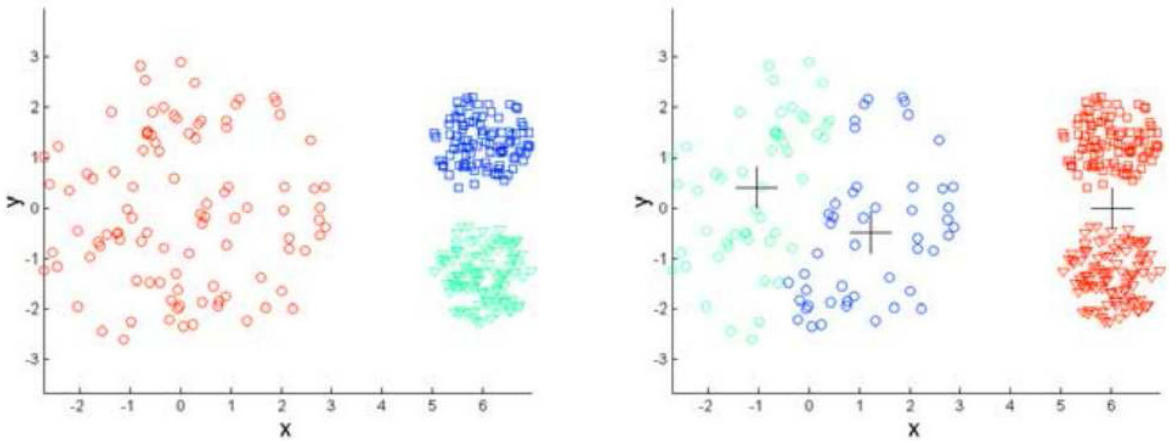
K-means limitations: non-spherical data



- Left: clustering we would like. Right: clustering K-means produces.
- Problem is that the clusters are not spherical and they are too close.

Figure: Christof Monz (Queen Mary, Univ. of London)

K-means limitations: unequal density



- Left: clustering we would like. Right: clustering K-means produces.
- Hard to address this with K-means.

Figure: David Barber

How can we improve this algorithm?

- K-medoids: robustness to outliers, works with arbitrary dissimilarities and data types
  - Simple idea: instead of computing the mean, select the data point which minimizes the average distance to all other points in the cluster
  - Problem: increased computational complexity

How can we improve this algorithm?

- K-medoids: robustness to outliers, works with arbitrary dissimilarities and data types
  - Simple idea: instead of computing the mean, select the data point which minimizes the average distance to all other points in the cluster
  - Problem: increased computational complexity
- Gaussian mixture models: generalizes this to allow Gaussian-shaped clusters (as opposed to hyperspheres!)

How can we improve this algorithm?

- K-medoids: robustness to outliers, works with arbitrary dissimilarities and data types
  - Simple idea: instead of computing the mean, select the data point which minimizes the average distance to all other points in the cluster
  - Problem: increased computational complexity
- Gaussian mixture models: generalizes this to allow Gaussian-shaped clusters (as opposed to hyperspheres!)
- More general mixture models: can handle discrete observations and missing data
- Mixed-membership models: for when "hard" clustering isn't appropriate (e.g. a "customer" may be better modeled as a mixture of multiple groups...)

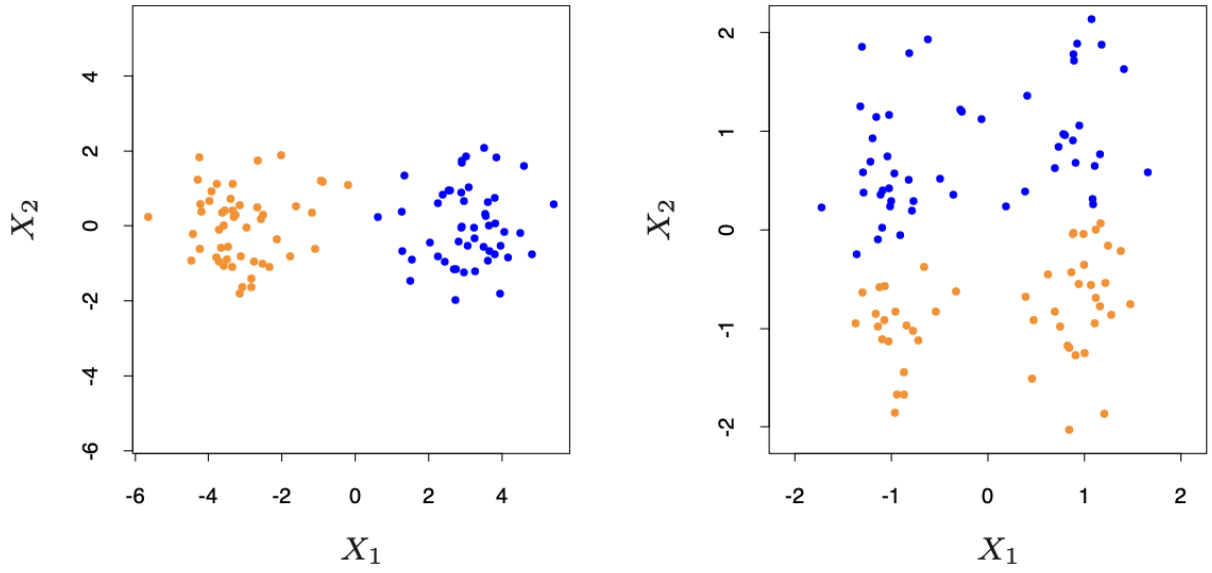
Practical challenges

- This only finds a local optima! Need to run K-means with different initial guesses for the cluster centers...
- K-means is generally fast, but in high dimensions finding the nearest cluster center can be slow (but we can use our fast nearest neighbor methods!)
- No great options for selecting K, other than to try different values and see when increasing no longer "helps"
- The cluster center is not necessarily close to an actual datapoint. This is particularly weird for e.g. one-hot encoded categorical data! (K-medoids does fix this though...)

Sensitivity to scaling

- Suppose each data-vector contains two attributes x_1 and x_2. x_1 represents the temperature in centigrade, and x_2 the distance.
- If we represent the distance in inches, then the Euclidean distance will be dominated by the difference in distance.
- If we represent the distance in miles, the distance will be most likely dominated by the difference in temperature.
- Rescaling / normalizing the dimensions one way around this. But...

Re-scaling dimensions can also be harmful



. . . "standardization" can remove otherwise obvious structure.

Figure: ESL

K-means as feature extractor

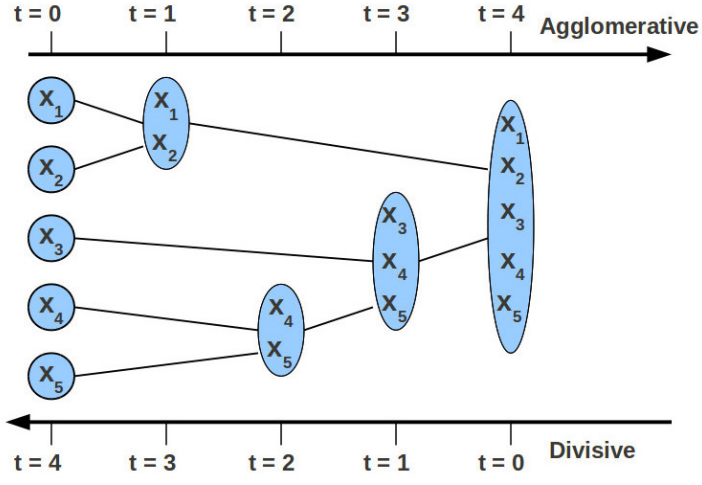
One interesting usage of K-means is as a way of defining a non-linear feature space.

- **Cluster membership**: Categorical feature
- **Distance to cluster centers**: K real-valued features

Can be used to construct useful feature spaces, particularly for low-dimensional data.

Agglomerative clustering

Hierarchical clustering



Hierarchical clustering forms a tree:
children are contained within their parent cluster.

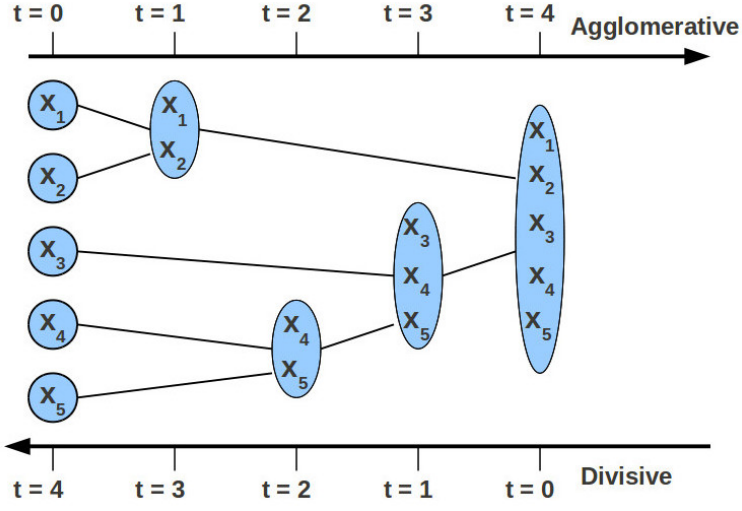
Two directions for clustering

Agglomerative (bottom-up)

- Start with each example in its own singleton cluster
- At each stage, greedily merge 2 most similar clusters

Divisive (top-down)

- Start with all examples in the same cluster
- At each stage, remove the 'outsiders' from the least cohesive cluster



Agglomerative clustering

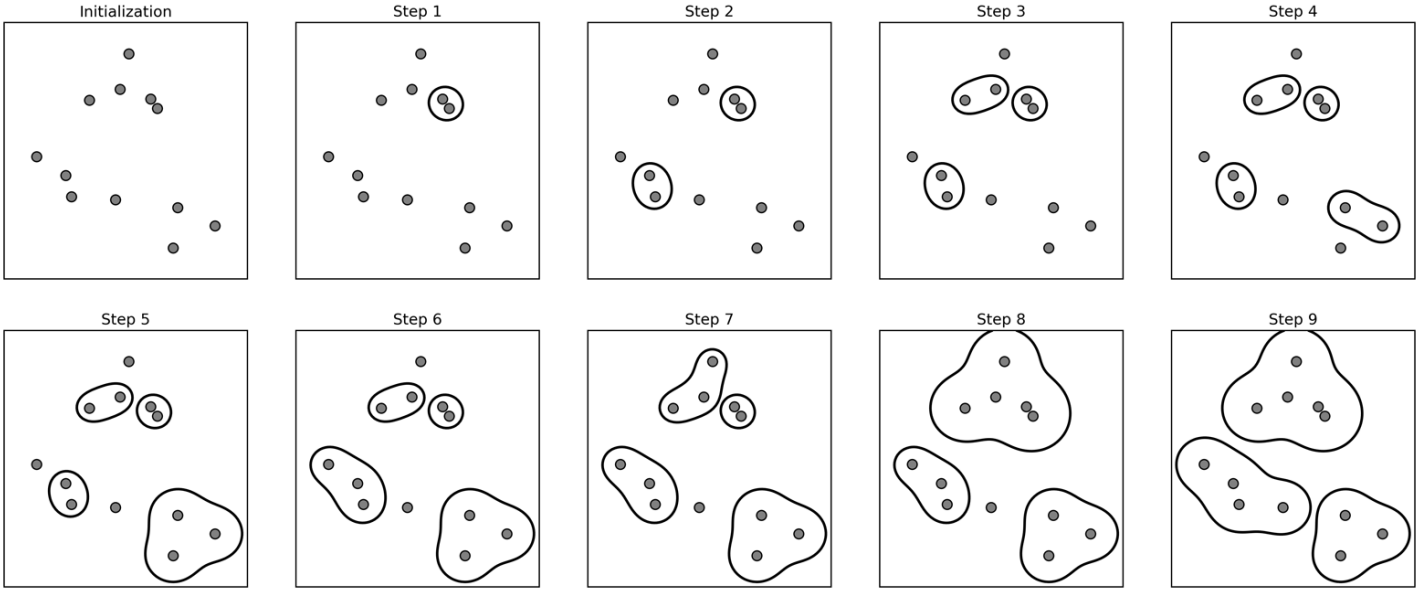
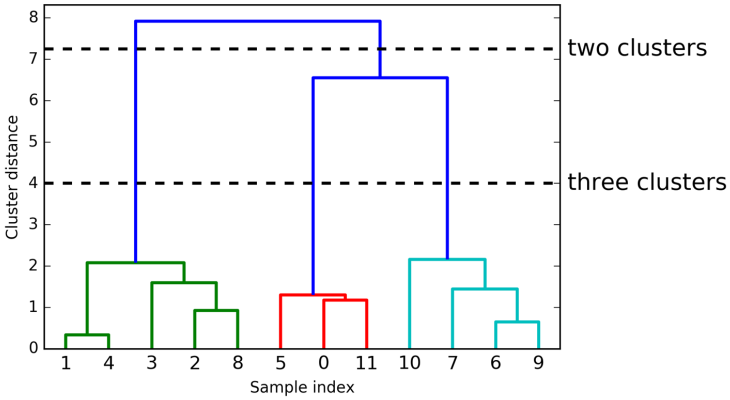
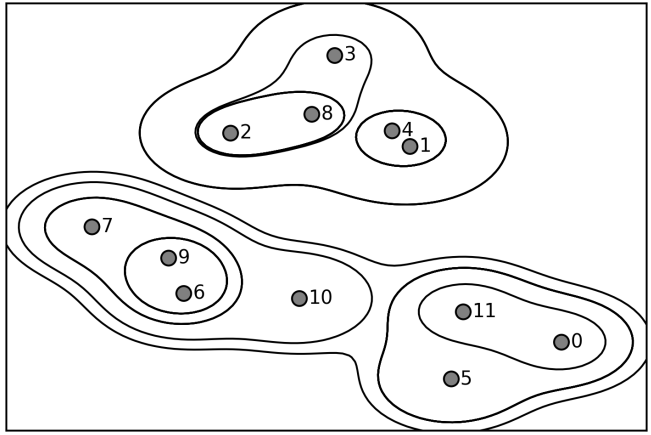


Figure: Andreas Müller

Dendograms





Figures: Andreas Müller

Deciding what clusters to merge

Given a dissimilarity $d(\mathbf{x}, \mathbf{x}')$ between two points $\mathbf{x}, \mathbf{x}'$, there are a handful of different approaches to defining the dissimilarity $D(G, H)$ between existing clusters G, H, i.e. between sets of points.

These are known as different **linkage rules**.

Deciding what clusters to merge

Given a dissimilarity $d(\mathbf{x}, \mathbf{x}')$ between two points $\mathbf{x}, \mathbf{x}'$, there are a handful of different approaches to defining the dissimilarity $D(G, H)$ between existing clusters G, H, i.e. between sets of points.

These are known as different **linkage rules**.

At each step of the algorithm, we merge whichever pair of clusters minimize the cluster dissimilarity $D(\cdot, \cdot)$.

Deciding what clusters to merge

- **Single Linkage**, also called the "nearest neighbor" approach, joins clusters based on their **minimum** distance between any two points:

$$D_{SL}(G, H) = \min_{\mathbf{x} \in G, \mathbf{x}' \in H} d(\mathbf{x}, \mathbf{x}')$$

- **Complete Linkage** (or "furthest neighbor") joins clusters based on their **maximum** distance between any two points:

$$D_{CL}(G, H) = \max_{\mathbf{x} \in G, \mathbf{x}' \in H} d(\mathbf{x}, \mathbf{x}')$$

- **Group Average Linkage** joins based on the average distance of points:

$$D_{GA}(G, H) = \frac{1}{|G||H|} \sum_{\mathbf{x} \in G} \sum_{\mathbf{x}' \in H} d(\mathbf{x}, \mathbf{x}')$$

Linkage rule behavior

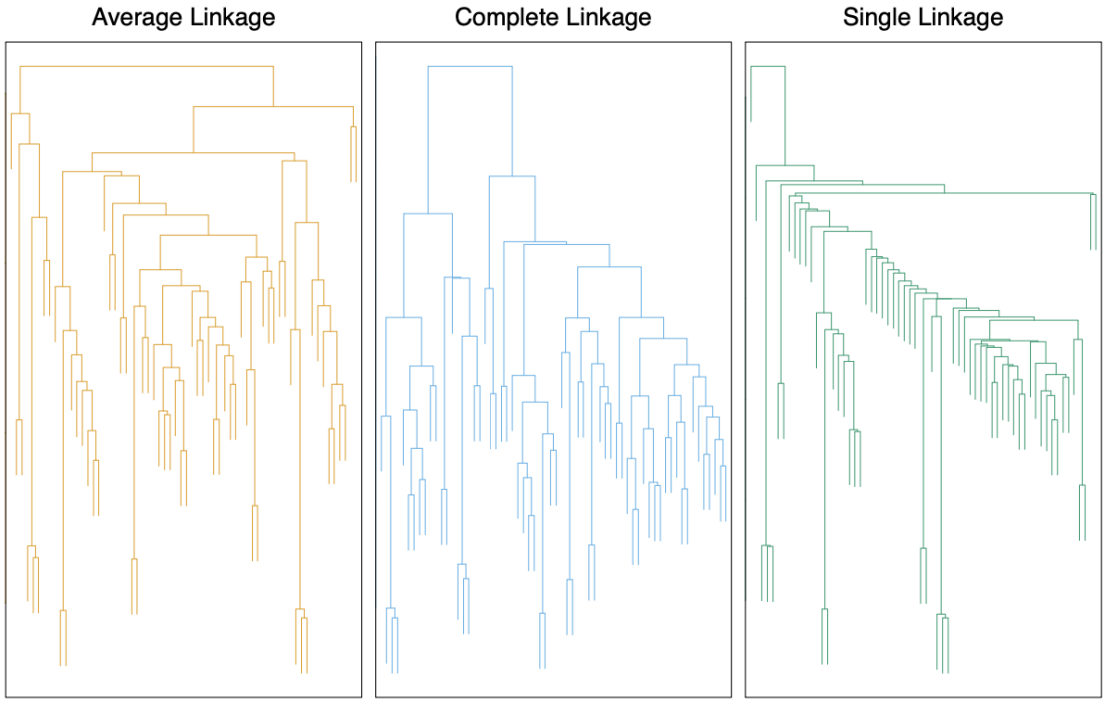


Figure: ESL

Deciding what clusters to merge

For real-valued data specifically, with the Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, a different common choice is:

- **Ward's Linkage** looks at how much the within-cluster variances would increase after the merge.

Deciding what clusters to merge

For real-valued data specifically, with the Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, a different common choice is:

- **Ward's Linkage** looks at how much the within-cluster variances would increase after the merge. Letting $\boldsymbol{\mu}$ denote cluster means,

$$D_{WL}(G, H) = \frac{1}{|G| + |H|} \sum_{\mathbf{x}_i \in G \cup H} \|\mathbf{x}_i - \boldsymbol{\mu}_{G \cup H}\|^2$$

Deciding what clusters to merge

For real-valued data specifically, with the Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, a different common choice is:

- **Ward's Linkage** looks at how much the within-cluster variances would increase after the merge. Letting $\boldsymbol{\mu}$ denote cluster means,

$$D_{WL}(G, H) = \frac{1}{|G| + |H|} \sum_{\mathbf{x}_i \in G \cup H} \|\mathbf{x}_i - \boldsymbol{\mu}_{G \cup H}\|^2 - \frac{1}{|G|} \sum_{\mathbf{x}_i \in G} \|\mathbf{x}_i - \boldsymbol{\mu}_G\|^2$$

Deciding what clusters to merge

For real-valued data specifically, with the Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, a different common choice is:

- **Ward's Linkage** looks at how much the within-cluster variances would increase after the merge. Letting $\boldsymbol{\mu}$ denote cluster means,

$$D_{WL}(G, H) = \frac{1}{|G| + |H|} \sum_{\mathbf{x}_i \in G \cup H} \|\mathbf{x}_i - \boldsymbol{\mu}_{G \cup H}\|^2 - \frac{1}{|G|} \sum_{\mathbf{x}_i \in G} \|\mathbf{x}_i - \boldsymbol{\mu}_G\|^2 - \frac{1}{|H|} \sum_{\mathbf{x}_i \in H} \|\mathbf{x}_i - \boldsymbol{\mu}_H\|^2.$$

Deciding what clusters to merge

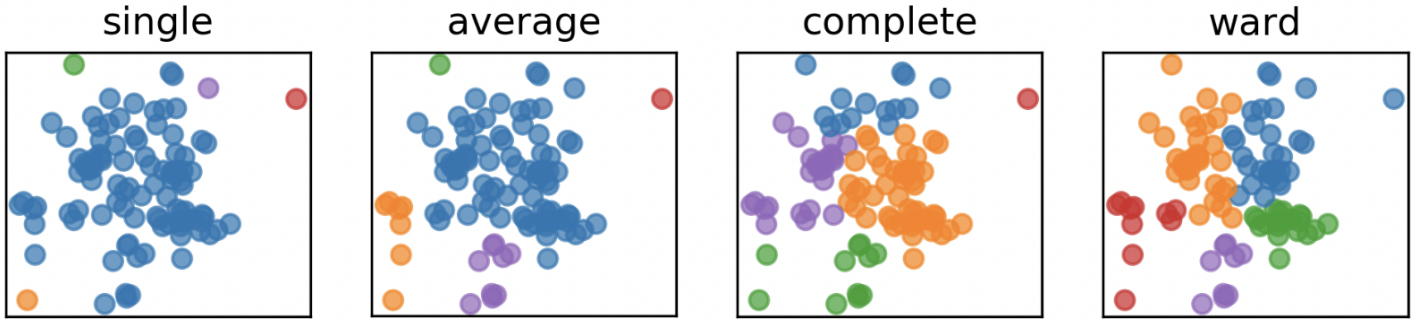
For real-valued data specifically, with the Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, a different common choice is:

- **Ward's Linkage** looks at how much the within-cluster variances would increase after the merge. Letting $\boldsymbol{\mu}$ denote cluster means,

$$D_{WL}(G, H) = \frac{1}{|G| + |H|} \sum_{\mathbf{x}_i \in G \cup H} \|\mathbf{x}_i - \boldsymbol{\mu}_{G \cup H}\|^2 - \frac{1}{|G|} \sum_{\mathbf{x}_i \in G} \|\mathbf{x}_i - \boldsymbol{\mu}_G\|^2 - \frac{1}{|H|} \sum_{\mathbf{x}_i \in H} \|\mathbf{x}_i - \boldsymbol{\mu}_H\|^2.$$

(This has some resemblance to K-means. . . !)

Consequences on cluster size distributions




```
single :   [96  1  1  1  1]
average :  [82  9  7  1  1]
complete : [50 24 14 11  1]
ward :     [31 30 20 10  9]
```

Ward's clustering tends to give more "evenly" sized clusters.

Figure: Andreas Müller

Mixture models

Probabilistic models

Basic idea: start with a simple probability distribution $p(\mathbf{x}|\theta)$, with parameters θ.

- Examples: Gaussian, Bernoulli, ...

Probabilistic models

Basic idea: start with a simple probability distribution $p(\mathbf{x}|\theta)$, with parameters θ.

- Examples: Gaussian, Bernoulli, ...

Define a more complex distribution as a weighted sum of several of these distributions (indexed by k):

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

Probabilistic models

Basic idea: start with a simple probability distribution $p(\mathbf{x}|\theta)$, with parameters θ.

- Examples: Gaussian, Bernoulli, ...

Define a more complex distribution as a weighted sum of several of these distributions (indexed by k):

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

Here, the values $\pi_1, \ldots, \pi_K$ are the **mixture weights**, corresponding to the relative contribution of the different **mixture components** $p(\mathbf{x}|\theta_k)$.

Probabilistic models

Basic idea: start with a simple probability distribution $p(\mathbf{x}|\theta)$, with parameters θ.

- Examples: Gaussian, Bernoulli, ...

Define a more complex distribution as a weighted sum of several of these distributions (indexed by k):

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

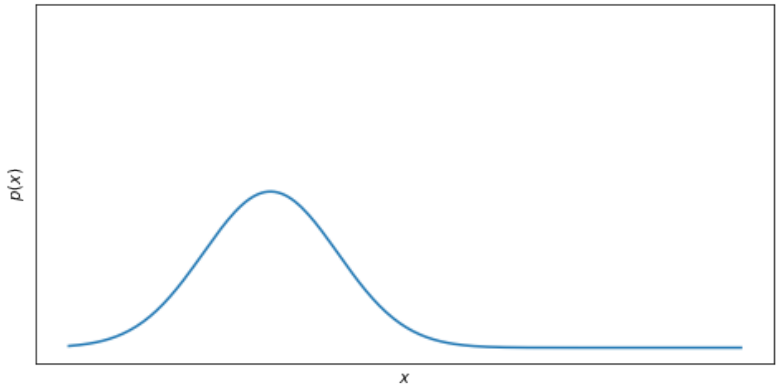
Here, the values $\pi_1, \ldots, \pi_K$ are the **mixture weights**, corresponding to the relative contribution of the different **mixture components** $p(\mathbf{x}|\theta_k)$.

Typically, this is a convex combination: $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$.

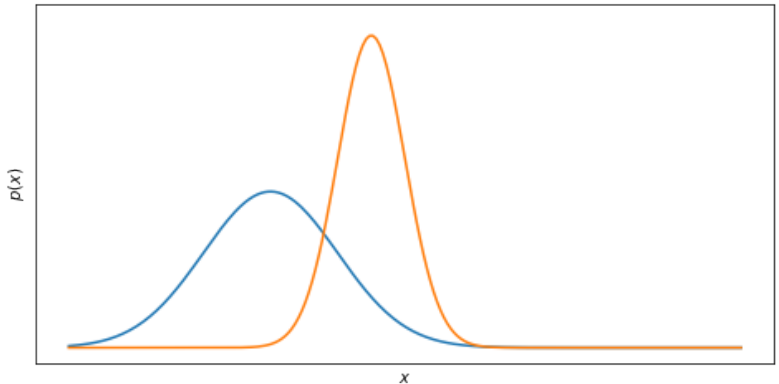
Gaussian mixture models

To make things more concrete, for the moment we will focus on Gaussian mixture models, where each mixture component is modeled by a Gaussian $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$.

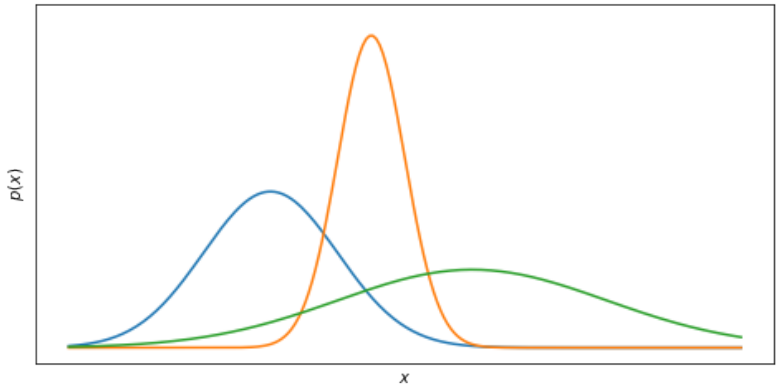
Gaussian mixture cartoon



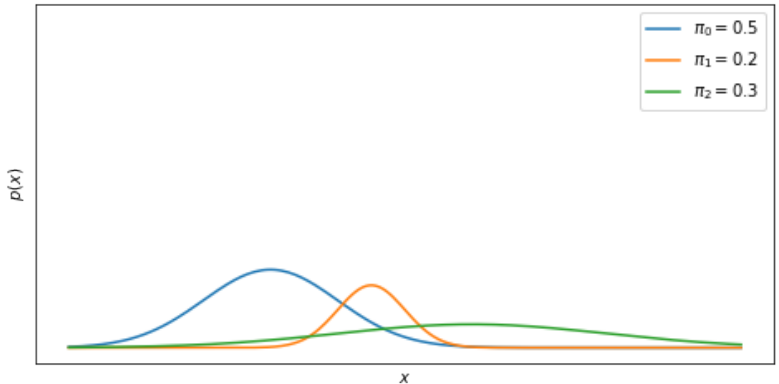
Gaussian mixture cartoon



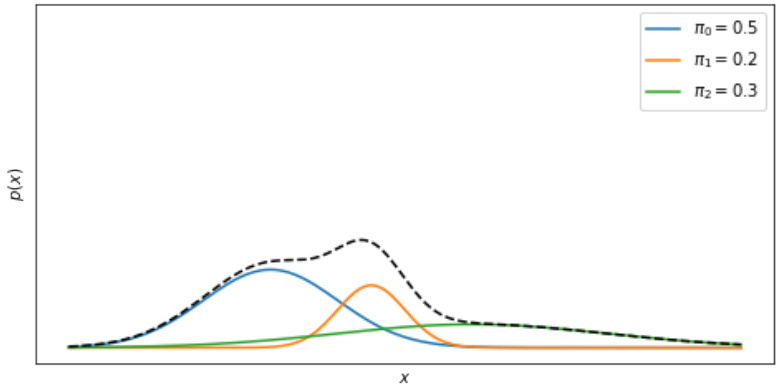
Gaussian mixture cartoon



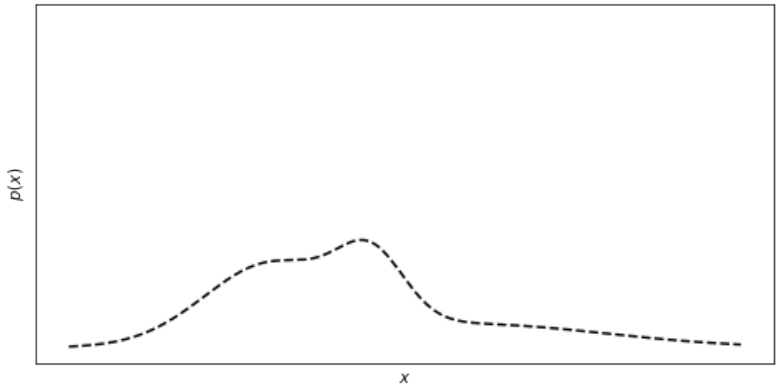
Gaussian mixture cartoon



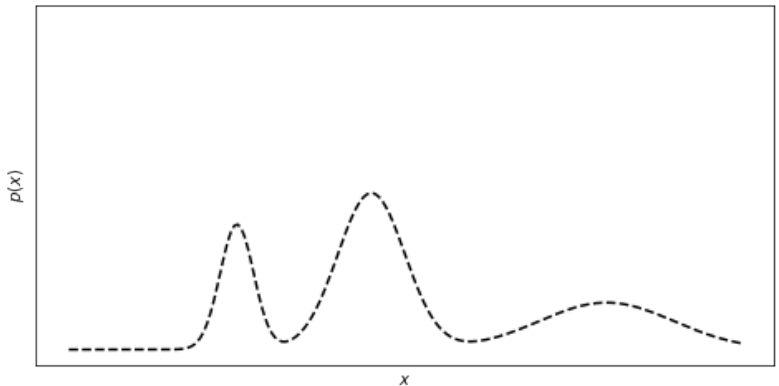
Gaussian mixture cartoon



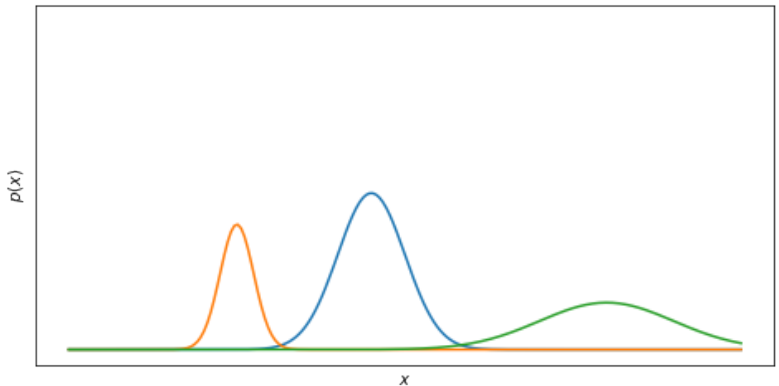
Gaussian mixture cartoon



Relevance to clustering



Relevance to clustering



Mixtures as latent variable models

Since the values of π_k are nonnegative and sum to one, we can also view them as a probability distribution over the different components, and write the model as

$$p(z_n) = \text{Discrete}(\boldsymbol{\pi})$$
$$p(\mathbf{x}_n | z_n = k) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Mixtures as latent variable models

Since the values of π_k are nonnegative and sum to one, we can also view them as a probability distribution over the different components, and write the model as

$$p(z_n) = \text{Discrete}(\boldsymbol{\pi})$$
$$p(\mathbf{x}_n | z_n = k) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Each data point $\mathbf{x}_n$ then is distributed

$$p(\mathbf{x}_n) = \sum_k p(z_n = k) p(\mathbf{x}_n | z_n = k) = \sum_k \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

Mixtures as latent variable models

If we frame the mixture model as

$$p(z_n) = \text{Discrete}(\boldsymbol{\pi}), \qquad p(\mathbf{x}_n | z_n = k) = \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

then z_n is a **latent random variable** which corresponds to an assignment of the datapoint $\mathbf{x}_n$ to a particular mixture component (or "cluster") from $1, \ldots, K$.

Mixtures as latent variable models

If we frame the mixture model as

$$p(z_n) = \text{Discrete}(\boldsymbol{\pi}), \qquad p(\mathbf{x}_n|z_n = k) = \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

then z_n is a **latent random variable** which corresponds to an assignment of the datapoint $\mathbf{x}_n$ to a particular mixture component (or "cluster") from $1, \ldots, K$.

There are two things we need to do:

Mixtures as latent variable models

If we frame the mixture model as

$$p(z_n) = \text{Discrete}(\boldsymbol{\pi}), \qquad p(\mathbf{x}_n|z_n = k) = \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

then z_n is a **latent random variable** which corresponds to an assignment of the datapoint $\mathbf{x}_n$ to a particular mixture component (or "cluster") from $1, \ldots, K$.

There are two things we need to do:

- Infer the posterior distribution $p(z_n = k|\mathbf{x}_n)$ for each data point. This corresponds to a probability distribution over which cluster $\mathbf{x}_n$ is associated with — akin to a "soft" version of K-means.

Mixtures as latent variable models

If we frame the mixture model as

$$p(z_n) = \text{Discrete}(\boldsymbol{\pi}), \qquad p(\mathbf{x}_n|z_n = k) = \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

then z_n is a **latent random variable** which corresponds to an assignment of the datapoint $\mathbf{x}_n$ to a particular mixture component (or "cluster") from $1, \ldots, K$.

There are two things we need to do:

- Infer the posterior distribution $p(z_n = k|\mathbf{x}_n)$ for each data point. This corresponds to a probability distribution over which cluster $\mathbf{x}_n$ is associated with — akin to a "soft" version of K-means.
- Estimate the parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ which correspond to cluster means and cluster covariances.

Gaussian mixtures: intuition

We'll go over the math behind estimating Gaussian mixture models (GMMs) in the next lecture. Algorithmically, it's simple enough to understand:

Gaussian mixtures: intuition

We'll go over the math behind estimating Gaussian mixture models (GMMs) in the next lecture. Algorithmically, it's simple enough to understand:

Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$. (Consider running K-means for $\boldsymbol{\mu}$!)

Gaussian mixtures: intuition

We'll go over the math behind estimating Gaussian mixture models (GMMs) in the next lecture. Algorithmically, it's simple enough to understand:

Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$. (Consider running K-means for $\boldsymbol{\mu}$!)

- Soft-assign each point $\mathbf{x}_n$ to a cluster k, based on $p(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_K)$.

Gaussian mixtures: intuition

We'll go over the math behind estimating Gaussian mixture models (GMMs) in the next lecture. Algorithmically, it's simple enough to understand:

Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$. (Consider running K-means for $\boldsymbol{\mu}$!)

- Soft-assign each point $\mathbf{x}_n$ to a cluster k, based on $p(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_K)$.
  - *(Analogous to hard cluster-assignment based on nearest mean in K-means)*

Gaussian mixtures: intuition

We'll go over the math behind estimating Gaussian mixture models (GMMs) in the next lecture. Algorithmically, it's simple enough to understand:

Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$. (Consider running K-means for $\boldsymbol{\mu}$!)

- Soft-assign each point $\mathbf{x}_n$ to a cluster k, based on $p(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_K)$.
  - *(Analogous to hard cluster-assignment based on nearest mean in K-means)*
- Update cluster parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ by computing a mean and covariance of data points $\mathbf{x}_n$, weighted by the probability that $z_n = k$

Gaussian mixtures: intuition

We'll go over the math behind estimating Gaussian mixture models (GMMs) in the next lecture. Algorithmically, it's simple enough to understand:

Initialize $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$. (Consider running K-means for $\boldsymbol{\mu}$!)

- Soft-assign each point $\mathbf{x}_n$ to a cluster k, based on $p(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_K)$.
  - *(Analogous to hard cluster-assignment based on nearest mean in K-means)*
- Update cluster parameters $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$ by computing a mean and covariance of data points $\mathbf{x}_n$, weighted by the probability that $z_n = k$
  - *(Analogous to re-computing cluster centers in K-means)*

Gaussian mixture estimation

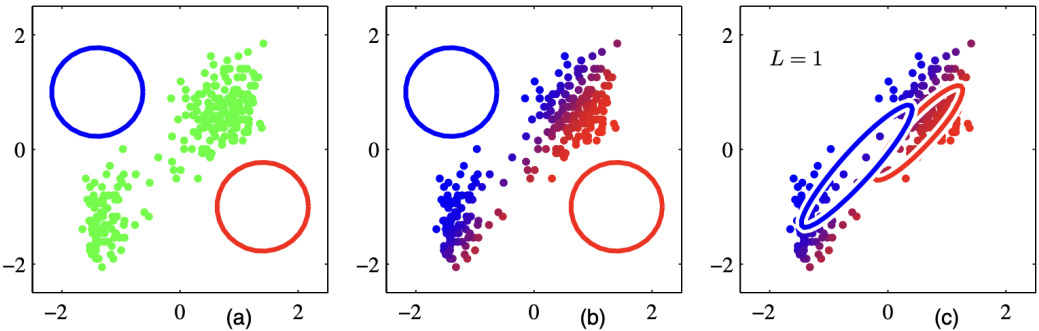


Figure: PRML

Gaussian mixture estimation

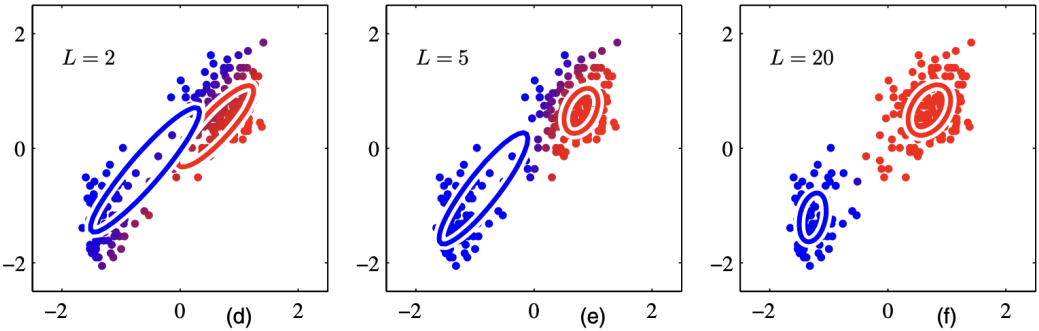


Figure: PRML

GMM or K-means?

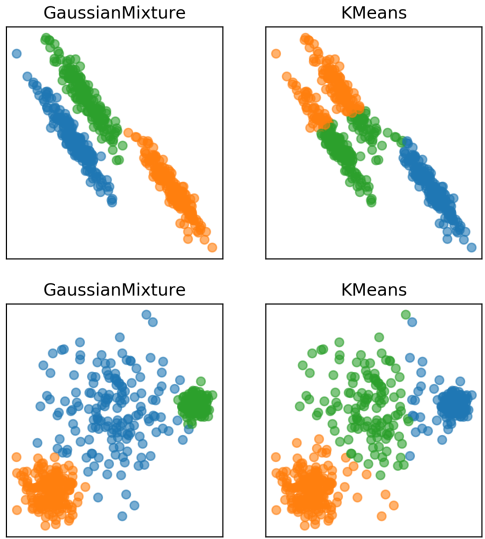


Figure: Andreas Müller

Thoughts on mixture models

Good things:

- Natural probabilistic model for clustering; generalizes K-means nicely
- Easy to expand to other data types (e.g. mixtures of Bernoullis, etc); different data types or likelihoods per-dimension are fine
- Choosing the number K can be done by looking at the likelihood on held-out test data, or using a Bayesian approach

Thoughts on mixture models

Good things:

- Natural probabilistic model for clustering; generalizes K-means nicely
- Easy to expand to other data types (e.g. mixtures of Bernoullis, etc); different data types or likelihoods per-dimension are fine
- Choosing the number K can be done by looking at the likelihood on held-out test data, or using a Bayesian approach

Less good things:

- Only can really find "compact" clusters
- Non-convex objective; initialization dependent
- GMMs are sensitive to outliers — consider using mixtures of Student-t distributions instead