Unsupervised Clustering (Pt. 2)

Machine Learning for Biomedical Data

Scott Doyle / scottdoy@buffalo.edu

Recap


Recap: Why Use Unlabeled Data?

- Samples are cheap to collect, costly to label.
- Clustering gives you a free(ish) look at the structure of your data.
- Clustering does not preclude performing hands-on labeling later.
- Unsupervised methods adapt to new trends in the data over time.
- Some methods learn features as well as class labels.

Recap: Component Densities and Mixing Parameters

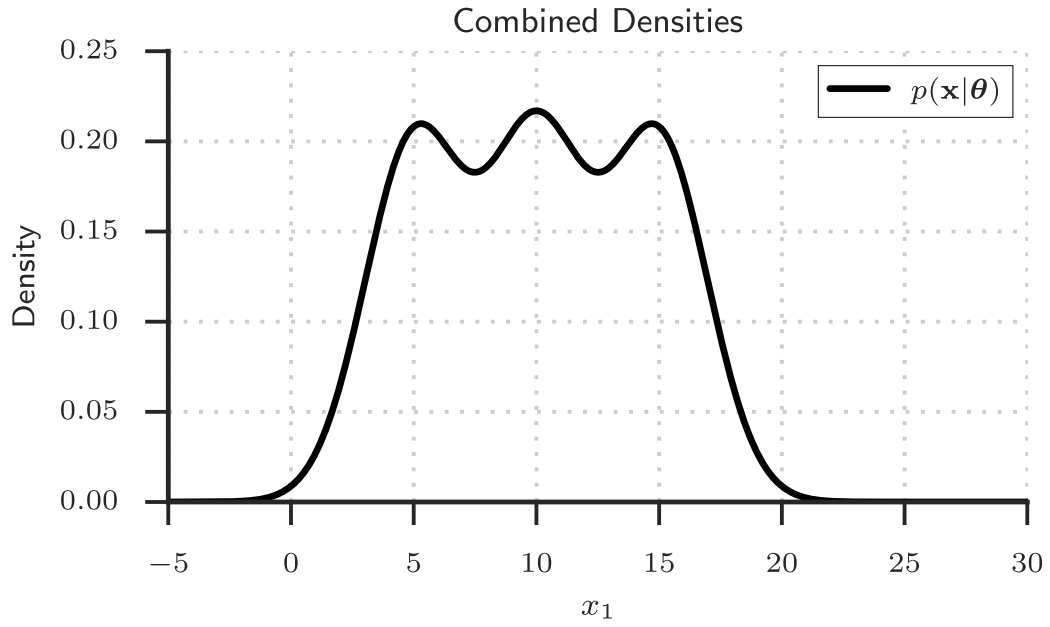
$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

In this form, $p(\mathbf{x}|\boldsymbol{\theta})$ is known as a **mixture density**.

Conditional densities $p(\mathbf{x}|\omega_j, \boldsymbol{\theta_j})$ are the **component densities**.

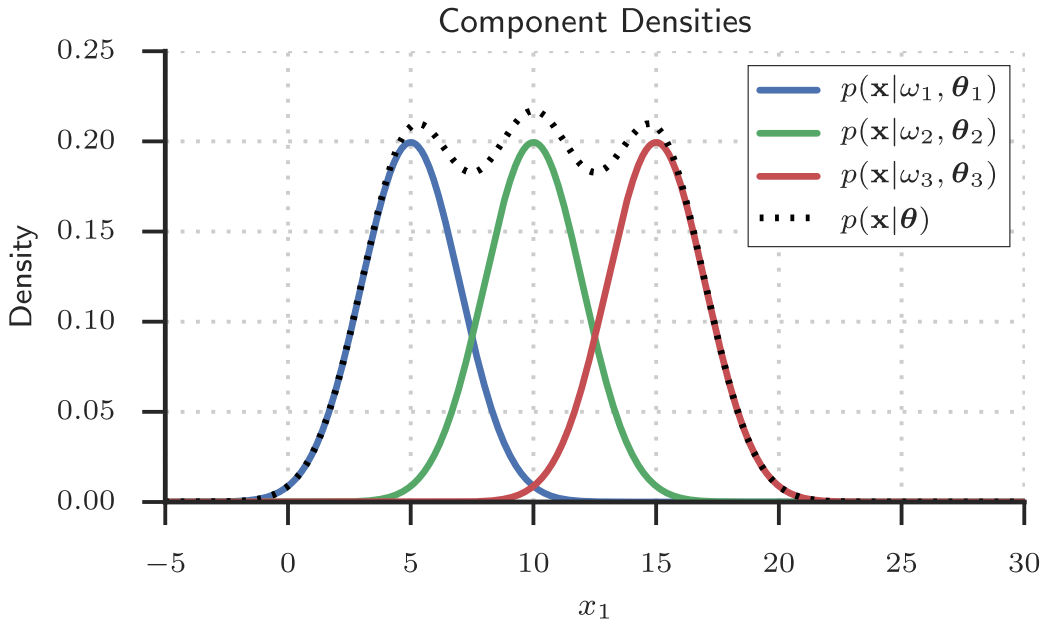
Priors $P(\omega_j)$ are the **mixing parameters**.

Recap: Component Densities and Mixing Parameters



Observed Sample Distribution

Recap: Component Densities and Mixing Parameters



Underlying Component Distributions

Recap: Normal Mixtures and Additional Assumptions

We've already assumed we know the form of each mixture density (namely, they are Gaussian).

There are four parameters that we may not know:

- $\boldsymbol{\mu}_i$, the multivariate mean;
- $\boldsymbol{\Sigma}_i$, the covariance matrix;
- $P(\omega_i)$, the prior probability; and
- c, the total number of classes.

We CAN evaluate the system if we don't know anything.

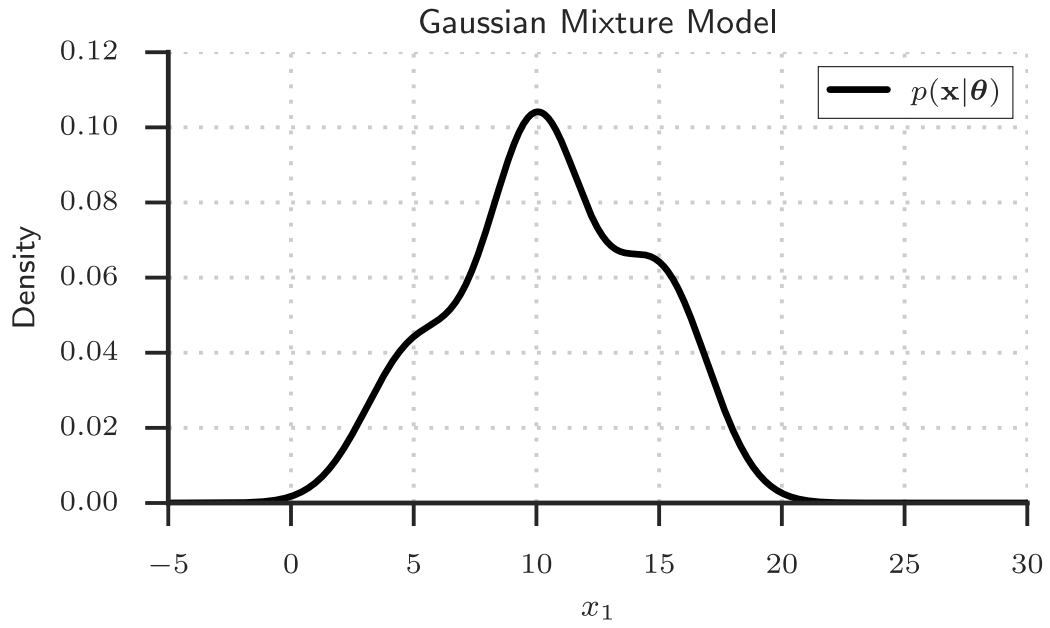
Recap: Estimating our Parameter Sets for Clustering

Our strategy for finding c and $P(\omega_i)$ is simply to estimate them from the domain (similar to how we estimate $P(\omega_i)$ in the Bayesian case).

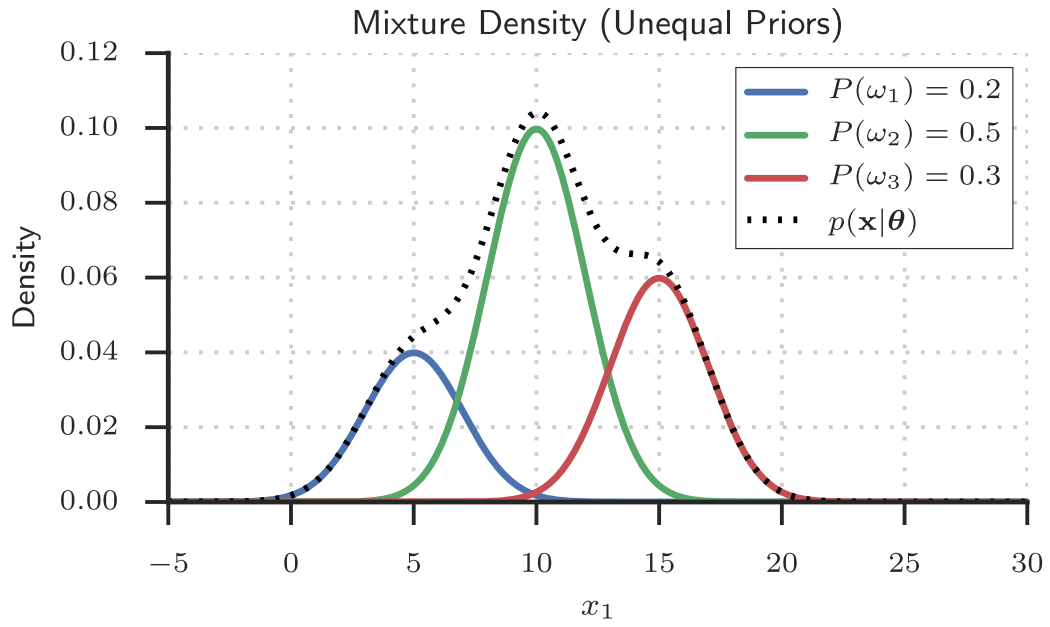
c can be optimized by looking at the clustering criteria (later in this lecture).

Typically (as in Bayes), we select non-informative $P(\omega_i)$ – what happens if we select unequal priors?

Recap: Unequal Prior Values



Recap: Unequal Prior Values



Recap: MLE: Estimating θ_i

For $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$, we find the derivative of our mixture likelihoods with respect to the parameters, set equal to zero, and iteratively find the **most likely** parameter set that would give us our training data.

If our mixture density is this:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j) P(\omega_j)$$

And we assume that each $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ is Gaussian, then we can differentiate the natural logarithm with respect to each parameter in turn and calculate the maximum likelihood estimate.

Recap: MLE: Solving for Σ

There's a lot of tricky math and derivations, but at the end of the day we can get an estimate for $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$:

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) = \frac{|\hat{\boldsymbol{\Sigma}}_i|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_i)^T\hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_i)\right]\hat{P}(\omega_i)}{\sum_{j=1}^{c}|\hat{\boldsymbol{\Sigma}}_j|^{-\frac{1}{2}} \exp\left[-\frac{1}{2}(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_j)^T\hat{\boldsymbol{\Sigma}}_j^{-1}(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_j)\right]\hat{P}(\omega_j)}$$

With this long, ugly thing, we can estimate the likelihood that a point $\mathbf{x}_k$ belongs to ω_i.

Simple explanation: if the squared Mahalanobis distance, $(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^T\hat{\boldsymbol{\Sigma}}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)$, is small, then $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ is large.

Recap: k-Means Clustering

If we replace the Mahalanobis with the squared Euclidean distance $|\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i|^2$, we can find the mean $\hat{\boldsymbol{\mu}}_m$ nearest to $\mathbf{x}_k$.

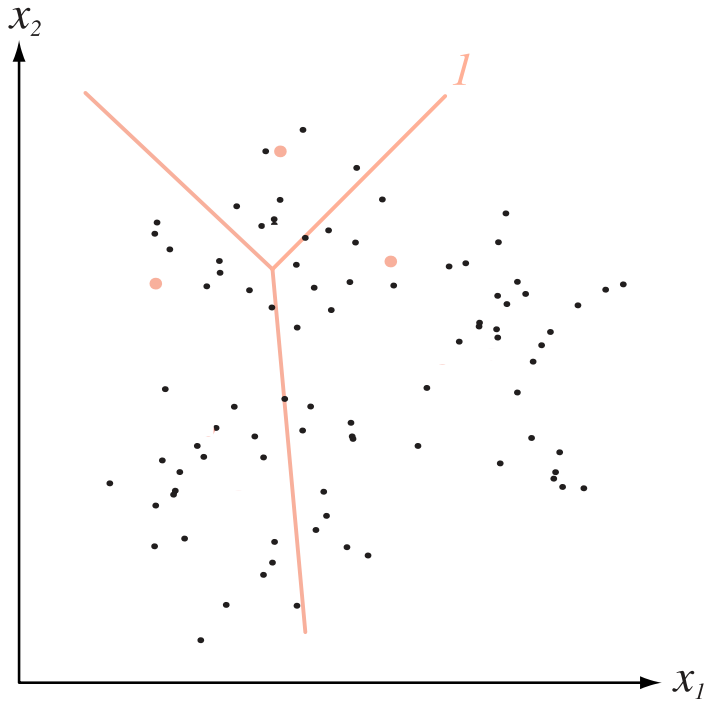
Thus we can approximate $\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ as:

$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \simeq \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise.} \end{cases}$$

Then we can plug this into the equation we got before and solve for $\hat{\boldsymbol{\mu}}_1, \cdots, \hat{\boldsymbol{\mu}}_c$.

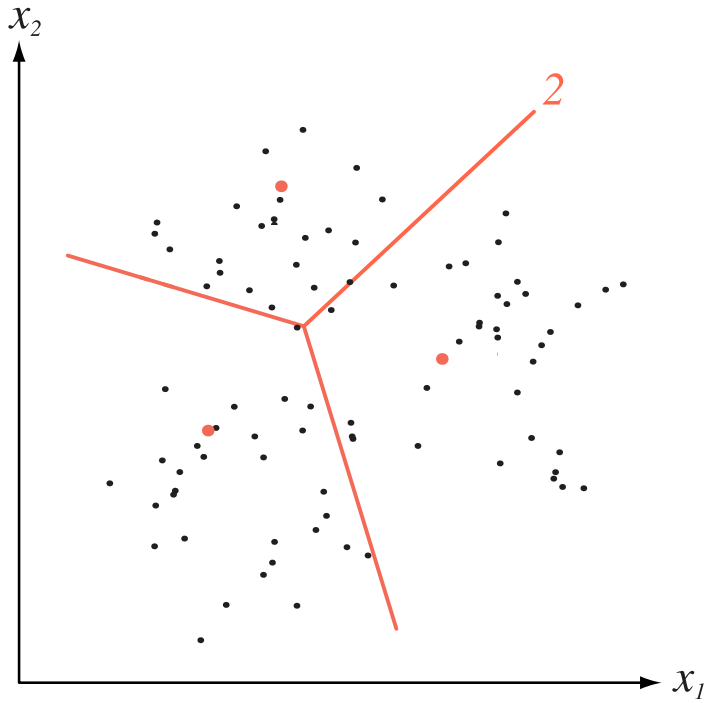
We can “initialize” by selecting c class centroids at random from the unlabeled data, and then iterating.

Recap: k-Means In Sample Space



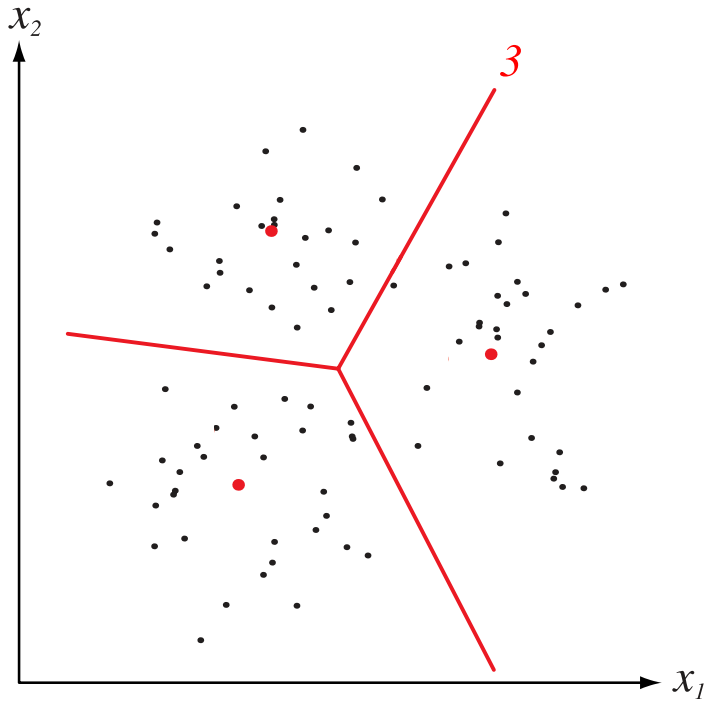
k-Means Sample Space

Recap: k-Means In Sample Space



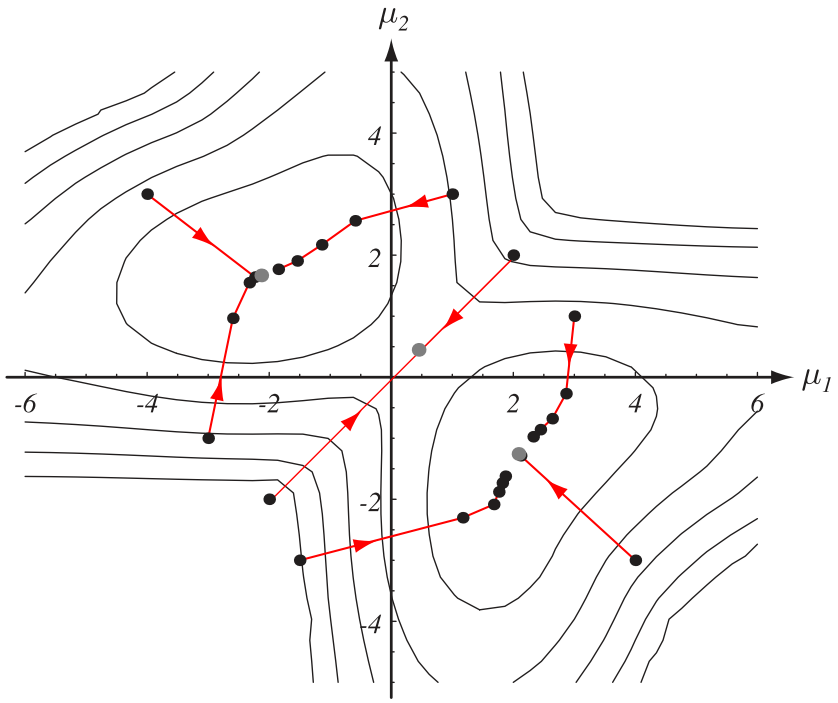
k-Means Sample Space

Recap: k-Means In Sample Space



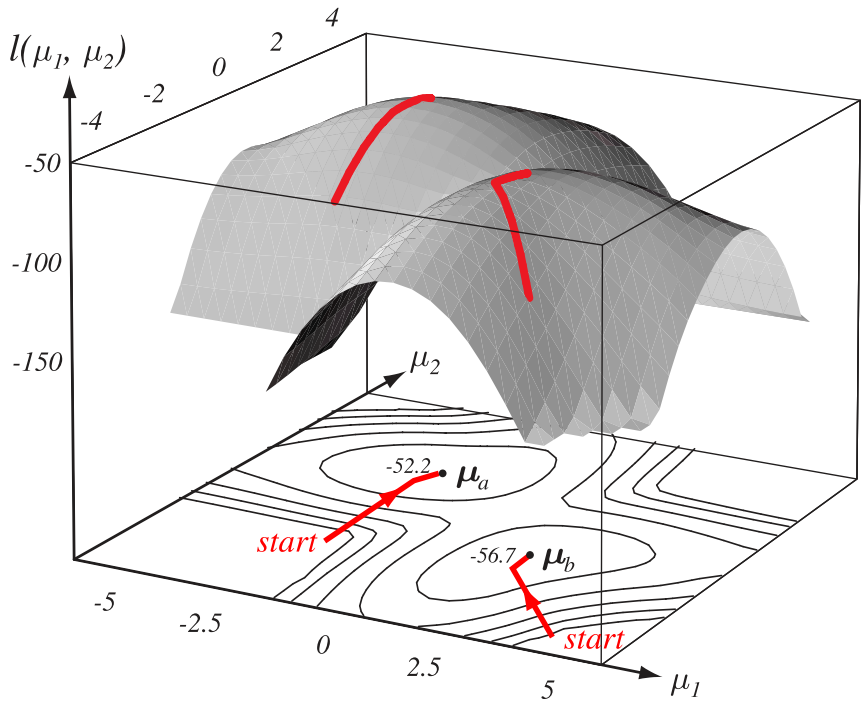
k-Means Sample Space

Recap: k-Means Hill-climbing

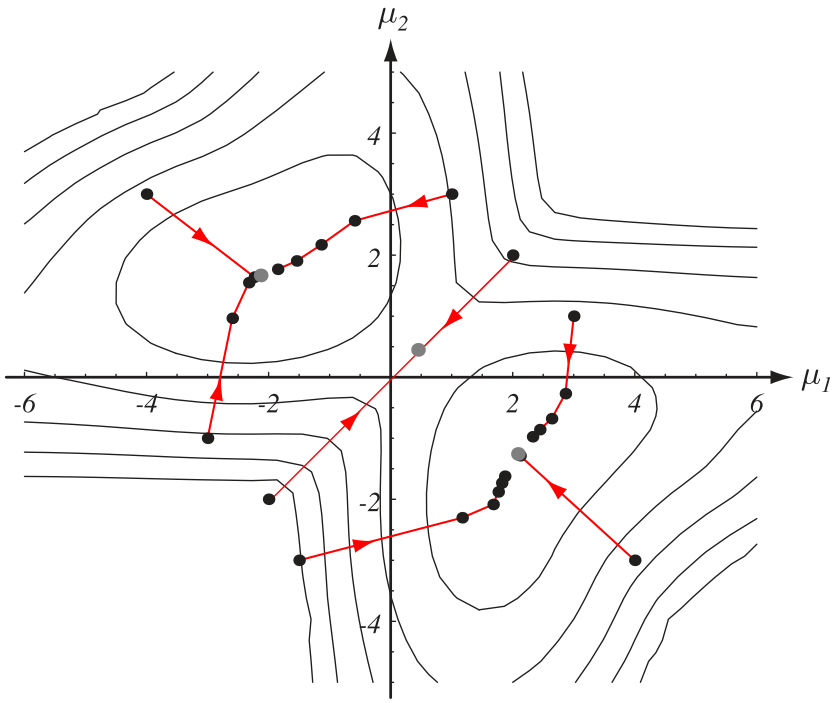


Stoachastic hill-climbing in k-means.

Recap: Comparing MLE and k-Means



MLE Example



k-Means Example

Comparison of MLE and k-Means. Since the overlap between the components is

Recap: k-Means Summary

k-Means is a staple of unsupervised clustering methods.

It is simple and fast.

When does it fail?

- If we are wrong about the number of clusters, we will converge on parameters that don't "mean" anything.
- If the clusters are too close to one another, there may not be enough samples to properly "ascend" to the true value of μ.

Remember: **everything** is dependent on your features!

Data Description and Clustering

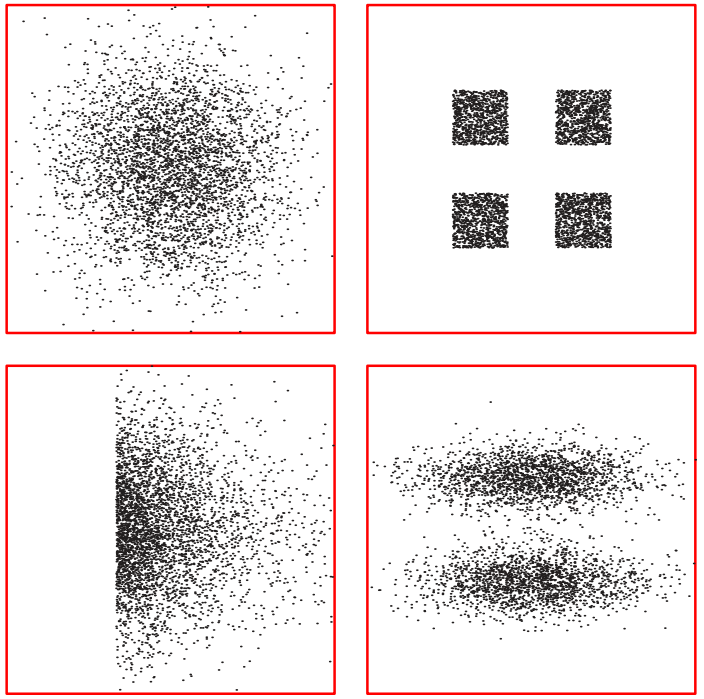
How Well Did We Cluster?

k-Means finds the parameters $\boldsymbol{\theta}$ of the underlying processes that control our samples.

Clusters and **Classes** are NOT synonymous!

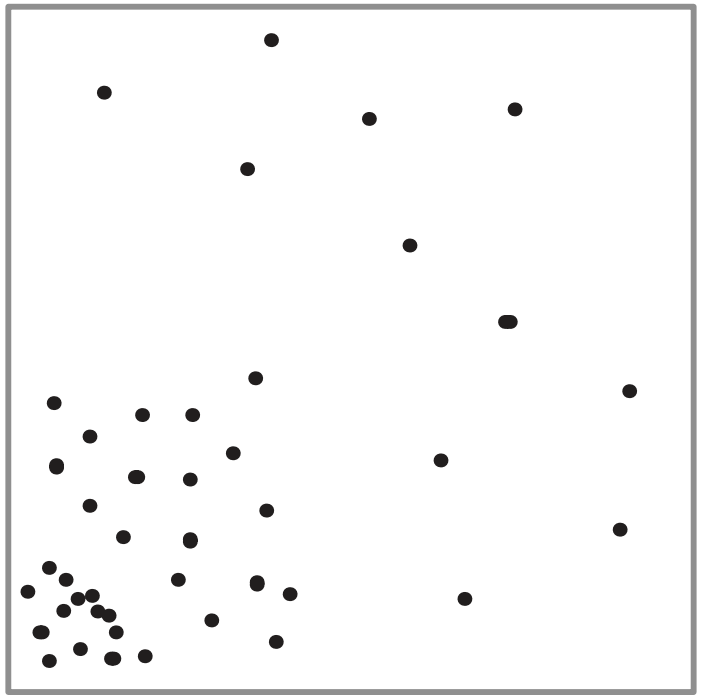
- Some classes are multi-modal: “Atypical” nuclei can be too small OR too large.
- Some features are bad: If $\boldsymbol{\theta}_1 \approx \boldsymbol{\theta}_2$, the feature cannot distinguish ω_1 and ω_2.

Examples of Misleading Parameters



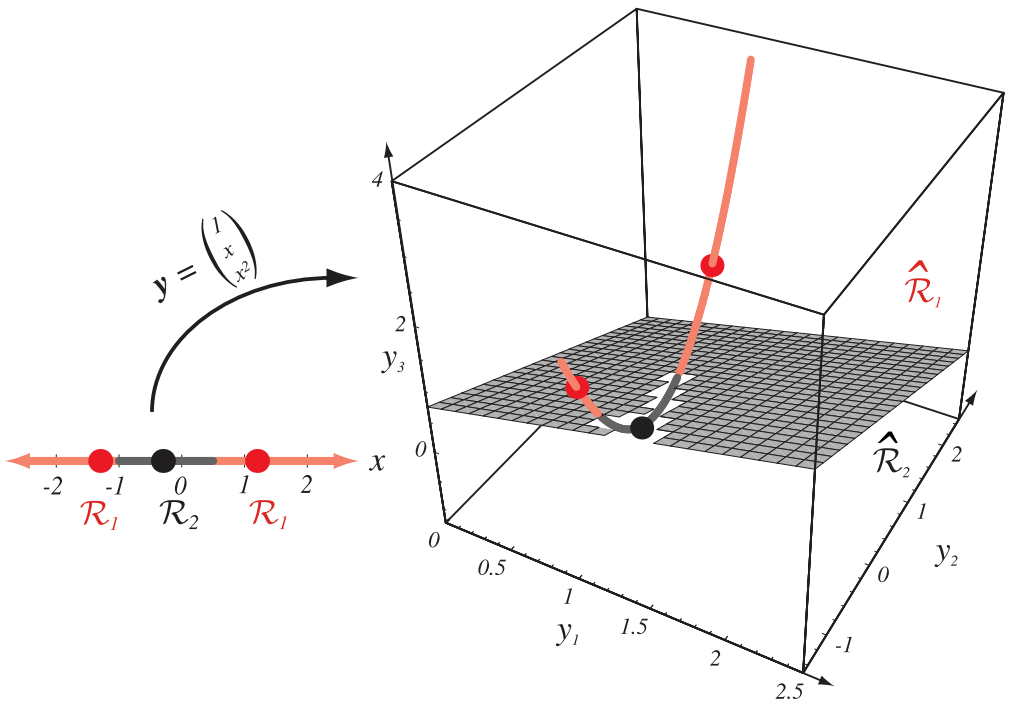
Distributions with equal μ and Σ.

Identifying the “True” Clusters



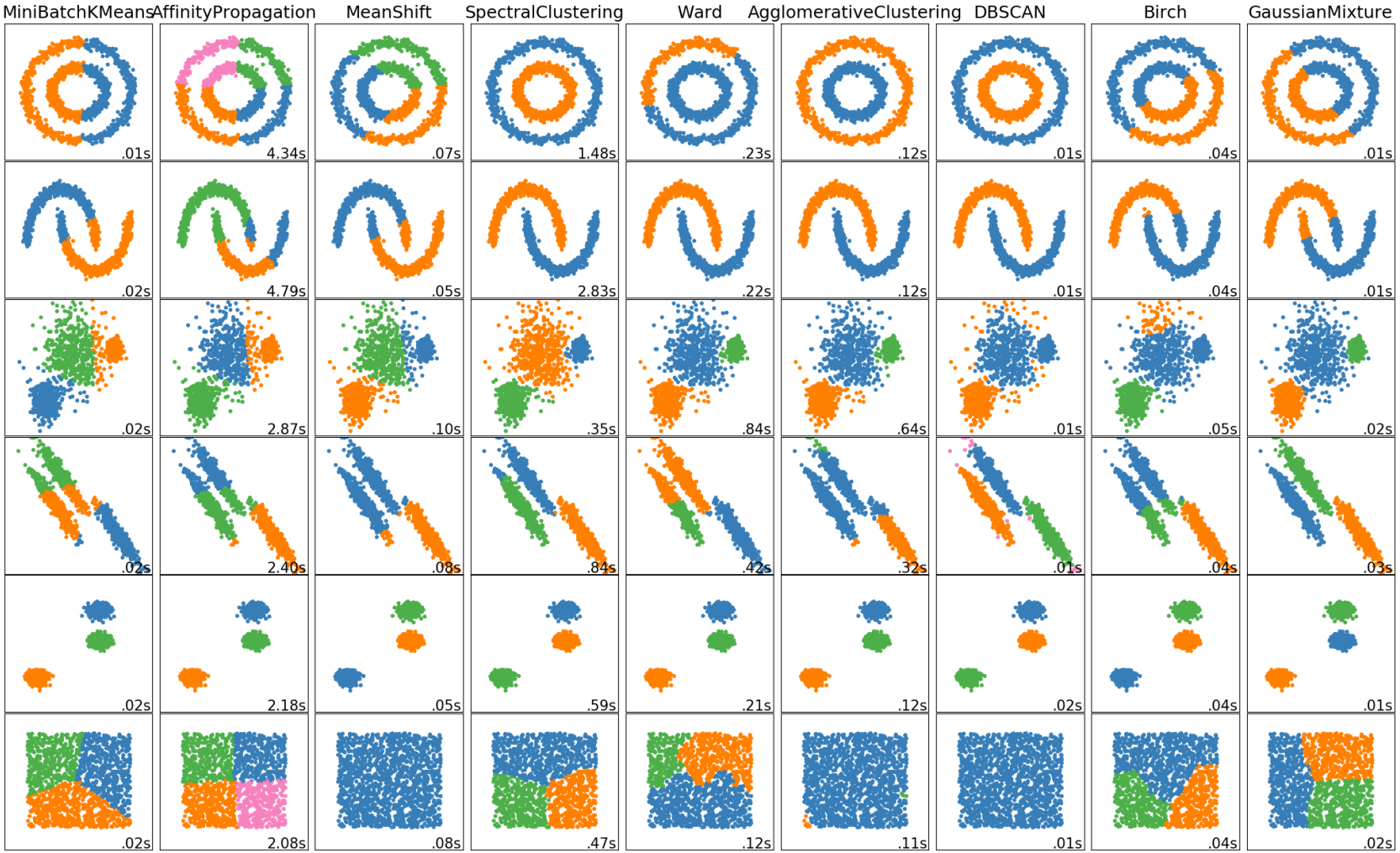
How many clusters are in this data?

Clustering vs. Classification



- A class is a label that **you** decide on, and is not necessarily synonymous with clusters.
- If we don't have class labels, we **cannot** say that $\mathcal{R}_1$ on the left is the same decision region as $\mathcal{R}_1$ on the right.

Illustration of Clustering Methods



The First Major Issue: Defining Similarity

Unlabeled samples rely **completely** on their descriptive features.

We need to quantitatively say that samples in one cluster are “more similar” to each other than they are to samples in another cluster.

This is a **similarity metric**.

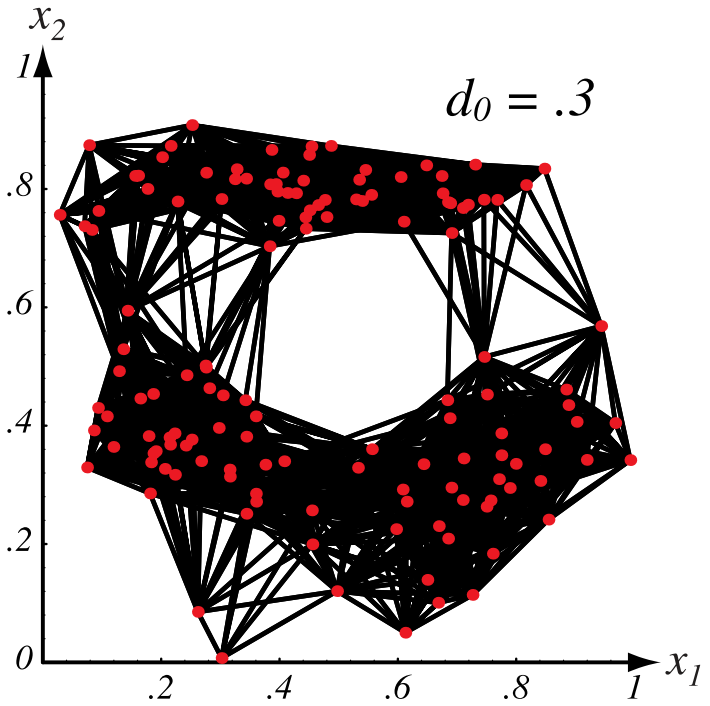
Distance as Inverse Similarity

Distance may represent the inverse of **similarity**: High distance = low similarity

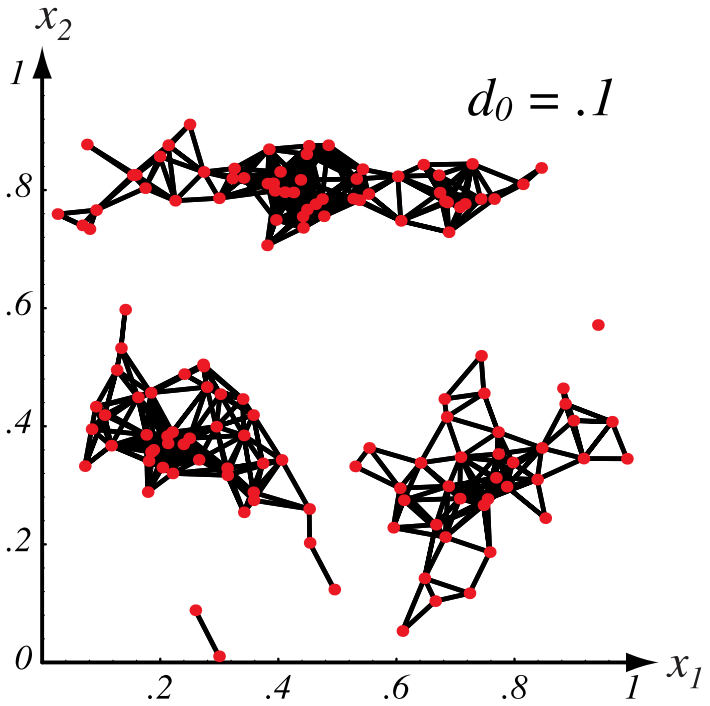
In Euclidean space, $\mathbf{x}_a$ is **more similar** to $\mathbf{x}_b$ than $\mathbf{x}_c$ if $|\mathbf{x}_a - \mathbf{x}_b|^2 < |\mathbf{x}_a - \mathbf{x}_c|^2$.

We can cluster points by setting a threshold d_0, where two points $\mathbf{x}_a$ and $\mathbf{x}_b$ belong to the same cluster if $|\mathbf{x}_a - \mathbf{x}_b|^2 < d_0$.

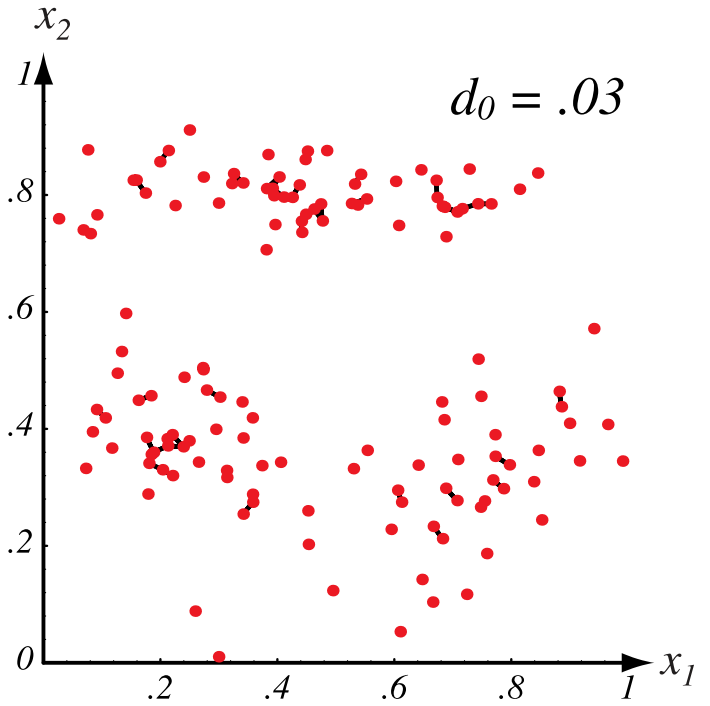
Examples of Distance Thresholding: High Threshold



High Threshold Cluster



Medium Threshold Cluster



Low Threshold Cluster

Choosing a Similarity Metric

Discrete Metric:

$$d(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 & \text{if } \mathbf{x} \neq \mathbf{y} \\ 0 & \text{if } \mathbf{x} = \mathbf{y} \end{cases}$$

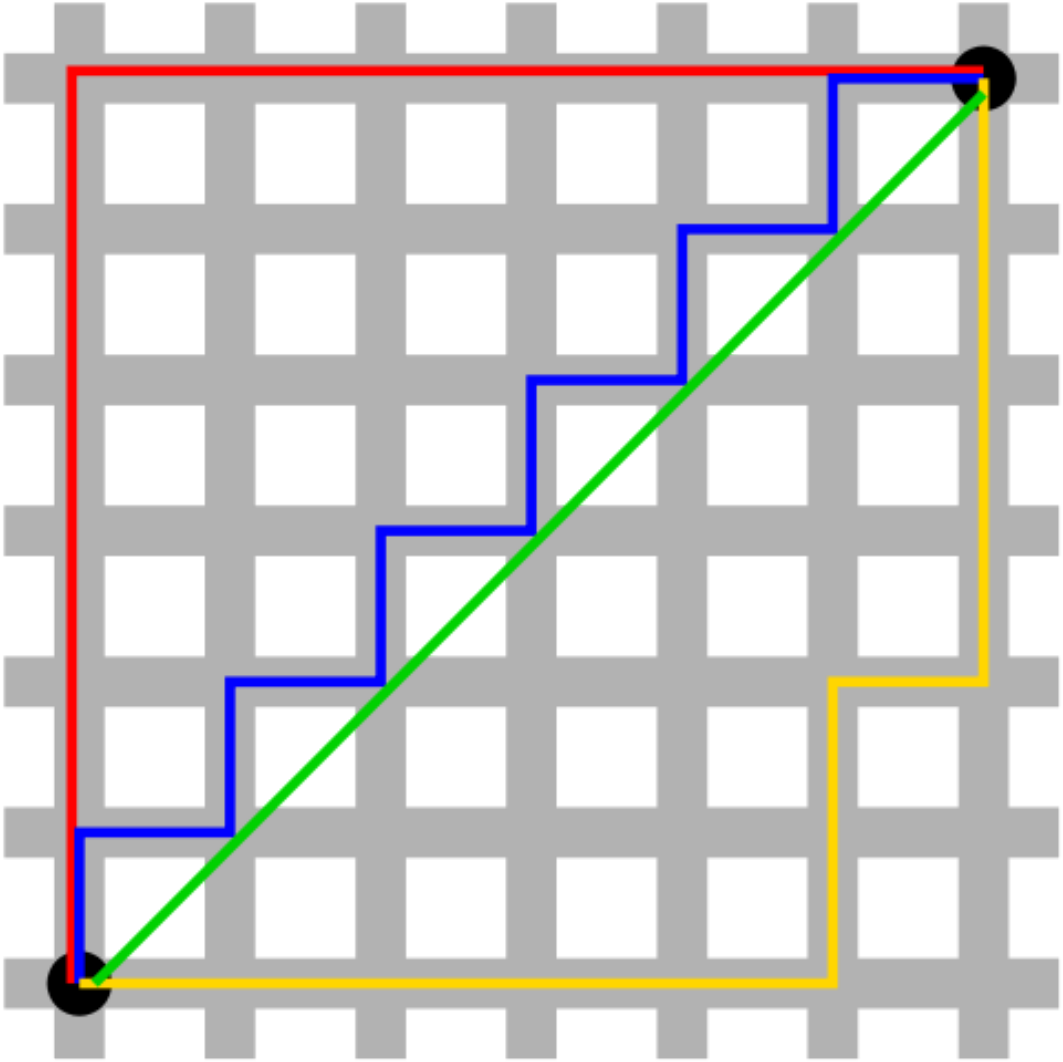
Euclidean Metric:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_d - y_d)^2}$$

Taxicab Metric:

$$d(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{d} |x_i - y_i|$$

Illustration of Distances



Choosing a Similarity Metric: Invariance to Transforms

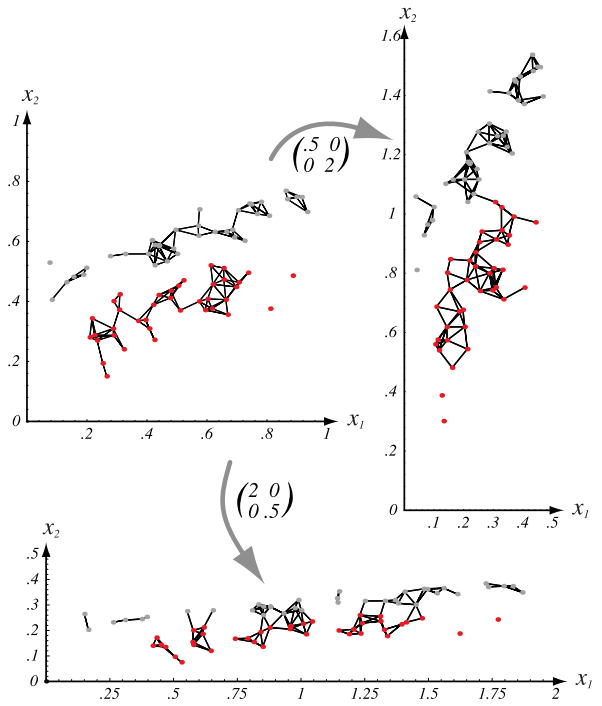
Euclidean distance is a good first choice:

$$D(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_{k=1}^{d} (x_k - x'_k)^2}$$

- **Isotropy**: distances in all dimensions must be equivalent.
- **Smoothness**: distances in all feature ranges are equivalent.
- **Linearity**: data should be observed throughout the feature space.

Euclidean distance is robust to rigid transformations of the feature space (rotation and translation), but are NOT robust to **arbitrary linear transformations** which distort distance relationships.

Rotation's Effect on Cluster Groupings



Cluster Scaling

Alternative Distance Metrics

We generalize from the Euclidean to the **Minkowski** metric:

$$D(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{d} |x_k - x'_k|^q \right)^{\frac{1}{q}}$$

If $q = 2$, then this is Euclidean; if $q = 1$, this is the **city block** metric.

You can also look at **Mahalanobis** distance, $(\mathbf{x} - \mathbf{x}')^T \Sigma^{-1} (\mathbf{x} - \mathbf{x}')$, which depends on the data itself to define the distance.

We can even abandon distance and define an arbitrary symmetric function $s(\mathbf{x}, \mathbf{x}')$ as some measurement of "similarity", e.g. the angle between two vectors:

$$s(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x}^T \mathbf{x}'}{|\mathbf{x}||\mathbf{x}'|}$$

Criterion Functions

The Second Major Issue: Evaluation

Suppose we've got our unlabeled training set $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$.

We want to divide this into exactly c disjoint subsets $\mathcal{D}_1, \ldots, \mathcal{D}_c$, which represent cluster memberships.

All samples in $\mathcal{D}_a$ should be more alike to each other than to those in $\mathcal{D}_b$, $a \neq b$.

Let's define ourselves a **criterion function** that is simply used to evaluate how good a given partition is; our task will then be to find a partition that "extremizes" our function.

Sum-of-Squared-Error Criterion

The **sum-of-squared-error** criterion is defined as the difference between the samples and the mean of the assigned cluster:

$$J_e = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} |\mathbf{x} - \mathbf{m}_i|^2$$

where:

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$$

The optimal partition is the one that minimizes J_e.

The idea is that it measures the error incurred if all of the samples in $\mathcal{D}_i$ were represented by the cluster center $\mathbf{m}_i$.

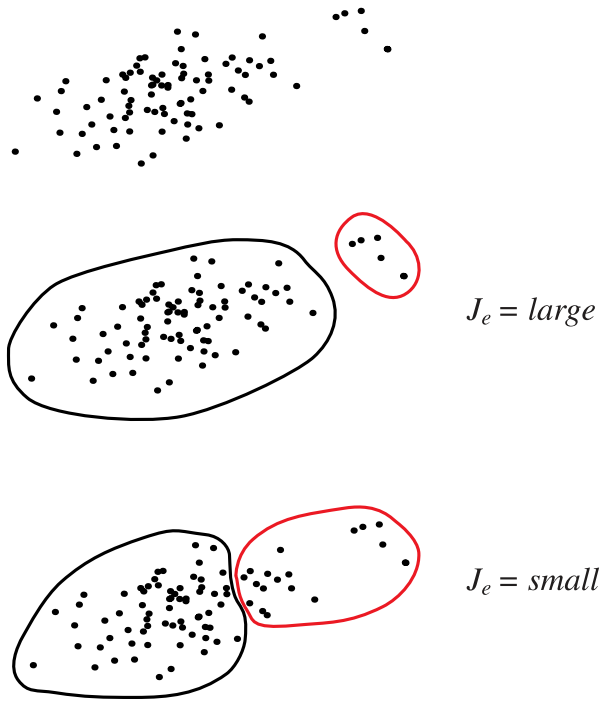
Sum-of-Squared-Error Criterion

This works very well if we have clusters that are:

- Equally evenly spread
- Far apart from each other

If one cluster is spread out and one is small, this criterion may end up trying to “even out” the differences by selecting a non-optimal partition.

Sum-of-Squared-Error Criterion



Sum-of-Squared Error Failure

Related Minimum Variance Criterion

We can wrangle the mean vectors out of the expression for Sum-of-Squared-Error and get the equivalent expression:

$$J_e = \frac{1}{2} \sum_{i=1}^{c} n_i \bar{s}_i$$

Where $\bar{s}_i$ can be formed into whatever kind of similarity function we want:

$$\bar{s}_i = \frac{1}{n_i^2} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_i} |\mathbf{x} - \mathbf{x}'|^2$$

$$\bar{s}_i = \frac{1}{n_i^2} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_i} s(\mathbf{x}, \mathbf{x}')$$

$$\bar{s}_i = \min_{\mathbf{x}, \mathbf{x}' \in \mathcal{D}_i} s(\mathbf{x}, \mathbf{x}')$$

Scatter Criteria

We can calculate a bunch of values that relate to the “scatter” of the data in each cluster.

We’ll come back to these when we discuss PCA, but here is a list of six quantities that are useful to know.

Mean Vectors and Scatter Matrices

Name	Equation
Mean vector for the ith cluster	$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x}$
Total mean vector	$\mathbf{m} = \frac{1}{n} \sum_{\mathcal{D}} \mathbf{x} = \frac{1}{n} \sum_{i=1}^{c} n_i \mathbf{m}_i$
Scatter matrix for the ith cluster	$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$
Within-cluster scatter matrix	$\mathbf{S}_W = \sum_{i=1}^{c} \mathbf{S}_i$
Between-cluster scatter matrix	$\mathbf{S}_B = \sum_{i=1}^{c} n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^T$
Total scatter matrix	$\mathbf{S}_T = \sum_{\mathbf{x} \in \mathcal{D}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T$
	$\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$

Trace Criterion

We need a scalar measure of the "size" of the scatter matrix, so we know if one set of points is more or less scattered than another.

One measure is the **trace** of the within-cluster scatter matrix, which is the sum of its diagonal elements.

Minimizing this turns out to be the sum-of-squared-error criterion:

$$\mathrm{Tr}[\mathbf{S}_W] = \sum_{i=1}^{c} \mathrm{Tr}[\mathbf{S}_i] = \sum_{i=1}^{c} \sum_{\mathbf{x} \in \mathcal{D}_i} |\mathbf{x} - \mathbf{m}_i|^2 = J_e$$

Note that $\mathrm{Tr}[\mathbf{S}_T]$ is independent of the partitioning (i.e. it doesn't change), and this is equal to $\mathrm{Tr}[\mathbf{S}_W] + \mathrm{Tr}[\mathbf{S}_B]$.

Therefore, minimizing $\mathrm{Tr}[\mathbf{S}_W]$ also maximizes $\mathrm{Tr}[\mathbf{S}_B]$.

Determinant Criterion

The determinant measures the square of the scattering volume.

$\mathbf{S}_B$ is singular if the number of clusters is less than or equal to the number of dimensions, so we don't want to use it for our criterion function.

If we assume that $\mathbf{S}_W$ is nonsingular, we can have:

$$J_d = |\mathbf{S}_W| = |\sum_{i=1}^{c} \mathbf{S}_i|$$

The trace and determinant criteria do not need to be the same, although they often are.

Invariant Criterion

We may want to look at the ratio of between-cluster and within-cluster matrices, $\mathbf{S}_W^{-1}\mathbf{S}_B$, as an optimal partition would ideally have small clusters that are also spread far apart.

This can be done by finding partitions that result in the eigenvalues of $\mathbf{S}_W^{-1}\mathbf{S}_B$ are large.

Since the trace of a matrix is the sum of its eigenvalues, we can maximize a criterion function as:

$$\mathrm{Tr}\left[\mathbf{S}_W^{-1}\mathbf{S}_B\right] = \sum_{i=1}^{d} \lambda_i$$

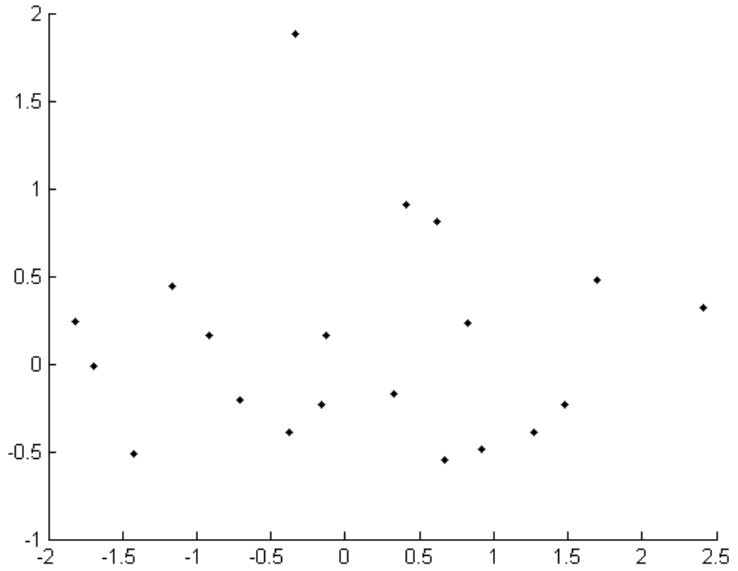
Invariant Criterion

By using the relation $\mathbf{S}_T = \mathbf{S}_W + \mathbf{S}_B$, we can derive the following invariant criterion functions:

$$J_f = \mathrm{Tr}\left[\mathbf{S}_T^{-1}\mathbf{S}_W\right] = \sum_{i=1}^{d} \frac{1}{1+\lambda_i}$$

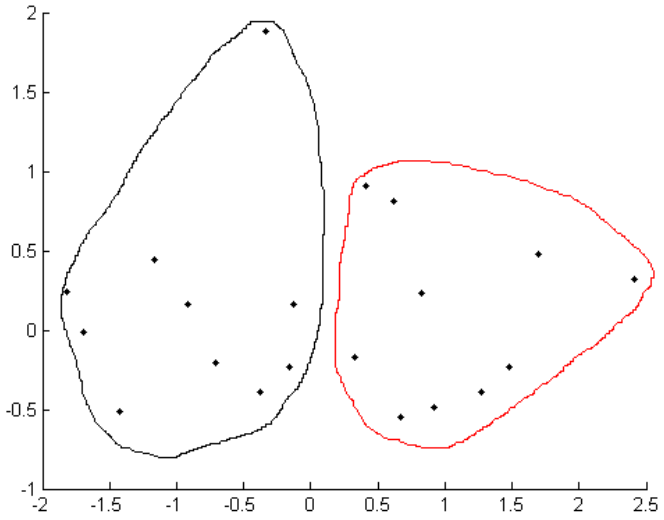
$$\frac{|\mathbf{S}_W|}{|\mathbf{S}_T|} = \prod_{i=1}^{d} \frac{1}{1+\lambda_i}$$

Comparing Criterion Examples

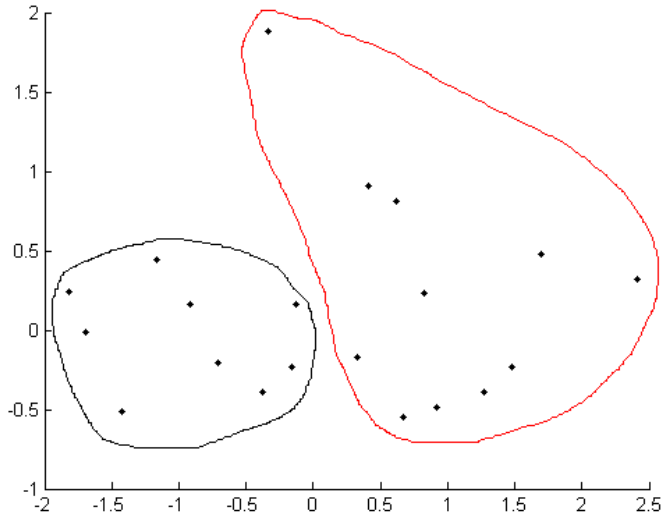


Unknown Number of Clusters

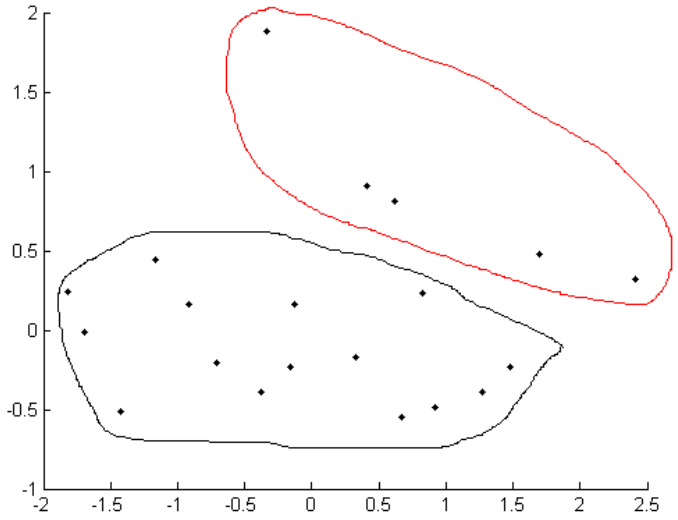
Comparing Criterion Examples: Two Clusters



$J_e, c = 2$

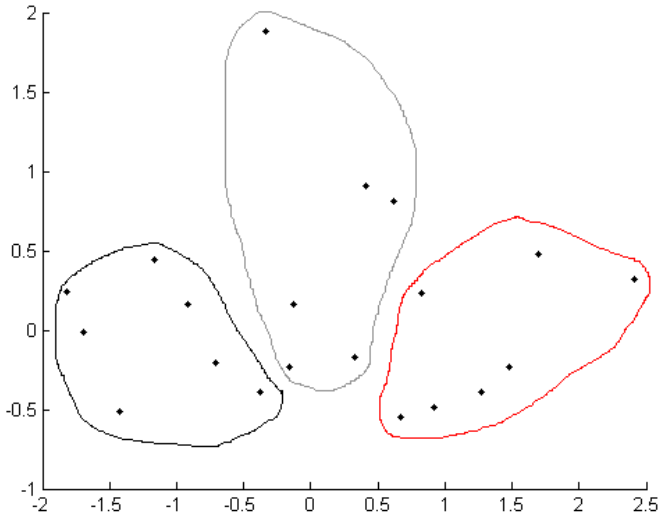


$J_d, c = 2$

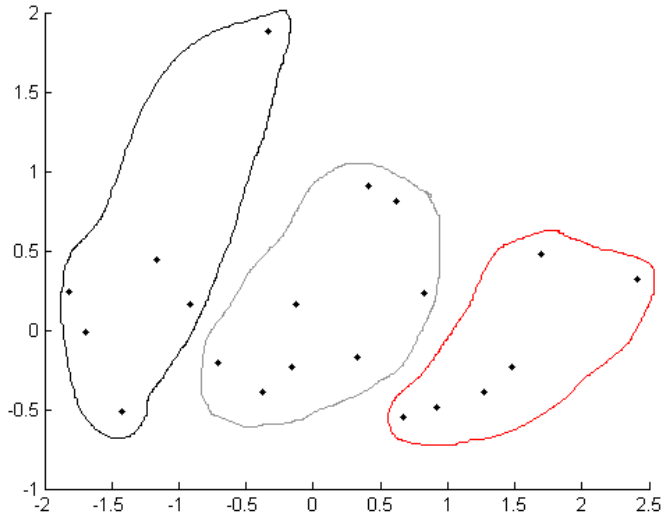


$J_f, c = 2$

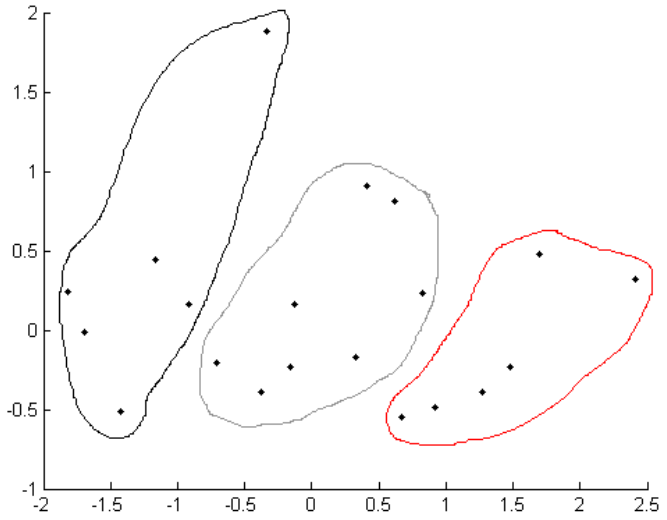
Comparing Criterion Examples: Three Clusters



$J_e, c = 3$



$J_d, c = 3$



$J_f, c = 3$

Hierarchical Clustering

Introduction to Hierarchical Clustering

So far, we've been interested in **disjoint subsets**; that is, $\mathcal{D}_a \cup \mathcal{D}_b = \emptyset$.

However, remember that our classes are things that **we** define, and some objects can be associated with more than one class.

- kingdom = animal
- phylum = Chordata
- ...
- family = Salmonidae
- genus = Oncorhynchus
- species = Oncorhynchus kisutch

Hierarchical Definitions

Data description is **flat** in disjoint partitioning.

In hierarchical clustering, we have a **sequence** of partitions. The approach is:

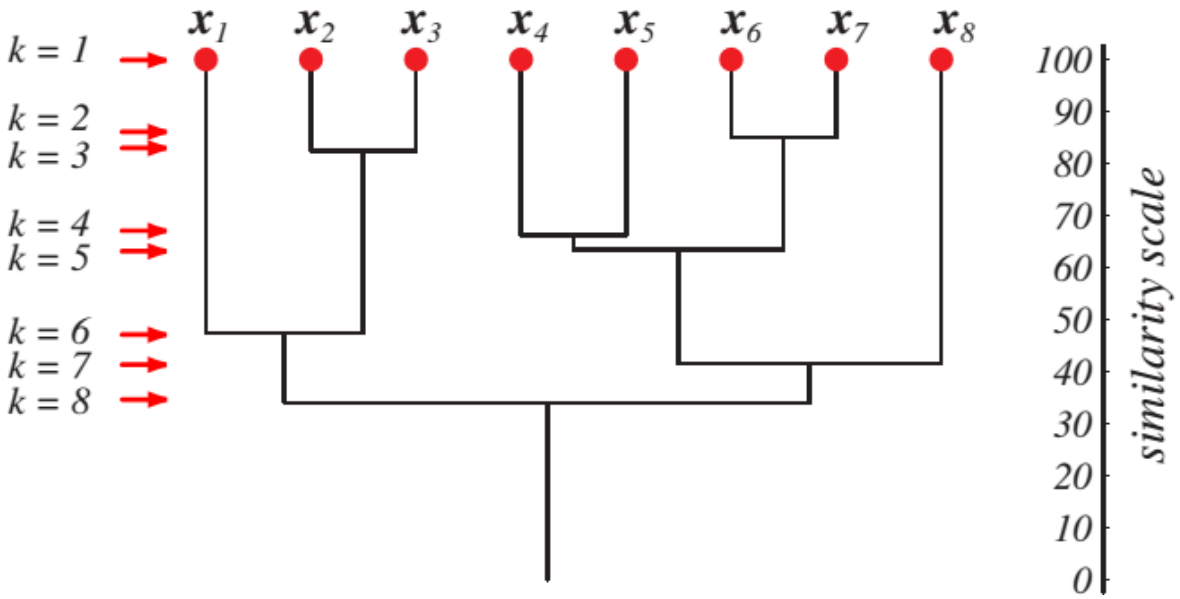
1. Partition the data into n clusters (each data point is its own cluster).
2. Merge clusters one at a time based on some criteria until we only have 1 cluster.

Level k in the sequence is when $c = n - k + 1$.

We select a minimum number of clusters, c, to stop the algorithm. If we set $c = 1$, we get a dendrogram.

In this setup, every two samples $\mathbf{x}$ and $\mathbf{x}'$ will be grouped eventually.

Dendrogram Tree



Dendrogram Tree

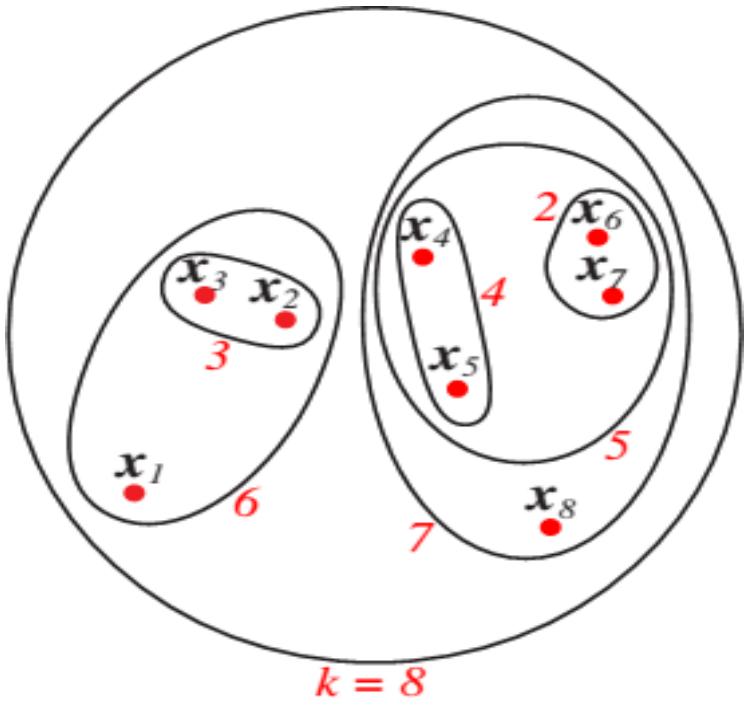
Cluster similarity is visualized as the distance from one level to the next.

Large distances imply here is a natural clustering at that level.

We can visualize this setup as text brackets:

$$\{\{\mathbf{x}_1, \{\mathbf{x}_2, \mathbf{x}_3\}\}, \{\{\{\mathbf{x}_4, \mathbf{x}_5\}, \{\mathbf{x}_6, \mathbf{x}_7\}\}, \mathbf{x}_8\}\}.$$

Alternative Representation: Venn Diagrams



Venn Diagram

Venn diagrams can also be used to see these relationships.

This is less quantitative, but represents how the different samples are grouped in the feature space.

Two Approches to Hierarchical Clustering

There are two related approaches to hierarchical clustering:

- **Agglomerative**: bottom-up, starting with n singleton clusters and grouping up.
- **Divisive**: top-down, start with 1 all-encompassing cluster and then splitting.

We will focus on the former, which typically has a simpler (but oft-repeated) calculation at each hierarchical level.

Agglomerative Clustering

Algorithm 1 Agglomerative Hierarchical Clustering

Input: Target clusters c

Set current clusters to number of points n

repeat

Decrease number of current clusters by 1

Find the two nearest clusters, D_i and D_j

Merge D_i and D_j

until Current cluster number equals c

return c clusters

Agglomerative Procedure

Define measurements of cluster relationships:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} |\mathbf{x} - \mathbf{x}'|$$

$$d_{max}(\mathcal{D}_i, \mathcal{D}_j) = \max_{\substack{\mathbf{x} \in \mathcal{D}_i \\ \mathbf{x}' \in \mathcal{D}_j}} |\mathbf{x} - \mathbf{x}'|$$

$$d_{avg}(\mathcal{D}_i, \mathcal{D}_j) = \frac{1}{n_i n_j} \sum_{\mathbf{x} \in \mathcal{D}_i} \sum_{\mathbf{x}' \in \mathcal{D}_j} |\mathbf{x} - \mathbf{x}'|$$

$$d_{mean}(\mathcal{D}_i, \mathcal{D}_j) = |\mathbf{m}_i - \mathbf{m}_j|$$

Computational Complexity

Suppose we have n samples in d-dimensional space; we want c clusters with d_{min}.

We'll need to calculate $n(n-1)$ distances, each of which is an $O(d)$ calculation.

For the first step, the complexity is $O(n(n-1)(d+1)) = O(n^2 d)$.

At each step we are merging points, so the number of calculations goes down.

For each step, we just need to calculate $n(n-1) - \hat{c}$ "unused" distances in the list.

The full time complexity is thus $O(cn^2 d)$.

Nearest-Neighbor Algorithm

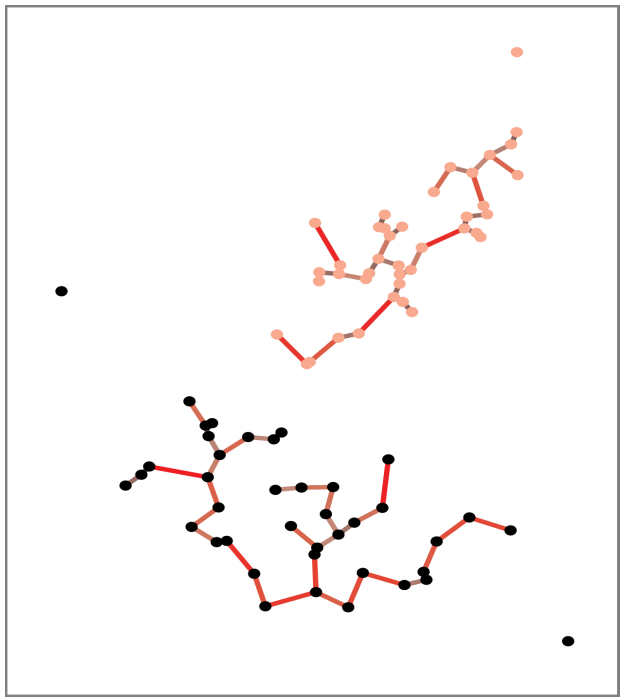
When the distance between clusters is $d_{min}(\cdot, \cdot)$, it's called the **nearest-neighbor** clustering algorithm.

Imagine each $\mathbf{x}$ is a node in a graph, with edges connecting nodes in the same cluster, $\mathcal{D}_i$. The merging of $\mathcal{D}_i$ and $\mathcal{D}_j$ corresponds to adding an edge between the nearest pair of nodes in $\mathcal{D}_i$ and $\mathcal{D}_j$.

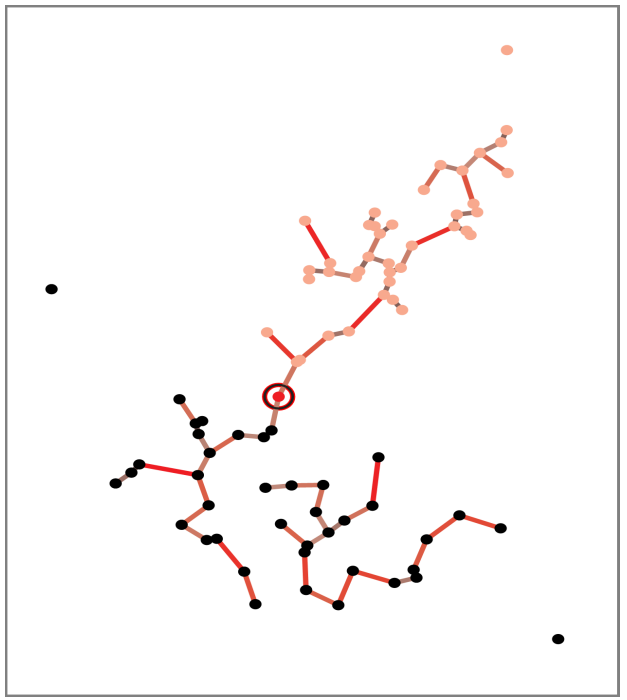
Because of the way we add edges, we won't have any closed loops or circuits. Therefore it generates a tree – this is a **spanning tree** is when all the subsets are linked.

Moreover, this algorithm will create a **minimum spanning tree**: the sum of edge lengths is lower than for all spanning trees.

Example of Chaining Effect



Nearest-Neighbor Clustering



Single Sample Added

Stepwise-Optimal Clustering

Previously we looked at the “nearest” clusters and merged them.

As we saw in the discussion about clustering evaluation, we need a way to define the “nearest” (i.e. most-similar) samples.

To do this, we can replace “nearest” with a criterion function which changes the least with each merger.

This gives us a new algorithm, and a new “distance” whose minimum value indicates the optimal merger:

$$d_e(\mathcal{D}_i, \mathcal{D}_j) = \sqrt{\frac{n_i n_j}{n_i + n_j}} |\mathbf{m}_i - \mathbf{m}_j|$$

Principal Component Analysis

Dimensionality Reduction

Component analysis is used both for dimensionality reduction and clustering.

We will start with linear techniques, and these will lend naturally to nonlinear approaches later.

There are two common linear approaches to component analysis: Principal Component Analysis and Multiple Discriminant Analysis.

PCA seeks a projection to **represent** the data, while MDA seeks a projection to **separate** the data.

Principal Component Analysis: Basic Approach

We have a d-dimensional mean $\boldsymbol{\mu}$ and a $d \times d$ covariance $\boldsymbol{\Sigma}$ for our data set $\mathcal{D}$.

Compute eigenvectors and eigenvalues of $\boldsymbol{\Sigma}$ and sort columns by decreasing eigenvalues.

Choose the k largest eigenvalues, and form a $d \times k$ matrix $\mathbf{A}$ of the k associated eigenvectors.

The data is then projected onto the k-dimensional subspace according to:

$$\mathbf{x}' = \mathbf{F}_1(\mathbf{x}) = \mathbf{A}^T(\mathbf{x} - \boldsymbol{\mu})$$

PCA in Detail

Imagine we want to represent all samples in $\mathcal{D}$ with a single vector $\mathbf{x}_0$.

$\mathbf{x}_0$ should minimize the sum of the squared distances between itself and each sample:

$$J_0(\mathbf{x}_0) = \sum_{k=1}^{n} |\mathbf{x}_0 - \mathbf{x}_k|^2$$

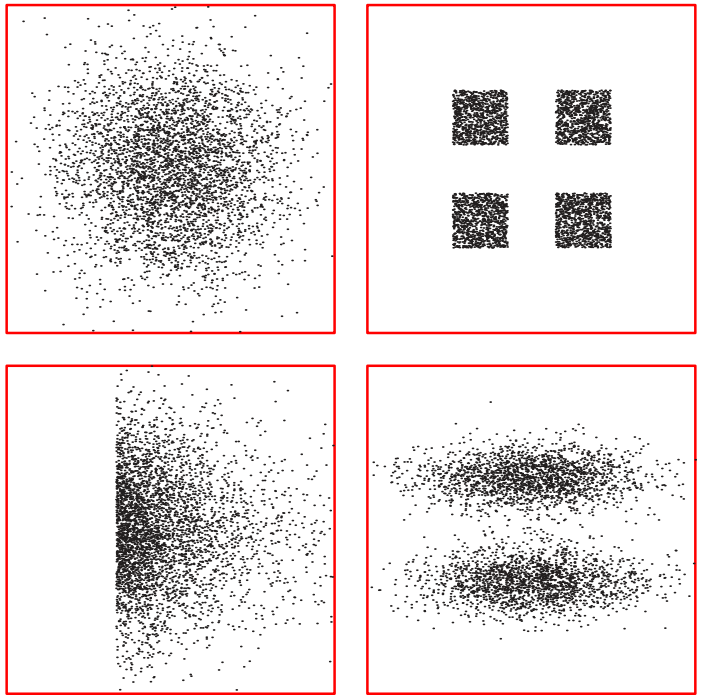
We can assume that this criterion function will be minimized by the sample mean:

$$\mathbf{x}_0 = \mathbf{m} = \frac{1}{n} \sum_{k=1}^{n} \mathbf{x}_k$$

Dimensionality of Our Representation

The sample mean is a “zero-dimensional” representation of the data, and can correspond to many different sample distributions, as we’ve seen.

Similar Means Can Represent Different Distributions



Distributions with equal μ and Σ.

Dimensionality of Our Representation

Let's say we want to "project" the data onto a **one**-dimensional representation (a line) running through the sample mean.

If $\mathbf{e}$ is a unit vector in the direction of the desired line, each sample can be written as:

$$\mathbf{x}_k = \mathbf{m} + a_k \mathbf{e}$$

... where a_k represents the distance of sample point $\mathbf{x}_k$ from the mean $\mathbf{m}$.

Obtaining the Optimal Set of Coefficients

We can get optimal coefficients $a_1, \ldots, a_n$ by minimizing the criterion:

$$
\begin{aligned}
J_1(a_1, \cdots, a_n, \mathbf{e}) &= \sum_{k=1}^{n} \left| (\mathbf{m} + a_k \mathbf{e}) - \mathbf{x}_k \right|^2 \\
&= \sum_{k=1}^{n} \left| a_k \mathbf{e} - (\mathbf{x}_k - \mathbf{m}) \right|^2 \\
&= \sum_{k=1}^{n} a_k^2 |\mathbf{e}|^2 - 2 \sum_{k=1}^{n} a_k \mathbf{e}^T (\mathbf{x}_k - \mathbf{m}) + \sum_{k=1}^{n} |\mathbf{x}_k - \mathbf{m}|^2
\end{aligned}
$$

Get the derivative with respect to a_k, set it to zero, and solve:

$$
a_k = \mathbf{e}^T (\mathbf{x}_k - \mathbf{m})
$$

Return to the Scatter Matrix

We have the distances to the line, but what direction is it in?

It passes through the mean in infinitely many directions.

Recall our scatter matrices that we've been using:

$$\mathbf{S} = \sum_{k=1}^{n} (\mathbf{x}_k - \mathbf{m})(\mathbf{x}_k - \mathbf{m})^T$$

If we plug our equation for a_k into the criterion function J_1, we get:

$$J_1(\mathbf{e}) = -\mathbf{e}^T \mathbf{S} \mathbf{e} + \sum_{k=1}^{n} |\mathbf{x}_k - \mathbf{m}|^2$$

Solving for the Direction of $\mathbf{e}$

$$J_1(\mathbf{e}) = -\mathbf{e}^T\mathbf{S}\mathbf{e} + \sum_{k=1}^{n} |\mathbf{x}_k - \mathbf{m}|^2$$

We can solve for the optimal $\mathbf{e}$ using the method of Lagrange multipliers, subject to $|\mathbf{e}| = 1$:

$$\underbrace{u}_{L(\mathbf{e},\lambda)} = \underbrace{\mathbf{e}^T\mathbf{S}\mathbf{e}}_{f(\mathbf{e})} - \lambda\underbrace{(\mathbf{e}^T\mathbf{e} - 1)}_{g(\mathbf{e})}$$

Differentiate with respect to $\mathbf{e}$, set to zero, and we get $\mathbf{S}\mathbf{e} = \lambda\mathbf{e}$.

Since $\mathbf{e}^T\mathbf{S}\mathbf{e} = \lambda\mathbf{e}^T\mathbf{e} = \lambda$, maximizing $\mathbf{e}^T\mathbf{S}\mathbf{e}$ involves selecting the eigenvector corresponding to the largest eigenvalue of $\mathbf{S}$.

Extending to Multiple Dimensions

One-dimensional representations are kind of boring.

If we want to obtain a d'-dimensional representation, where $d' \leq d$, we can estimate $\mathbf{x}$ as:

$$\mathbf{x} = \mathbf{m} + \sum_{i=1}^{d'} a_i \mathbf{e}_i$$

This makes our criterion function:

$$J_{d'} = \sum_{k=1}^{n} \left| \left(\mathbf{m} + \sum_{i=1}^{d'} a_{ki} \mathbf{e}_i \right) - \mathbf{x}_k \right|^2$$

Why It's Called PCA

That criterion function is minimized when $\mathbf{e}_1, \ldots, \mathbf{e}_{d'}$ are the d' eigenvectors of the scatter matrix with the largest eigenvalues.

These are also orthogonal, since $\mathbf{S}$ is real and symmetric.

In linear algebra, they form a **basis** for representing $\mathbf{x}$.

The coefficients a_i are the components of $\mathbf{x}$ in that basis, and are called the **principal components**.

Parting Words

Drawbacks to PCA

PCA is a fairly simple approach, but it does the job in linear spaces.

PCA seeks an **optimal projection**: that is, it tries to represent the variation in the data.

If our subspace is non-linear, we need additional approaches to properly separate out our data (as we saw in the criterion functions).

If we know that our data is coming from multiple sources, then we may want to seek an **independent component** set that splits apart those signals.

Next Class

We will wrap up our discussion of clustering with **dimensionality reduction**, which is a way to visualize and handle data that exists in too high of a dimension.

We will also cover **nonlinear** methods of reduction, which do not rely on a Euclidean space distribution of the samples.

After that, we will begin a discussion of deep learning and artificial intelligence with our first lecture on neural networks.