Unsupervised Clustering (Pt. 1)

Machine Learning for Biomedical Data

Scott Doyle / scottdoy@buffalo.edu

Recap Last Lectures

Recap: Nature of Training Samples

A training set $\mathcal{D}$ is a **limited random sampling** of the feature space.

Supervised Learning

$\mathcal{D} = \{\mathbf{x}_i, y_i\}$, where $y_i \in \{\omega_1, \omega_2, \ldots, \omega_c\}$ is the i-th class label.

Unsupervised Learning

$\mathcal{D} = \{\mathbf{x}_i\}$, where we just have the values for $\mathbf{x}$, but no class label.

Recap: Supervised Classification

We've seen several approaches to classification using class labels:

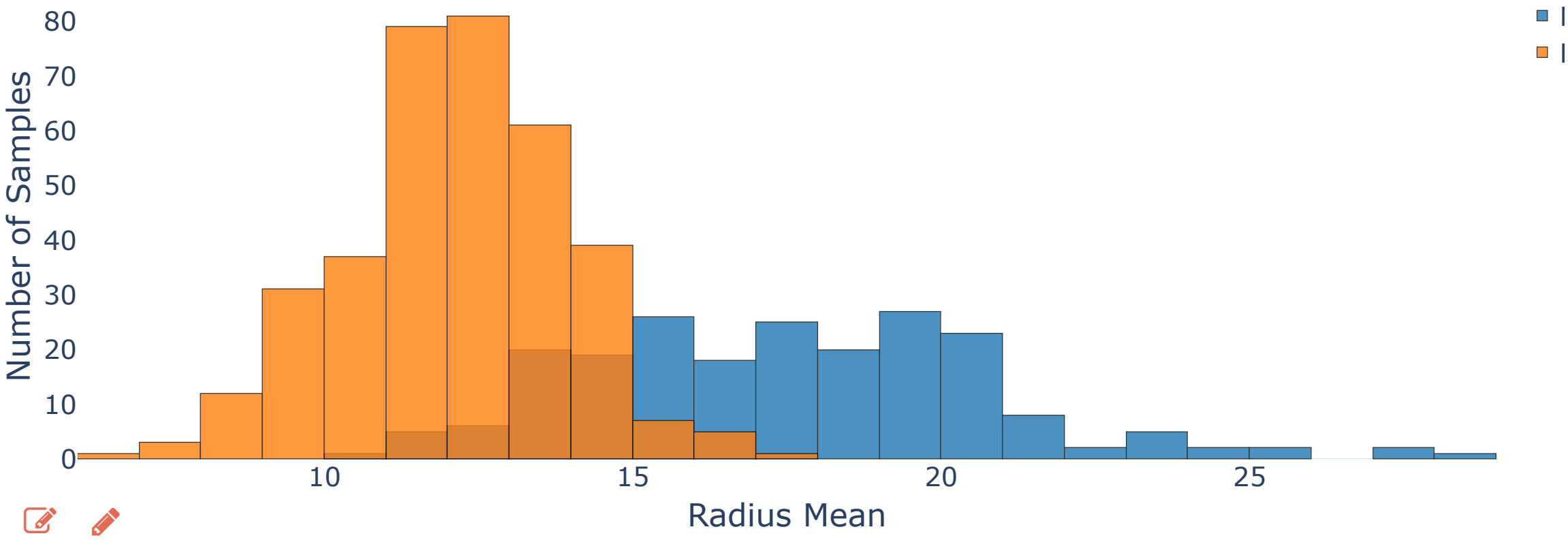
- **Bayes:** Use $\mathcal{D}$ to predict $p(\omega_i|\mathbf{x})$
- **Decision Tree:** Split $\mathcal{D}$ using optimal feature thresholds
- **Linear Discriminants / SVMs:** Build a set of functions $g_i(\mathbf{x})$ such that the predicted label is $\hat{y} = \omega_j$, where $j = \arg\max_i [g_i(\mathbf{x})]$
- **k-Nearest Neighbor:** Predicted label $\hat{y}$ is the most-common label of the k-nearest neighbors of $\mathbf{x}$ according to the feature space

Recap: Parametric Methods

$\mathcal{D}$ may come from an underlying **parameterized** distribution:

- **Normal:** $\theta = (\mu, \Sigma)$ (mean, covariance / variance)
- **Log-normal:** $\theta = (\sigma, \mu)$ (shape, log-scale)
- **Binomial:** $\theta = (n, p)$ (trials, success probability)
- **Gamma:** $\theta = (k, \theta)$ or (α, β) or (k, μ) (shape, scale/rate/mean)
- **Weibull:** $\theta = (k, \lambda, \theta)$ (shape, scale, location)
- **Poisson:** $\theta = (\lambda)$ (mean / variance)

Recap: Parametric Methods



Recap: Choosing your Distribution

Each of these has its own form!

Choose the one that:

- Describes your data
- Has the fewest parameters
- Makes intuitive sense, given the source of the feature
- (Is the Normal Distribution)

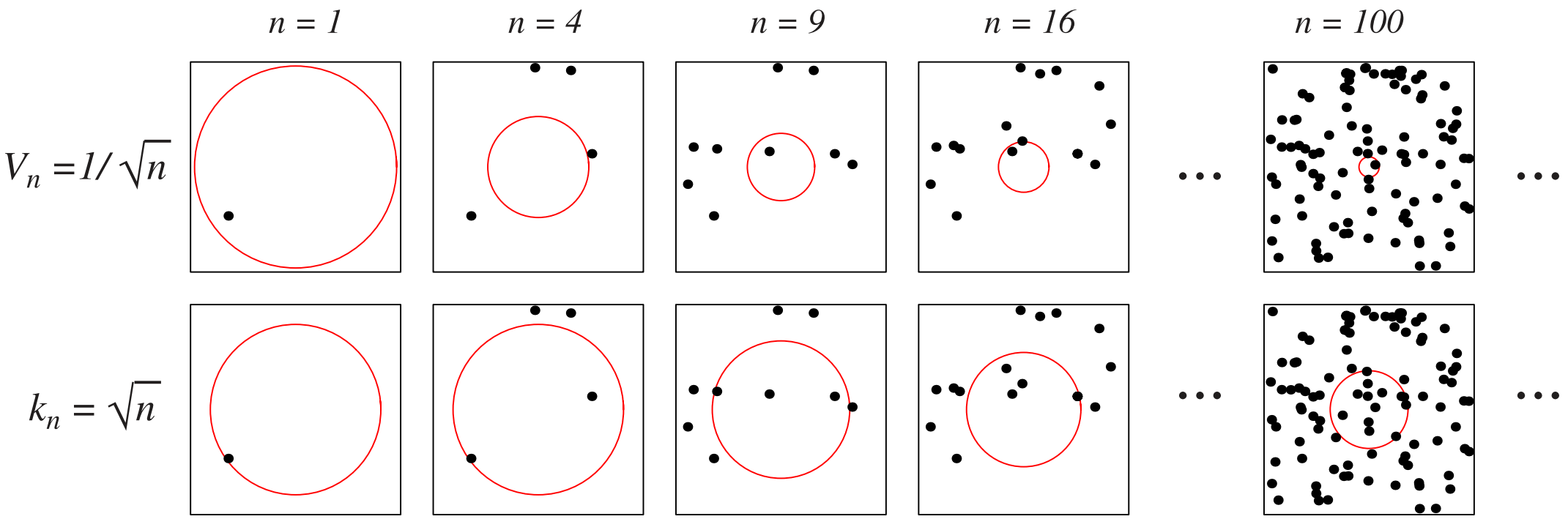
Recap: Nonparametric methods

What if:

- … we don’t know what our parametric form should be?
- … our samples don’t come from a single distribution?
- … we have way too few samples to even estimate our parameters?

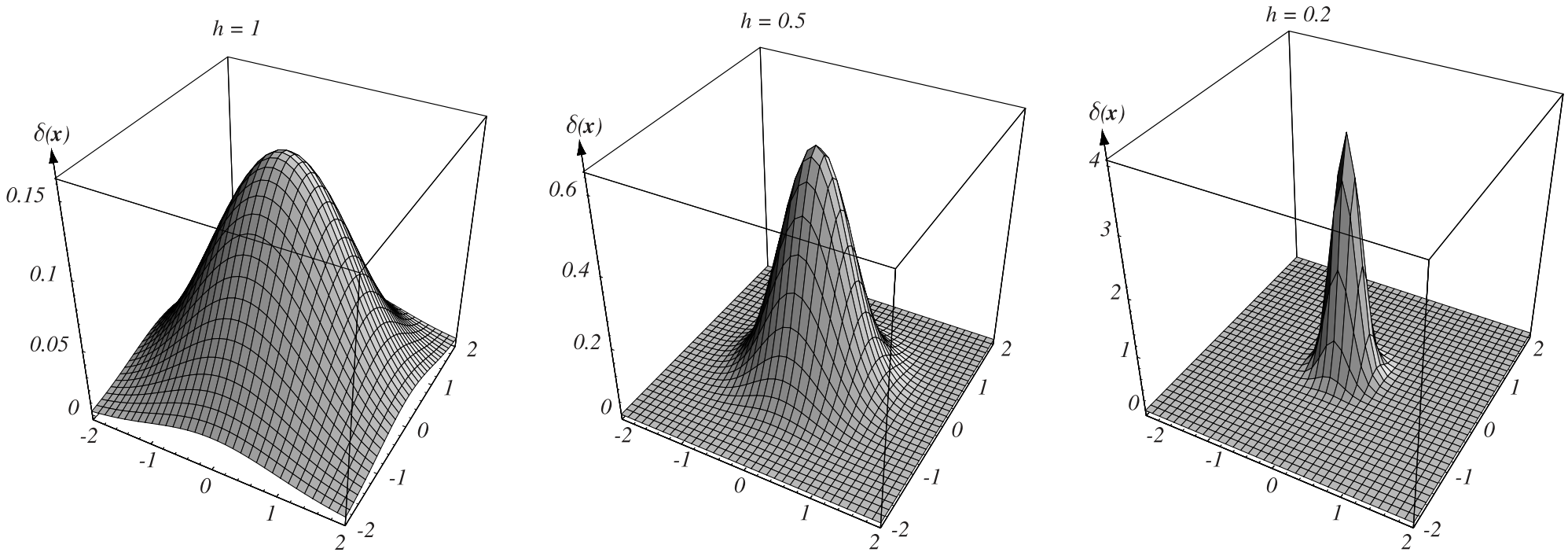
In these cases, we need non-parametric methods.

Recap: Two Methods for Finding Densities



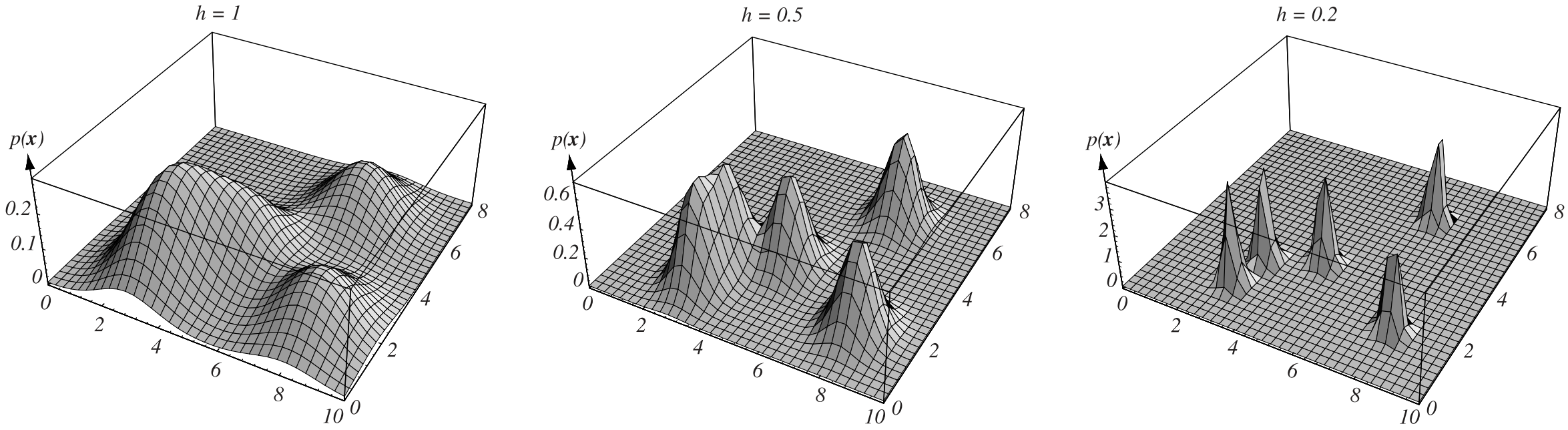
Parzen Windows (top) and k-Nearest Neighbors (bottom) for estimating density at a point as $N \to \infty$.

Recap: Parzan Windows Width



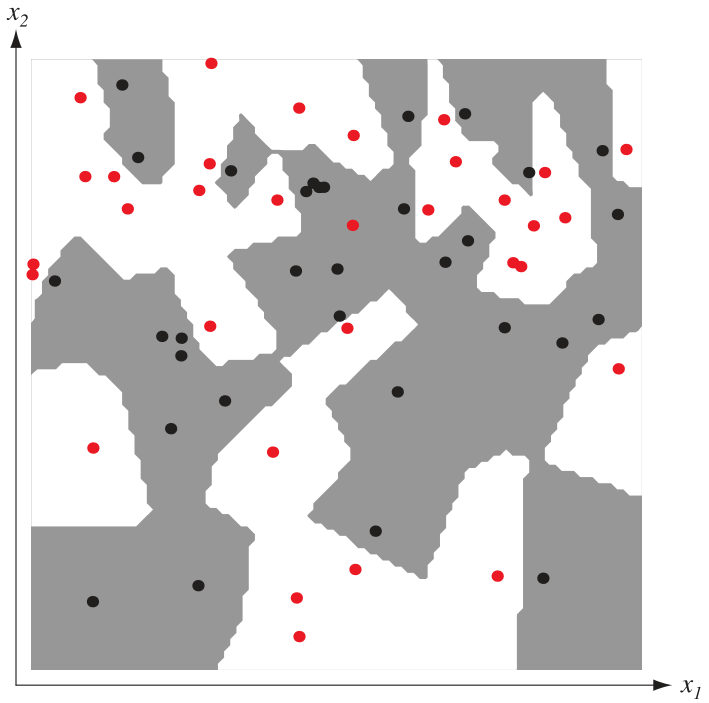
Effect of h (width) on the "window" $\delta_n(\mathbf{x})$.

Recap: Parzen Windows Width

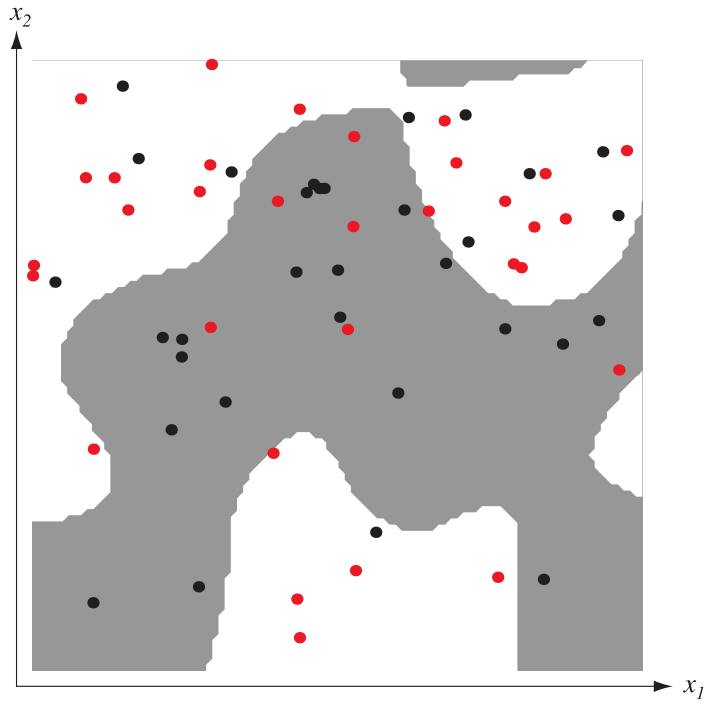


Effect of h (width) on the density estimates $p_n(\mathbf{x})$.

Recap: Classification With Parzen Windows



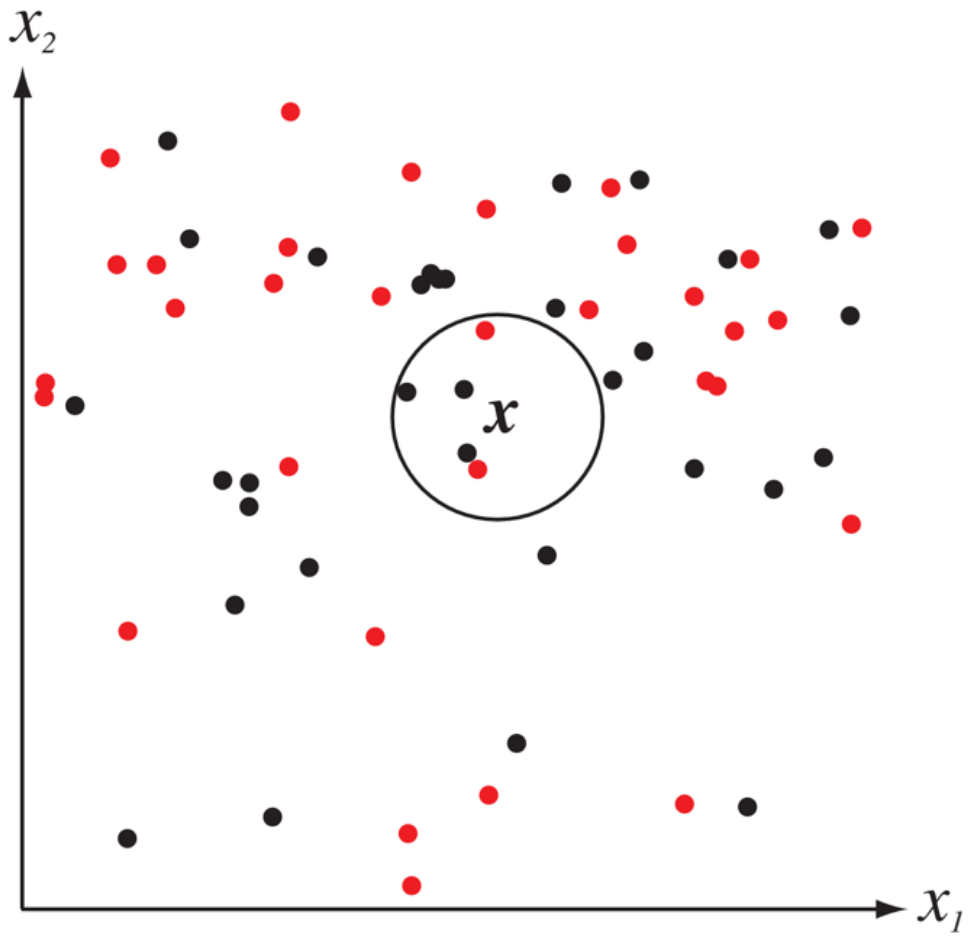
Small h



Large h

Classification using Parzen Windows. If h is small, we risk over training. If h is large

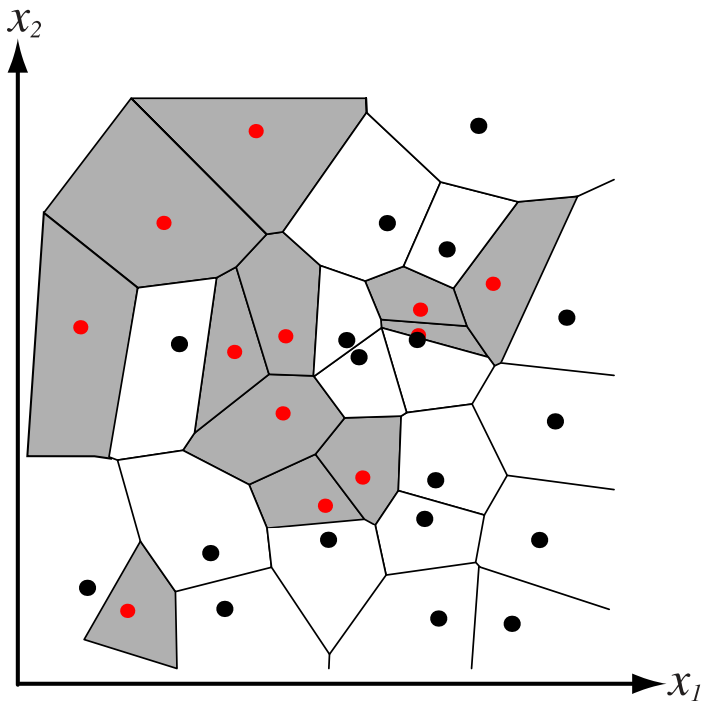
Recap: Classification with kNN

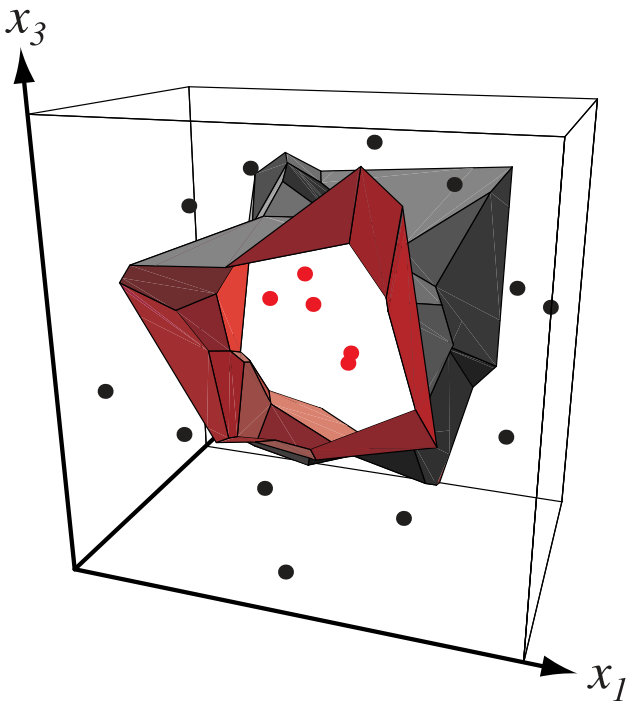


Classification with kNN is so simple, it essentially “skips” the consideration of

densities and just labels samples by counting the neighbors that belong to each class.

Recap: Classification with kNN





Clustering: Mixture Densities

Clustering Introduction

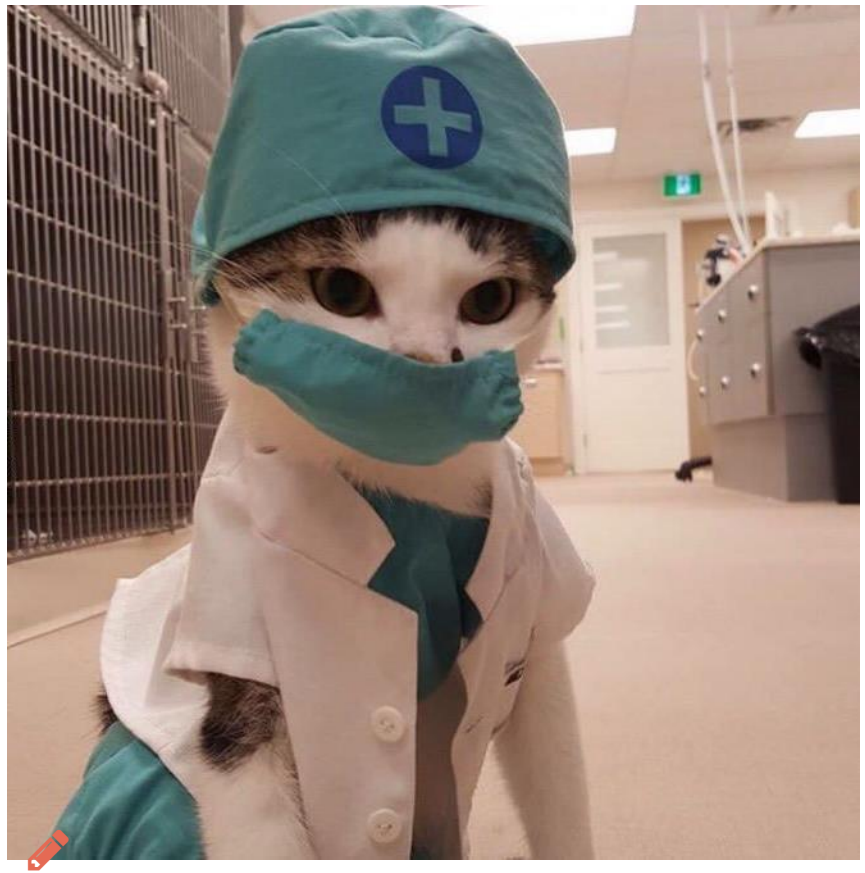
In almost all situations, gathering class labels is **hard**.

We are often presented with datasets having many unlabeled samples:

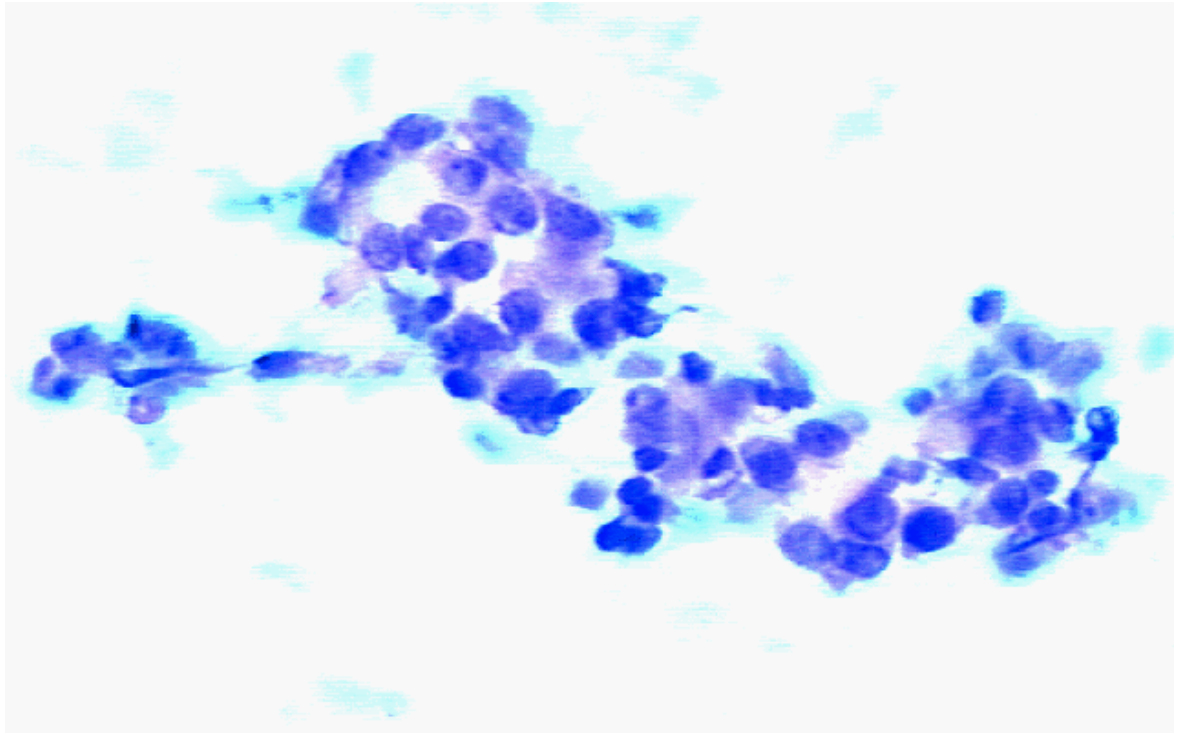
- Uploaded photos (Facebook, Google+, the NSA)
- Recorded telephone audio (Call centers, speech therapy, the NSA again)
- Recorded video segments (Security firms, video game cameras, also the NSA)

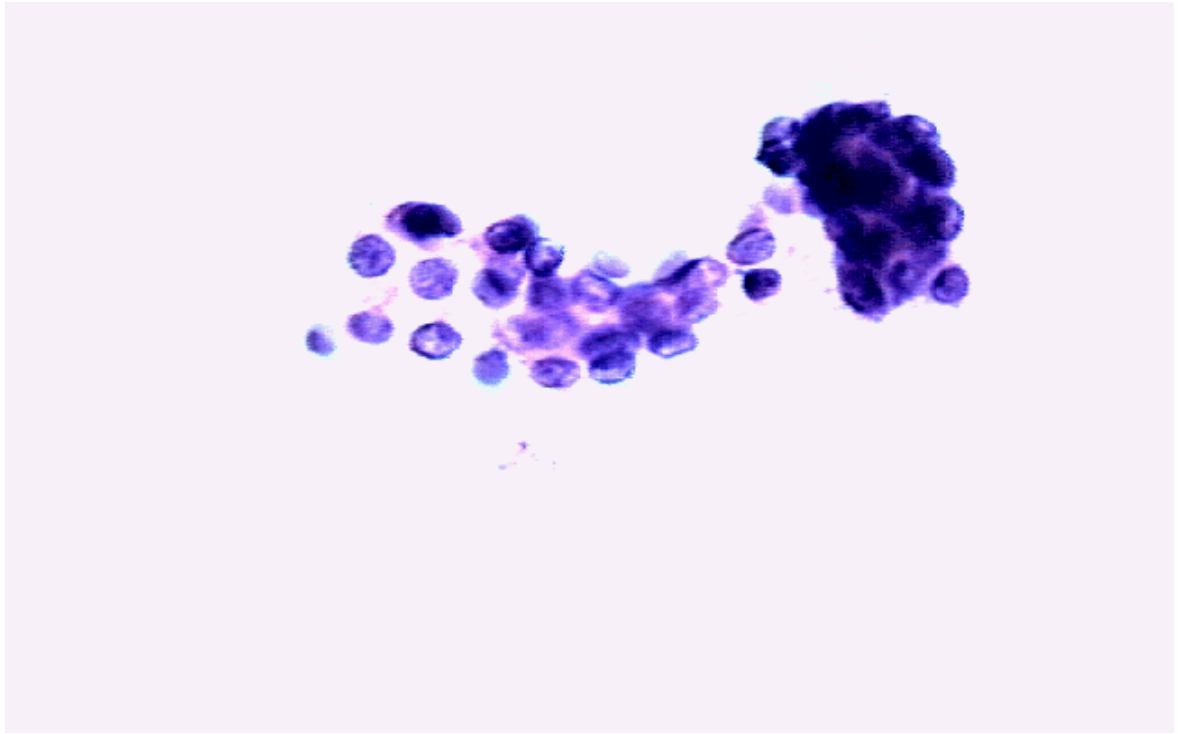
For some kinds of problems, you can **crowd-source** labeling.

Cost of Acquiring Labels



Cost of Acquiring Labels





Why Use Unlabeled Data?

Without labels, can we learn about categories?

- You can cluster, and then perform “hands-on” labeling.
- You can get a “free” look at the data before doing labeling, to get an “unbiased” look at the structure of the feature space.
- You can perform “weak” supervised classification, a.k.a. **semi-supervised** learning

Semi-Supervised Learning

From Zhou, Z. “A Brief Introduction to Weakly Supervised Learning”:

- **Incomplete Supervision:** Only a small subset of the available data is labeled
- **Inexact Supervision:** Only coarse-level annotations are provided
- **Inaccurate Supervision:** Labels may be incorrectly assigned

Known Knowns, Known Unknowns

As ever, we start with some basic assumptions:

- We know the number of classes, c.
- We know the priors, $P(\omega_j)$, for $j = 1,\ c$.
- We know the form of $p(\mathbf{x}|\omega_j, \boldsymbol{\theta}_j)$ (e.g. Gaussian).
- We do NOT know the parameter vectors $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_c$.
- We do NOT know the actual category labels for the samples.

Mixture Densities

We can start by modeling the observed density, $p(\mathbf{x}|\boldsymbol{\theta})$, as a mixture of c different class densities:

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

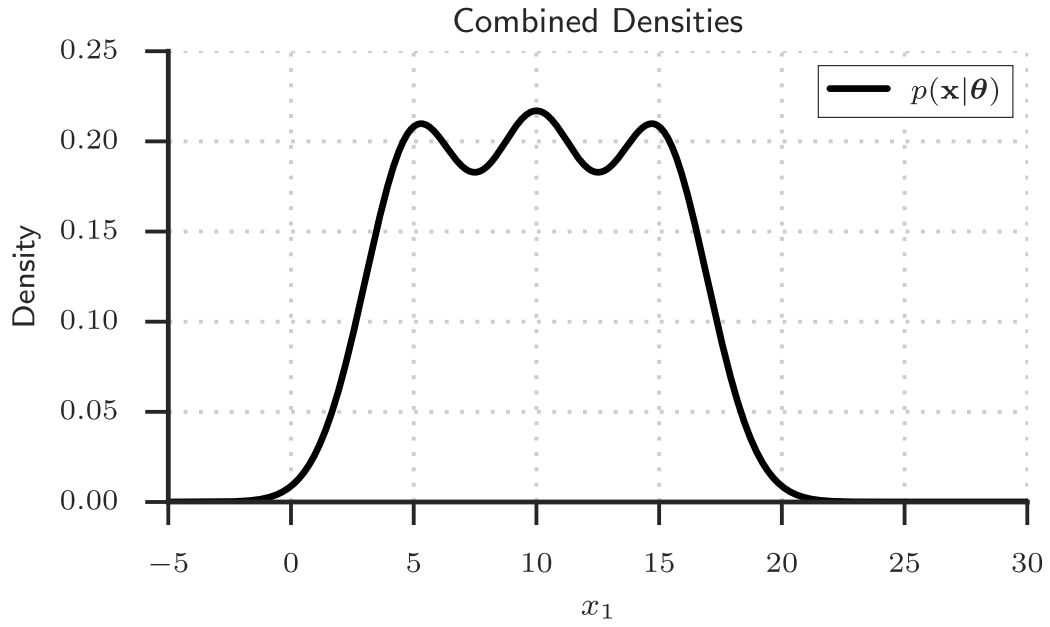
... where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_c)^T$.

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**.

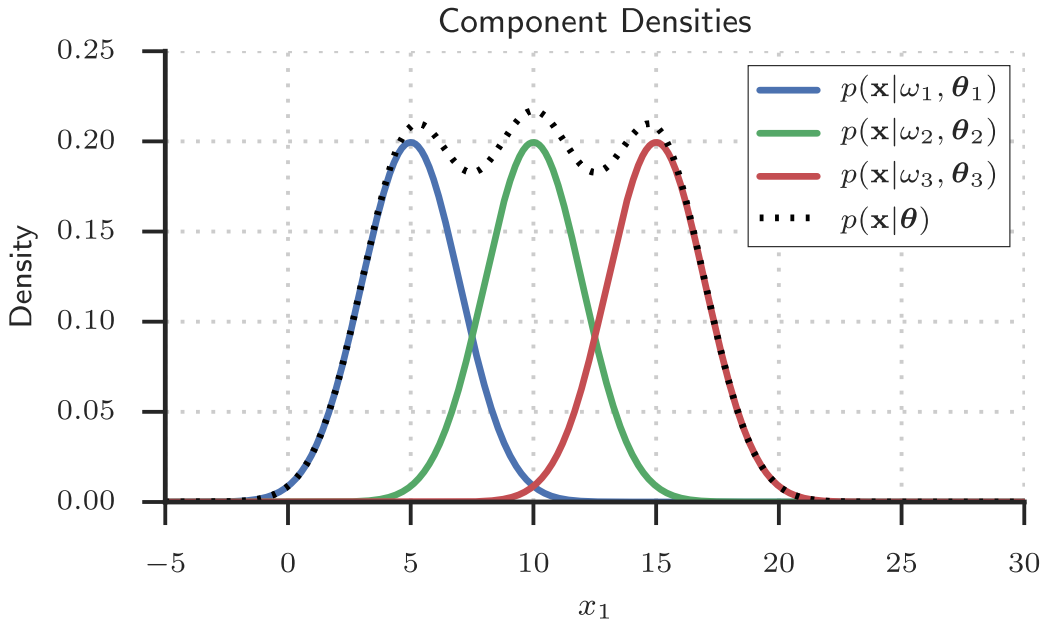
Priors $P(\omega_j)$ are the **mixing parameters**.

Component Densities and Mixing Parameters



Observed Sample Distribution

Component Densities and Mixing Parameters



Underlying Component Distributions

Identifiability of a Density

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

Our unknown is our set of parameters $\boldsymbol{\theta}$, so that's what we want to estimate.

Once we have our parameter sets, we can "un-mix" the density, figure out what classes are the largest contributors at each point of $\mathbf{x}$, and then classify new points accordingly.

One valid question: if we have an infinite number of samples, and we know the underlying form of $p(\mathbf{x}|\boldsymbol{\theta})$, then is $\boldsymbol{\theta}$ unique?

A density $p(\mathbf{x}|\boldsymbol{\theta})$ is **identifiable** if $\boldsymbol{\theta} \neq \boldsymbol{\theta}'$ implies that there exists an $\mathbf{x}$ such that $p(\mathbf{x}|\boldsymbol{\theta}) \neq p(\mathbf{x}|\boldsymbol{\theta}')$.

Maximum Likelihood Estimation

Maximum Likelihood Estimates

Suppose $p(\mathbf{x}|\boldsymbol{\theta})$ gives us a set $\mathcal{D} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$

The likelihood of observing a specific $\mathcal{D}$ is the joint density:

$$p(\mathcal{D}|\boldsymbol{\theta}) = \prod_{k=1}^{n} p(\mathbf{x}_k|\boldsymbol{\theta})$$

Maximizing our Dataset Probability

If we're looking to find $\hat{\boldsymbol{\theta}}$ which maximizes $p(\mathcal{D}|\boldsymbol{\theta})$, we have to do the whole log-likelihood / gradient thing:

$$l = \sum_{k=1}^{n} \ln p(\mathbf{x}_k|\boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} \frac{1}{p(\mathbf{x}_k|\boldsymbol{\theta})} \nabla_{\boldsymbol{\theta}_i} \left[\sum_{j=1}^{c} p(\mathbf{x}_k|\omega_j, \boldsymbol{\theta}_j) P(\omega_j) \right]$$

Maximum Likelihood Estimates

If we introduce the posterior probability, we can write in terms of the component densities:

$$P(\omega_i|\mathbf{x}_k,\boldsymbol{\theta}) = \frac{p(\mathbf{x}_k|\omega_i,\boldsymbol{\theta}_i)P(\omega_i)}{p(\mathbf{x}_k|\boldsymbol{\theta})}$$

Then we can rewrite the previous derivative as:

$$\nabla_{\boldsymbol{\theta}_i} l = \sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k,\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i,\boldsymbol{\theta}_i)$$

As always, we set this to zero and solve for the class-specific parameters $\boldsymbol{\theta}_i$.

Normal Mixtures: Estimating μ

Normal Mixtures and Additional Assumptions

We've already assumed we know the form of the density (namely, that it's 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.

Just like linear discriminants, for simplicity we start by assuming that we know three of these: $\boldsymbol{\Sigma}_i$, $P(\omega_i)$, and c.

Case 1: Unknown Mean Vectors

Once again, we take the log-likelihood of the Gaussian for simplicity:

$$\ln p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = -\ln\left[(2)^{\frac{d}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}\right] - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$$

... take the derivative...

$$\nabla_{\boldsymbol{\mu}_i} \ln p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$$

... and drop it into the old MLE equation for finding $\hat{\boldsymbol{\theta}}$, which, to refresh your memory, is:

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \boldsymbol{\theta})\nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \boldsymbol{\theta}_i) = 0$$

Case 1: Unknown Mean Vectors

Thus we have:

$$\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}})\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i) = 0$$

If we multiply by $\boldsymbol{\Sigma}_i$ and moving around terms, we are left with:

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\mu}})\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\mu}})}$$

In other words, $\hat{\boldsymbol{\mu}}_i$ is a weighted average of the samples.

The weight for the kth sample is an estimate of how likely it is that $\mathbf{x}_k$ belongs to the ith class.

Does this equation give us $\hat{\boldsymbol{\mu}}_i$?

Explicit Solution or Iterative Estimates?

Unfortunately, we can't solve $\hat{\boldsymbol{\mu}}_i$ explicitly.

What we CAN do is perform an iterative search: Select an initial value for $\hat{\boldsymbol{\mu}}_i(0)$, then solve:

$$\hat{\boldsymbol{\mu}}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))}$$

This is essentially a **gradient ascent** method, meaning that eventually we will converge to a point where $\hat{\boldsymbol{\mu}}_i(j+1) \approx \hat{\boldsymbol{\mu}}_i(j)$

As with all iterative approaches, once the gradient is zero, we are only ensured that we've reached a **local** maximum.

Example Solution for Finding μ_i

Let's say we have the following two-component, one-dimensional normal mixture:

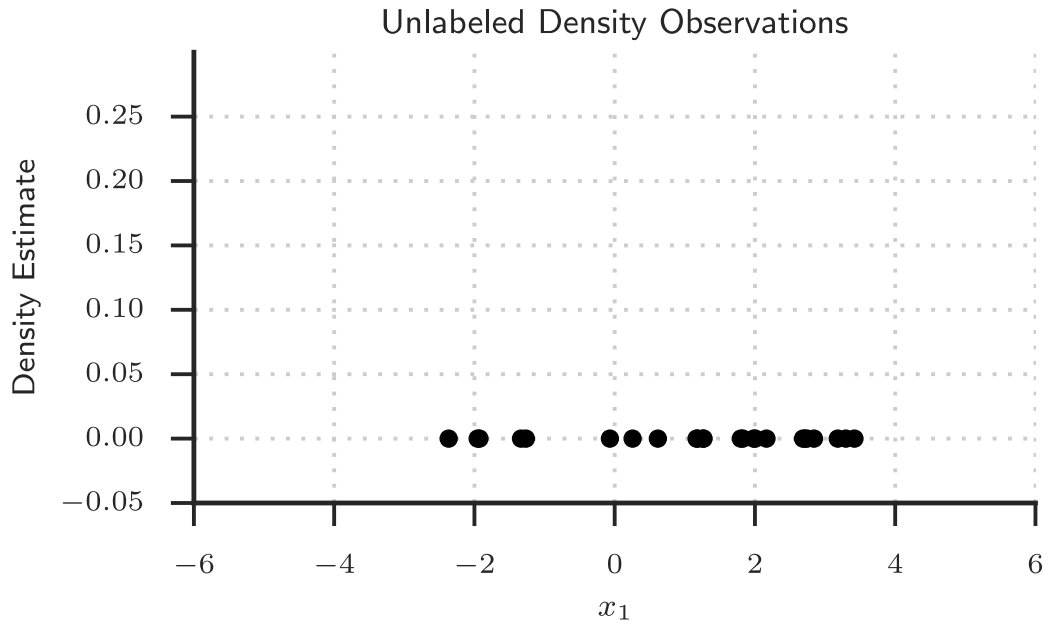
$$p(x|\mu_1, \mu_2) = \underbrace{\frac{1}{3\sqrt{2}} \exp\left[-\frac{1}{2}(x-\mu_1)^2\right]}_{\omega_1} + \underbrace{\frac{2}{3\sqrt{2}} \exp\left[-\frac{1}{2}(x-\mu_2)^2\right]}_{\omega_2}$$

We pull a set of $k = 25$ samples sequentially from this mixture.

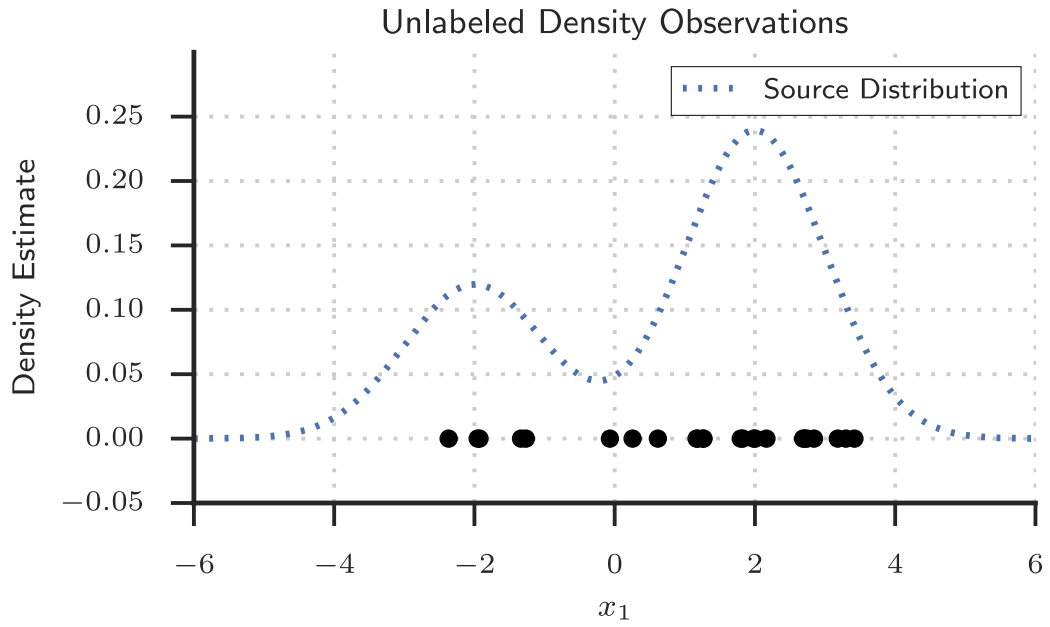
These samples are used to calculate the log-likelihood function:

$$l(\mu_1, \mu_2) = \sum_{k=1}^{n} \ln p(x_k|\mu_1, \mu_2)$$

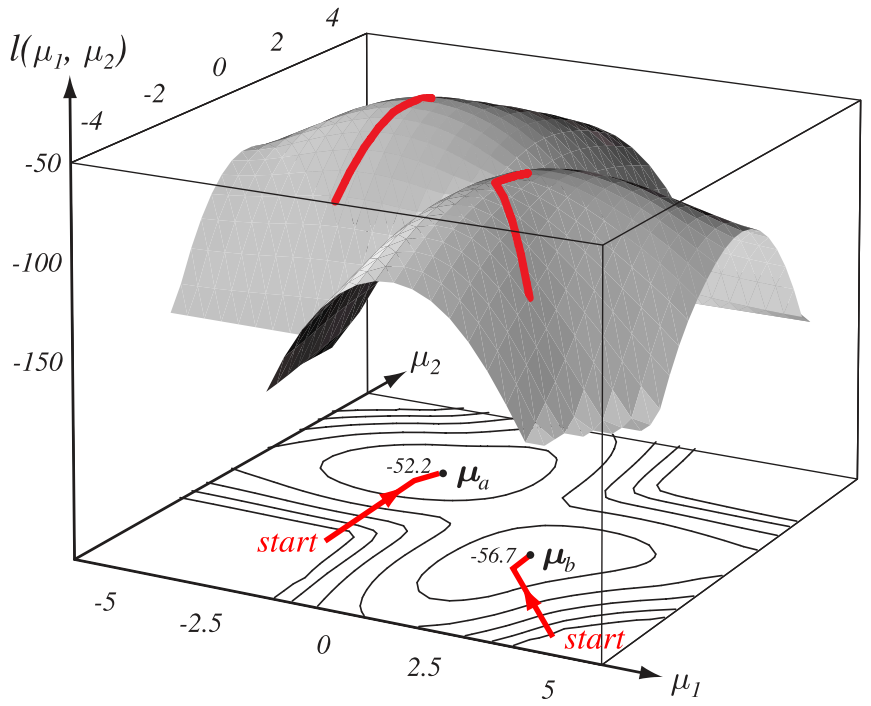
Example Solution for Finding μ_i: Illustration



Example Solution for Finding μ_i: Illustration



Example Solution for Finding μ_i: Illustration



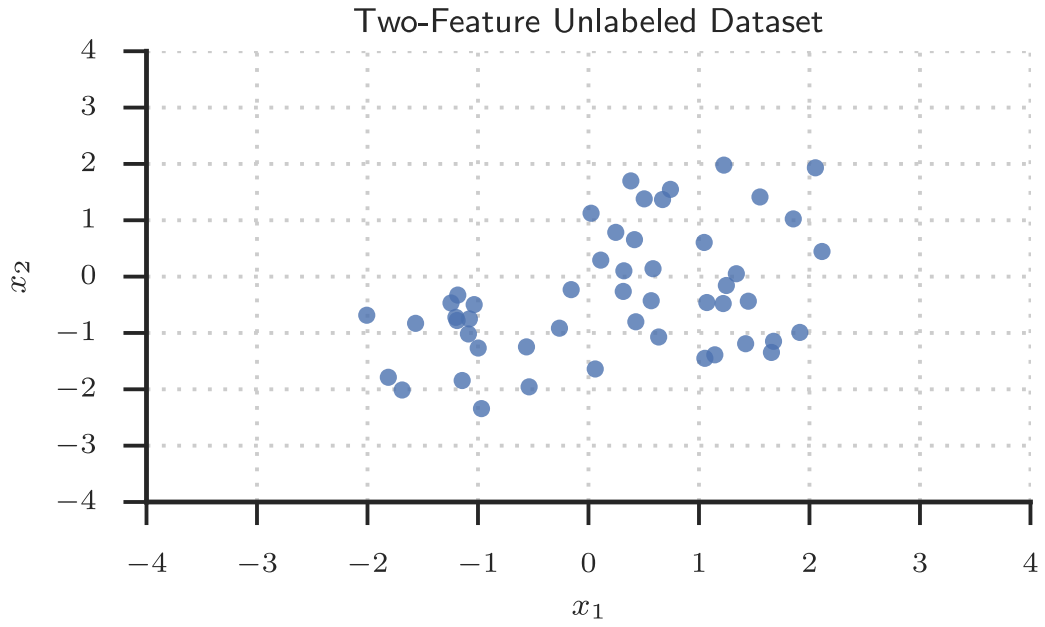
2D Mixture Density

Unlabeled Densities: Importance of Sample Size

As ever, it’s important to have a significant number of samples to estimate your parameters.

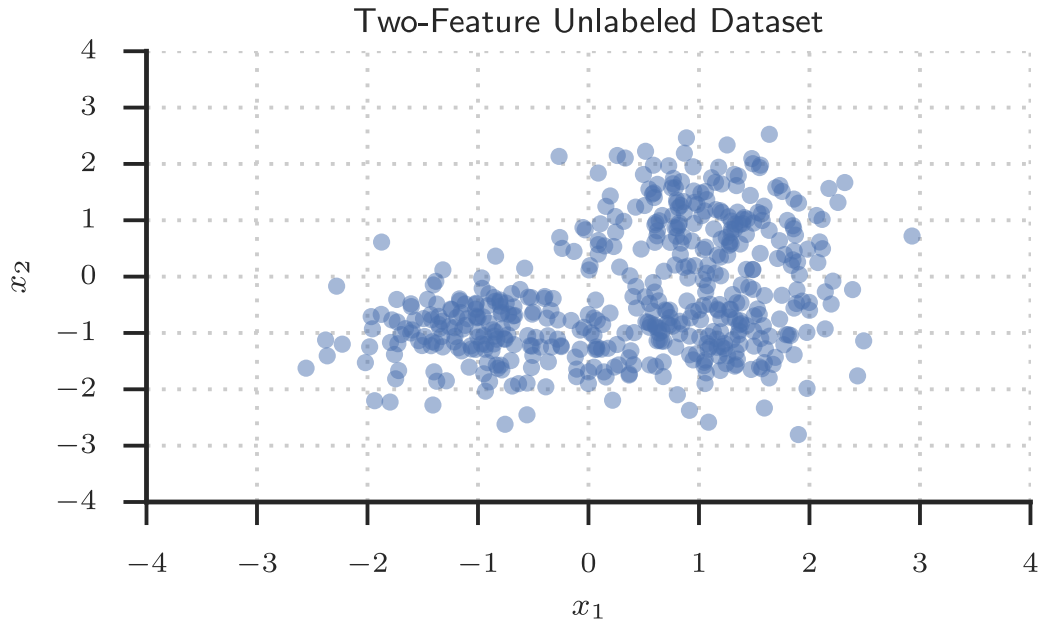
What would it look like if we had insufficient samples for this problem?

Unlabeled Densities: Few Samples



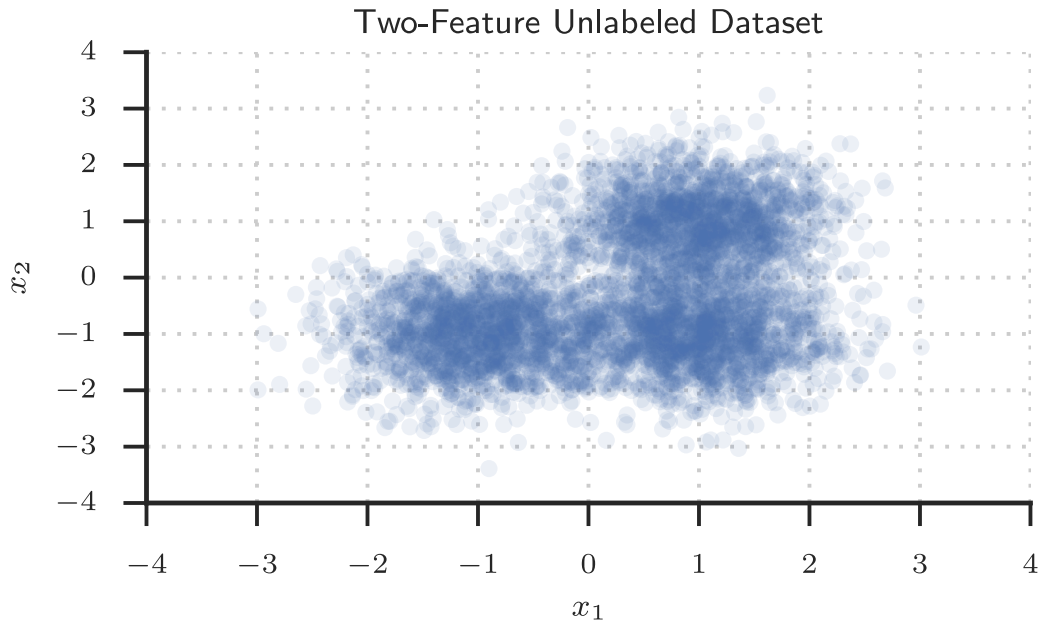
Low Number of Observations

Unlabeled Densities: More Samples



Larger Number of Observations

Unlabeled Densities: Lots of Samples



Lots of Observations

Normal Mixtures: Estimating Σ

Case 2: Unconstrained Covariance

If no constraints are placed on $\boldsymbol{\Sigma}_i$, then we have a problem...

Let's say our two-component mixture is given like so:

$$p(x|\mu,\sigma^2) = \underbrace{\frac{1}{2\sqrt{2}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]}_{\text{First Component}} + \underbrace{\frac{1}{2\sqrt{2}}\exp\left[-\frac{1}{2}x^2\right]}_{\text{Second Component}}$$

In other words, the second component has $\mu = 0, \sigma = 1$.

Now let's set the mean of the first component to $\mu = x_1$. For $p(x_1|\mu,\sigma^2)$, the first term's exponential goes to 0, and so we have:

$$p(x_1|\mu,\sigma^2) = \frac{1}{2\sqrt{2}\sigma} + \frac{1}{2\sqrt{2}}\exp\left[-\frac{1}{2}x_1^2\right]$$

Case 2: Unconstrained Covariance

For $x_k \neq \mu$, the first term's exponential remains. Thus:

$$p(x_k|\mu,\sigma^2) \geq \frac{1}{2\sqrt{2}}\exp\left[-\frac{1}{2}x_k^2\right]$$

The joint probability for $x_1,\cdots,x_n$ is the product of all of the above, so with some rearranging we have:

$$p(x_1,\cdots,x_n|\mu,\sigma^2) \geq \left[\frac{1}{\sigma}+\exp\left[-\frac{1}{2}x_1^2\right]\right]\frac{1}{(2\sqrt{2})^n}\exp\left[-\frac{1}{2}\sum_{k=2}^{n}x_k^2\right]$$

All Parameters Unknown: Exploding Equations with Small Variance

$$p(x_1, \cdots, x_n | \mu, \sigma^2) \geq \left[\frac{1}{\sigma} + \exp\left[-\frac{1}{2}x_1^2\right]\right] \frac{1}{(2\sqrt{2})^n} \exp\left[-\frac{1}{2}\sum_{k=2}^{n} x_k^2\right]$$

What's wrong with this equation? Specifically, look at σ...

If σ is small, the equation explodes. That is to say, the MLE solution is **singular**.

Fixing Exploding Equations

So how do we get around this? Put some constraints on $\boldsymbol{\Sigma}_i$!

Recall the likelihood equation when finding $\boldsymbol{\mu}_i$:

$$\ln p(\mathbf{x}|\omega_i, \boldsymbol{\mu}_i) = -\ln\left[(2)^{\frac{d}{2}}|\boldsymbol{\Sigma}_i|^{\frac{1}{2}}\right] - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)$$

Just like before, we want to differentiate and set to zero, but this time we have to differentiate with respect to both $\boldsymbol{\mu}_i$ **and** $\boldsymbol{\Sigma}_i$.

Solving for Everything

Remember the relation we had previously:

$$\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \nabla_{\boldsymbol{\theta}_i} \ln p(\mathbf{x}_k|\omega_i, \hat{\boldsymbol{\theta}}_i) = 0$$

I am skipping a lot of the math, but we end up with the following:

$$\hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$$

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \mathbf{x}_k}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})}$$

$$\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_i)^t}{\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})}$$

Explanations for Everything

Okay, so what do those mean?

$\hat{P}(\omega_i) = \frac{1}{n}\sum_{k=1}^{n}\hat{P}(\omega_i\|\mathbf{x}_k,\hat{\boldsymbol{\theta}})$	The likelihood of ω_i is the fraction of samples from ω_i
$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{k=1}^{n}\hat{P}(\omega_i\|\mathbf{x}_k,\hat{\boldsymbol{\theta}})\mathbf{x}_k}{\sum_{k=1}^{n}\hat{P}(\omega_i\|\mathbf{x}_k,\hat{\boldsymbol{\theta}})}$	$\hat{\boldsymbol{\mu}}_i$ is the mean of those samples in ω_i
$\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{k=1}^{n}\hat{P}(\omega_i\|\mathbf{x}_k,\hat{\boldsymbol{\theta}})(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_i)(\mathbf{x}_k-\hat{\boldsymbol{\mu}}_i)^T}{\sum_{k=1}^{n}\hat{P}(\omega_i\|\mathbf{x}_k,\hat{\boldsymbol{\theta}})}$	$\hat{\boldsymbol{\Sigma}}_i$ is the corresponding covariance matrix

If $\hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}})$ is between 0.0 and 1.0, then all samples play a role in the estimates (not just those belonging to ω_i).

One More Equation

One final point: Above we've written $\sum_{k=1}^{n} \hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}})$ a lot.

For each of the above three equations, we can substitute Bayes equation here:

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

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

Explicit Description of Our Parameter Set

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

Much simpler, right?

This is just to illustrate that now, we DO have an explicit way to calculate each of our unknown parameters given our sample set.

Okay, So Now What?

Once again, we can take an iterative approach to solving for our parameter sets:

$$\hat{\boldsymbol{\mu}}_i(j+1) = \frac{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))\mathbf{x}_k}{\sum_{k=1}^{n} P(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\mu}}(j))}$$

k-Means Clustering

Mahalanobis Distance

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

The value of $\hat{P}(\omega_i|\mathbf{x}_k,\hat{\boldsymbol{\theta}})$ is high 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.

Intuitively, this just means that if a sample is close to the centroid of a class cluster, then the likelihood of it belonging to the associated class is high.

Calculating Probabilities Via Distance to Centroids

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:

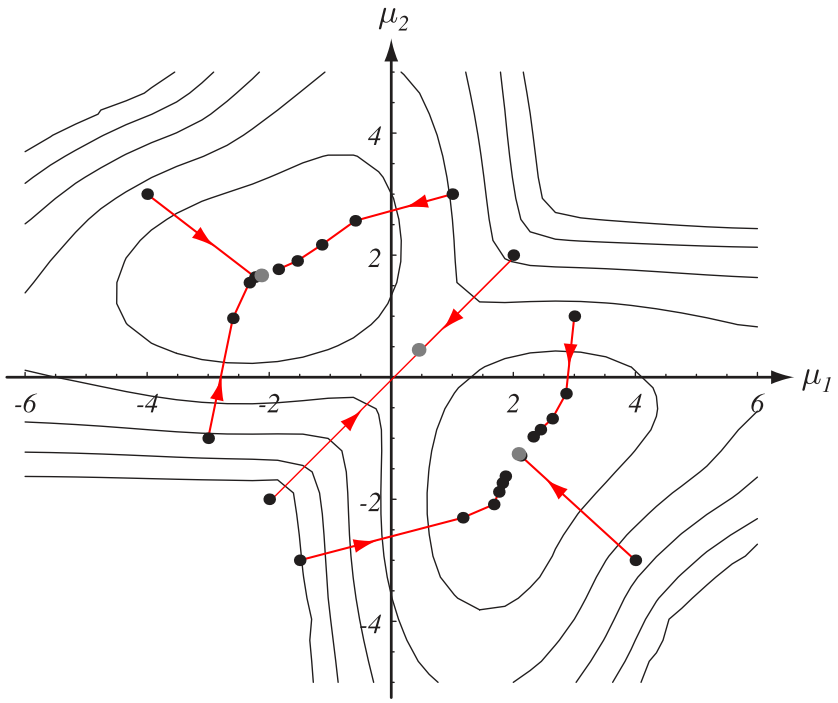
$$\hat{P}(\omega_i|\mathbf{x}_k, \hat{\boldsymbol{\theta}}) \simeq \begin{cases} 1 & \text{if } i = m \\ 0 & \text{otherwise.} \end{cases}$$

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.

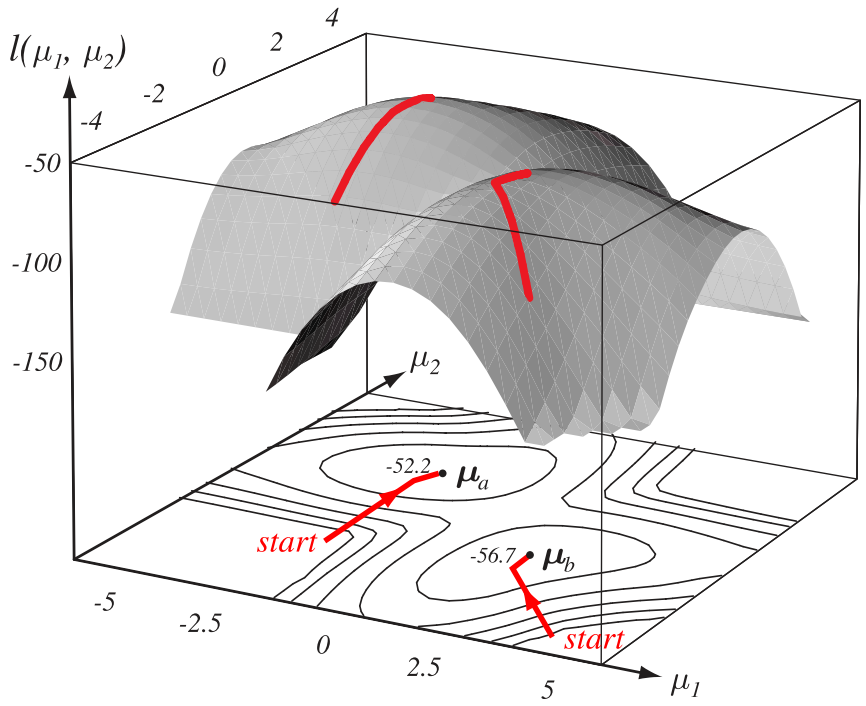
Algorithm for k-Means Clustering

k-Means Example: Hill-climbing

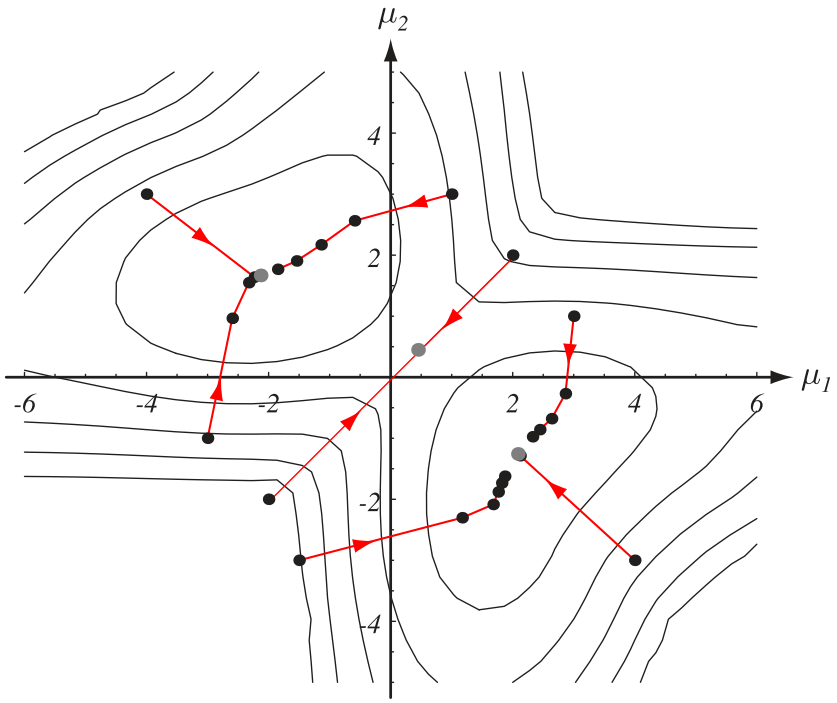


Stoachastic hill-climbing in k-means.

Comparing k-Means and MLE



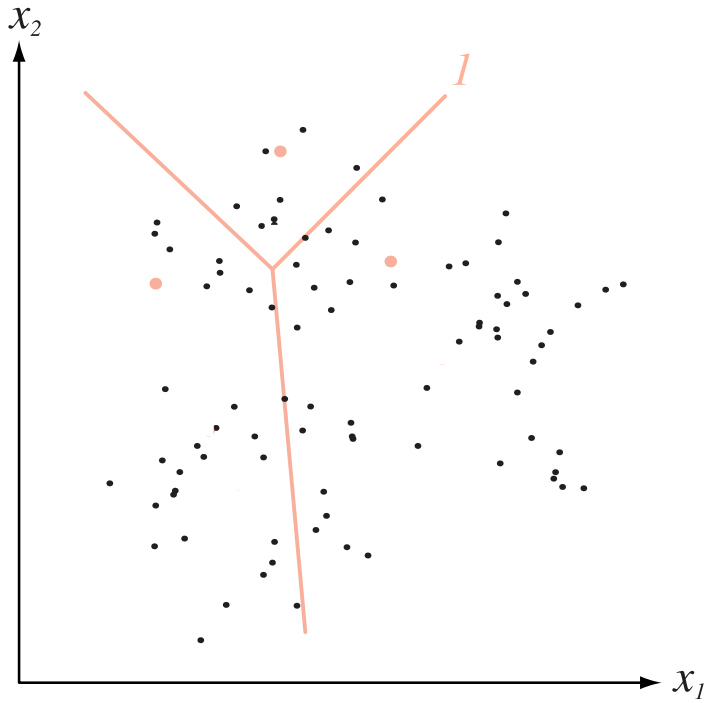
MLE Example



k-Means Example

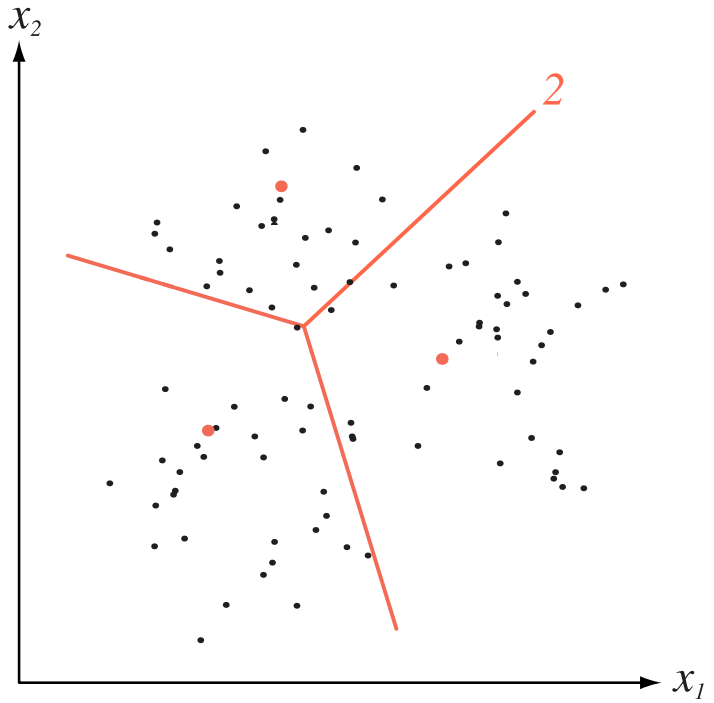
Comparison of MLE and k-Means. Since the overlap between the components is

k-Means In Sample Space



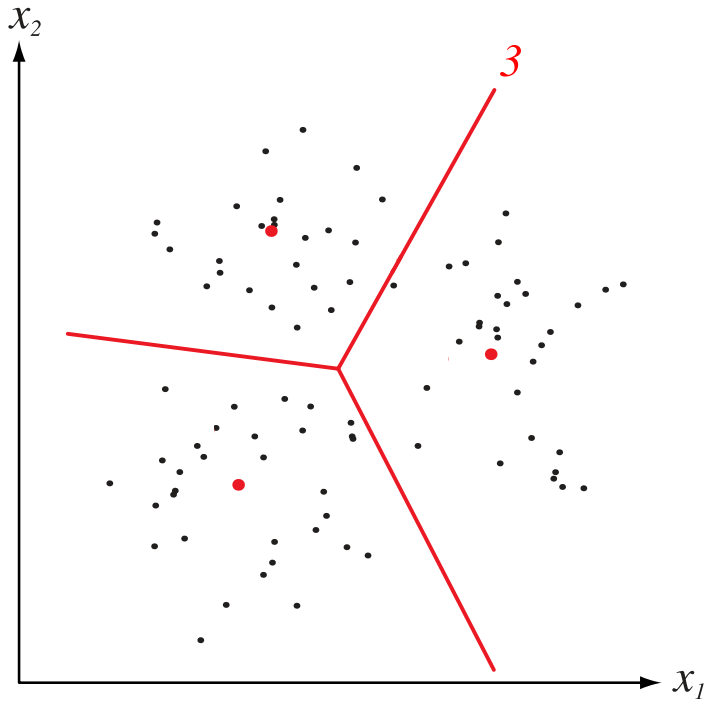
k-Means Sample Space

k-Means In Sample Space



k-Means Sample Space

k-Means In Sample Space



k-Means Sample Space

k-Means Summary

k-Means is a staple of unsupervised clustering methods.

It is simple and relatively fast, operating in $O(ndcT)$ time where n is the number of patterns, d is the number of features, c is the number of clusters and T is the number of iterations.

It typically finishes in a small number of iterations.

k-Means Caveats

When does it fail?

- If we are wrong about our number of classes, we will converge on two means 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!

Evaluating Clusters

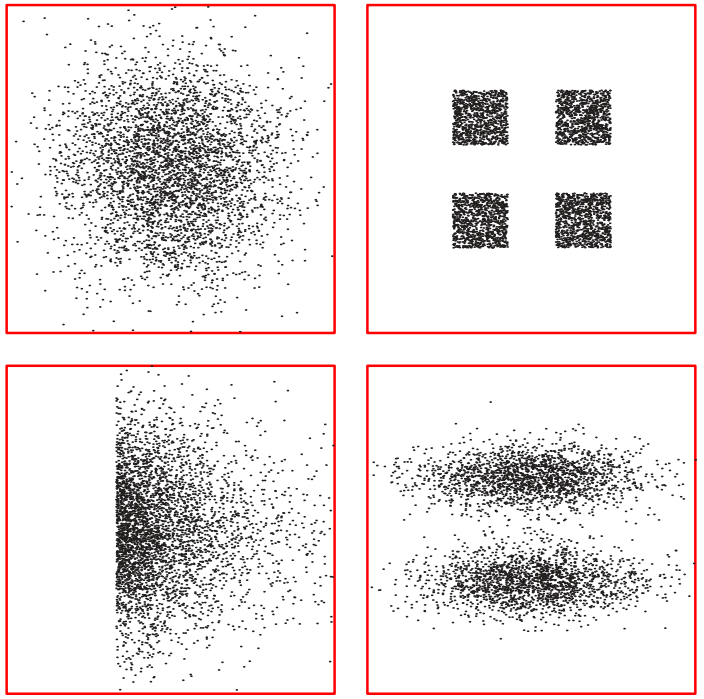
How Well Did We Cluster?

So far, we've been talking about **clusters** and **classes** as being synonymous, but that's clearly not the case.

An underlying density with multiple modes could represent the same class, while a unimodal density may represent multiple classes (i.e. it's a bad feature for discrimination).

We make assumptions about the form of the data, but clearly we could estimate perfectly valid parameter sets and completely miss the underlying structure of the data.

Examples of Misleading Parameters



Distributions with equal μ and Σ.

Similarity Measures

If we are interested in finding **subclasses** in our unlabeled data, we have to define what we mean.

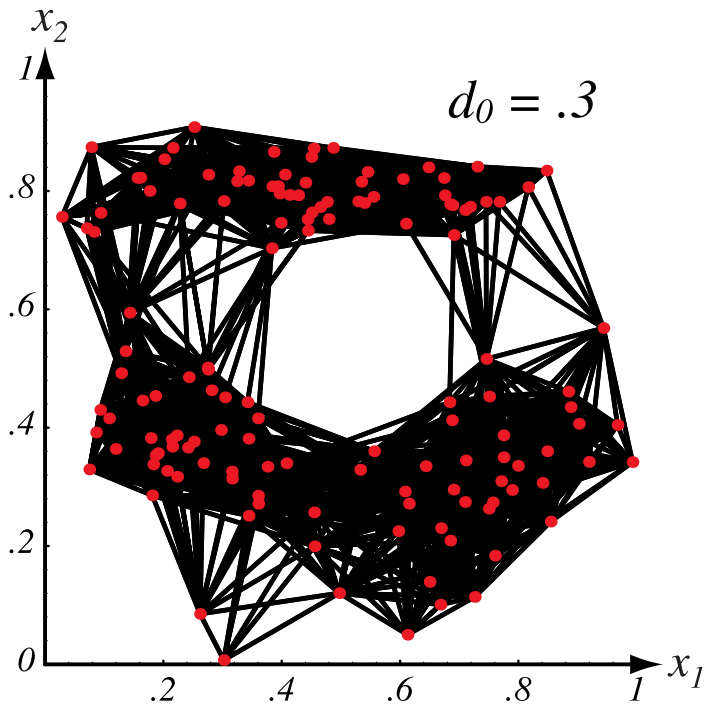
Finding “clusters” seems like a good place to start, but how do we define a cluster?

We need a way to say that samples in one cluster are more similar to each other than they are to samples in another cluster.

We also need a way to evaluate a given partitioning of the dataset, to say that Clustering Result A is better or worse than Clustering Result B.

We can even start by using a distance metric to define our clusters in the first place!

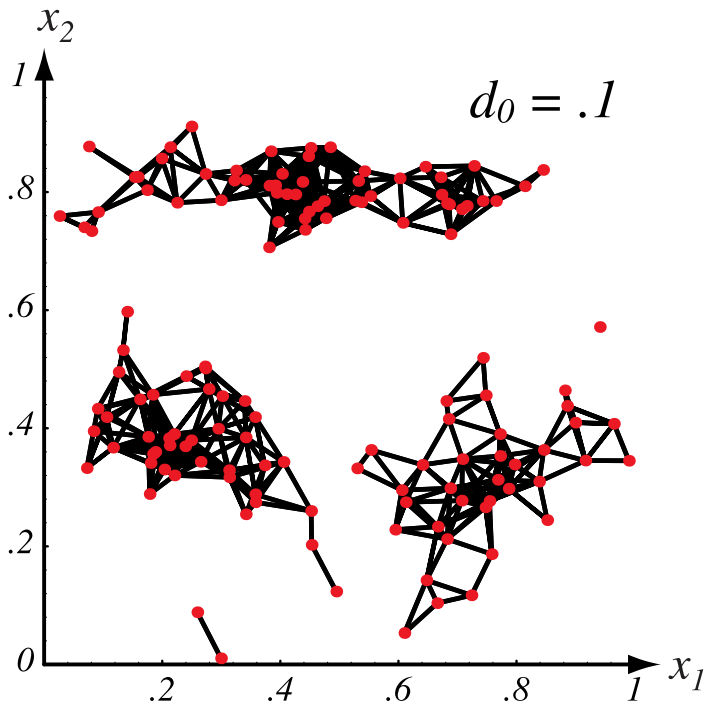
Examples of Distance Thresholding: High Threshold



High Threshold Cluster

Clustering with a high threshold means that almost everything is put into the same

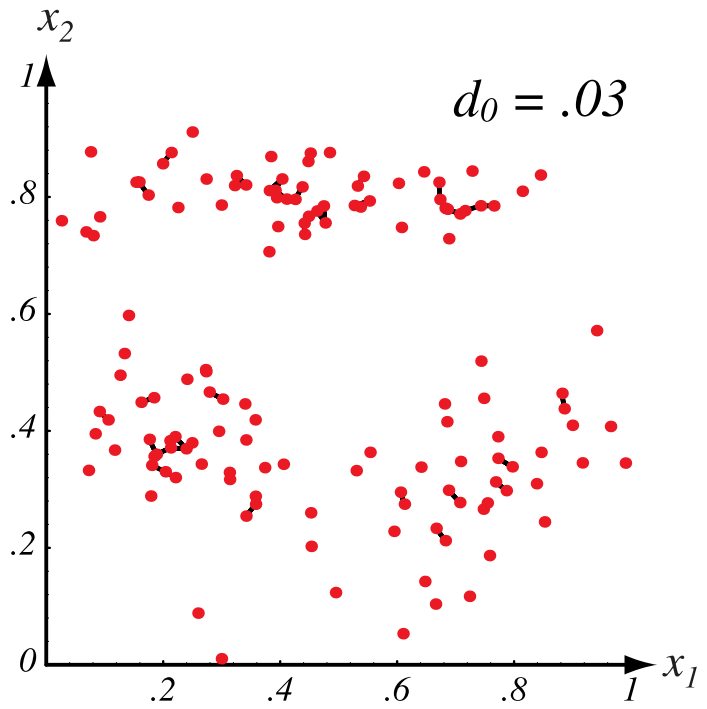
Examples of Distance Thresholding: Medium Threshold



Medium Threshold Cluster

Clustering with an intermediate threshold can reveal structure in your data, but it is up

Examples of Distance Thresholding: Low Threshold



Low Threshold Cluster

Clustering with a low threshold forces only the closest points to be associated with

Choosing a Similarity Metric: Invariance to Transforms

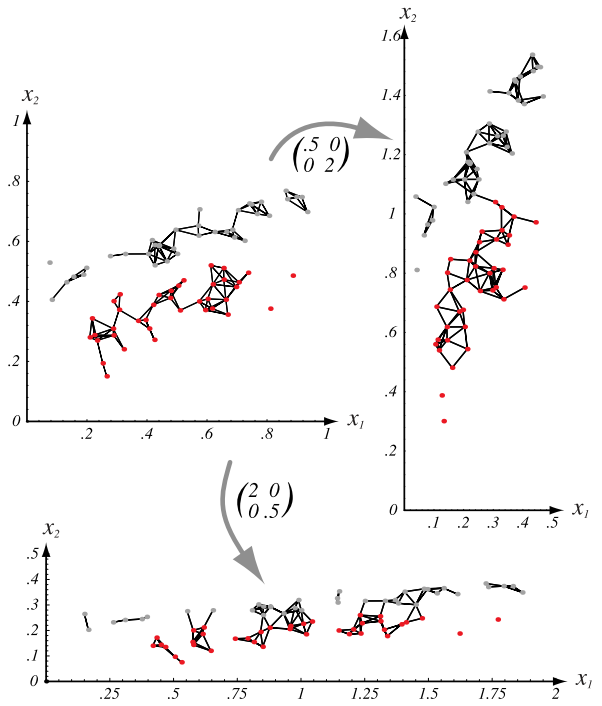
Generally speaking, we assume the correct distance metric is Euclidean (the shortest distance between two points is a straight line).

However, this is only justified if the space is **isotropic** (distances in one dimension are equivalent to those in another) and the data is evenly spread.

Euclidean distance is robust to **rigid** transformations of the feature space (rotation and translation).

It is NOT robust to arbitrary linear transformations which distort distance relationships.

Rotation's Effect on Cluster Groupings



Cluster Scaling

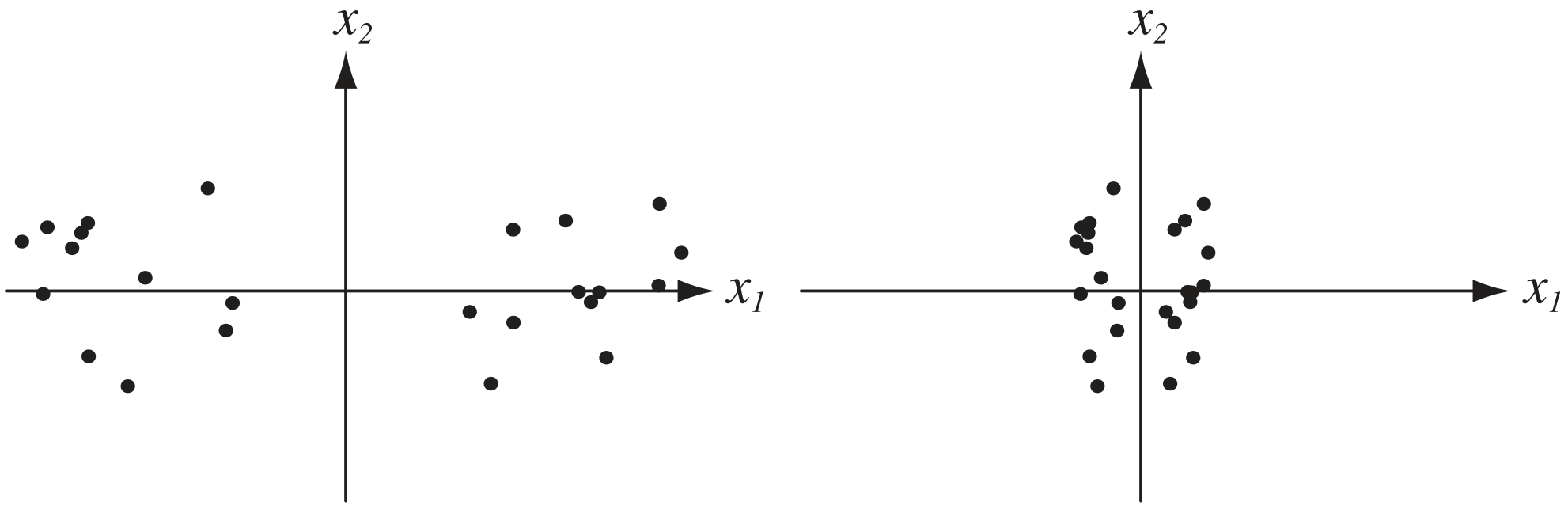
Choosing a Similarity Metric: Normalization

We can try to counteract the effects of non-isotropy in our feature space through **normalization**, where we translate and scale the data to have a zero mean and unit variance.

We can also rotate the coordinate space so that the axes coincide with the eigenvectors of the covariance matrix; this is part of **principal component analysis (PCA)** which we will cover shortly.

However, normalization has its own problems...

Normalization Sometimes Ruins Our Clusters



Normalization

Normalization can take a well-separated dataset with two processes (left) and

Parting Words

Benefits of Unsupervised Methods

Question: Will supervised methods always out-perform unsupervised methods?

As always, **it depends!** Having more information (like labels) is better, but over-fitting is always a danger.

Over-fitting isn't just fine-tuning your decision hyperplanes, it is also when you choose c to be too high, or if you pick too many mixtures for each of your densities.

Sometimes, clustering can give you insight above and beyond what you **thought** you knew!

- If you think you have two classes, and you find three clusters, what does that mean?
- If you have five classes and you find two clusters, what does THAT mean?

Next Class

Similarity Metric Selection

In most situations, the choice of similarity metric / distance function will be arbitrary.

In the literature, Euclidean distance metric is usually assumed, unless you have reason to believe otherwise.

With non-linear datasets, this may NOT true… but that's for next class.

We will also look at evaluating our clusters by calculating **criterion functions**: sum-of-squared-error, minimum variance, scatter matrices, trace / determinant criterions, and invariant criterions.

(We may not look at all of those, but enough to give you the idea.)