# Lecture 19: Cluster Analysis

## Reading: Sections 8.5, 14.3

**GU4241/GR5241 Statistical Machine Learning**

**Linxi Liu**

**March 30, 2017**

CNN and hidden layer

# Clustering

We assign a class to each sample in the data matrix. However, the class *is not an output variable*; we only use input variables.

Clustering is an **unsupervised** procedure, whose goal is to find homogeneous subgroups among the observations. It has wide applications in practice. Image segmentation, handwritten digit identification, vector quantization

In this lecture, we will discuss 2 algorithms:

- Hierarchical clustering
- EM algorithm

We have discussed:

- $K$-means clustering
- $K$-medoids clustering

# Handwritten digit identification

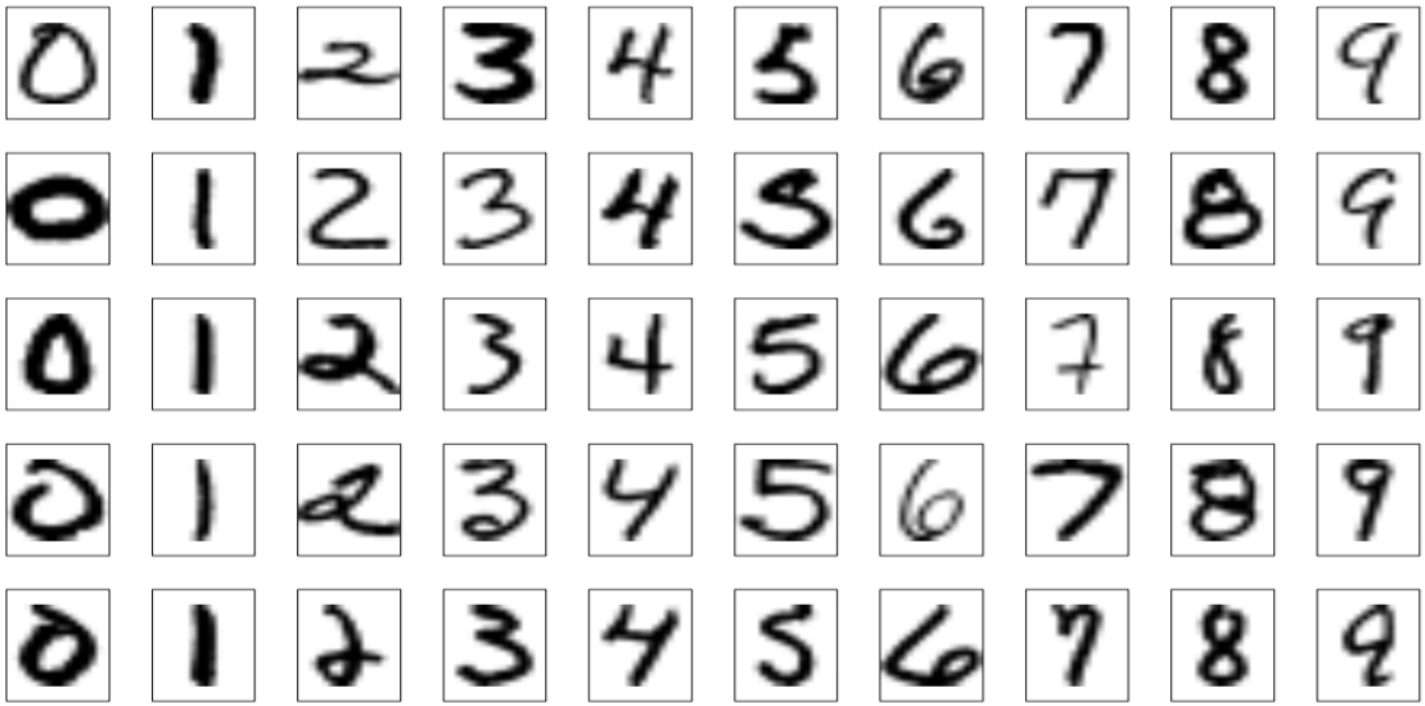
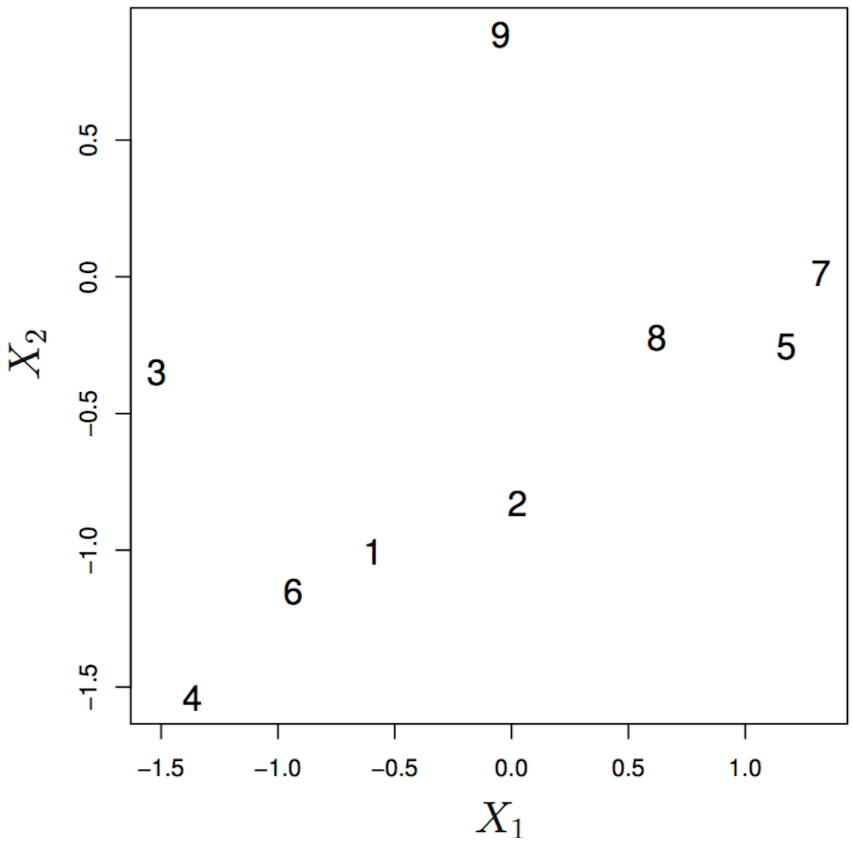


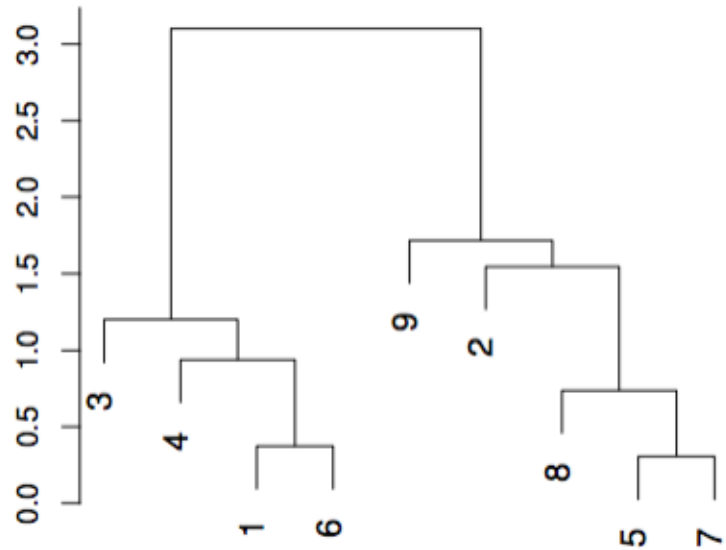
**FIGURE 11.9.** *Examples of training cases from ZIP code data. Each image is a* $16 \times 16$ *8-bit grayscale representation of a handwritten digit.*

# Image segmentation

# Hierarchical clustering

Most algorithms for hierarchical clustering are *agglomerative*.



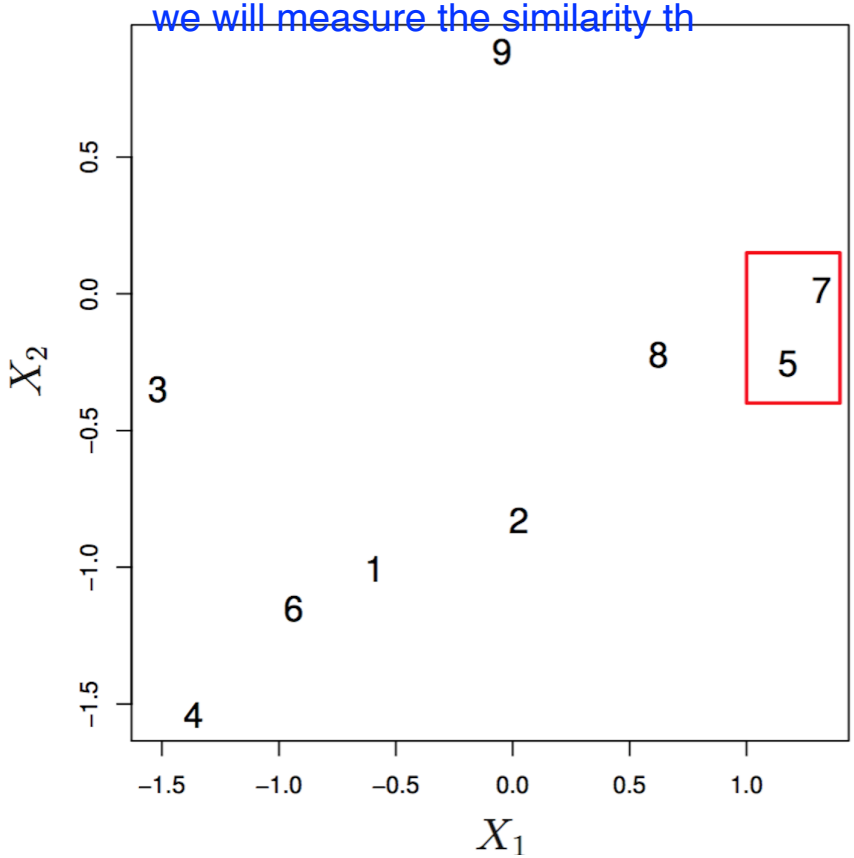



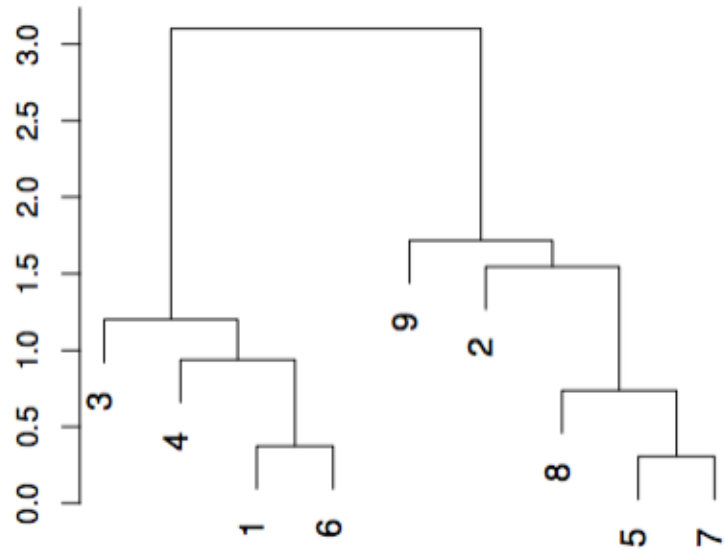

The output of the algorithm is a *dendogram*.

# Hierarchical clustering

Most algorithms for hierarchical clustering are *agglomerative*.

it two points are similar or closed to each other,
then we can merge the data set
we will measure the similarity th

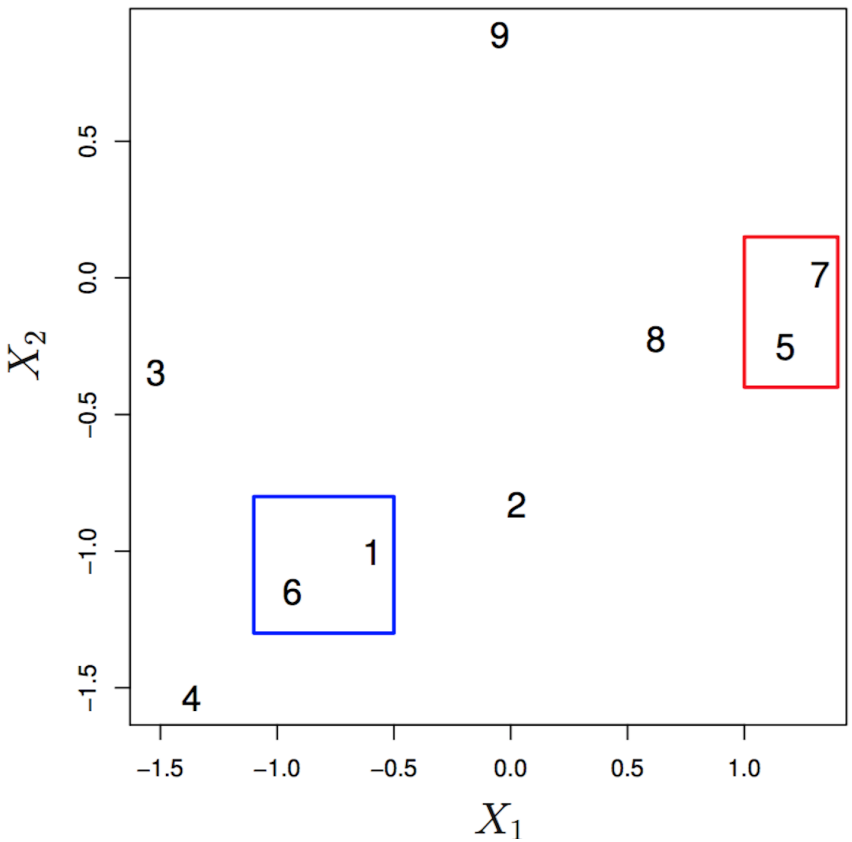


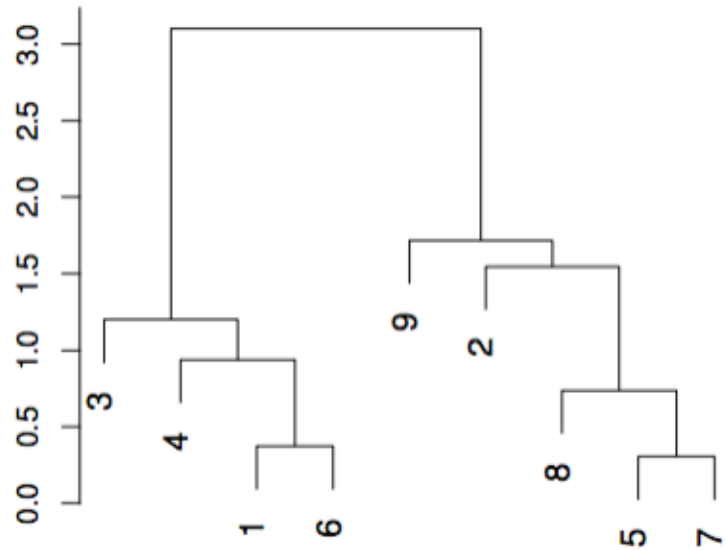




The output of the algorithm is a *dendogram*.

# Hierarchical clustering

Most algorithms for hierarchical clustering are *agglomerative*.



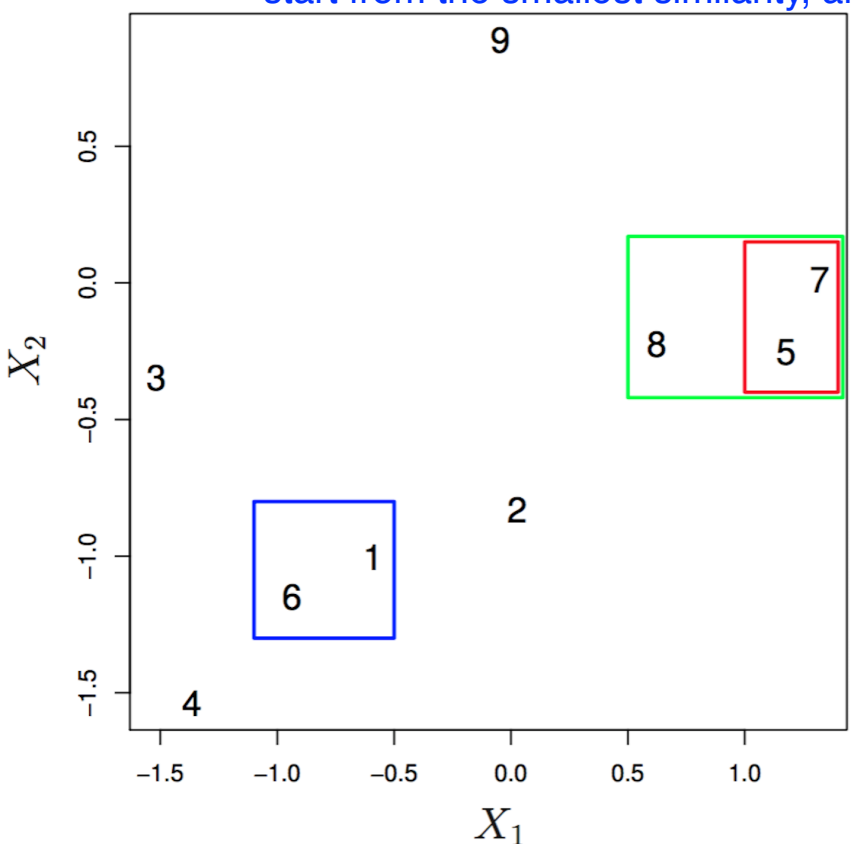



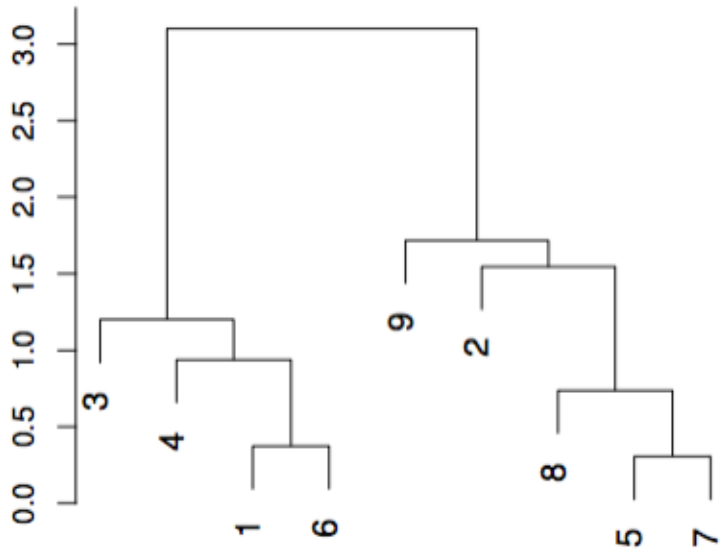

The output of the algorithm is a *dendogram*.

# Hierarchical clustering

Most algorithms for hierarchical clustering are *agglomerative*.

the first step, the similarity is smallest between 7&5
start from the smallest similarity, and then increase as a tree structure

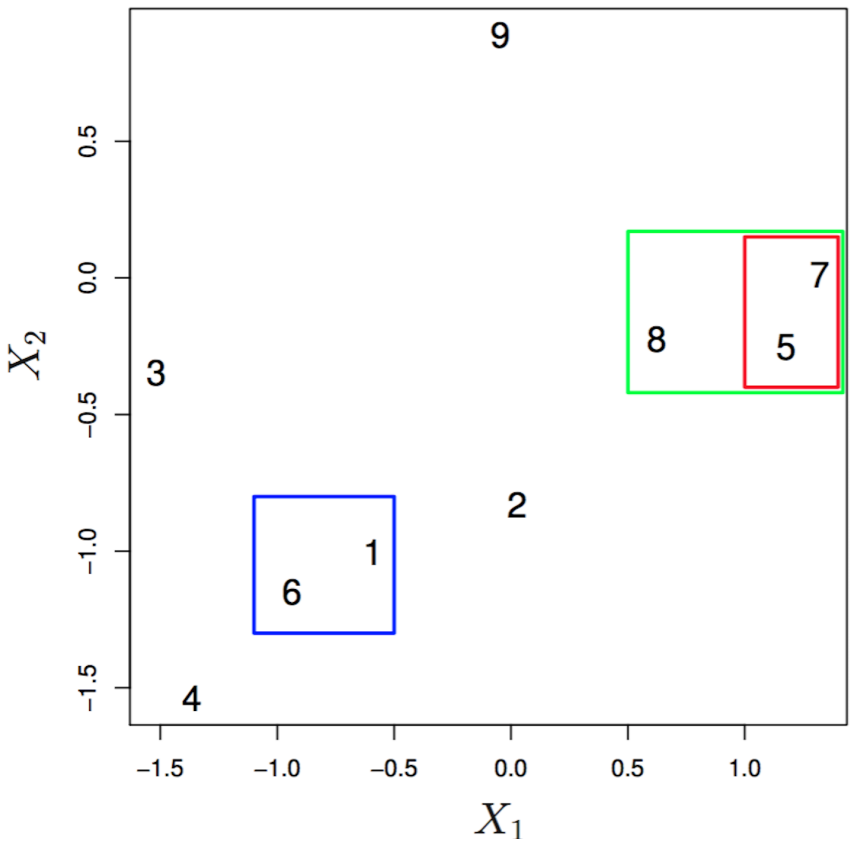


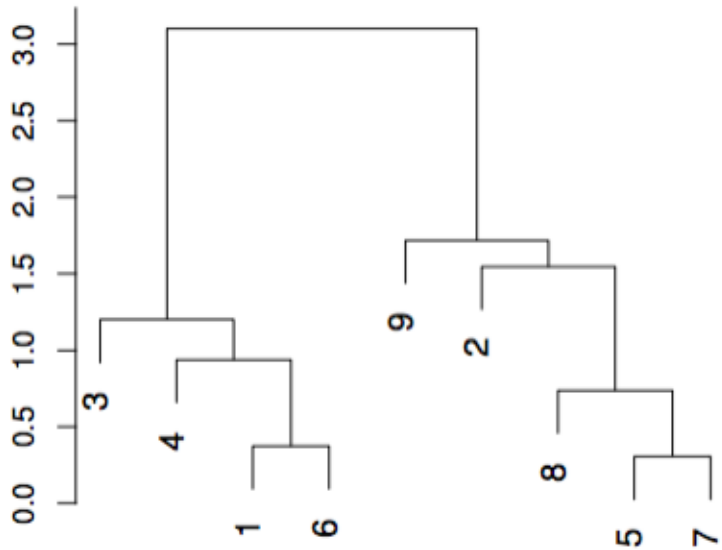




The output of the algorithm is a *dendogram*.

# Hierarchical clustering

Most algorithms for hierarchical clustering are *agglomerative*.

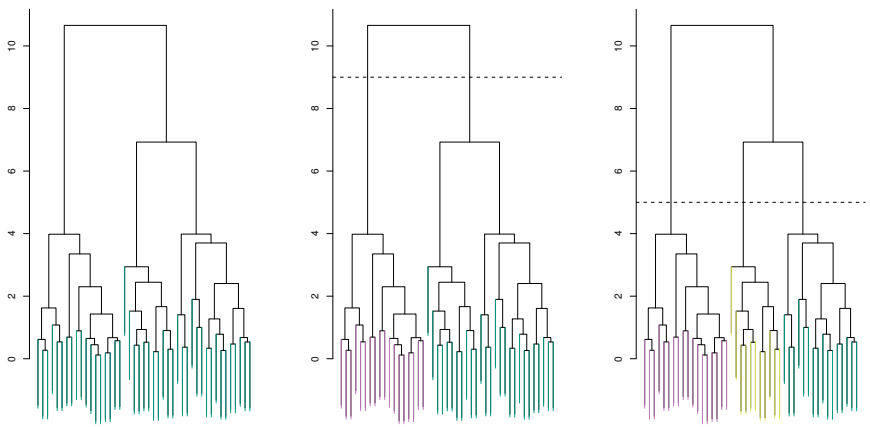



We must be careful about how we interpret the dendogram.

# Hierarchical clustering

if the number is 2, the cluster will reflect of the gender;
if the number of the 3, the result is different



ISL Figure 10.9

- The number of clusters is not fixed.
- Hierarchical clustering is not always appropriate.

  e.g. Market segmentation for consumers of 3 different nationalities.
  - Natural 2 clusters: gender
  - Natural 3 clusters: nationality

  These clusterings are not nested or hierarchical.

# Notion of distance between clusters

It is easy to measure the distance: we use uclidian distance
to measure the similarity distance of two points;
for the two cluster:
1. we will measure the distance between the largest distance
of the points in the two cluster, among all of the pairwise points;
2.

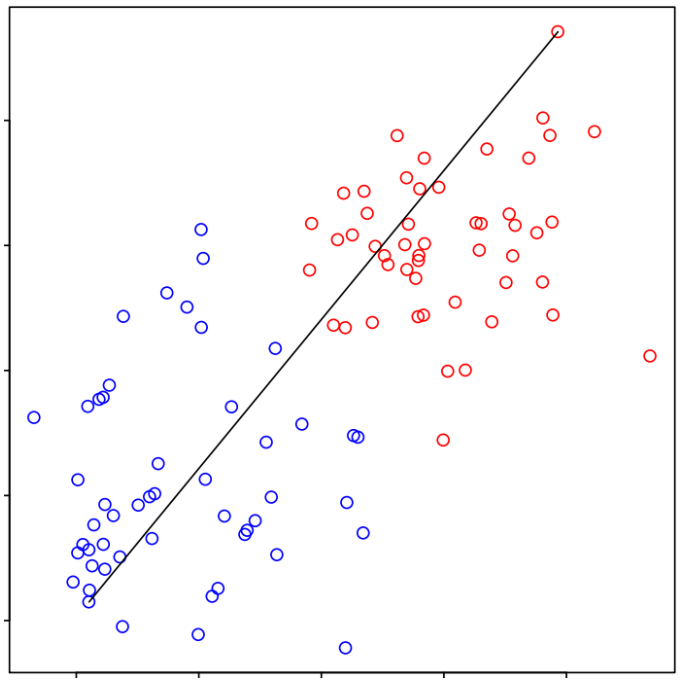
At each step, we link the 2 clusters that are "closest" to each other.

Hierarchical clustering algorithms are classified according to the notion of distance between clusters.

# Notion of distance between clusters

At each step, we link the 2 clusters that are "closest" to each other.

Hierarchical clustering algorithms are classified according to the notion of distance between clusters.

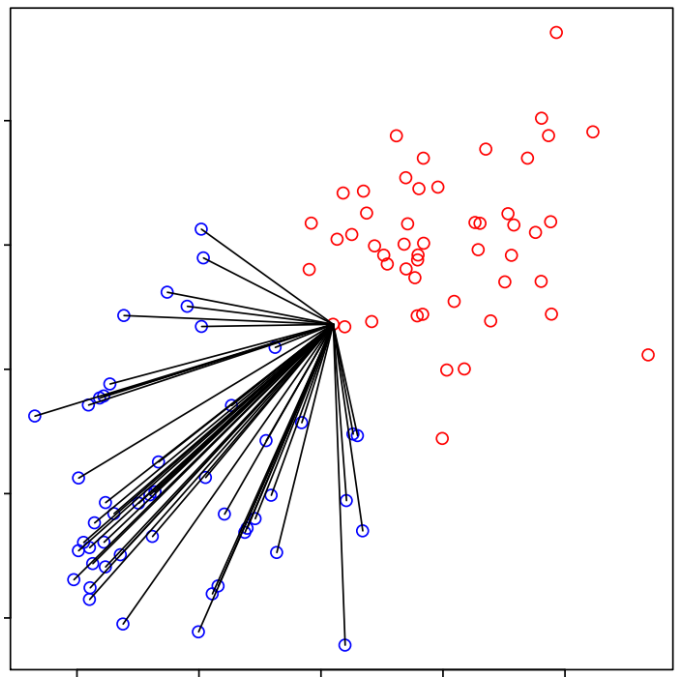


**Complete linkage:**
The distance between 2 clusters is the *maximum* distance between any pair of samples, one in each cluster.

# Notion of distance between clusters

At each step, we link the 2 clusters that are "closest" to each other.

Hierarchical clustering algorithms are classified according to the notion of distance between clusters.

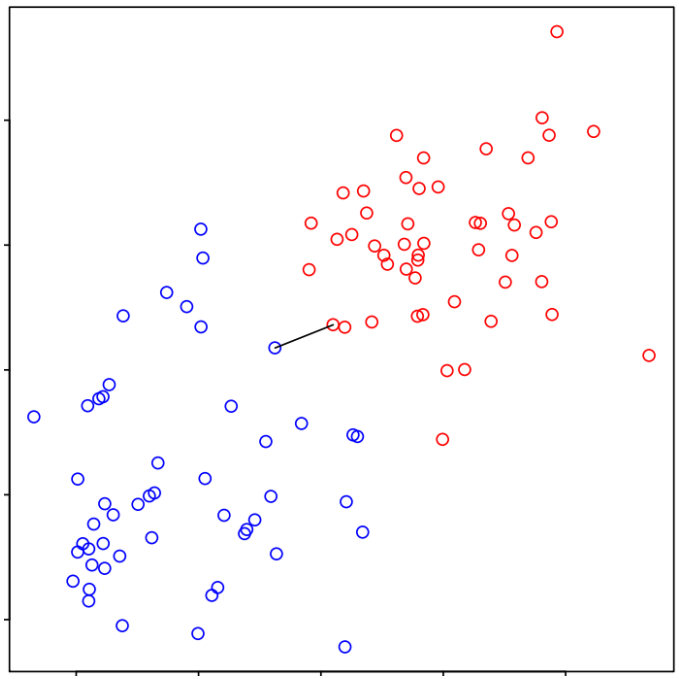


**Average linkage:**
The distance between 2 clusters is the average of all pairwise distances.

# Notion of distance between clusters

At each step, we link the 2 clusters that are "closest" to each other.

Hierarchical clustering algorithms are classified according to the notion of distance between clusters.

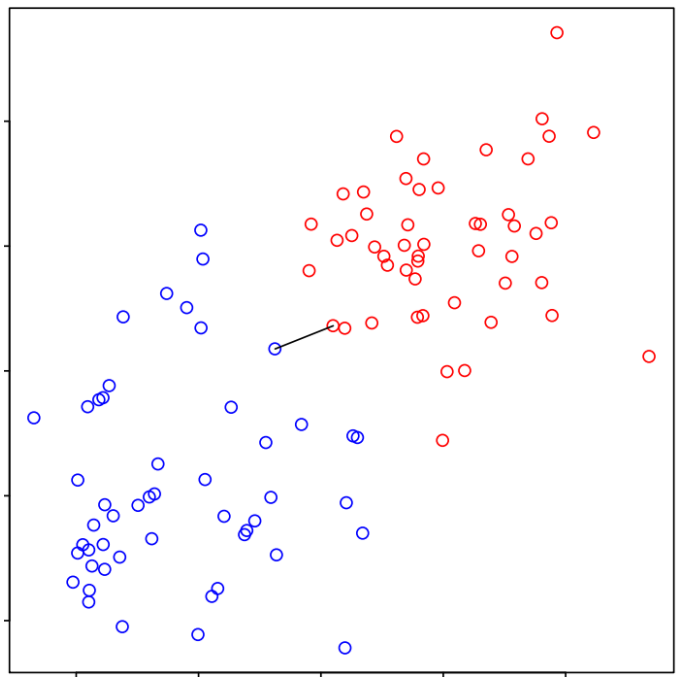


**Single linkage:**
The distance between 2 clusters is the *minimum* distance between any pair of samples, one in each cluster.

# Notion of distance between clusters

At each step, we link the 2 clusters that are "closest" to each other.

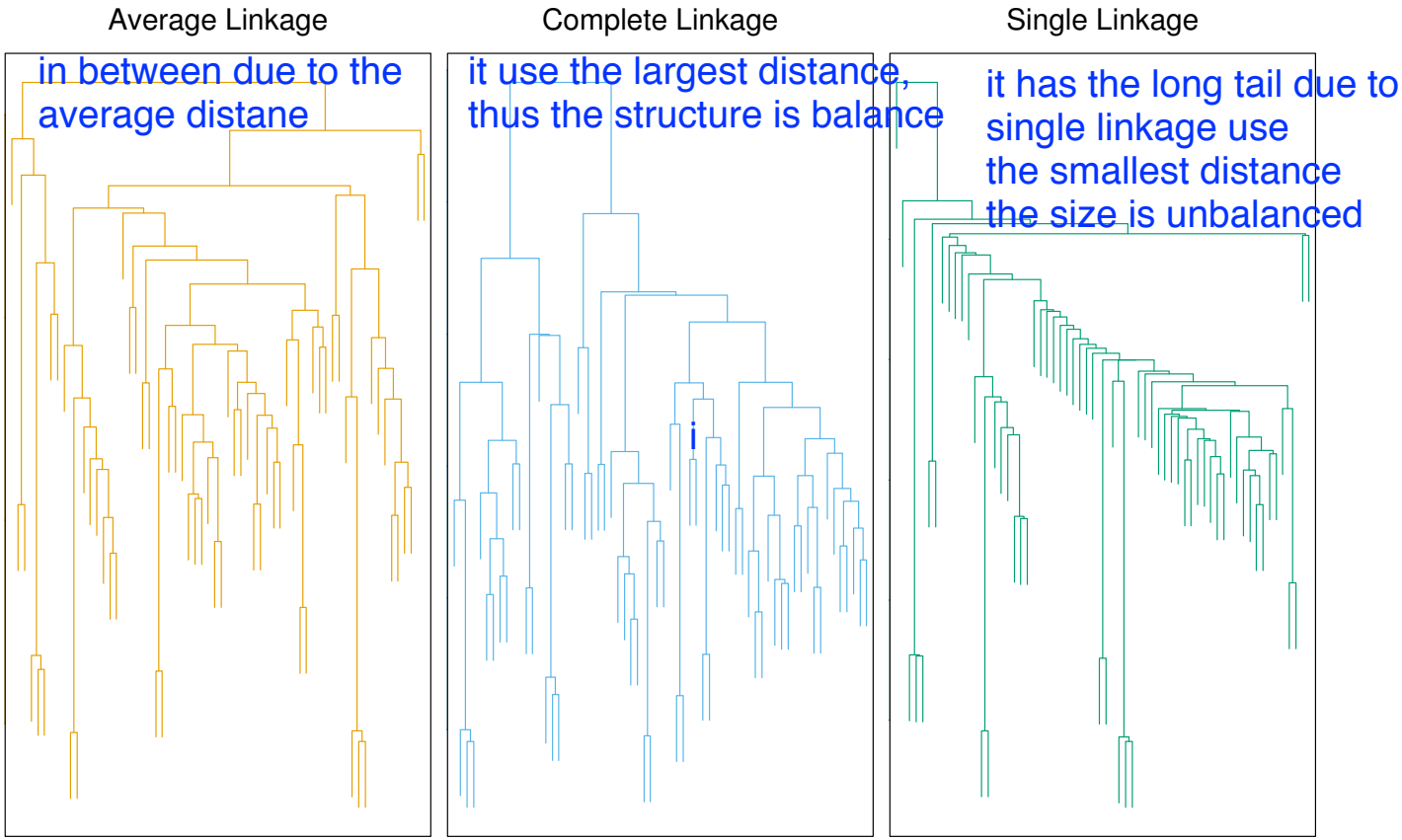
Hierarchical clustering algorithms are classified according to the notion of distance between clusters.



**Single linkage:**
The distance between 2 clusters is the *minimum* distance between any pair of samples, one in each cluster.

Suffers from the *chaining phenomenon*

# Example




ESL Figure 14.13

# Mixture Models

we want to combine different distribution togeter,
thus why we use mixture models

## Mixture

For a parametric model $p(x|\theta)$ and a probability density $q$, a distribution of the form

$$\pi(x) = \int_{\mathcal{T}} p(x|\theta)q(\theta)d\theta$$

the claction distribution:
P(X|Θ)
Θ = (μ,Δ^2)
the final model is π(x) above

is called a **mixture model**. The distribution given by $q$ is called the **mixing distribution**.

## Interpretation

Mixtures describe two-stage sampling procedures. We can generate samples from $\pi$ as follows:

1. Sample $\theta_i \sim q$.
2. Sample $X_i \sim p(\,.\,|\theta_i)$.

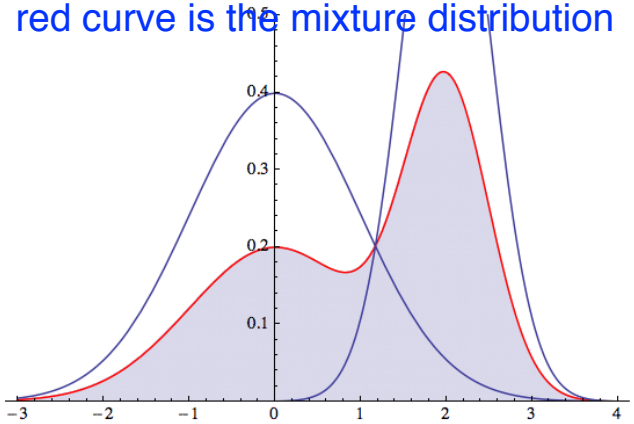
we choose one cluction distribution and then
we will sample q, then we choose particular Θ;

The distribution of a sample $x_1, \ldots, x_n$ generated in this manner has density $\pi$.

# Illustration

## Mixture of two Gaussians

red curve is the mixture distribution

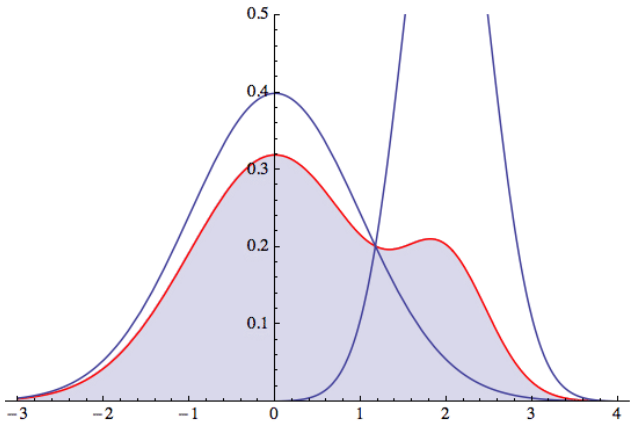



The curve outlined in red is the mixture

$$\pi(x) = 0.5g(x|0,1) + 0.5g(x|2,0.5) \ ,$$

where $g$ is the Gaussian density. The blue curves are the component densities.

## Influence of the weights




change weights, the 0.8 prob we want to sample from first componentnora

Here, the weights $c_1 = c_2 = 0.5$ above have been changed to $c_1 = 0.8$ and $c_2 = 0.2$. The component distributions are the same as above.

# Example: Continuous Mixture

## Example

We are mostly interested *discrete* mixing distributions, but $\theta$ can be continuous variable, as in the following example.

## Mixture components

1. Sample $\theta \sim \text{Gamma}(\alpha, \beta)$.
2. Regard $\theta$ as an inverse variance $\frac{1}{\sigma^2} := \theta$ and sample

normal +gamma = t distribution

$$X \sim \text{Normal}(0, \sigma)$$

## Mixture distribution

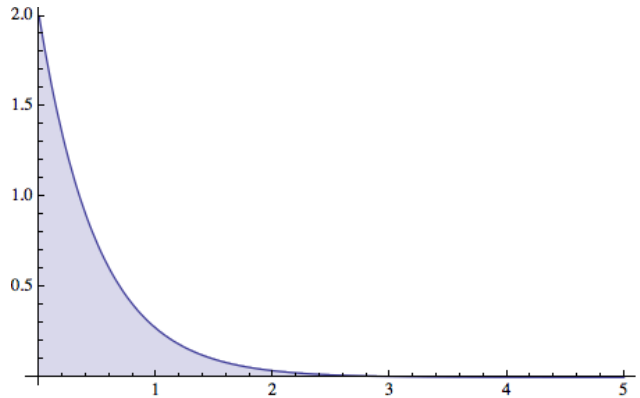
The distribution of $X$ is the mixture with density

$$\pi(x|0, \nu := \frac{\alpha}{\beta}, \tau := 2\alpha) = \int_{\mathbb{R}_+} p_{\text{Normal}}(x|0, 1/\theta) q_{\text{Gamma}}(\theta|\alpha, \beta) d\theta$$

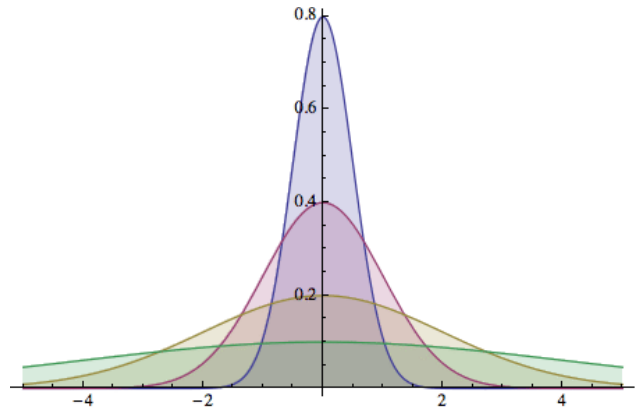
This is **Student's $t$-distribution** with parameters $0$ (the mean of the normal), $\nu$, $\tau$.

# Example: Continuous Mixture

## Mixture components

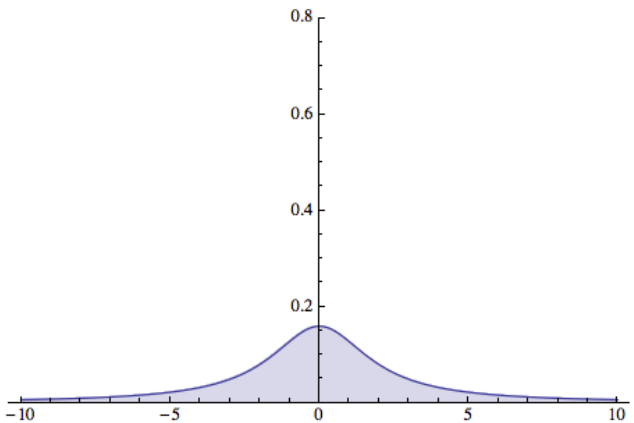



Gamma distribution

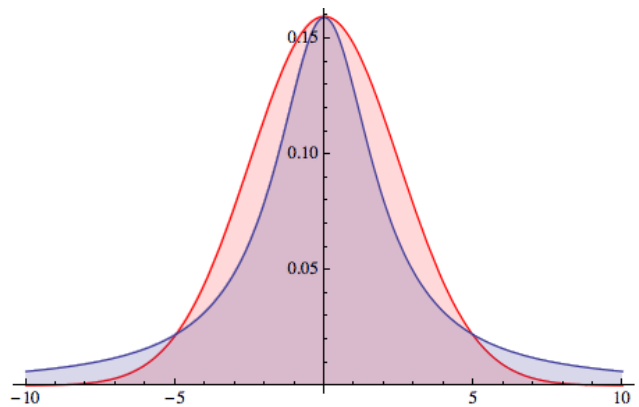



Normal distribution, different variances

## Mixture distribution




The mixture is a Student distribution. Mixing over different variances results in "heavy tails".




Comparison: Normal distribution (red) vs Student distribution (blue)

# Finite Mixtures

## Finite Mixture Model

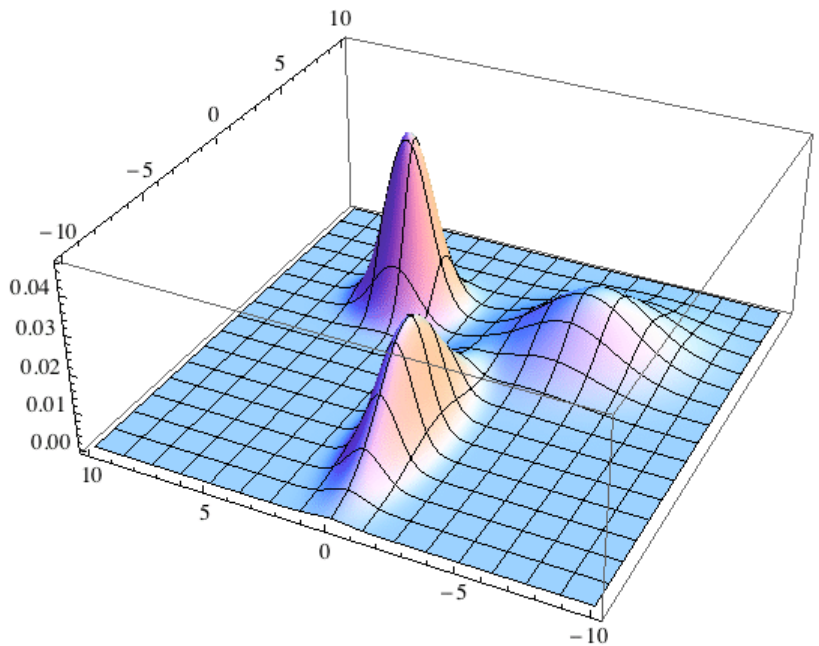
A **finite mixture model** is a distribution with density of the form

$$\pi(x) = \sum_{k=1}^{K} c_k p(x|\theta_k) \ ,$$

where $\sum_k c_k = 1$ and $c_k \geq 0$.

fisst we need to choose a component
based on the discreet distribution;
after we choose a particular sample
each component can from different distribution

## Example: Finite mixture of Gaussians



# Finite Mixtures

## Interpretation as mixture

A mixture is of the form

$$\pi(x) = \int_{\mathcal{T}} p(x|\theta) q(\theta) d\theta \ .$$

We choose $q$ as

$$q = \sum_{k=1}^{K} c_k \delta_{\theta_k}$$

for $K$ fixed values $\theta_k \in \mathcal{T}$. Recall that integration against the Dirac distribution $\delta_\theta$ "picks out" the function value at $\theta$.

The mixture with mixing distribution $q$ is therefore

$$\begin{aligned}\pi(x) &= \int_{\mathcal{T}} p(x|\theta) \Big( \sum_{k=1}^{K} c_k \delta_{\theta_k} \Big) d\theta = \sum_{k=1}^{K} c_k \int_{\mathcal{T}} p(x|\theta) \delta_{\theta_k} d\theta \\ &= \sum_{k=1}^{K} c_k p(x|\theta_k) \ .\end{aligned}$$

# Sampling

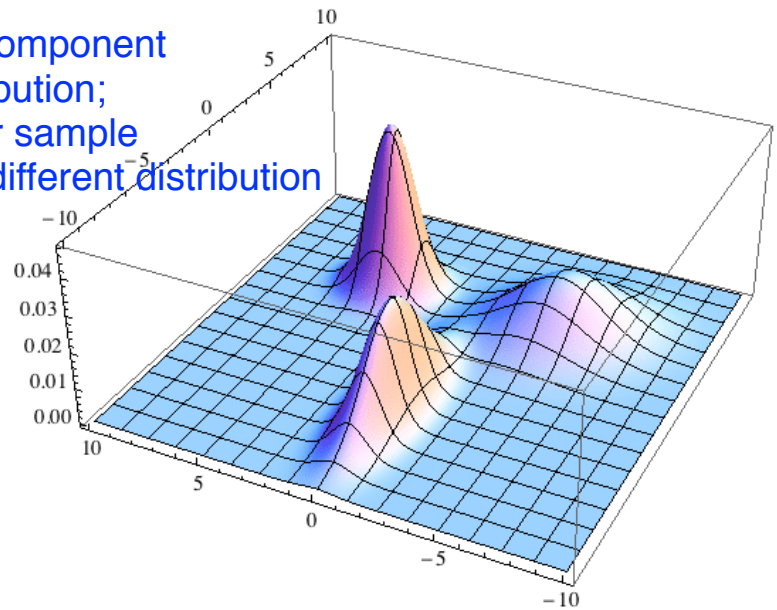
## Sampling from a finite mixture

For a finite mixture with fixed parameters $c_k$ and $\theta_k$, the two-step sampling procedure is:

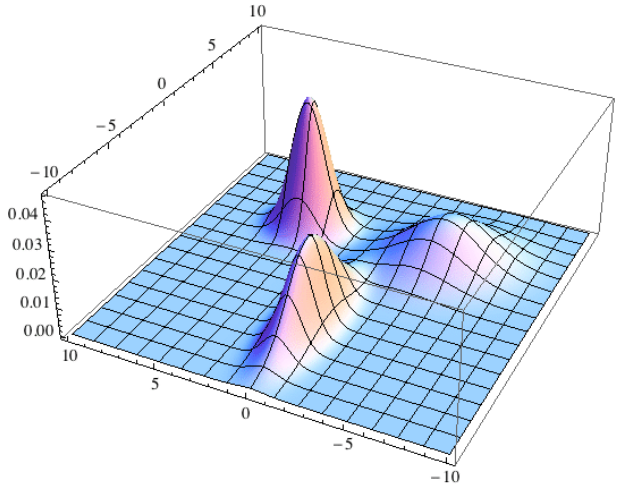
1. Choose a mixture component at random. Each component $k$ is selected with probability $c_k$.
2. Sample $x_i$ from $p(x|\theta_k)$.

**Note:** We always repeat both steps, i.e. for $x_{i+1}$, we choose again choose a (possibly different) component at random.

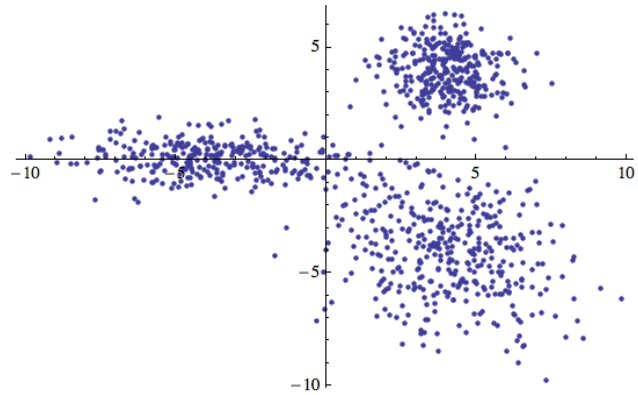
fisst we need to choose a component
based on the discreet distribution;
after we choose a particular sample
each component can from different distribution



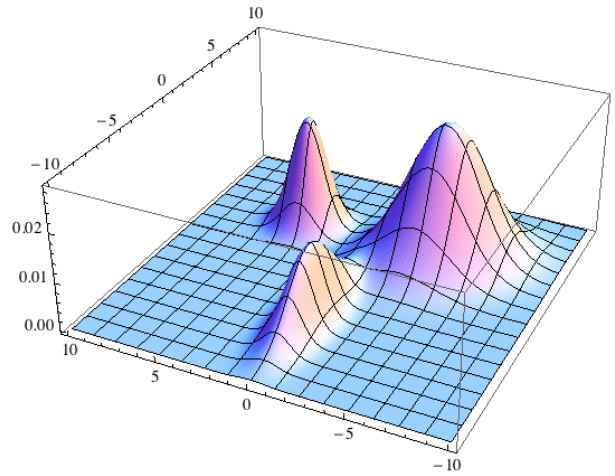
# Illustration: Mixture of Gaussian in 2D



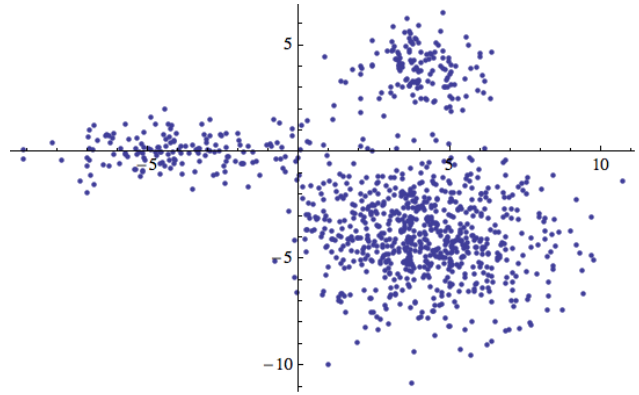

Plot of the mixture density.




A sample of size $1000$.




Same components as above, with weight of one component increased.




A sample of $1000$ points. Note how the relative size of one cluster has increased.

# Finite Mixtures and Clustering

the clustering problem becomes the prob parameter estimation problem

## Clustering with finite mixtures

if we have the mixture distribution;
then we want to know the likelyhood of each obsv
there are k clusters
then
L(xi) = Σ log(ΣCkp(xi | Θk))
the parameter of the model is the mixing weight;
vector c and Θ=(Θ1,Θ2,…Θk)

For a clustering problem with $K$ clusters,

$$p(x|\theta_k) = \text{model of cluster } k$$

The weight $c_k$ is the relative cluster size.

## Estimation problem

If $K$ is fixed and given, the unknown parameters of a mixture model are the weights $c_k$ and the cluster parameters $\theta_k$. The parameters of finite mixtures are estimated using a method known as the *EM algorithm*.

# Mixture Estimation

## Maximum likelihood for finite mixtures

Writing down the maximum likelihood problem is straightforward:

$$(\hat{\mathbf{c}}, \hat{\boldsymbol{\theta}}) := (\hat{c}_1, \ldots, \hat{c}_K, \hat{\theta}_1, \ldots, \hat{\theta}_K) = \arg\max_{\mathbf{c}, \boldsymbol{\theta}} \prod_{i=1}^{n} \Big( \sum_{k=1}^{K} c_k p(x_i | \theta_k) \Big)$$

The maximality equation for the logarithmic likelihood is

$$\frac{\partial}{\partial(\mathbf{c}, \boldsymbol{\theta})} \sum_{i=1}^{n} \log \Big( \sum_{k=1}^{K} c_k p(x_i | \theta_k) \Big) = 0$$

The component equation for each $\theta_k$ is:

$$\sum_{i=1}^{n} \frac{c_k \frac{\partial}{\partial \theta_k} p(x_i | \theta_k)}{\sum_{k=1}^{K} c_k p(x_i | \theta_k)} = 0$$

Solving this problem is analytically infeasible (note that we cannot multiply out the denominator, because of the sum over $i$). Even numerical solution is often difficult.

# Latent Variables

## Problems with ML estimation

- Solving the ML problem is difficult.
- For clustering, the maximum likelihood solution does not tell us *which* cluster generated each $x_i$.

## Cluster assignments

- The mixture assumption implies that each $x_i$ was generated from one component.
- For each $x_i$, we again use an **assignment variable** $m_i \in \{1, \ldots, K\}$ which encodes which cluster $x_i$ was sampled from.

we introduce a variable to let us know
which particular component it comes from
"m" = <1,…k>, we do not observe m

after we introducing more variables,
but it is easer to solve the problem

## Latent Variables

Since we do not know which component each $x_i$ was generated by, the values of the assignment variables are *unobserved*. Such variables whose values are not observed are called **latent variables** or **hidden variables**.

# Estimation With Latent Variables

## Latent variables as auxiliary information

If we knew the correct assignments $m_i$, we could:

- Estimate each component distribution $p(x|\theta_k)$ separately, using only the data assigned to cluster $k$.
- Estimate the cluster proportions $c_k$ as $\hat{c}_k := \frac{\#\text{points in cluster } k}{n}$.

## EM algorithm: Idea

The EM algorithm estimates values of the latent variables to simplify the estimation problem. EM altnernates between two steps:

1. Estimate assignments $m_i$ given current estimates of the parameters $c_k$ and $\theta_k$ ("E-step"). we try to guess the lables, the value of latent varaible at E step
2. Estimate parameters $c_i$ and $\theta_k$ given current estimates of the assignments ("M-step"). we assume we know the lables, we just use MLE because we can just estimate the element from same component

These two steps are iterated repeatedly.

# Representation of Assignments

We re-write the assignments as vectors of length $K$:

in practice , we will use a large full vector to indicate
the Mi, if kth element = 1
if M1 = 1 is the kth element of the full vector;
then the first component is from the kth cluster

$\mathbf{x}_i$ in cluster $k$ as $M_i := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \longleftarrow k$th entry

so $M_{ik} = 1$ if $x_i$ in cluster $k$, and $M_{ik} = 0$ otherwise.
We collect the vectors into a matrix

the sum of the row = 1
the sum of the col = cluster size

$$\mathbf{M} = \begin{pmatrix} M_{11} & \dots & M_{1K} \\ \vdots & & \vdots \\ M_{n1} & \dots & M_{nK} \end{pmatrix}$$

Note: Rows = observations, columns = clusters
Row sums = 1, column sums = cluster sizes.

aik is the poterior prob which used in the bayes rules
P(Mi = k | Xi) = Ck*P(xi | Θk)/ΣCi*P(Xi | Θi)
use aik to estimate Mik. instead of the guess of the lables between 0 and 1
we use probability aik to guess lables

E-Step

## Hard vs soft assignments

- The vectors $M_i$ are "hard assignments" with values in $\{0, 1\}$.
- EM computes "soft assignments" $a_{ik}$ with values in $[0, 1]$.
- Once the algorithm terminates, each point is assigned to a cluster by setting

$$m_i := \operatorname*{argmax}_k a_{ik}$$

The vectors $M_i$ are the latent variables in the EM algorithm. The $a_{ik}$ are there current estimates.

## Assignment probabilities

The soft assignments are computed as

$$a_{ik} := \frac{c_k p(x_i|\theta_k)}{\sum_{l=1}^{K} c_l p(x_i|\theta_l)} .$$

They can be interpreted as

$$a_{ik} := \mathbb{E}[M_{ik}|x_i, \mathbf{c}, \boldsymbol{\theta}] = \Pr\{x_i \text{ generated by component } k \,|\, \mathbf{c}, \boldsymbol{\theta}\}$$

# M-Step (1)

## Objective

The M-Step re-estimates $\mathbf{c}$ and $\boldsymbol{\theta}$. In principle, we use maximum likelihood within each cluster, but we have to combine it with the use of weights $a_{ik}$ instead "switch variables" $M_{ik}$.

## Cluster sizes

If we know which points belong to which cluster, we can estimate the cluster proportions $c_k$ by counting point:

$$\hat{c}_k = \frac{\#\text{ points in cluster } k}{n} = \frac{\sum_{i=1}^{n} M_{ik}}{n}$$

we use aik matrix to estimate the m

# M-Step (1)

## Objective

The M-Step re-estimates $\mathbf{c}$ and $\boldsymbol{\theta}$. In principle, we use maximum likelihood within each cluster, but we have to combine it with the use of weights $a_{ik}$ instead "switch variables" $M_{ik}$.

## Cluster sizes

If we know which points belong to which cluster, we can estimate the cluster proportions $c_k$ by counting point:

mean is the average, we replace aik to Mik

$$\hat{c}_k = \frac{\#\text{ points in cluster } k}{n} = \frac{\sum_{i=1}^{n} M_{ik}}{n}$$

Since we do not know $M_{ik}$, we substitute our current best guess, which are the expectations $a_{ik}$:

$$\hat{c}_k := \frac{\sum_{i=1}^{n} a_{ik}}{n}$$

SAME FOR COVARIANCE

# M-Step (2)

## Gaussian special case

The estimation of the component parameters $\theta$ depends on which distribution we choose for $p$. For now, we assume a Gaussian.

## Component parameters

We use maximum likelihood to estimate $\theta = (\mu, \Sigma)$. We can write the MLE of $\mu_k$ as

$$\hat{\mu}_k := \frac{1}{\#\text{ points in cluster } k} \sum_{i:\ x_i \text{ in } k} x_i = \frac{\sum_{i=1}^n M_{ik} x_i}{\sum_{i=1}^n M_{ik}}$$

Estimating M is the same for estimating indicator posterior

# M-Step (2)

## Gaussian special case

The estimation of the component parameters $\theta$ depends on which distribution we choose for $p$. For now, we assume a Gaussian.

## Component parameters

We use maximum likelihood to estimate $\theta = (\mu, \Sigma)$. We can write the MLE of $\mu_k$ as

$$\hat{\mu}_k := \frac{1}{\#\text{ points in cluster } k} \sum_{i:\ x_i \text{ in } k} x_i = \frac{\sum_{i=1}^n M_{ik} x_i}{\sum_{i=1}^n M_{ik}}$$

By substituting current best guesses ($=a_{ik}$) again, we get:

$$\hat{\mu}_k := \frac{\sum_{i=1}^n a_{ik} x_i}{\sum_{i=1}^n a_{ik}}$$

For the covariance matrices:

$$\hat{\Sigma}_k := \frac{\sum_{i=1}^n a_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^t}{\sum_{i=1}^n a_{ik}}$$

# Notation Summary

## Assignment probabilities

$$\mathbf{a} = \begin{pmatrix} a_{11} & \dots & a_{1K} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nK} \end{pmatrix} = \mathbb{E}\left[ \begin{pmatrix} M_{11} & \dots & M_{1K} \\ \vdots & & \vdots \\ M_{n1} & \dots & M_{nK} \end{pmatrix} \right] = \begin{pmatrix} \mathbb{E}[M_{11}] & \dots & \mathbb{E}[M_{1K}] \\ \vdots & & \vdots \\ \mathbb{E}[M_{n1}] & \dots & \mathbb{E}[M_{nK}] \end{pmatrix}$$

Rows = observations, columns = clusters.

## Mixture parameters

$\boldsymbol{\tau} = (\mathbf{c}, \boldsymbol{\theta})$ $\quad$ $\mathbf{c} =$ cluster proportions $\quad$ $\boldsymbol{\theta} =$ component parameters

## Iterations

$\theta^{(\mathrm{j})}$, $\mathbf{a}^{(\mathrm{j})}$ etc = values in $j$th iteration

# Summary: EM for Gaussian Mixture

## Gaussian special case

$\theta = (\mu, \Sigma)$ (mean & covariance) $p(x|\theta) = g(x|\mu, \Sigma)$ (Gaussian density)

## Algorithm

The EM algorithm for a finite mixture of Gaussians looks like this:

- **Initialize:** Choose random values $c_k^{(0)}$ and $\theta_k^{(0)}$.
- **E-Step:** Recompute the assignment weight matrix as

$$a_{ik}^{(j+1)} := \frac{c_k^{(j)} g(x_i|\theta_k^{(j)})}{\sum_{l=1}^{K} c_l^{(j)} g(x_i|\theta_l^{(j)})} .$$

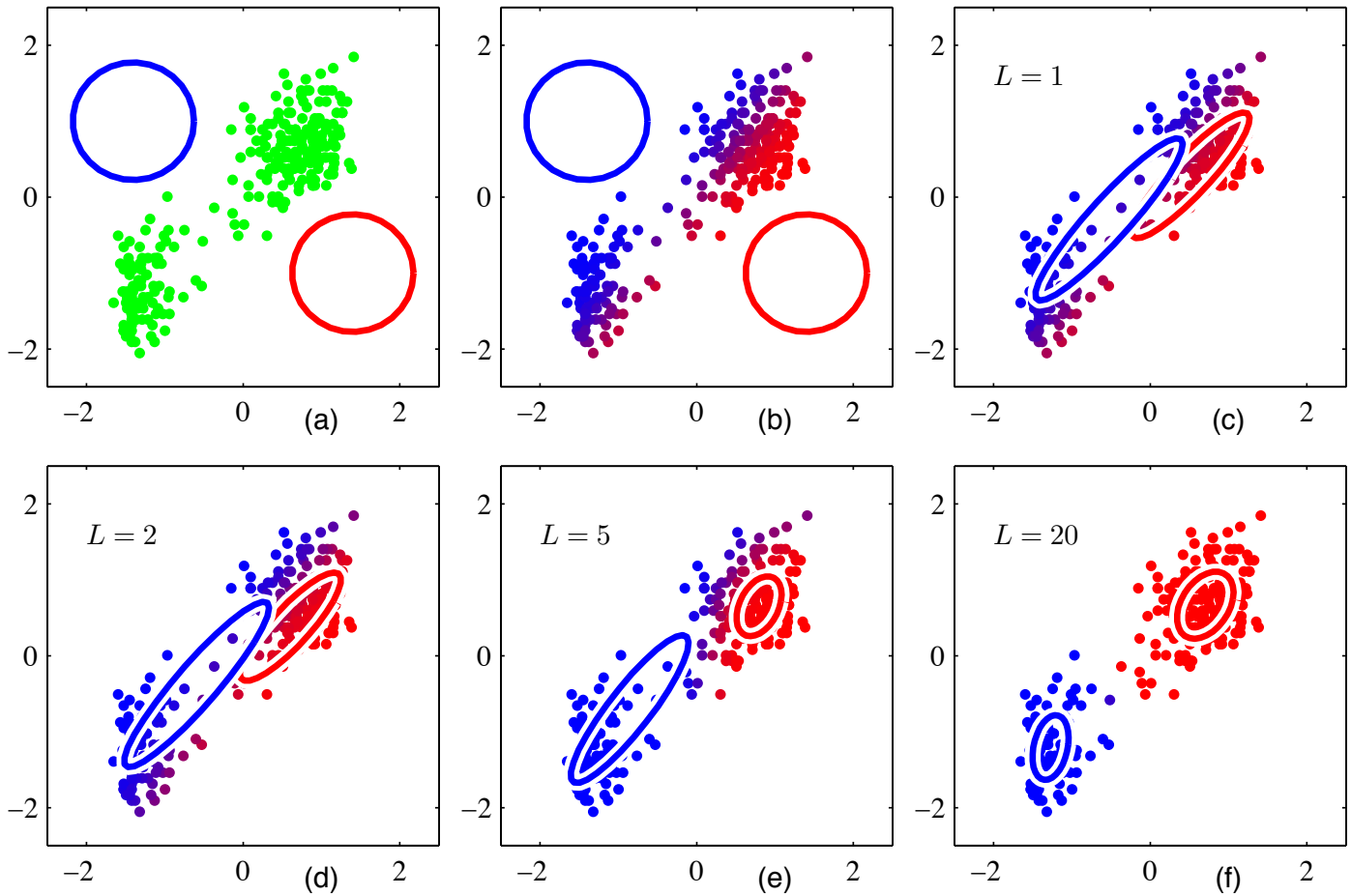
- **M-Step:** Recompute the proportions $c_k$ and parameters $\theta_k = (\mu_k, \Sigma_k)$ as

$$\mu_k^{(j+1)} := \frac{\sum_{i=1}^{n} a_{ik}^{(j+1)} x_i}{\sum_{i=1}^{n} a_{ik}^{(j+1)}}, \quad \Sigma_k^{(j+1)} := \frac{\sum_{i=1}^{n} a_{ik}^{(j+1)} (x_i - \mu_k^{(j+1)})(x_i - \mu_k^{(j+1)})^t}{\sum_{i=1}^{n} a_{ik}^{(j+1)}}$$

The E-Step and M-Step are repeated alternatingly until convergence criterion (e.g. threshold) satisfied.

# EM: Illustration

## EM for a mixture of two Gaussians




The algorithm fits both the mean and the covariance parameter.

# Convergence Properties

it similar to K means

## Log-likelihood

- It can be shown that the likelihood

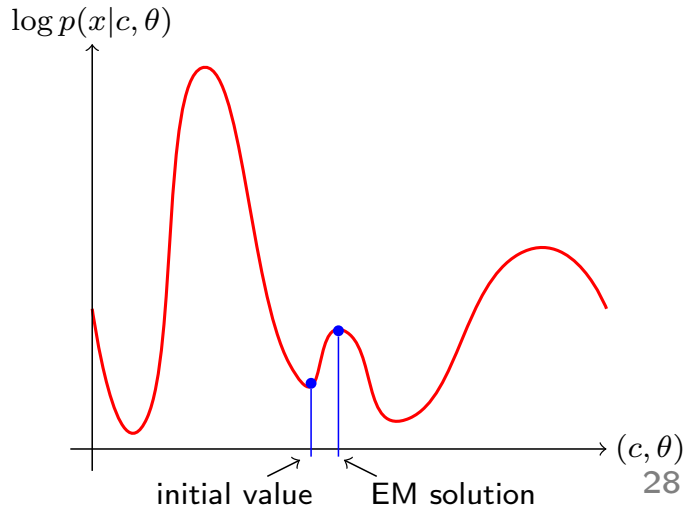
$$\prod_{i=1}^{n} \pi(x_i|\mathbf{c}, \boldsymbol{\theta})$$

always increases from each step to the next, unless $(\mathbf{c}, \boldsymbol{\theta})$ is already a stationary point.

- The theory guarantees only that the algorithm terminates at a stationary point. That point can be a saddle point rather than a maximum (very rare problem).

## The real problem: Local maxima

- EM is effectively a gradient method.
- The maxima it finds are **local maxima of the log-likelihood**.



# EM in Practice

## Comparing solutions

- If $(\mathbf{c}, \boldsymbol{\theta})$ and $(\mathbf{c}', \boldsymbol{\theta}')$ are two different EM solutions, we can always compute the log-likelihoods

$$\sum_i \log \pi(x_i|\mathbf{c}, \boldsymbol{\theta}) \quad \text{and} \quad \sum_i \log \pi(x_i|\mathbf{c}', \boldsymbol{\theta}')$$

  (no approximations or complications!).

- The solution with the higher likelihood is better.
- This is a very convenient feature of EM: Different solutions are comparable.

## Random restarts

In practice, the best way to use EM is often:

- Restart EM repeatedly with randomly chosen initial values.
- Compute the log-likelihoods of all solutions and compare them.
- Choose the solution achieving maximal log-likelihood.

# $K$-means clustering algorithm

special case of the EM;
it similar to use K Gaussian. the covariance matrix are the identity matrix

1. Assign each sample to a cluster from 1 to $K$ arbitrarily, e.g. at random.

2. Iterate these two steps until the clustering is constant:

   - Find the *centroid* of each cluster $\ell$; i.e. the average $\overline{x}_{\ell,:}$ of all the samples in the cluster:

$$\overline{x}_{\ell,j} = \frac{1}{|\{i : C(i) = \ell\}|} \sum_{i:C(i)=\ell} x_{i,j} \quad \text{for } j = 1, \ldots, p.$$

   - Reassign each sample to the nearest centroid.

# $K$-means: Gaussian Interpretation

## $K$ Gaussians

Consider the following algorithm:

- Suppose each $\mu_k$ is the expected value of a Gaussian density $p(x|\mu_k, \mathbb{I})$ with unit covariance. the form of the gaussian distribution: pdf
  the ll Xi - μrll^2 is the uclidy distance
- Start with $K$ randomly chose means and iterate:
  1. Assign each $x_i$ to the Gaussian under which it has the highest probability of occurence (more precisely: highest density value). which is the same to the x to the closest cluster
  2. Given the assignments, fit $p(x|\mu_k, \mathbb{I})$ by maximum likelihood estimation of $\mu_k$ from all points assigned to cluster $k$.

     in the EM, we use hard assimention

# $K$-means: Gaussian Interpretation

## Comparison to $K$-means

- Since the Gaussians are spherical with identical covariance, the density $p(x_i|\mu_k, \mathbb{I})$ is largest for the mean $\mu_k$ which is closest to $x_i$ in Euclidean distance.
- The maximum likelihood estimator of $\mu_k$ is

$$\hat{\mu}_k := \frac{1}{\left|\{i : C(i) = k\}\right|} \sum_{i:C(i)=k} x_i$$

This is precisely the $K$-means algorithm!

# Clustering is riddled with questions and choices

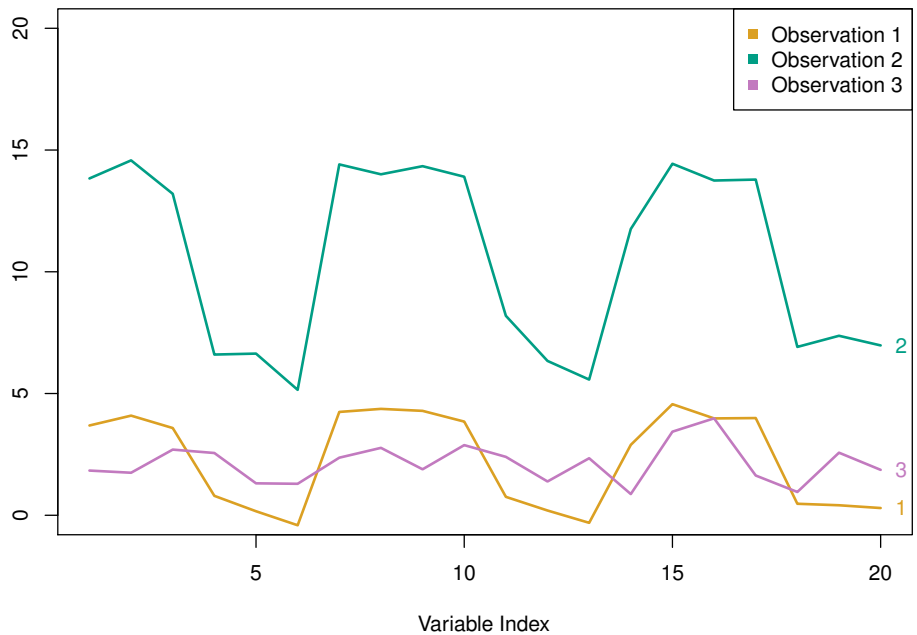
- Is clustering appropriate? i.e. Could a sample belong to more than one cluster?
  - Mixture models, soft clustering, topic models.
- Are the clusters robust?
  - Run the clustering on different random subsets of the data. Is the structure preserved?
  - Try different clustering algorithms. Are the conclusions consistent?
  - Most important: temper your conclusions.
- Should we scale the variables before doing the clustering.
  - Variables with larger variance have a larger effect on the Euclidean distance between two samples.

# Clustering is riddled with questions and choices

- Does Euclidean distance capture dissimilarity between samples?

  **Example:** Suppose that we want to cluster customers at a store for market segmentation.

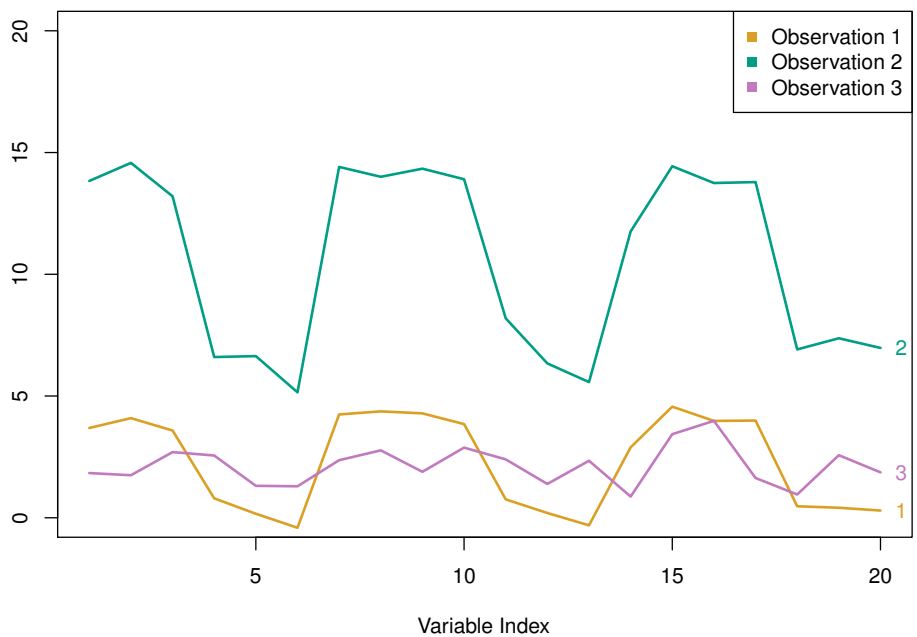
  - Samples are customers
  - Each variable corresponds to a specific product and measures the number of items bought by the customer during a year.



# Correlation distance

- Euclidean distance would cluster all customers who purchase few things (orange and purple).
- Perhaps we want to cluster customers who purchase *similar* things (orange and teal).
- Then, the **correlation distance** may be a more appropriate measure of dissimilarity between samples.

we need to scale the variables; each variable is a intensity, a number between 1-151, then we do not need to normalize/ scale them



# Fact of correlation distance

Correlation is defined by

$$\rho(x_i, x_{i'}) = \frac{\sum_j (x_{ij} - \bar{x}_i)(x_{i'j} - \bar{x}_{i'})}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2 \sum_j (x_{i'j} - \bar{x}_{i'})^2}},$$

where $\bar{x}_i$ = mean of obeservation $i$.

If observations are standardized:

$$x_{ij} \leftarrow \frac{x_{ij} - \bar{x}_i}{\sqrt{\sum_j (x_{ij} - \bar{x}_i)^2}},$$

then $2(1 - \rho(x_i, x_{i'})) = \sum_j (x_{ij} - x_{i'j}))^2$.