# Clustering & EM

## Lecture 16, 11/7/17

David Sontag

Massachusetts
Institute of
Technology

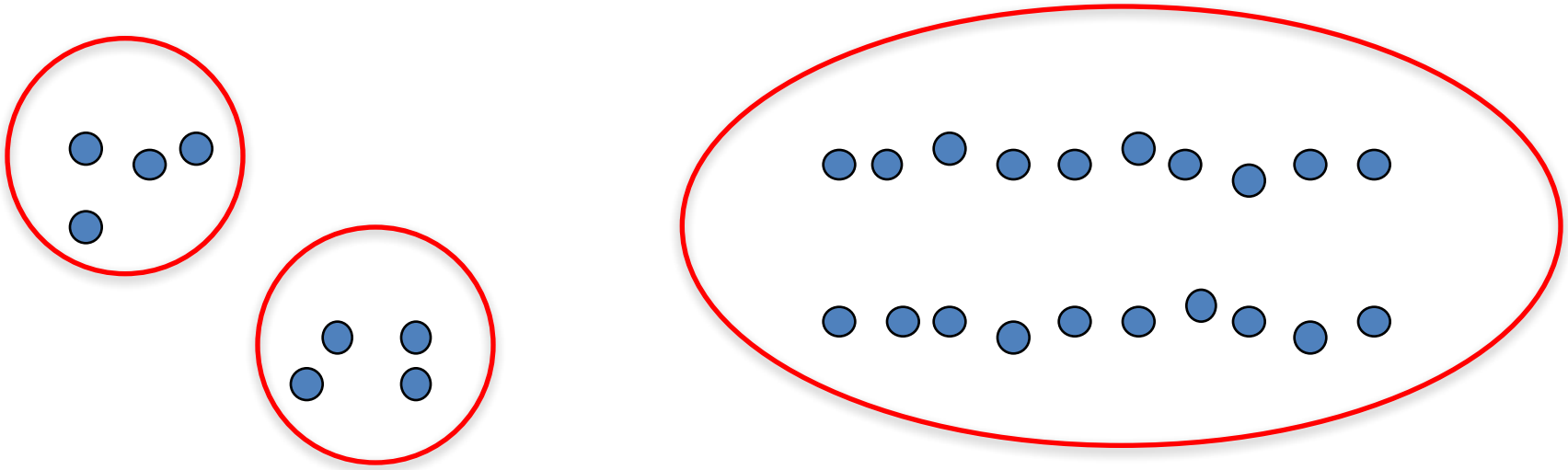
*Acknowledgement: several slides adapted from Luke Zettlemoyer, Vibhav Gogate, Carlos Guestrin, Andrew Moore, Dan Klein, Dan Weld*

# Course announcements

- HW3
  - Due next Tuesday, 11/14
  - At least as long as HW2; start now!
  - Reminder of policy: *you must write up your own solutions*
- No recitation this Friday

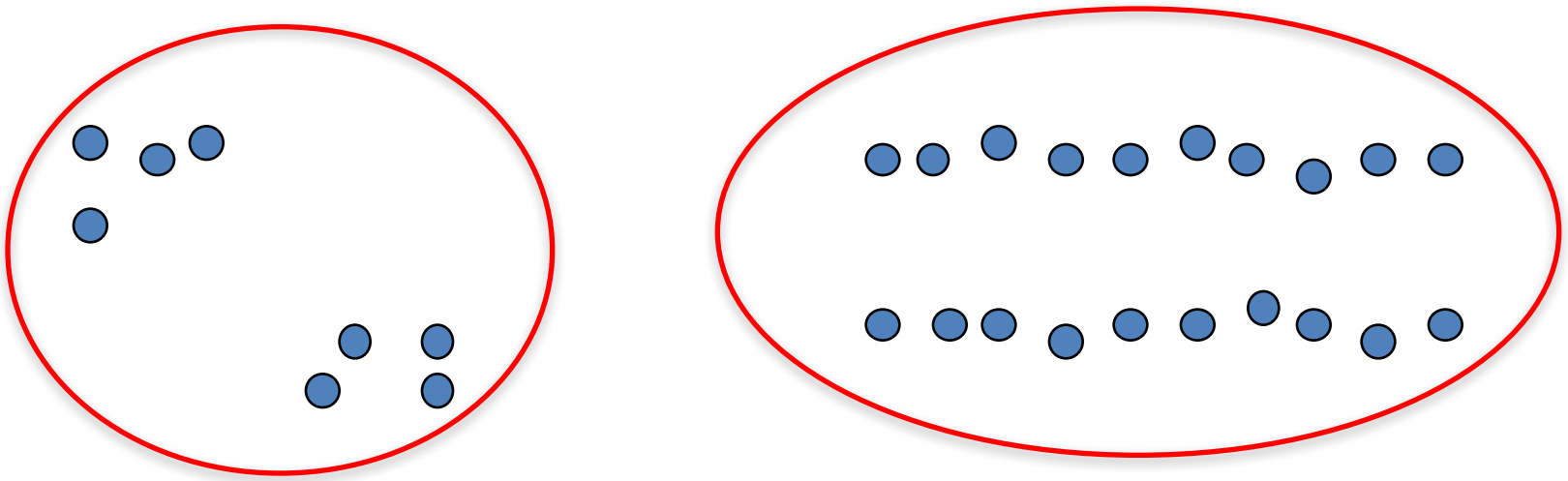
# Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



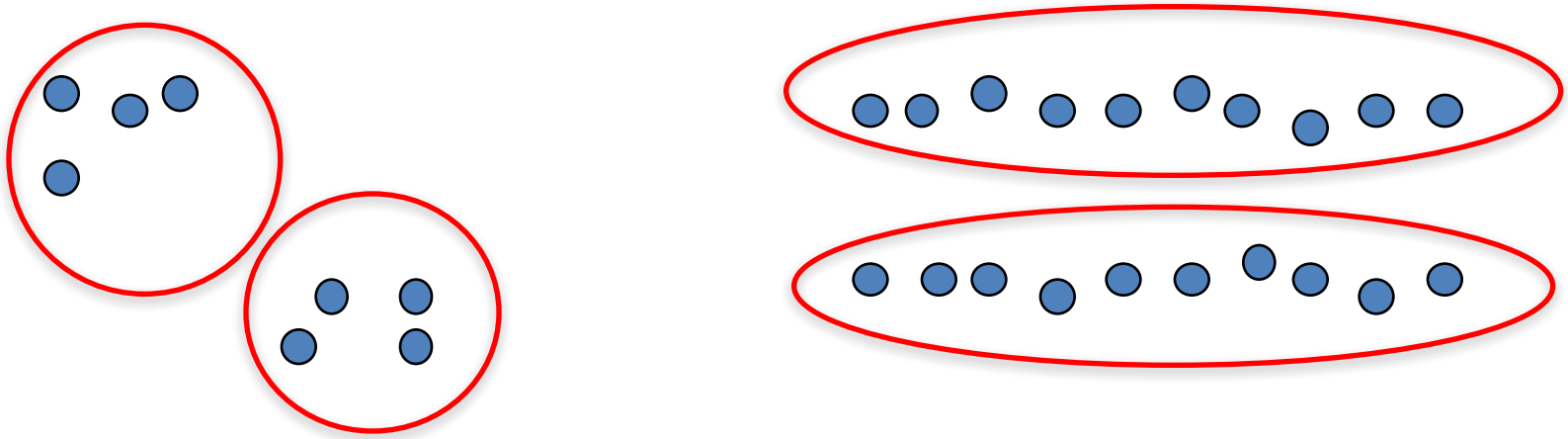
# Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



# Clustering

- Basic idea: group together similar instances
- Example: 2D point patterns



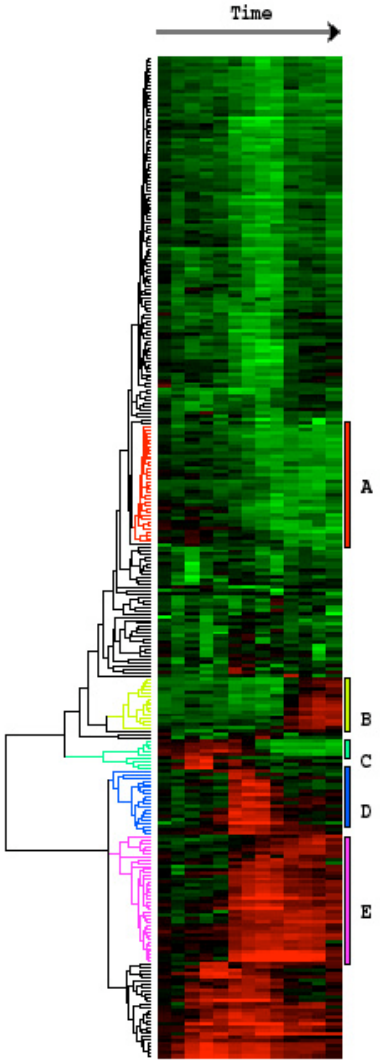
- What could "similar" mean?
  - One option: small Euclidean distance (squared)

$$\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered

# Clustering examples

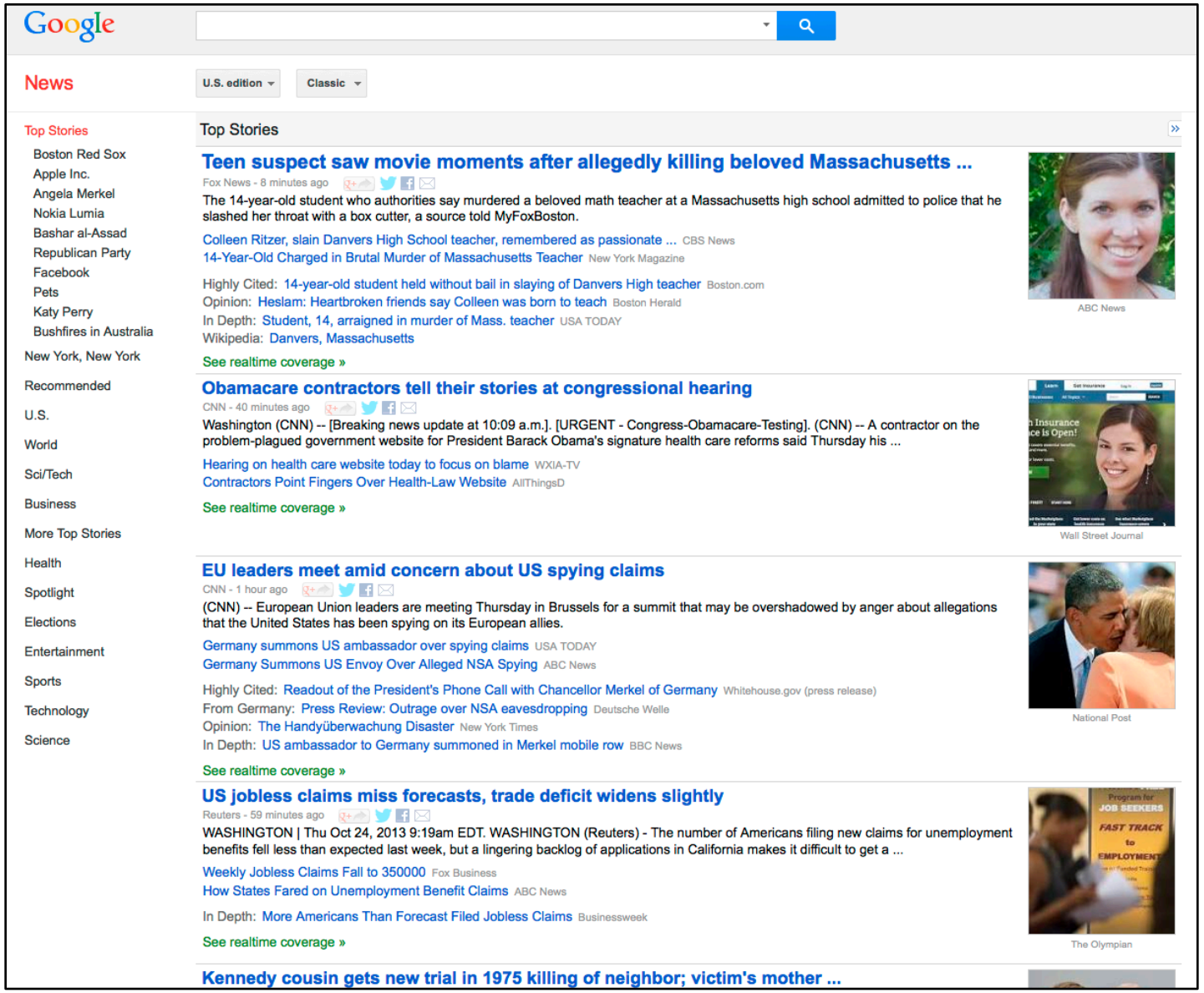
**Cluster gene expression data**




Eisen et al, PNAS 1998

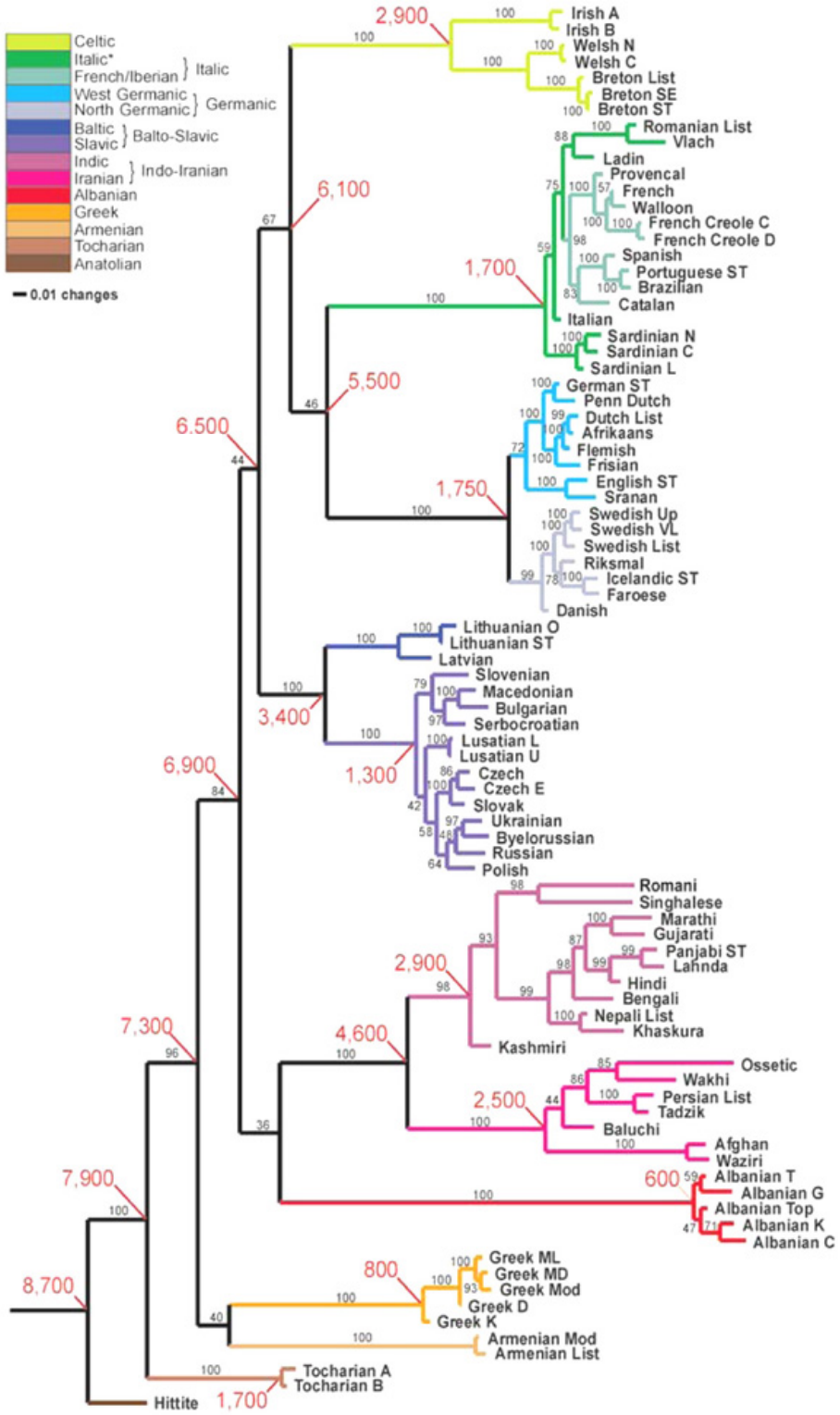
# Clustering examples

**Cluster news articles**



# Clustering examples

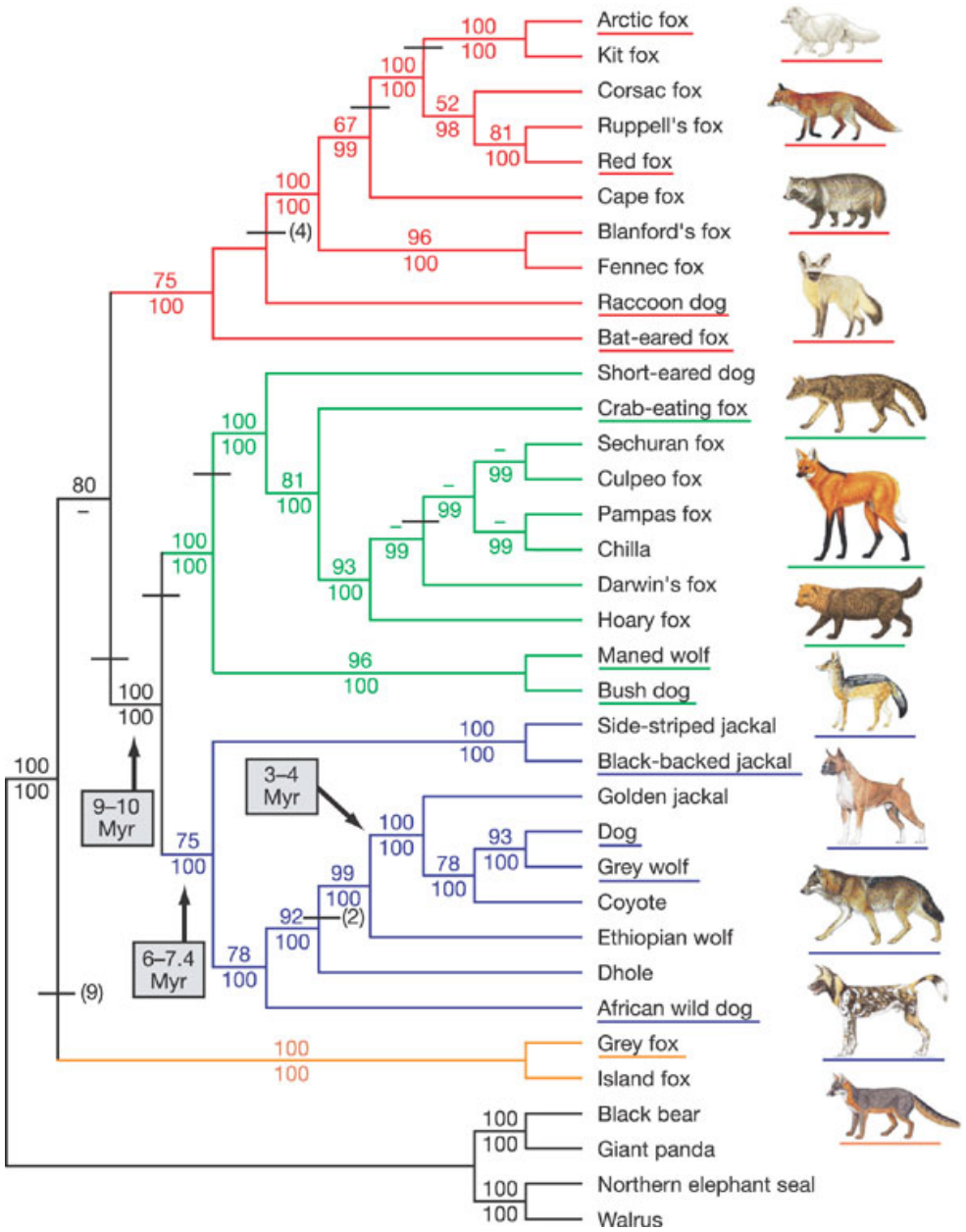
**Clustering languages**




[Image from dhushara.com]

# Clustering examples

**Clustering species ("phylogeny")**



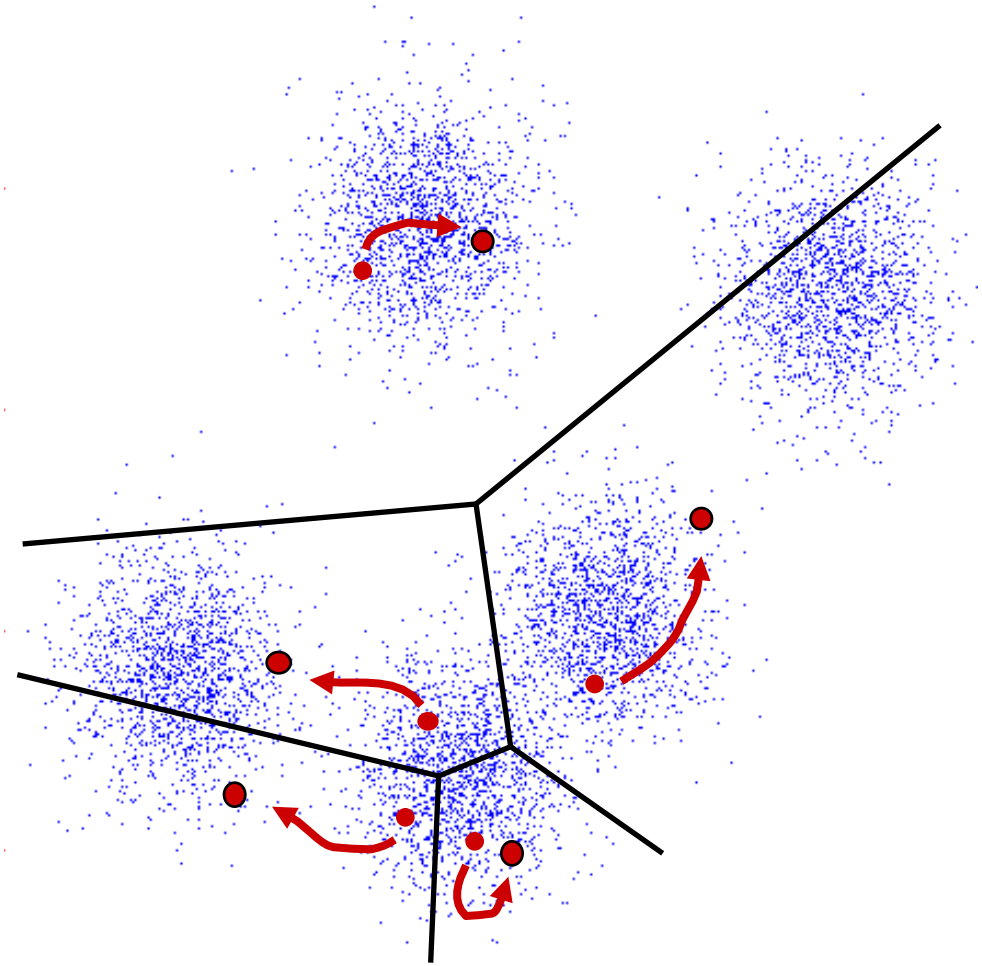

[Lindblad-Toh et al., Nature 2005]

# Unsupervised learning / clustering algorithms

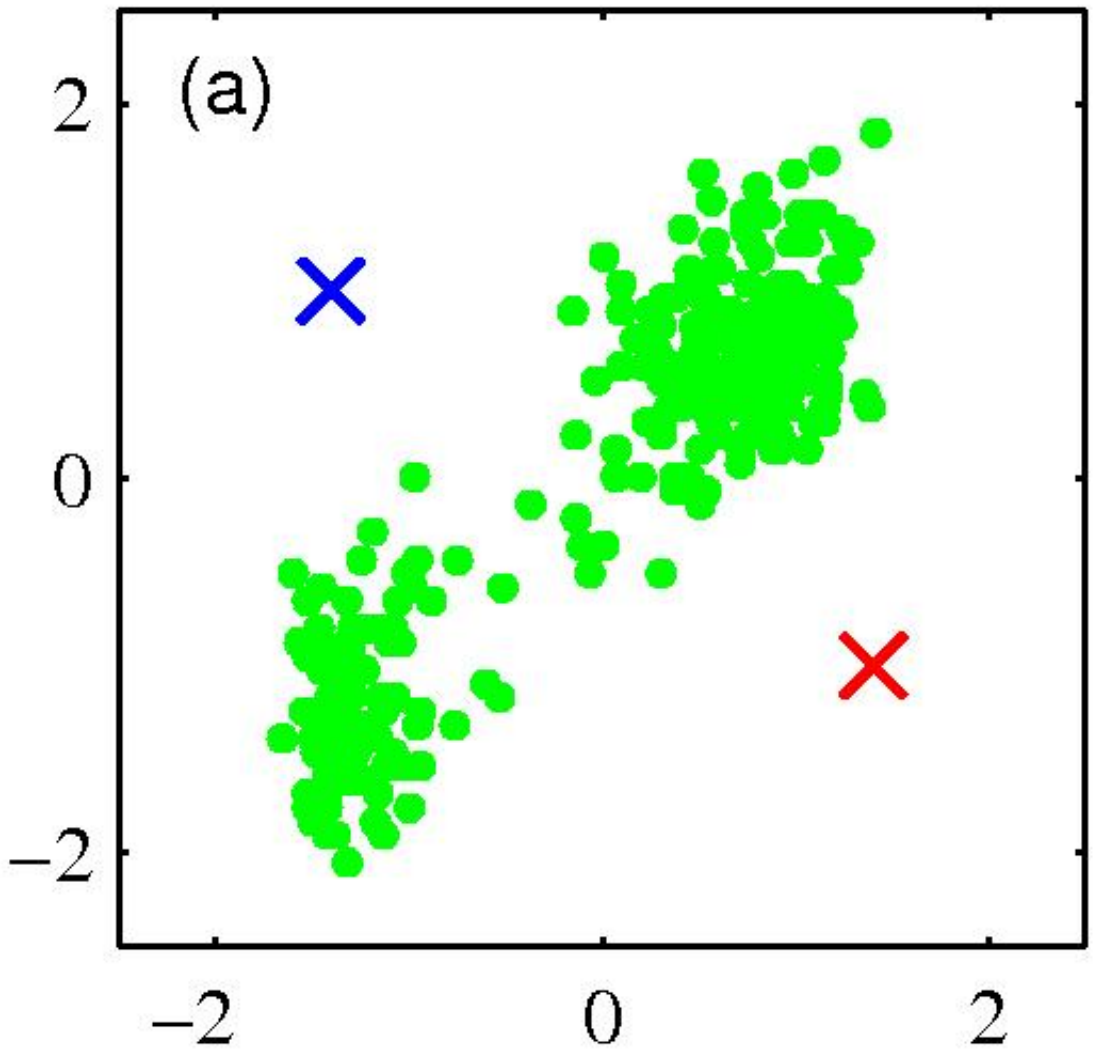
- **K-means**
  - Find natural groupings of data according to a distance function (e.g. Euclidean distance)
  - Simplest possible parametric method
- Non-parametric methods
  - Hierarchical/agglomerative
  - Spectral clustering
- Parametric methods (+model)
  - Gaussian mixture models
  - *Factor analysis, topic models, … (future lectures)*

# K-Means

- An iterative clustering algorithm
  - Initialize: Pick *K* random points as cluster centers
  - Alternate:
    1. Assign data points to closest cluster center
    2. Change the cluster center to the average of its assigned points
  - Stop when no points' assignments change



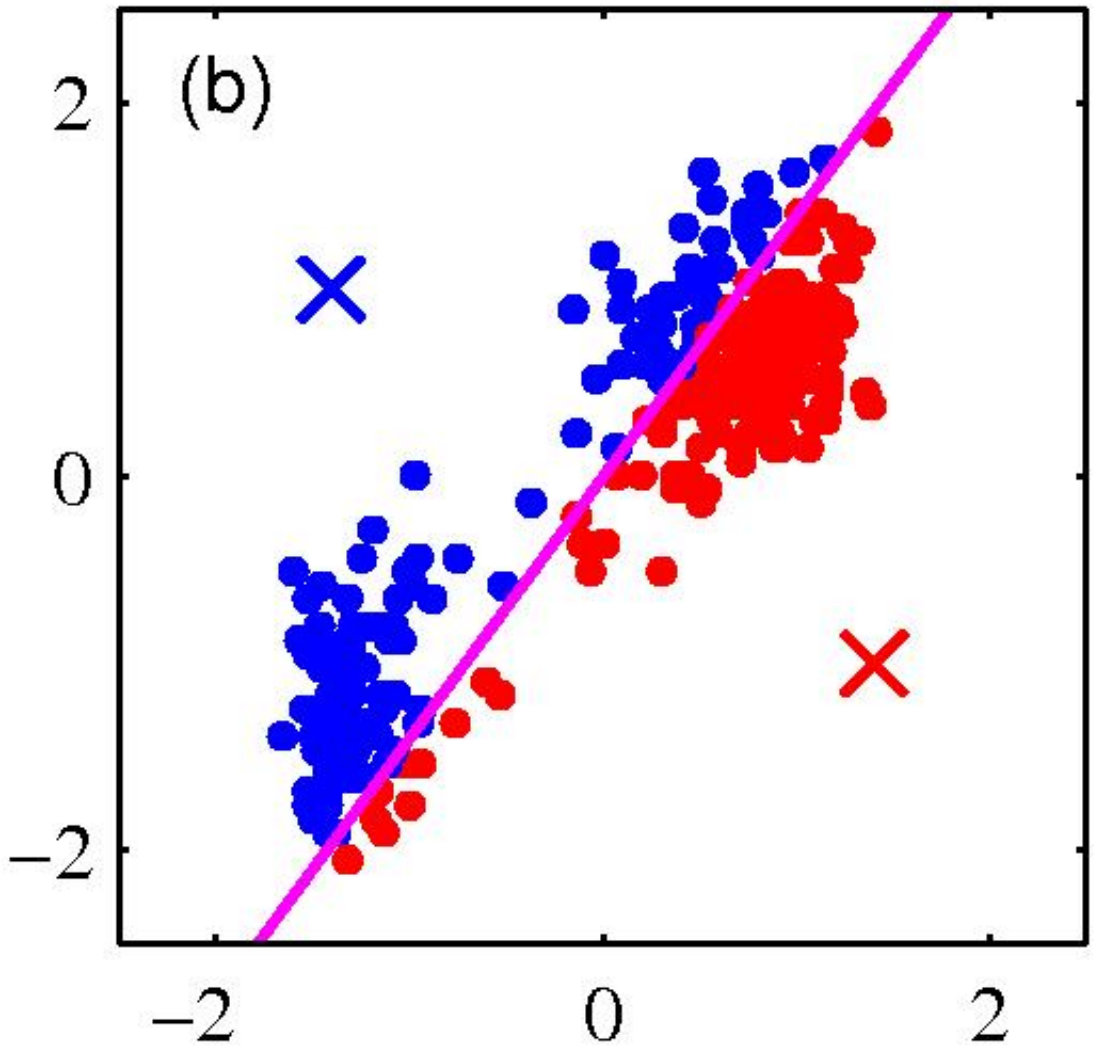
# K-means clustering: Example




- Pick *K* random points as cluster centers (means)

Shown here for *K*=2

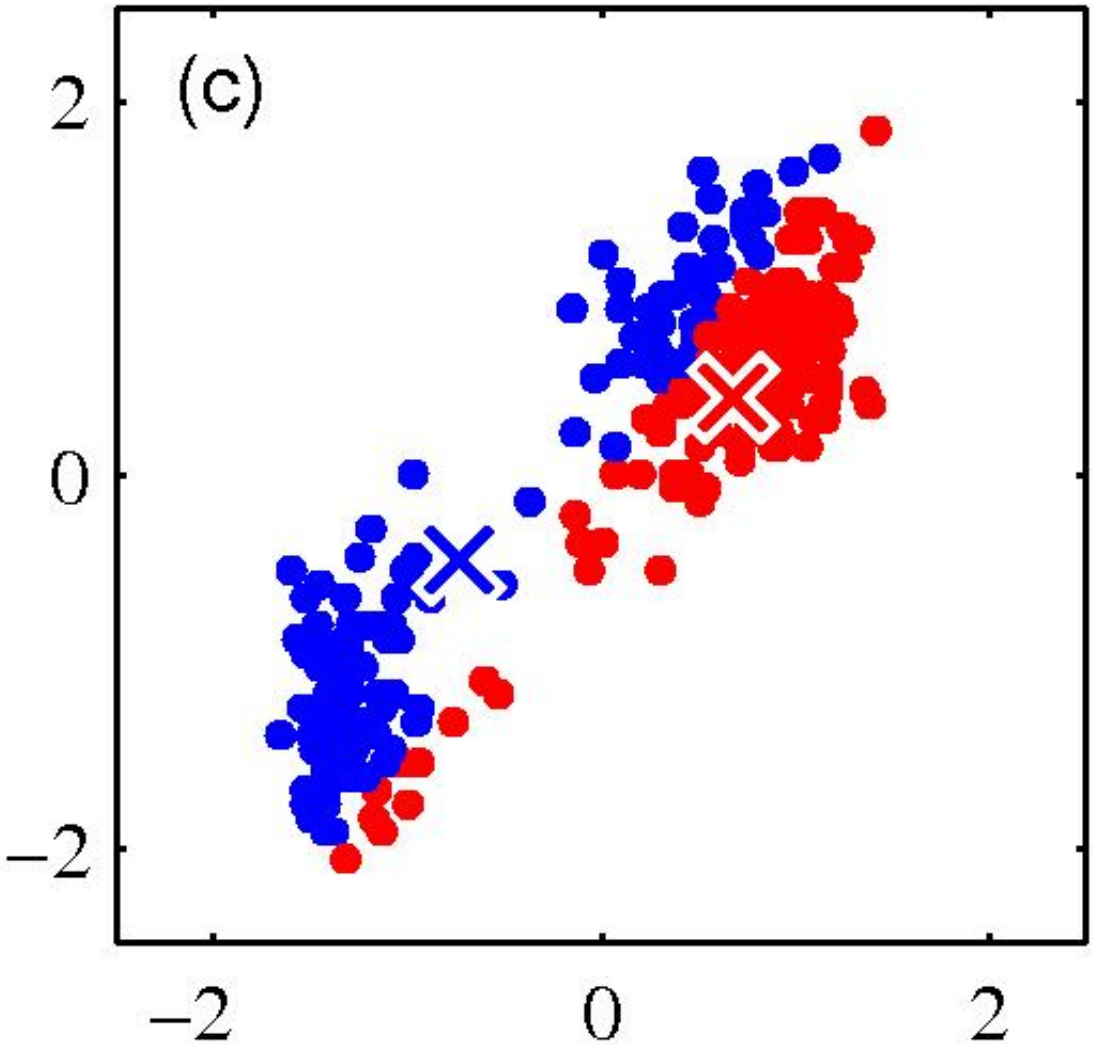
# K-means clustering: Example




Iterative Step 1

- Assign data points to closest cluster center

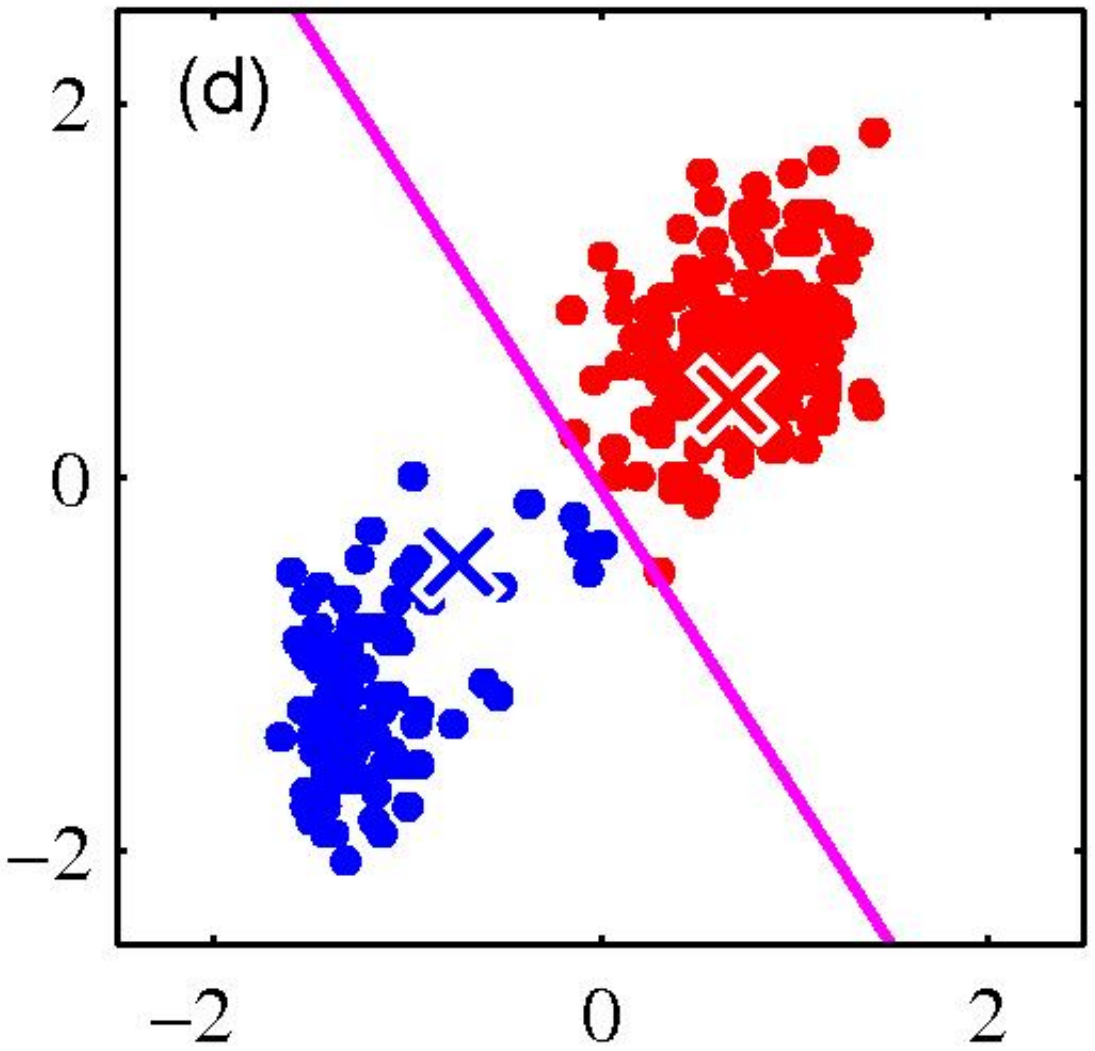
# K-means clustering: Example




Iterative Step 2

- Change the cluster center to the average of the assigned points

# K-means clustering: Example




- Repeat until convergence

# Properties of K-means **algorithm**

- Guaranteed to converge in a finite number of iterations

- Running time per iteration:
  1. Assign data points to closest cluster center

     O(KN) time

  2. Change the cluster center to the average of its assigned points

     O(N)

# Kmeans Convergence

**Objective**

$$\min_{\mu}\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$

1. Fix $\mu$, optimize *C:*

***Step 1 of kmeans***

$$\min_{C}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2=\min_{c}\sum_{i}^{n}\left|x_i-\mu_{x_i}\right|^2$$

2. Fix *C,* optimize $\mu$:

$$\min_{\mu}\sum_{i=1}^{k}\sum_{x\in C_i}|x-\mu_i|^2$$

   – Take partial derivative ~~of~~ with respect to $\mu_i$ and set to zero, we have

$$\mu_i=\frac{1}{|C_i|}\sum_{x\in C_i}x$$

***Step 2 of kmeans***

Kmeans takes an alternating optimization approach, each step is guaranteed to decrease the objective – thus guaranteed to converge

[Slide from Alan Fern]

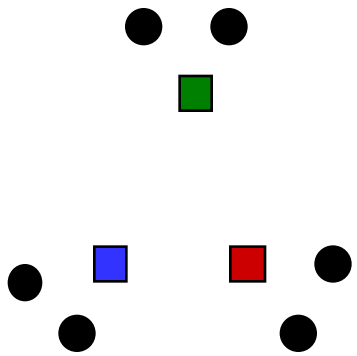
# Example: Vector quantization

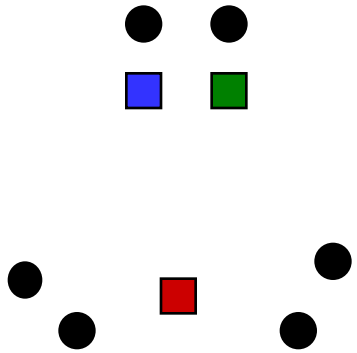
**FIGURE 14.9.** *Sir Ronald A. Fisher (*1890 – 1962*) was one of the founders of modern day statistics, to whom we owe maximum-likelihood, sufficiency, and many other fundamental concepts. The image on the left is a* 1024×1024 *grayscale image at* 8 *bits per pixel. The center image is the result of* $2 \times 2$ *block VQ, using* 200 *code vectors, with a compression rate of* 1.9 *bits/pixel. The right image uses only four code vectors, with a compression rate of* 0.50 *bits/pixel*

[Figure from Hastie *et al.* book]

# Initialization

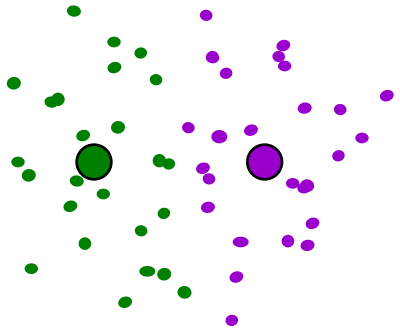
- K-means **algorithm** is a heuristic
  - Requires initial means
  - It does matter what you pick!
  - What can go wrong?
  - Various schemes for preventing this kind of thing: variance-based split / merge, initialization heuristics



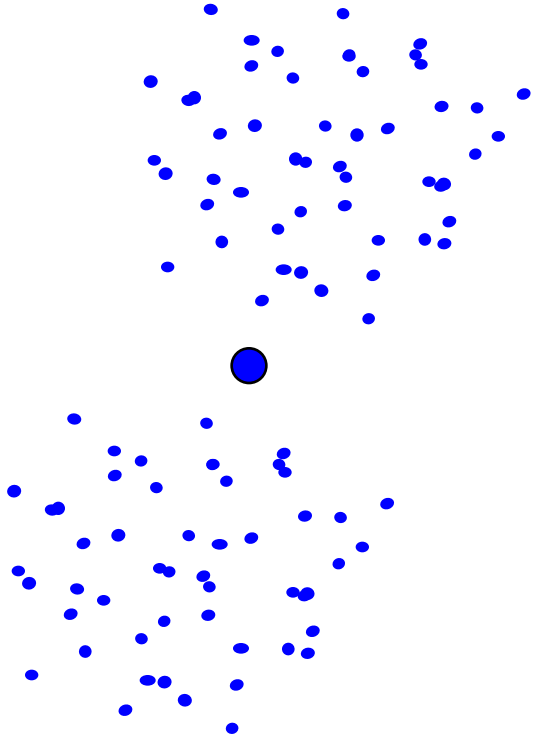


# K-Means Getting Stuck

A local optimum:

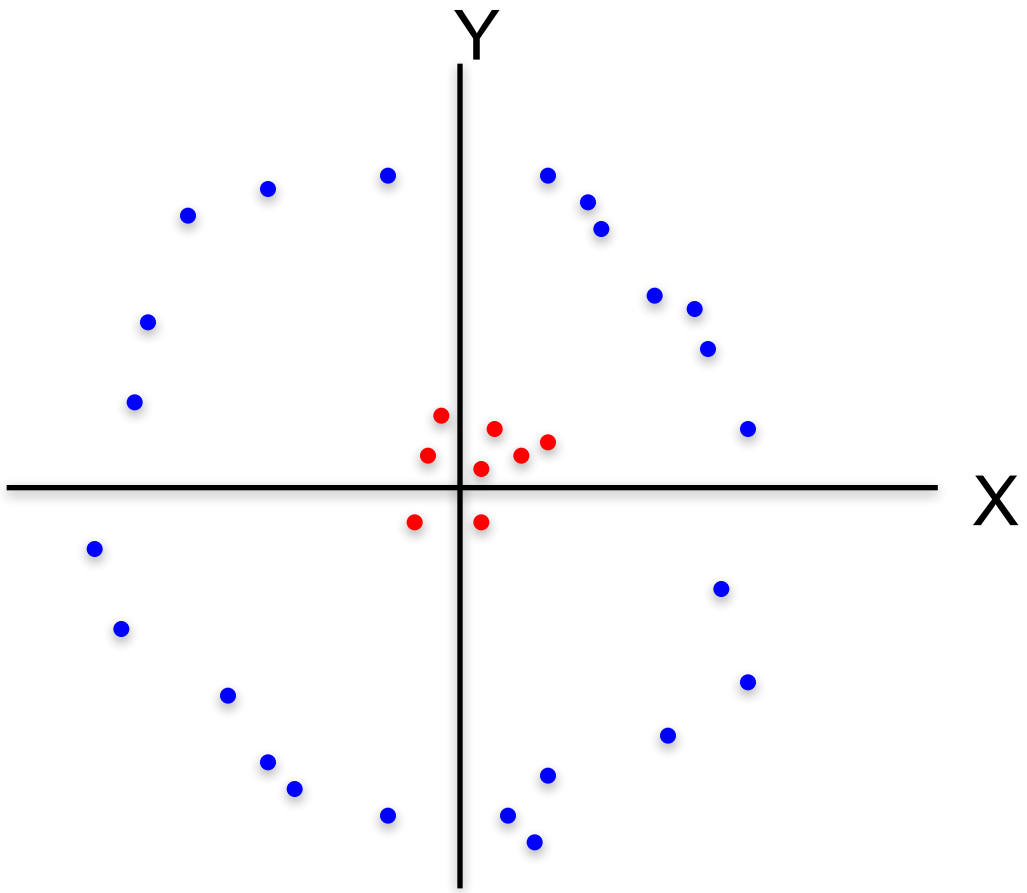


Would be better to have
one cluster here



… and two clusters here

# K-means not able to properly cluster




Euclidean distance is insufficient

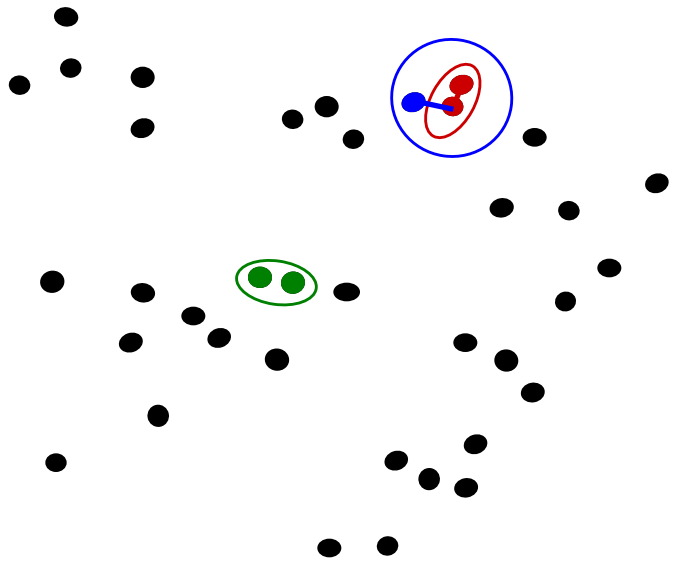
Can we find the data manifold?

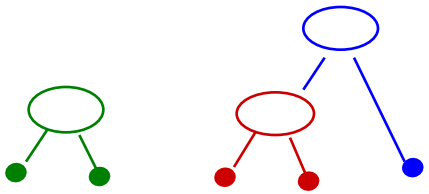
# Unsupervised learning / clustering algorithms

- K-means
  - Find natural groupings of data according to a distance function (e.g. Euclidean distance)
  - Simplest possible parametric method
- **Non-parametric methods**
  - Hierarchical/agglomerative clustering
  - Spectral clustering
- Parametric methods (+model)
  - Gaussian mixture models
  - *Factor analysis, topic models, … (future lectures)*

# Agglomerative Clustering

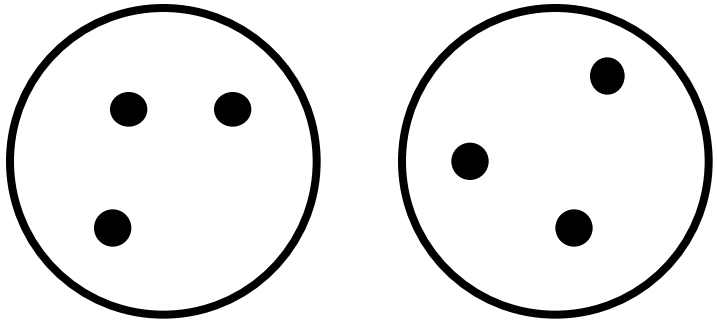
- Agglomerative clustering:
  - First merge very similar instances
  - Incrementally build larger clusters out of smaller clusters
- Algorithm:
  - Maintain a set of clusters
  - Initially, each instance in its own cluster
  - Repeat:
    - Pick the two closest clusters
    - Merge them into a new cluster
    - Stop when there's only one cluster left
- Produces not one clustering, but a family of clusterings represented by a dendrogram





# Agglomerative Clustering

- How should we define “closest” for clusters with multiple elements?

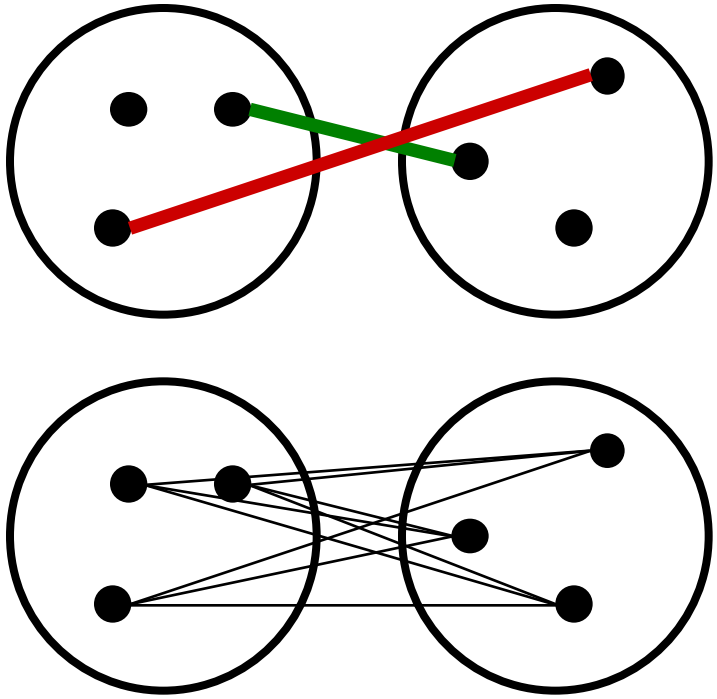


# Agglomerative Clustering

- How should we define "closest" for clusters with multiple elements?

- Many options:
  - Closest pair
    (single-link clustering)
  - Farthest pair
    (complete-link clustering)
  - Average of all pairs

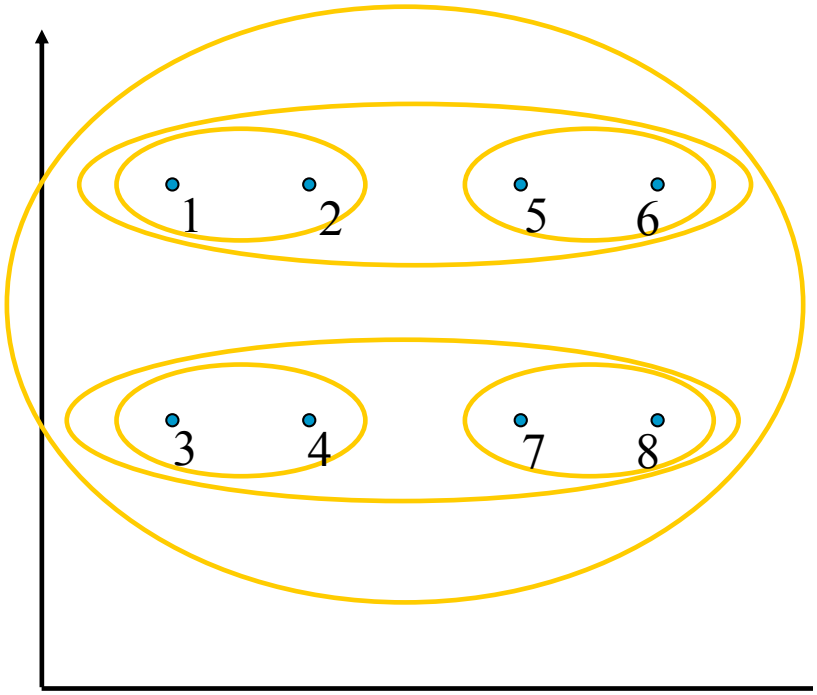
- Different choices create different clustering behaviors



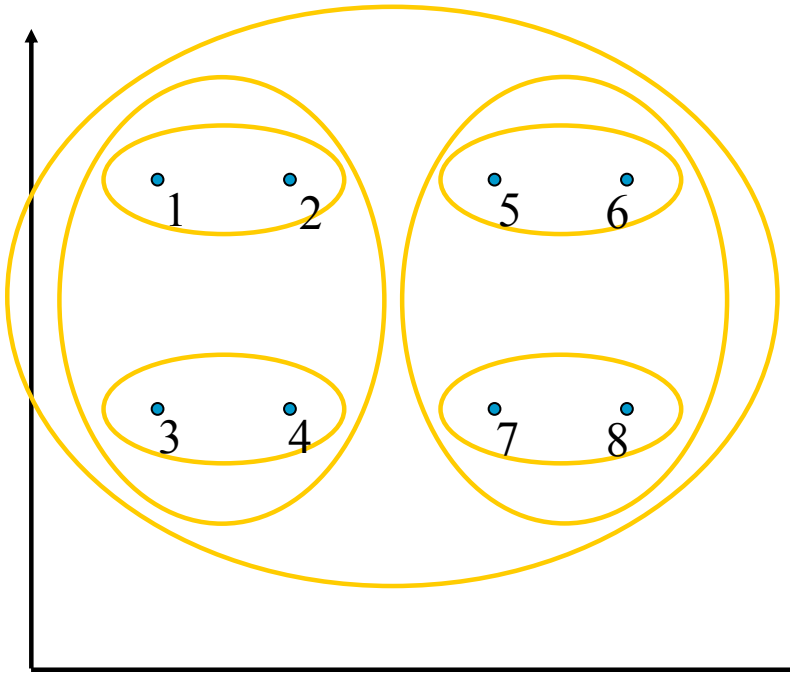
# Agglomerative Clustering

- How should we define "closest" for clusters with multiple elements?

Closest pair
(single-link clustering)

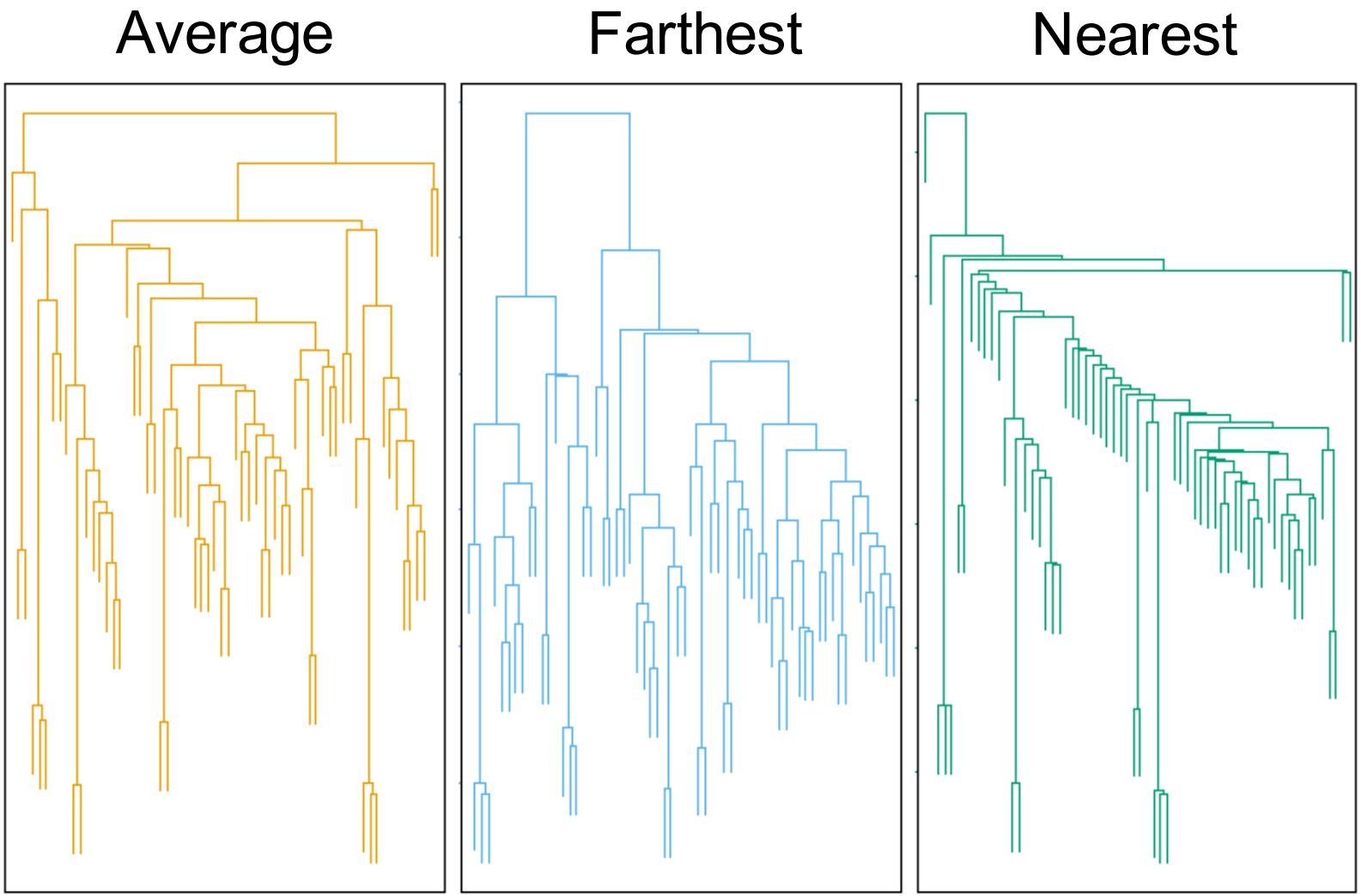



Farthest pair
(complete-link clustering)




[Figures from Thorsten Joachims]

# Clustering Behavior

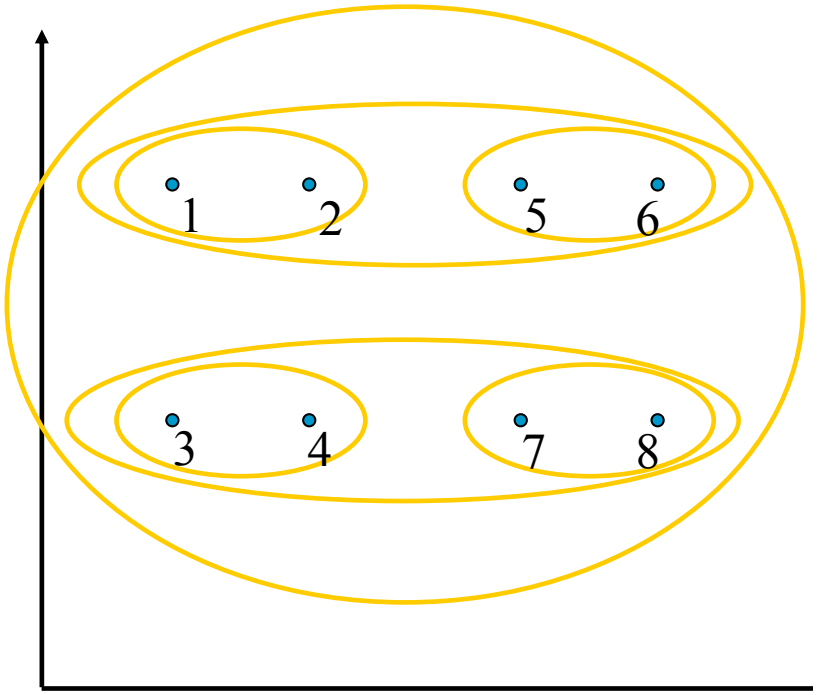



Mouse tumor data from [Hastie *et al.*]

# Agglomerative Clustering

When can this be expected to work?

Closest pair
(single-link clustering)




**Strong separation** property:
*All points are more similar to points in their own cluster than to any points in any other cluster*

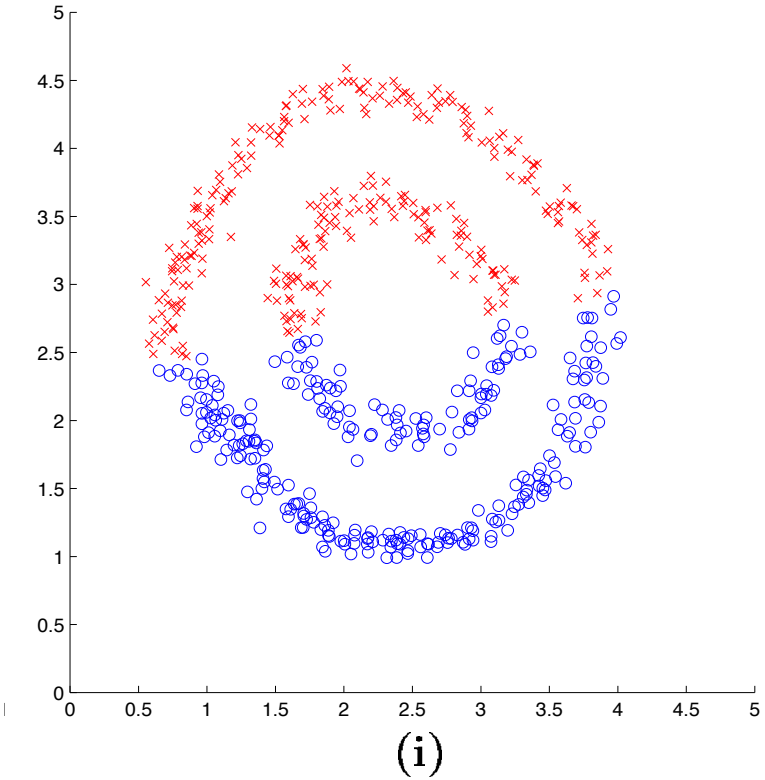
Then, the true clustering corresponds to some **pruning** of the tree obtained by single-link clustering!

Slightly weaker (stability) conditions are solved by average-link clustering

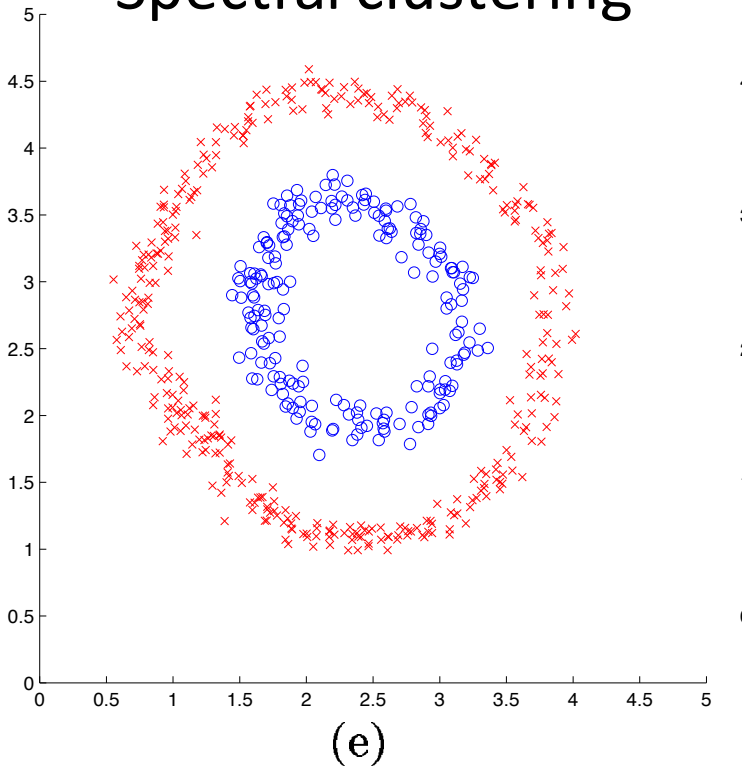
(Balcan et al., 2008)

# Spectral clustering

K-means

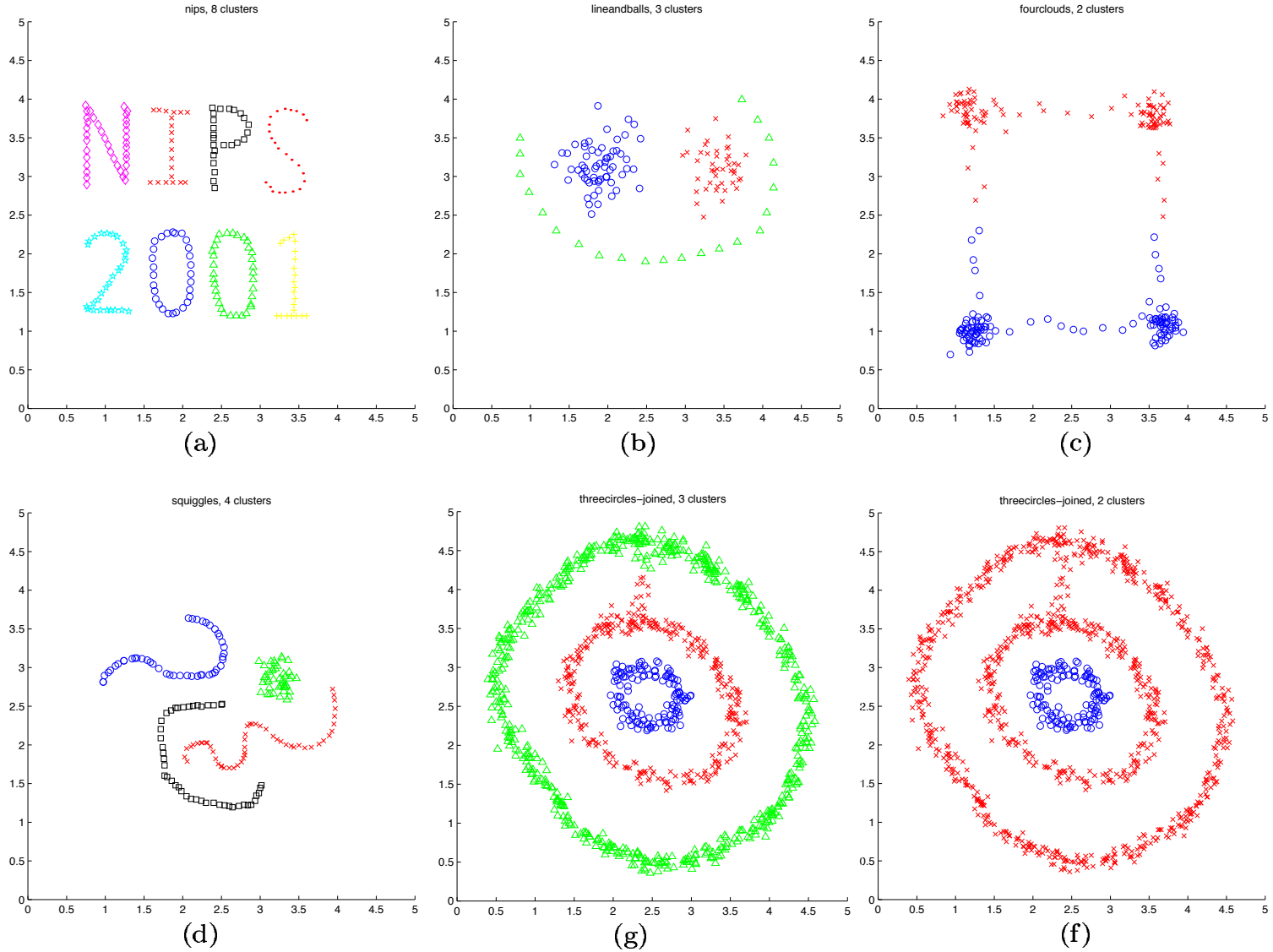



Spectral clustering




[Shi & Malik '00; Ng, Jordan, Weiss NIPS '01]

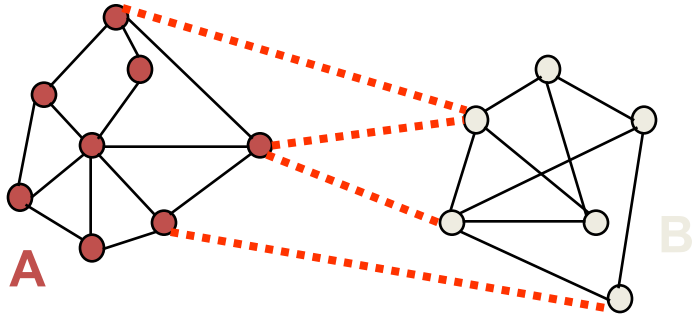
# Spectral clustering




[Figures from Ng, Jordan, Weiss NIPS '01]

# Spectral clustering – key idea

- Create a *weighted graph* on the data points by connecting nearby points:




$$w(i,j) = \exp \frac{-||x_i - x_j||^2}{\sigma^2}$$

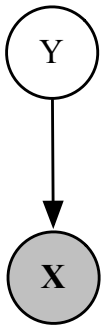
- Find a *balanced partition* of the graph (called a "normalized cut"), simultaneously attempting to:
  - Minimize weight of cut edges
  - Have large components (i.e., many nodes in each cluster)

# Unsupervised learning / clustering algorithms

- K-means
  - Find natural groupings of data according to a distance function (e.g. Euclidean distance)
  - Simplest possible parametric method
- Non-parametric methods
  - Hierarchical/agglomerative clustering
  - Spectral clustering
- **Parametric methods (+model)**
  - Gaussian mixture models
  - *Factor analysis, topic models, … (future lectures)*

# Clustering using probabilistic models

- General strategy:
  - Give a model of how the data might have been *generated*
  - Part of the model specifies cluster membership
  - Learn this model from data (***now with latent variables***)




Tell a *generative story* for data

$P(Y)P(X|Y)$

| Y | $X_1$ | $X_2$ |
|---|---|---|
| ?? | 0.1 | 2.1 |
| ?? | 0.5 | -1.1 |
| ?? | 0.0 | 3.0 |
| ?? | -0.1 | -2.0 |
| ?? | 0.2 | 1.5 |
| … | … | … |

# Clustering using probabilistic models

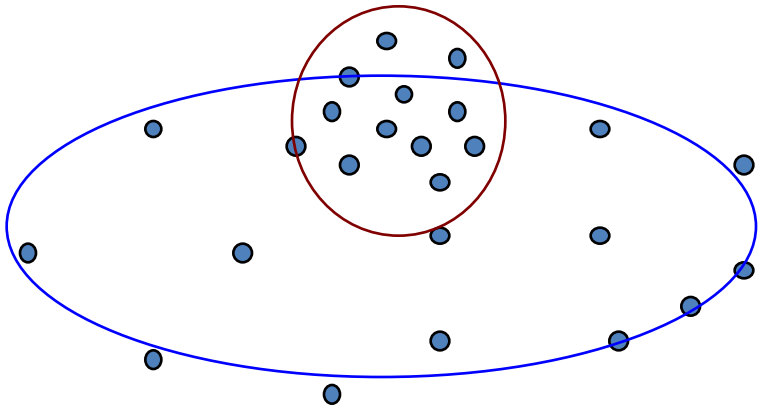
- General strategy:
  - Give a model of how the data might have been *generated*
  - Part of the model specifies cluster membership
  - Learn this model from data (*now with latent variables*)
- Advantages:
  - Naturally handles heterogeneous or missing data

| Y | $X_1$ | $X_2$ | $X_3$ |
|---|---|---|---|
| ?? | 0 | 2.123 | 3 |
| ?? | 1 | -1.101 | 5 |
| ?? | ? | 3.01 | 2 |

$$y \sim \text{Mult}(\pi)$$
$$x_1 \sim \text{Bernoulli}(p_y)$$
$$x_2 \sim \text{Gaussian}(\mu_y)$$
$$x_3 \sim \text{Poisson}(\lambda_y)$$

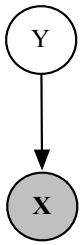
# Clustering using probabilistic models

- General strategy:
  - Give a model of how the data might have been *generated*
  - Part of the model specifies cluster membership
  - Learn this model from data (*now with latent variables*)
- Advantages:
  - Naturally handles heterogeneous or missing data
  - Allows for overlapping clusters



# Clustering using probabilistic models

- General strategy:
  - Give a model of how the data might have been *generated*
  - Part of the model specifies cluster membership
  - Learn this model from data (*now with latent variables*)
- Advantages:
  - Naturally handles heterogeneous or missing data
  - Allows for overlapping clusters
  - Can be jointly optimized as part of semi-supervised learning



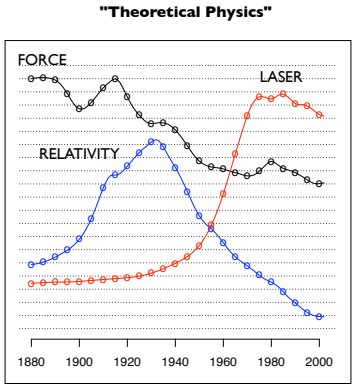

$$\underbrace{\sum_{(\mathbf{x},y)\sim\widetilde{p}_l} \mathcal{L}(\mathbf{x},y) + \sum_{\mathbf{x}\sim\widetilde{p}_u} \mathcal{U}(\mathbf{x})}_{\text{Likelihood of observed data}} + \underbrace{\alpha \cdot \mathbb{E}_{\widetilde{p}_l(\mathbf{x},y)}\left[-\log q_\phi(y|\mathbf{x})\right]}_{\text{Predictive performance (for data points with } y \text{ observed)}}$$

(e.g., Kingma et al., NIPS '14)

# Clustering using probabilistic models

- General strategy:
  - Give a model of how the data might have been *generated*
  - Part of the model specifies cluster membership
  - Learn this model from data (*now with latent variables*)
- Advantages:
  - Naturally handles heterogeneous or missing data
  - Allows for overlapping clusters
  - Can be jointly optimized as part of semi-supervised learning
  - Can be easily extended to allow for multiple cluster memberships, time-varying clusters, etc.

(e.g., Blei & Lafferty, ICML '06)



# Gaussian Mixture Models

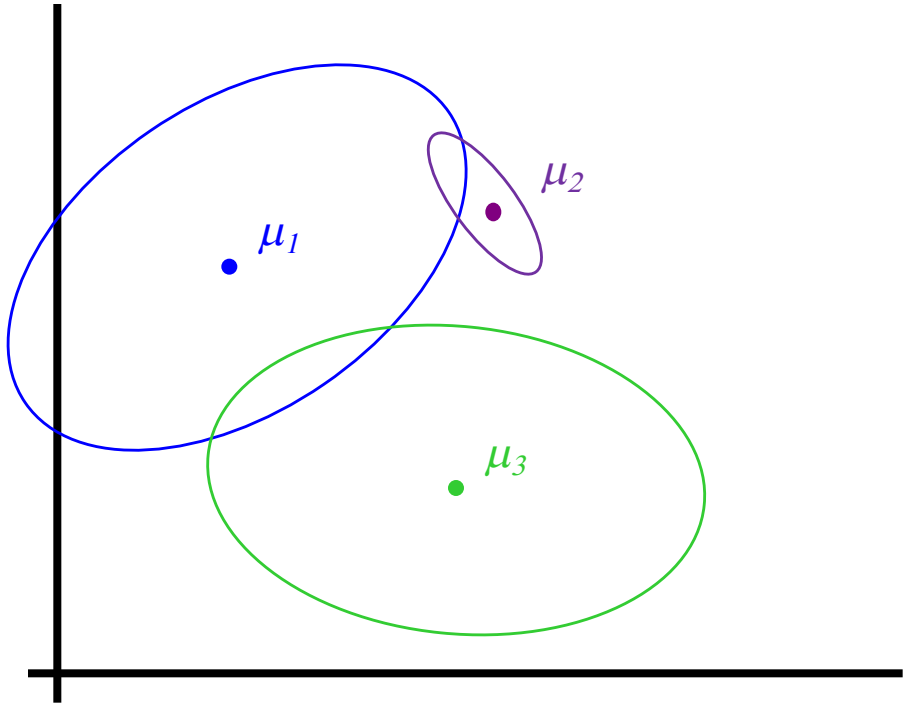
- P(Y): There are k components
- P(X|Y): Each component generates data from a **multivariate Gaussian** with mean $\mu_i$ and covariance matrix $\Sigma_i$

Each data point assumed to have been sampled from a ***generative process*:**

1. Choose component i with probability *P(y=i)* *[Multinomial]*
2. Generate datapoint ~ N($m_i$, $\Sigma_i$)

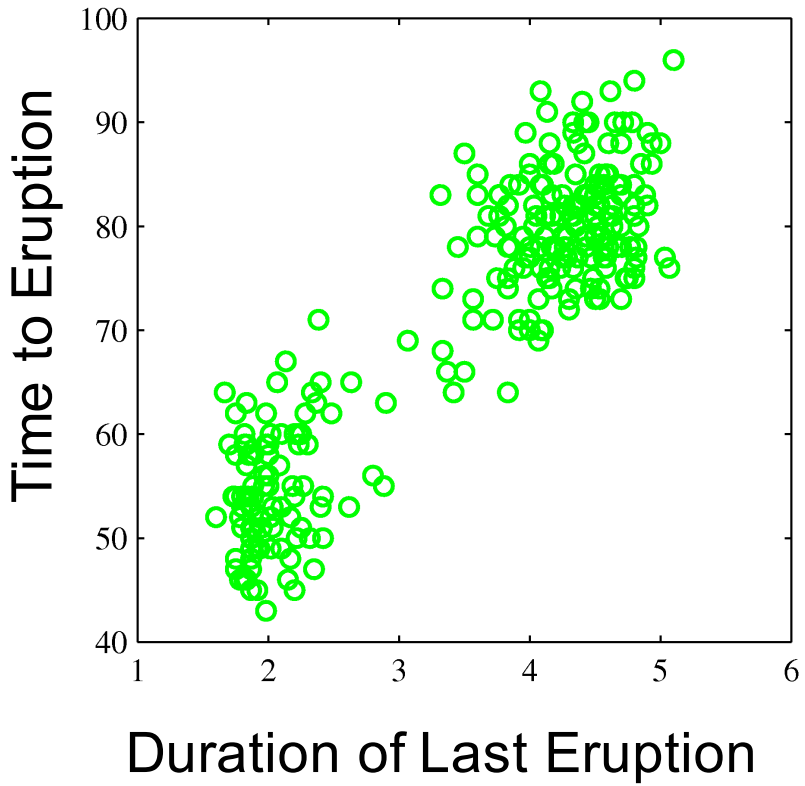
$$P(X = \mathbf{x}_j \mid Y = i) = \frac{1}{(2\pi)^{m/2} \| \Sigma_i \|^{1/2}} \exp\left[ -\frac{1}{2} \left( \mathbf{x}_j - \mu_i \right)^T \Sigma_i^{-1} \left( \mathbf{x}_j - \mu_i \right) \right]$$

***By fitting this model (unsupervised learning), we can learn new insights about the data***



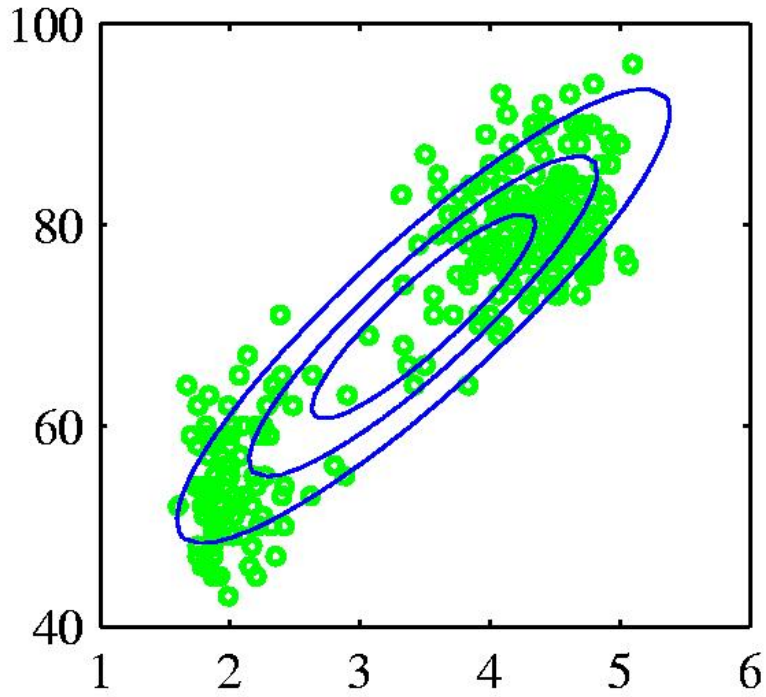
# Modelling eruption of geysers

## Old Faithful Data Set

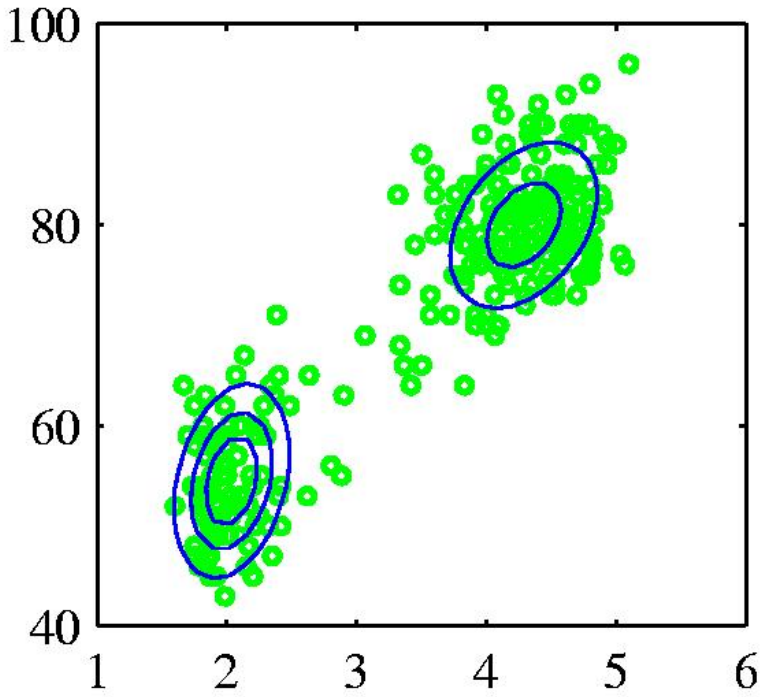


# Modelling eruption of geysers

## Old Faithful Data Set




Single Gaussian




Mixture of two Gaussians

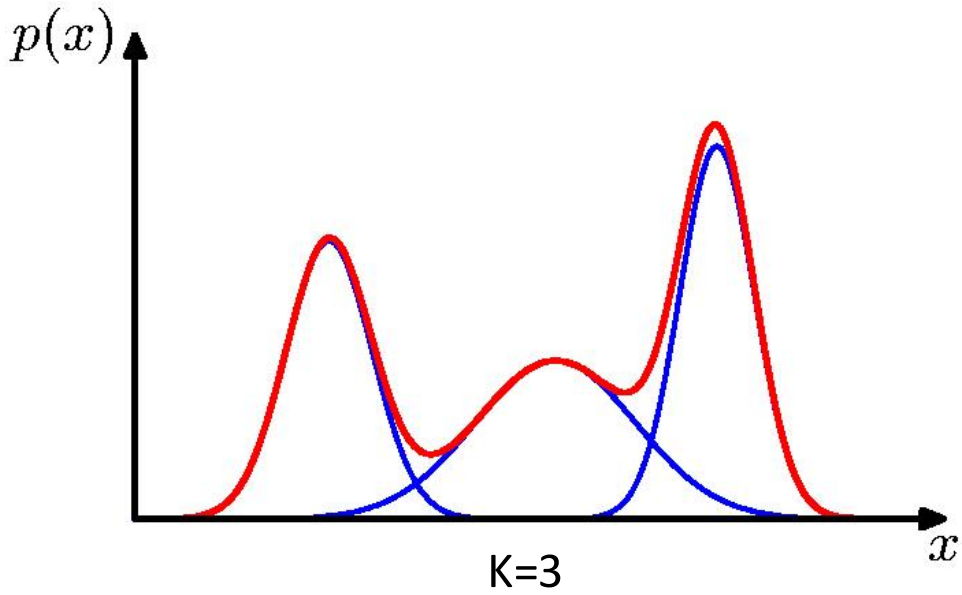
# Marginal distribution for mixtures of Gaussians

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Component

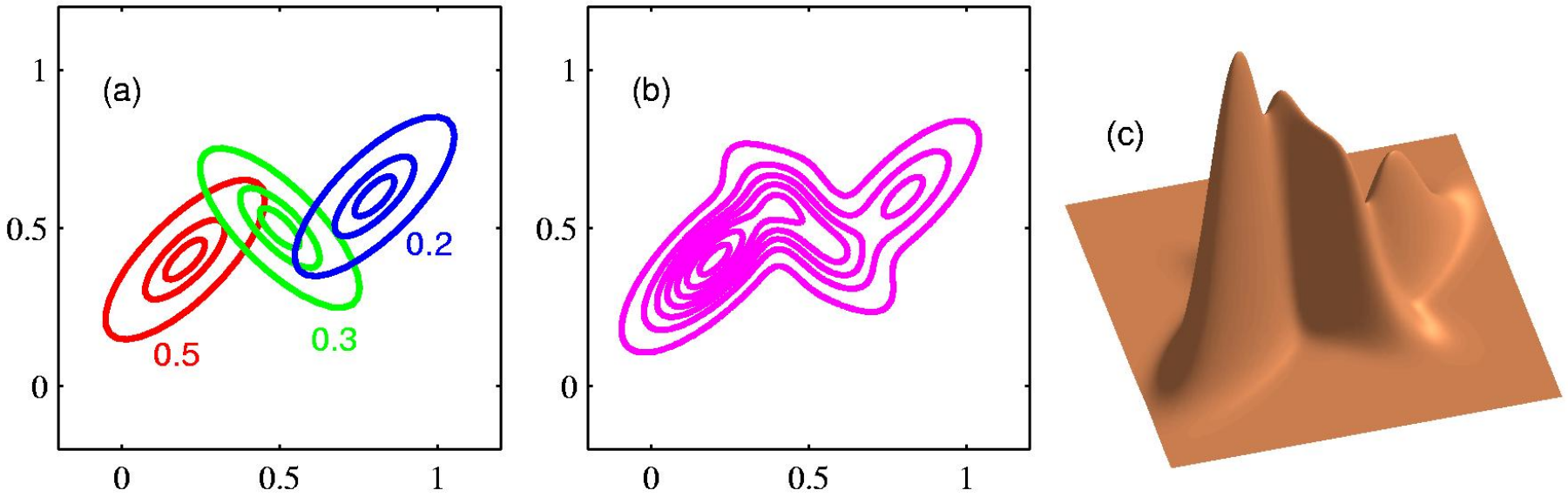
Mixing coefficient

$$\forall k : \pi_k \geqslant 0 \qquad \sum_{k=1}^{K} \pi_k = 1$$



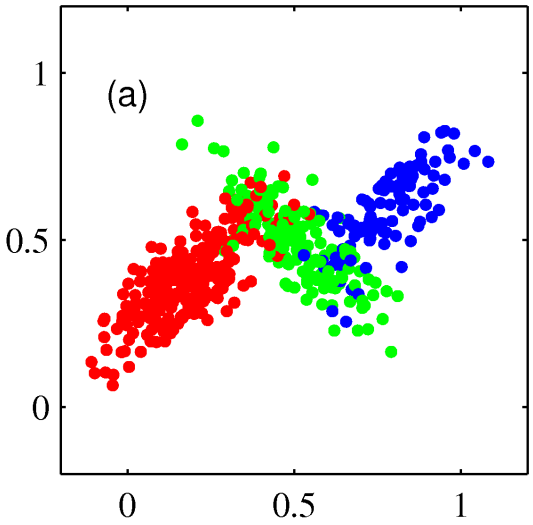

K=3

# Marginal distribution for mixtures of Gaussians

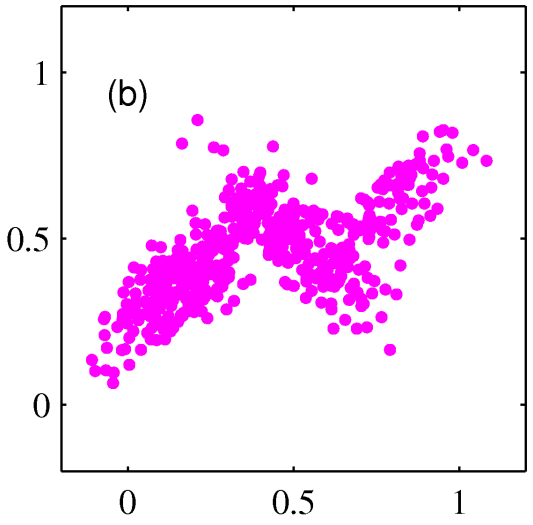


# Learning mixtures of Gaussians

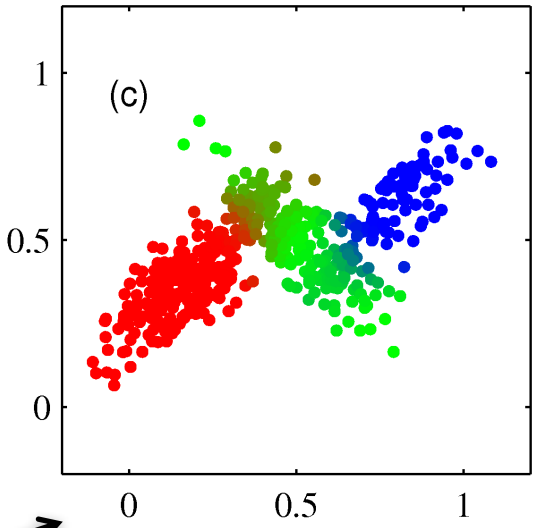
Original data (hypothesized)




Observed data (y missing)




Inferred y's (learned model)




Shown is the *posterior probability* that a point was generated from i$^{th}$ Gaussian: $\Pr(Y = i \mid x)$

# ML estimation in supervised setting

- Univariate Gaussian

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu})^2$$

- Mixture of Multivariate Gaussians

ML estimate for each of the Multivariate Gaussians is given by:

$$\mu^k_{ML} = \frac{1}{n}\sum_{j=1}^{n} x_n \qquad \Sigma^k_{ML} = \frac{1}{n}\sum_{j=1}^{n}\left(\mathbf{x}_j - \mu^k_{ML}\right)\left(\mathbf{x}_j - \mu^k_{ML}\right)^T$$

**Just sums over *x* generated from the *k*'th Gaussian**

# What about with unobserved data?

- Maximize ***marginal likelihood***:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax} \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j)$
- Almost always a hard problem!
  - Usually no closed form solution
  - Even when lgP(X,Y) is convex, lgP(X) generally isn't…
  - Many local optima

# The EM Algorithm

- A clever method for maximizing marginal likelihood:
  - $\text{argmax}_\theta \prod_j P(x_j) = \text{argmax}_\theta \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j)$
  - Based on coordinate descent. Easy to implement (eg, no line search, learning rates, etc.)
- Alternate between two steps:
  1. Compute an expectation
  2. Compute a maximization
- Not magic: ***still optimizing a non-convex function with lots of local optima***
  - The computations are just easier (often, significantly so)

# EM: Two Easy Steps

**Objective:** $\text{argmax}_\theta \lg \prod_j \sum_{k=1}^{K} P(Y_j=k, x_j ; \theta) = \sum_j \lg \sum_{k=1}^{K} P(Y_j=k, x_j; \theta)$

**Data: $\{x_j \mid j=1 .. n\}$**

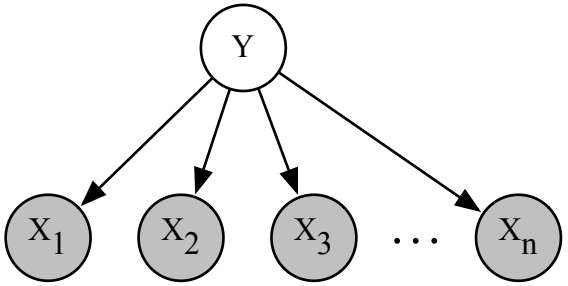
- **E-step**: Compute expectations to "fill in" missing y values according to current parameters, $\theta$
  - For all examples j and values k for $Y_j$, compute: $P(Y_j=k \mid x_j; \theta)$

- **M-step**: Re-estimate the parameters with "weighted" MLE estimates
  - Set $\theta^{new} = \text{argmax}_\theta \sum_j \sum_k P(Y_j=k \mid x_j ;\theta^{old}) \lg P(Y_j=k, x_j ; \theta)$

Takes the place of knowing $Y_j$

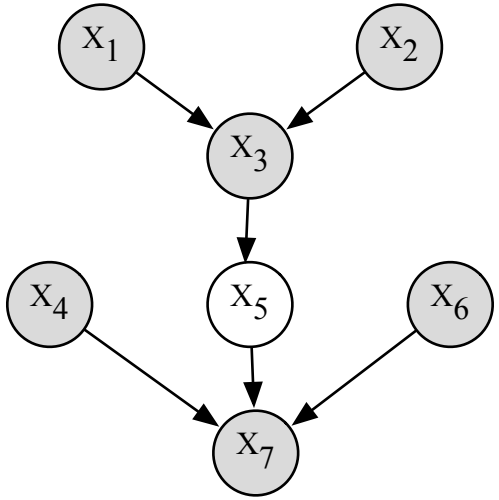
**Particularly useful when the E and M steps have closed form solutions**

# E-step: Inference



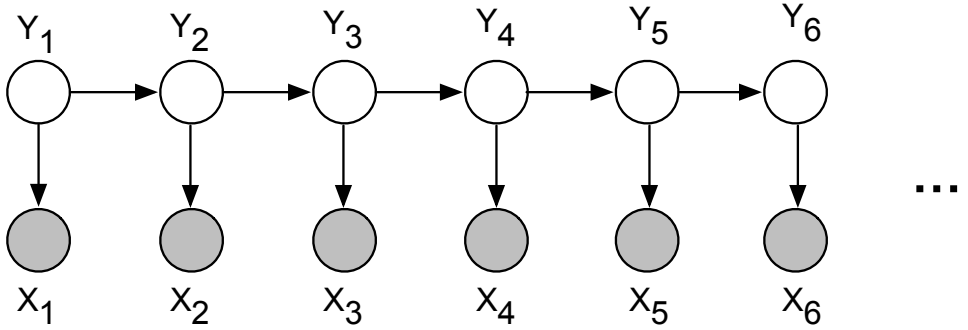

$$p(y \mid x_1, \ldots x_n) = \frac{p(y)p(x \mid y)}{p(x)} \quad \textbf{Bayes' rule}$$

$$= \frac{p(y)\prod_{i=1}^{n} p(x_i \mid y)}{\sum_{\hat{y}} p(\hat{y})\prod_{i=1}^{n} p(x_i \mid \hat{y})}$$

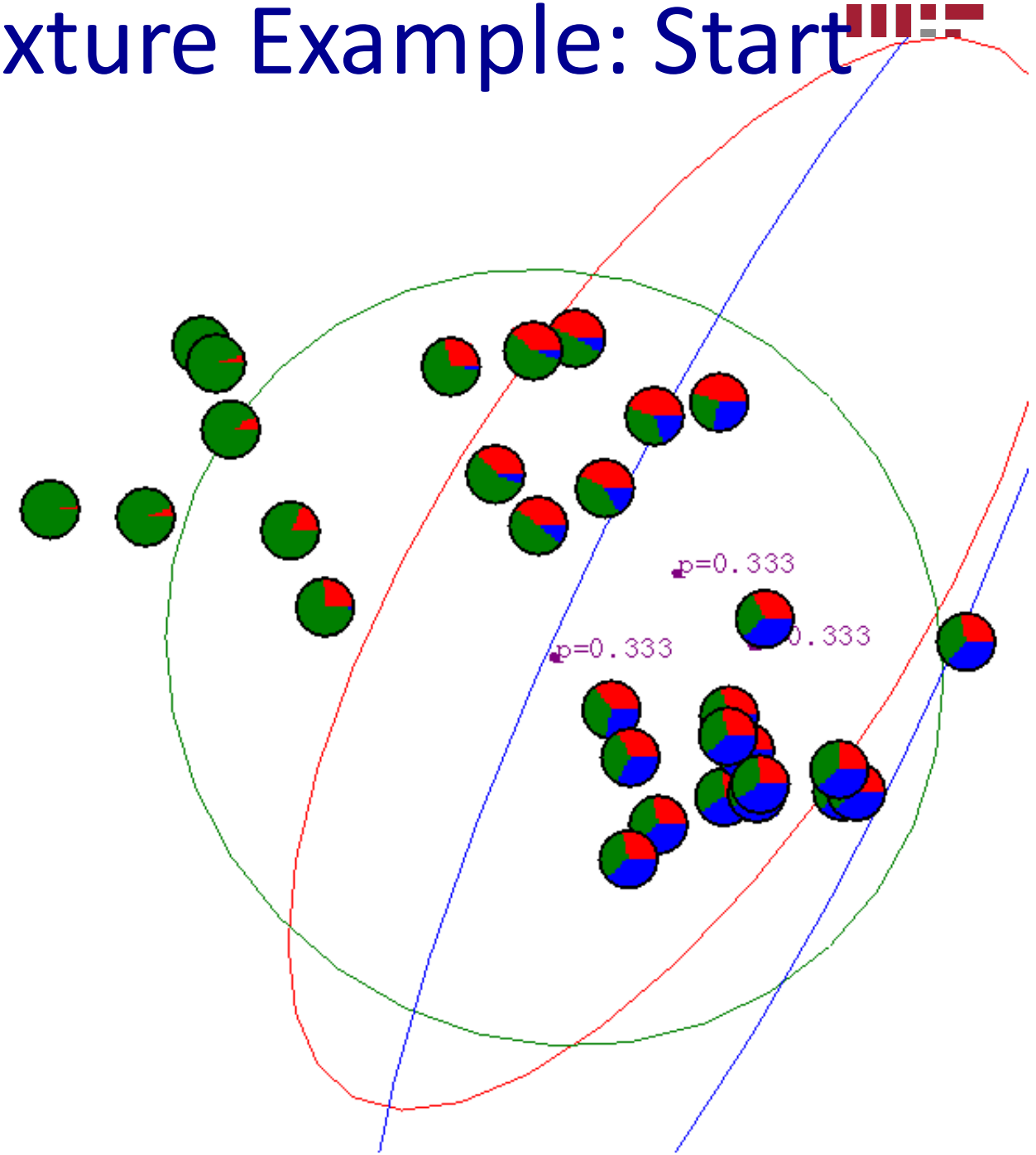



$$p(x_5 \mid x_{-5}) \propto p(x_5 \mid x_3)p(x_7 \mid x_4, x_5, x_6)$$

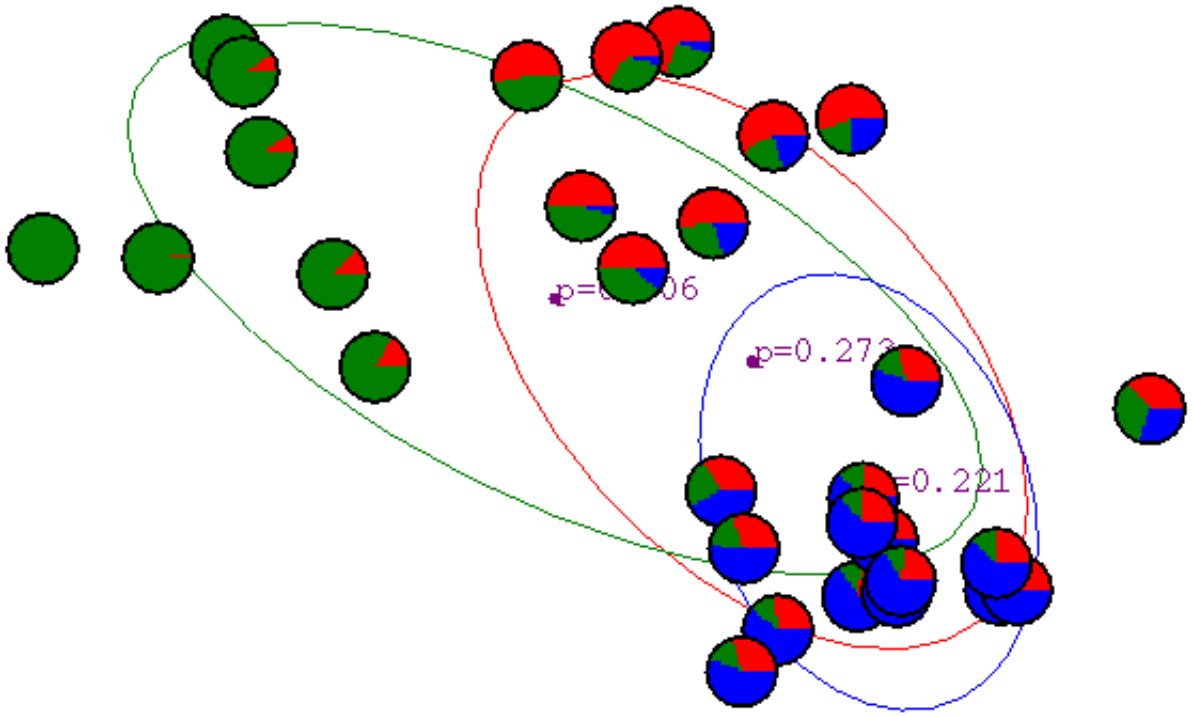
**Markov blanket of $x_5$ consists of $x_3$, $x_4$, $x_6$, $x_7$**



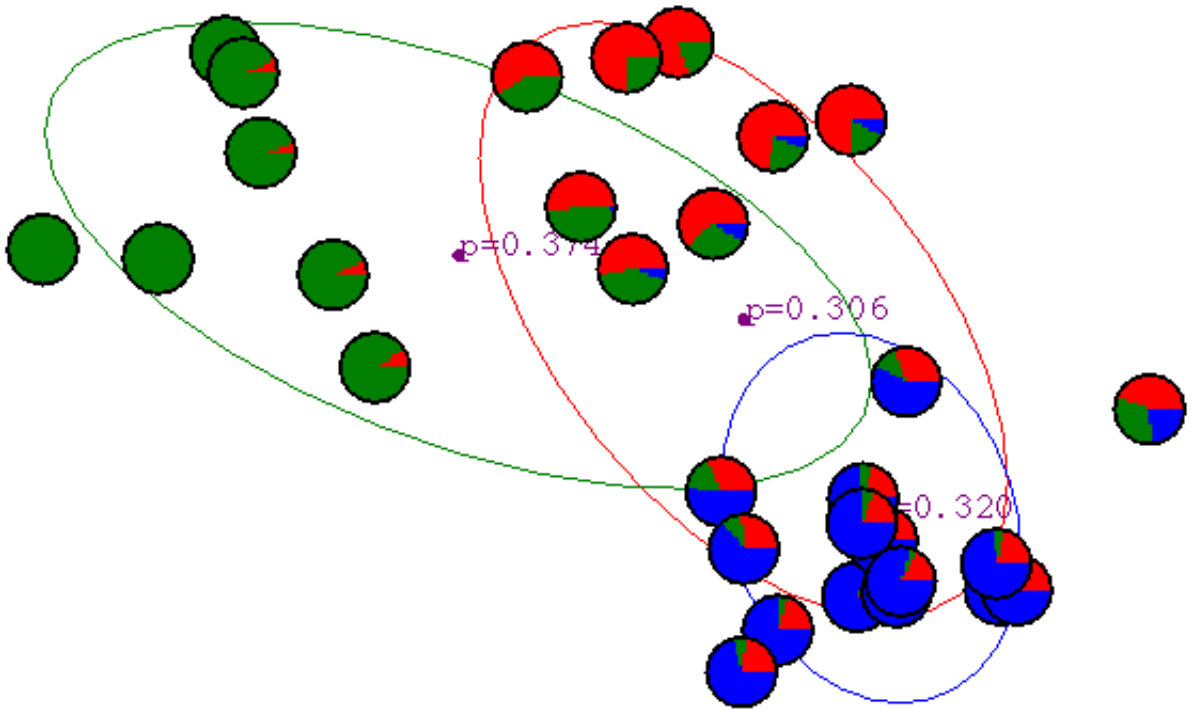
# Gaussian Mixture Example: Start



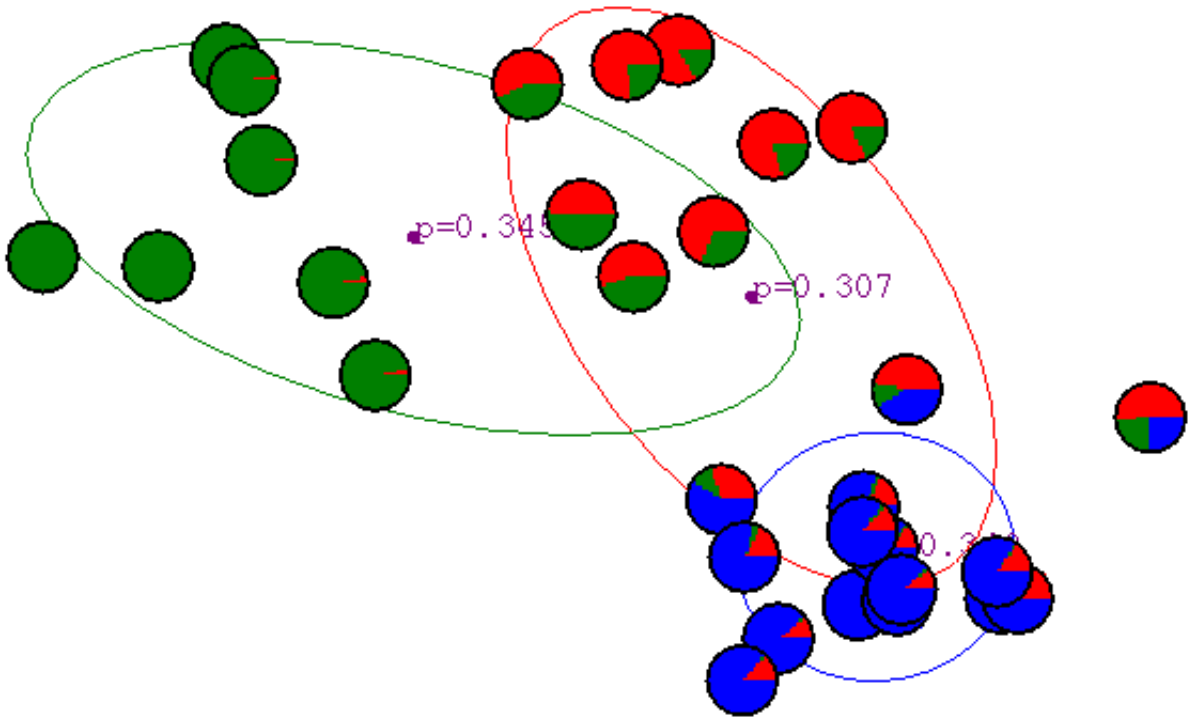
# After first iteration



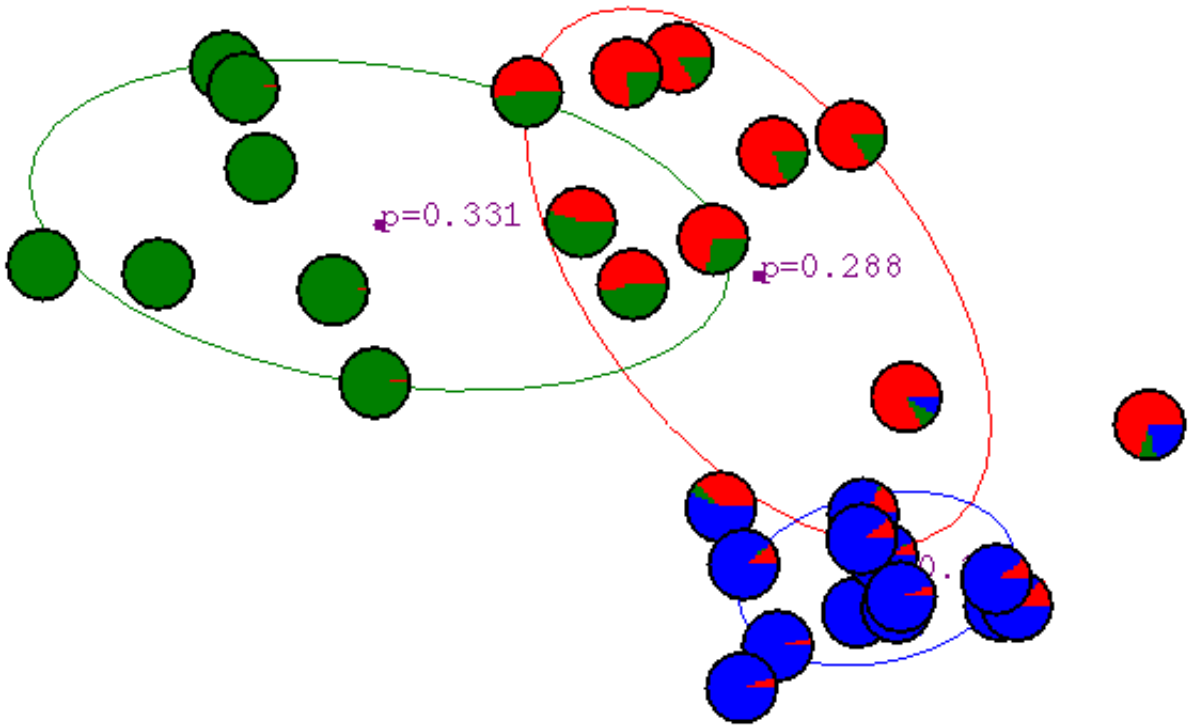
# After 2nd iteration



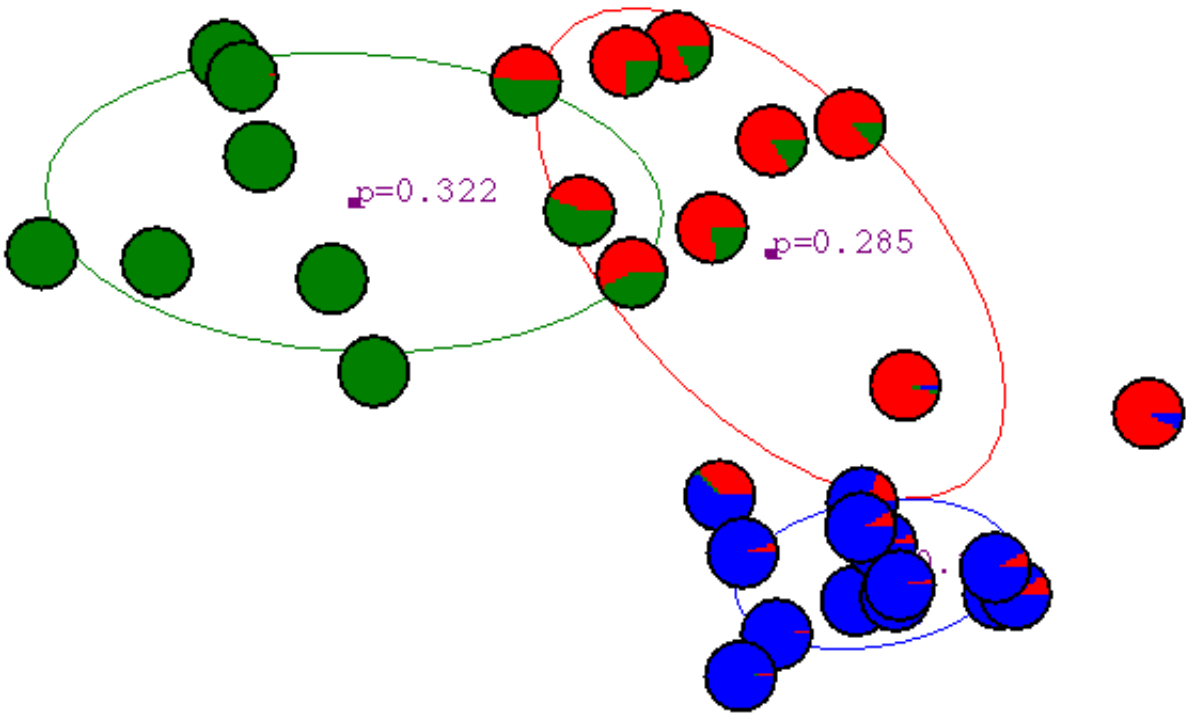
# After 3rd iteration



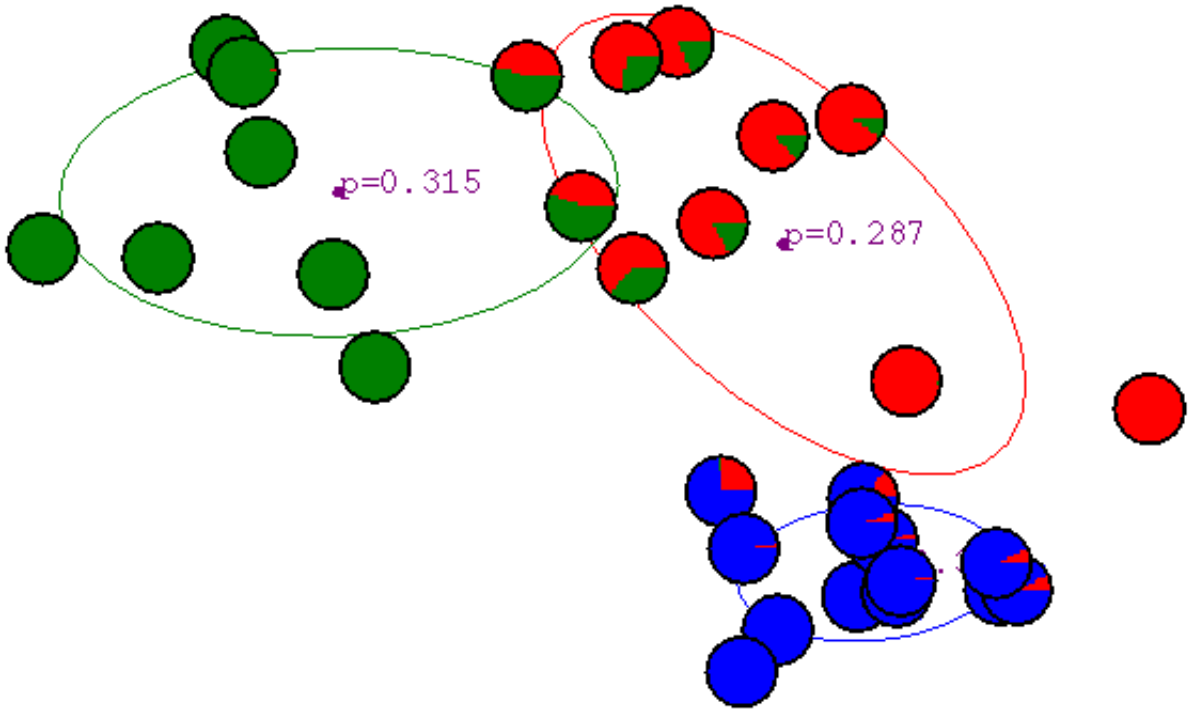
# After 4th iteration



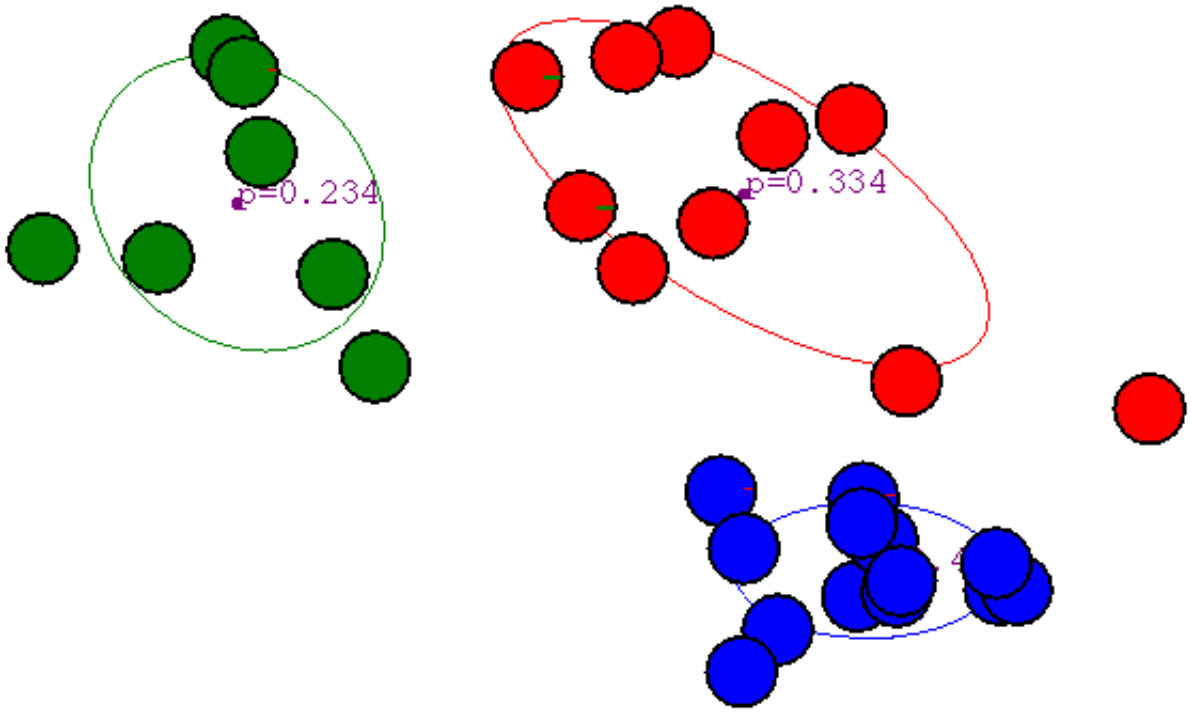
# After 5th iteration



# After 6th iteration



# After 20th iteration



# EM for GMMs: only learning means (1D)

**Iterate:** On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \}$$

**E-step**

Compute "expected" classes of all datapoints

$$P(Y_j = k | x_j, \mu_1 \ldots \mu_K) \propto \exp\left(-\frac{1}{2\sigma^2}(x_j - \mu_k)^2\right) P(Y_j = k)$$

**M-step**

Compute most likely new **μ**s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P(Y_j = k | x_j)\, x_j}{\sum_{j=1}^{m} P(Y_j = k | x_j)}$$

# What if we do hard assignments?

**Iterate:** On the *t*'th iteration let our estimates be

$$\lambda_t = \{ \mu_1^{(t)}, \mu_2^{(t)} \ldots \mu_K^{(t)} \}$$

**E-step**

Compute "expected" classes of all datapoints

$$P(Y_j = k | x_j, \mu_1 \ldots \mu_K) \propto \exp\left(-\frac{1}{2\sigma^2}(x_j - \mu_k)^2\right) P(Y_j = k)$$

**M-step**

Compute most likely new **μ**s given class expectations

$$\mu_k = \frac{\sum_{j=1}^{m} P(Y_j = k | x_j)\, x_j}{\sum_{j=1}^{m} P(Y_j = k | x_j)}$$

$$\mu_k = \frac{\sum_{j=1}^{m} \delta(Y_j = k, x_j)\, x_j}{\sum_{j=1}^{m} \delta(Y_j = k, x_j)}$$

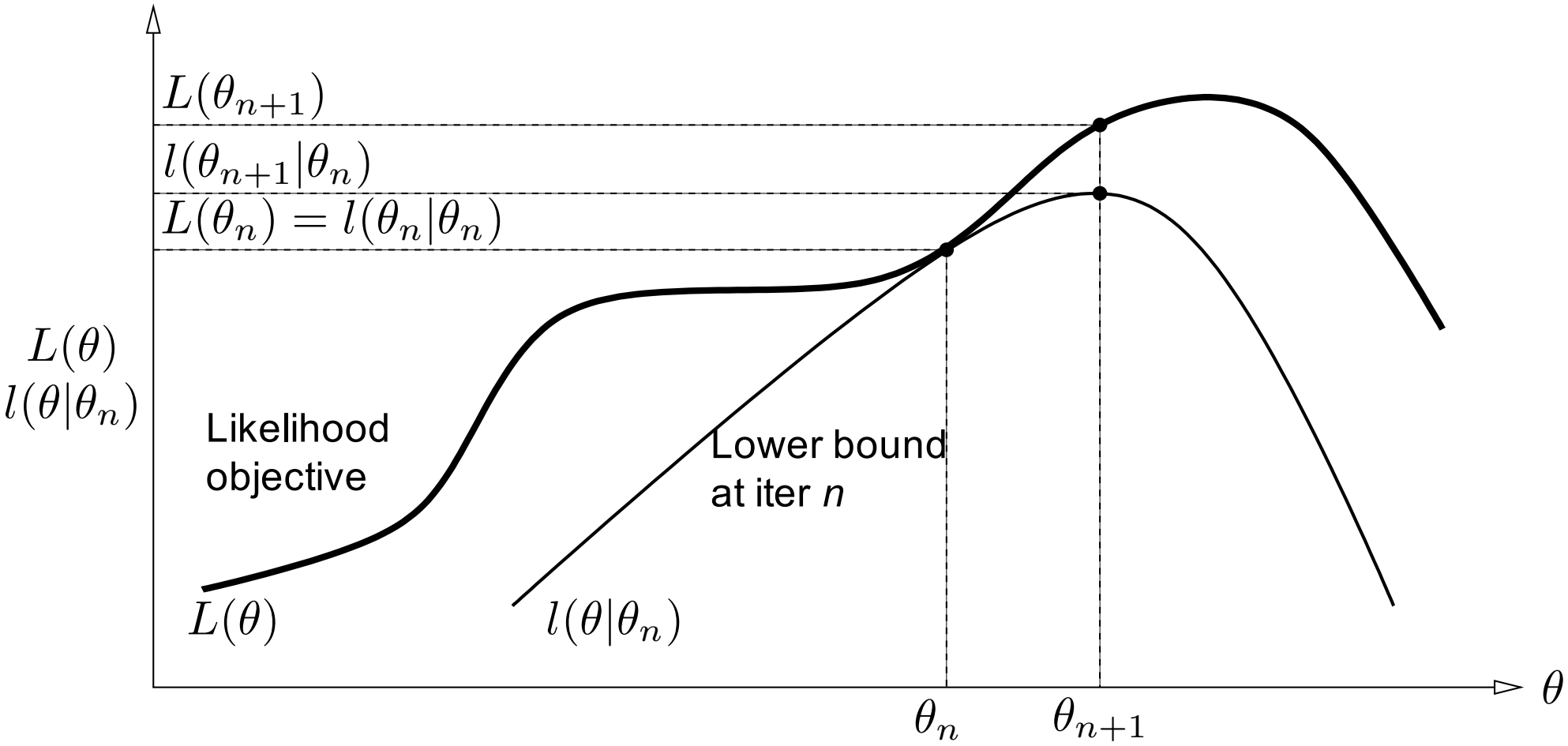
$\delta$ represents hard assignment to "most likely" or nearest cluster

Equivalent to k-means clustering algorithm

# Properties of EM

- One can prove that:
  - EM converges to a local maxima
  - Each iteration improves the log-likelihood
- How? (Same as k-means)
  - Likelihood objective instead of k-means objective
  - M-step can never decrease likelihood

# EM pictorially




(Figure from tutorial by Sean Borman)

# What you should know

- Different approaches to clustering – k-means, hierarchical, spectral, mixture of Gaussians
- Expectation maximization (EM):
  - How to learn maximum likelihood parameters in the case of unlabeled data
  - Relation of EM for mixture of Gaussians to K-means
    - Two step algorithm, just like K-means
    - Hard / soft clustering
    - Probabilistic model
- Remember, EM can get stuck in local minima,
  - And empirically it ***DOES***