Data Mining & Machine Learning

CS37300
Purdue University

November 20, 2017

Public Leaderboard Private Leaderboard

This leaderboard is calculated with approximately 30% of the test data.
The final results will be based on the other 70%, so the final standings may be different.

Raw Data Refresh

#	△1w	Team Name	Kernel	Team Members	Score	Entries	Last
1	▲4	**Yoda**			0.90530	27	7d
2	▼1	**General Grievous**			0.87980	6	18d
3	▼1	**Luke Skywalker**			0.85997	18	2d
4	▲2	**Count Dooku**			0.85066	11	3d
5	▼2	**Captain Rex**			0.84621	6	5d
6	▲4	**Bossk**			0.84216	14	21h
7	▼3	**Revan**			0.83407	12	23d
8	▼1	**Cad Bane**			0.81991	18	1mo
9	▼1	**Dengar**			0.81222	4	16d
10	▼1	**Darth Vader**			0.81019	13	10d
11	new	**IG 88**			0.79643	4	20h
12	▼1	**Mace Windu**			0.77498	3	14d
13	▼1	**Anakin Solo**			0.77296	2	2d
14	▼1	**Ki-Adi-Mundi**			0.76811	2	1mo
15	▼1	**Kyp Durron**			0.73613	5	8d
16	▼1	**Shaak Ti**			0.70133	2	1mo
17	▼1	**Admiral Thrawn**			0.65520	2	1mo
18	▼1	**Clone Commander Cody**			0.63901	8	13d

Ideal thanksgiving project
(entire family can join with ideas)

ENDS Dec 1st (10 days)

K-means Recap

Algorithm 2.1 The `k-means` algorithm

Input: Dataset D, number clusters k
Output: Set of cluster representatives C, cluster membership vector $\mathbf{m}$

```
/* Initialize cluster representatives C */
Randomly choose k data points from D
Use these k points as initial set of cluster representatives C
repeat
    /* Data Assignment */
    Reassign points in D to closest cluster mean
    Update m such that m_i is cluster ID of ith point in D
    /* Relocation of means */
    Update C such that c_j is mean of points in jth cluster
until convergence of objective function
```

$$\sum_{i=1}^{N}(argmin_j ||\mathbf{x_i} - \mathbf{c_j}||_2^2)$$

Score function: $$wc(C) = \sum_{k=1}^{K} wc(C_k) = \sum_{k=1}^{K} \sum_{x(i) \in C_k} d(x(i), r_k)^2$$

Algorithm details

- Does it terminate?
  - Yes, the objective function decreases on each iteration. It usually converges quickly.
- Does it converge to an optimal solution?
  - No, the algorithm terminates at a local optima which depends on the starting seeds.
- What is the time complexity?
  - $O(k \cdot n \cdot i)$, where i is the number of iterations

K-means

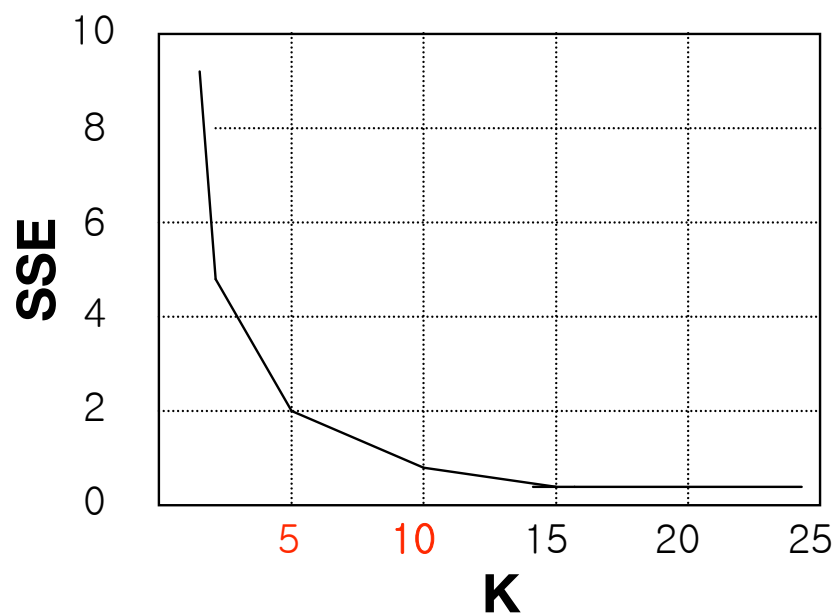
- Strengths:
  - Relatively efficient
  - Finds spherical clusters
- Weaknesses:
  - Terminates at local optimum (sensitive to initial seeds)
  - Applicable only when mean is defined
  - Need to specify k
  - Susceptible to outliers/noise

Variations

- Selection of initial centroids
  - Run with multiple random selections, pick result with best score
  - Use hierarchical clustering to identify likely clusters and pick seeds from distinct groups
  - Select first seed randomly and then pick successive points that are farthest away
- Algorithm modifications:
  - Recompute centroid after each point is assigned
  - Allow for merge and split of clusters (e.g., if cluster becomes empty, start a new one from randomly selected point)

Variations

- When mean is undefined
  - K-medioids: use one of the data points as cluster center
  - K-modes: uses categorical distance measure and frequency-based update method
- How to select k?
  - Plot objective function (within cluster SSE) as a function of k, look for knee in plot



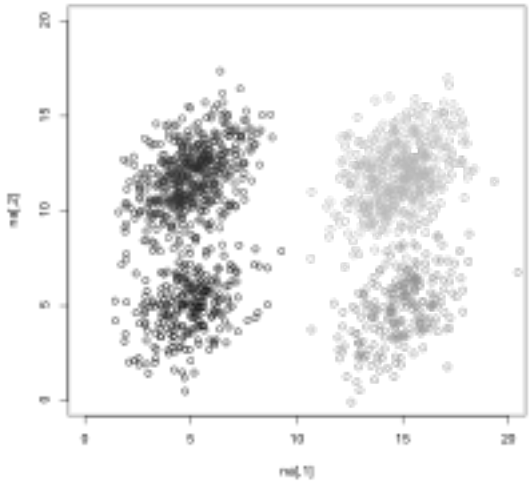
K-means summary

- Knowledge representation
  - K clusters are defined by canonical members (e.g., centroids)
- Model space the algorithm searches over?
  - *All possible partitions of the examples into k groups*
- Score function?
  - *Minimize within-cluster Euclidean distance*
- Search procedure?
  - *Iterative refinement correspond to greedy hill-climbing*

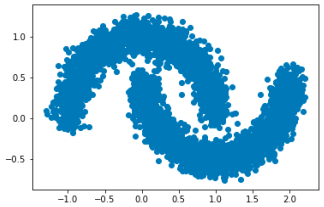
Probabilistic Models

Descriptive Modeling through Modeling Distribution

- Model the probability distribution $p(x)$

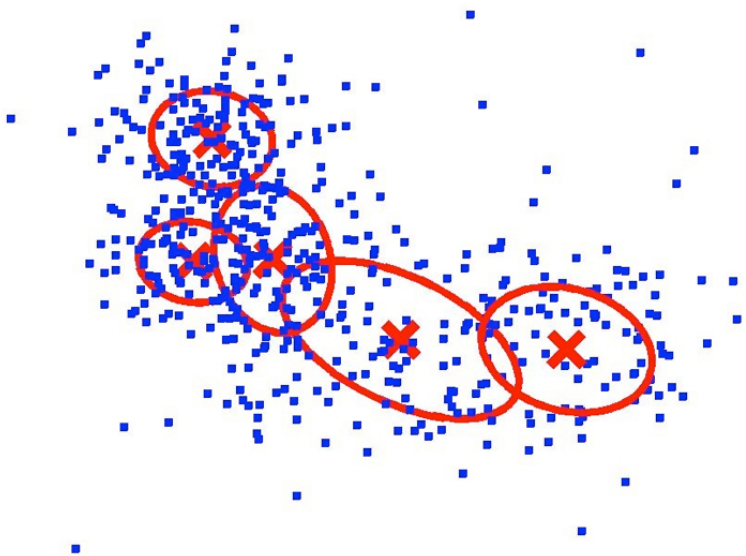


- Hard to describe using a single probability distribution
  - We will use a mixture of simple distributions
  - *Drawback*: data is may **not** be mixture of simple distributions:



- K-means is just an approximate solution (heuristic) of a specific type of mixture model (Gaussian Mixture Model)

Probabilistic mixture model



$$p(x) = \sum_{k=1}^{K} w_k \, p(x; \theta_k)$$

parameters describe cluster k model

probability of point belonging to cluster k

probability of observing x

probability of x being generated from cluster k

Mixture models

- How to learn the model from data?
- We don't know the mixing coefficients ($w_{1...k}$) or the component parameters (θ)
- Solution:
  - Interpret mixing coefficients as prior probabilities of cluster membership
  - Use **Expectation-Maximization** algorithm to estimate model *(Dempster, Laird, Rubin, 1977)*

$$p(x) = \sum_{k=1}^{K} \underbrace{w_k}_{p(k)} \, p(x; \theta_k)$$

Generative process (revisited)

- Assume that the data are generated from a mixture of k multi-dimensional Gaussians, where each component is has parameters: $N_k(\mu_k, \Sigma_k)$

covariance matrix (next slide)

average (vector)

- For each data point:
  - Pick component Gaussian randomly with probability p(k)
  - Draw point from that Gaussian randomly by sampling from: $N_k(\mu_k, \Sigma_k)$

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

$$= \sum_{k=1}^{K} p(k)p\Big(x|x \sim N(\mu_k, \Sigma_k)\Big)$$

Multidimensional Gaussian

- A multi-dimensional Gaussians, for data with p dimensions is specified as follows

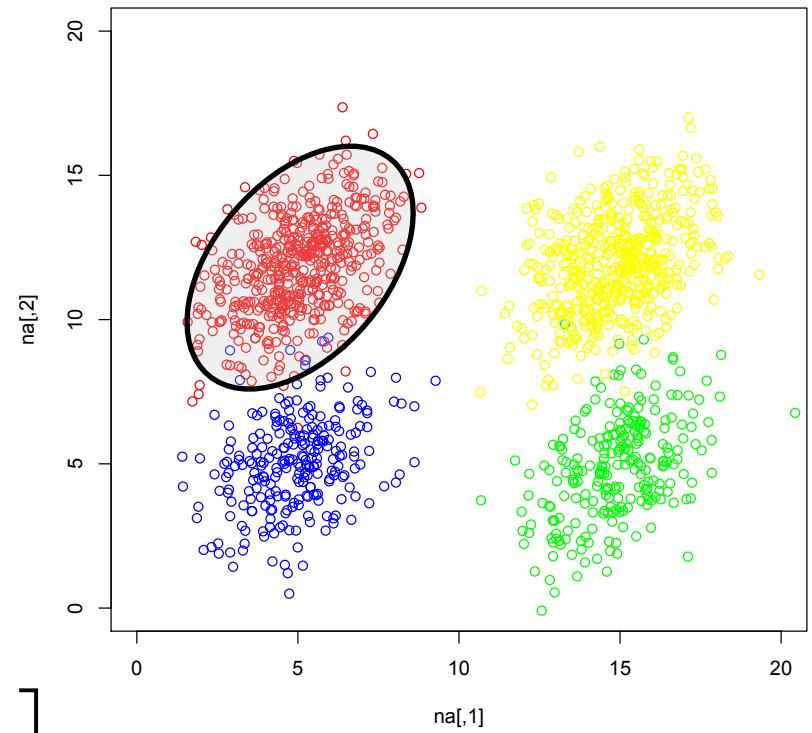
$$x \sim \mathcal{N}(\mu, \Sigma)$$

where:

$$\mu = \Big(E[X_1], ..., E[X_p]\Big)$$

$$\Sigma = \begin{bmatrix} Var(X_1) & ... & Cov(X_1, X_p) \\ ... & ... & ... \\ Cov(X_1, X_p) & ... & Var(X_p) \end{bmatrix}$$

$$p(\mathbf{x}) = p(x_1, ..., x_p) = \frac{1}{\sqrt{(2\pi)^p |\Sigma|}} exp\left(-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right)$$

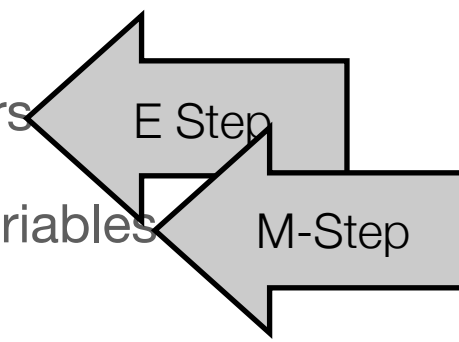


Learning the model from data

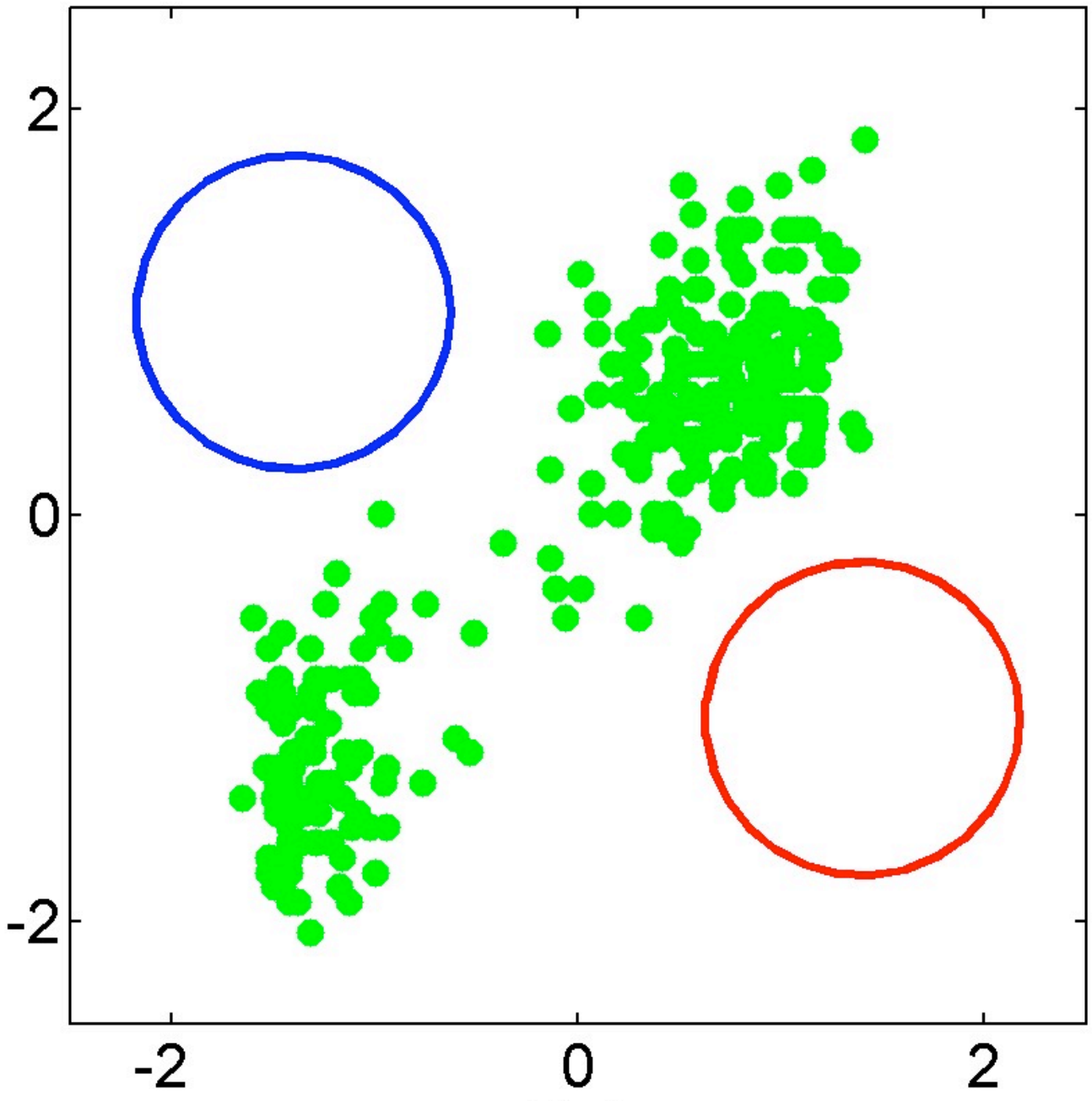
- We want to invert this process
- Given the dataset, find the parameters
  - Mixing coefficients $p(k)$
  - Component means and covariance matrix $N_k(\mu_k, \Sigma_k)$
- If we knew which component generated each point then the MLE solution would involve fitting each component distribution to the appropriate cluster points
- Problem: the cluster memberships are **hidden**

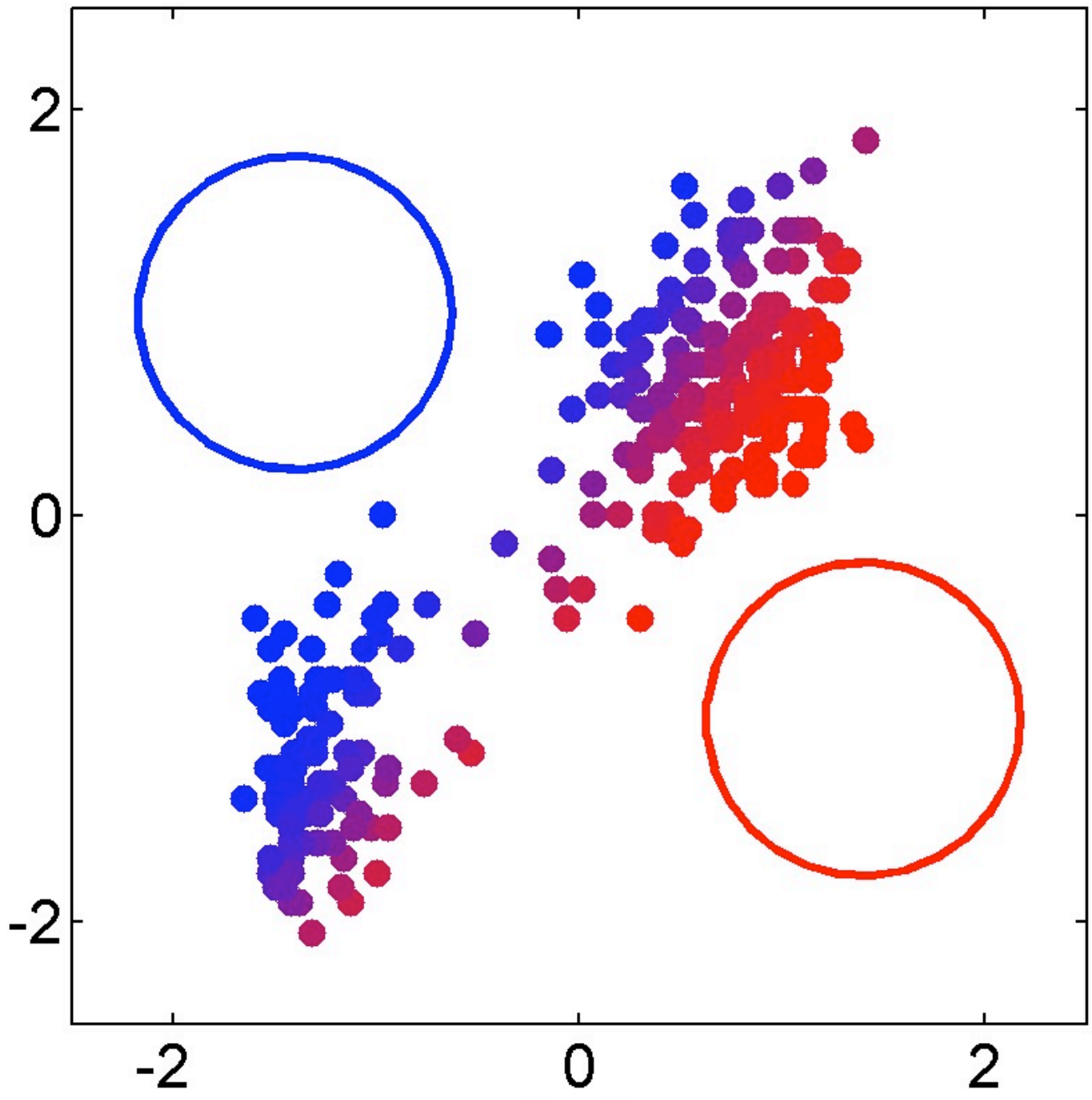
Expectation-maximization (EM) algorithm

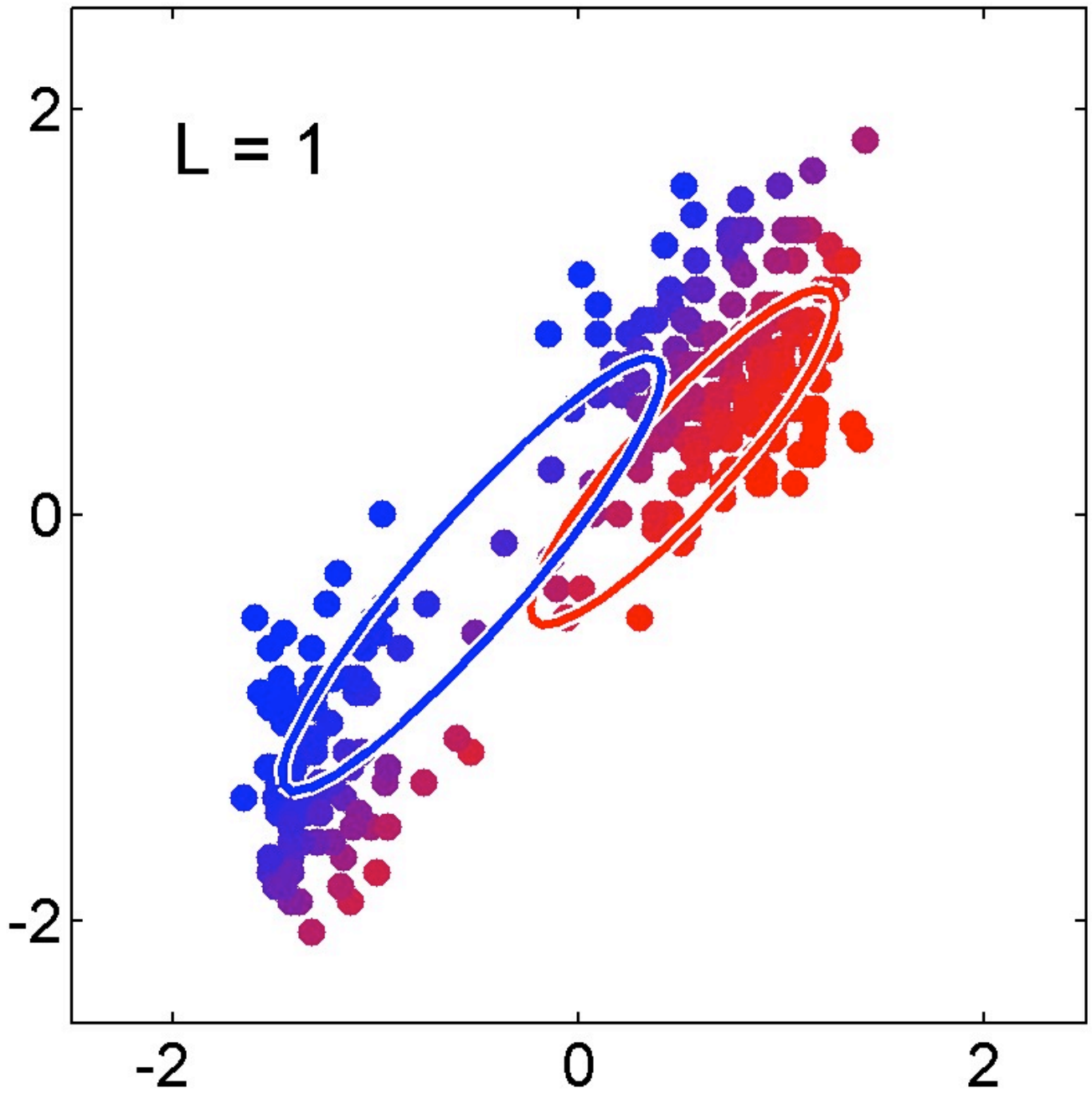
- Popular algorithm for parameter estimation in data with hidden/unobserved values
  - Hidden variables=cluster membership
- Basic idea
  - Initialize hidden variables and parameters
  - Predict values for hidden variables given current parameters
  - Estimate parameters given current prediction for hidden variables
  - Repeat

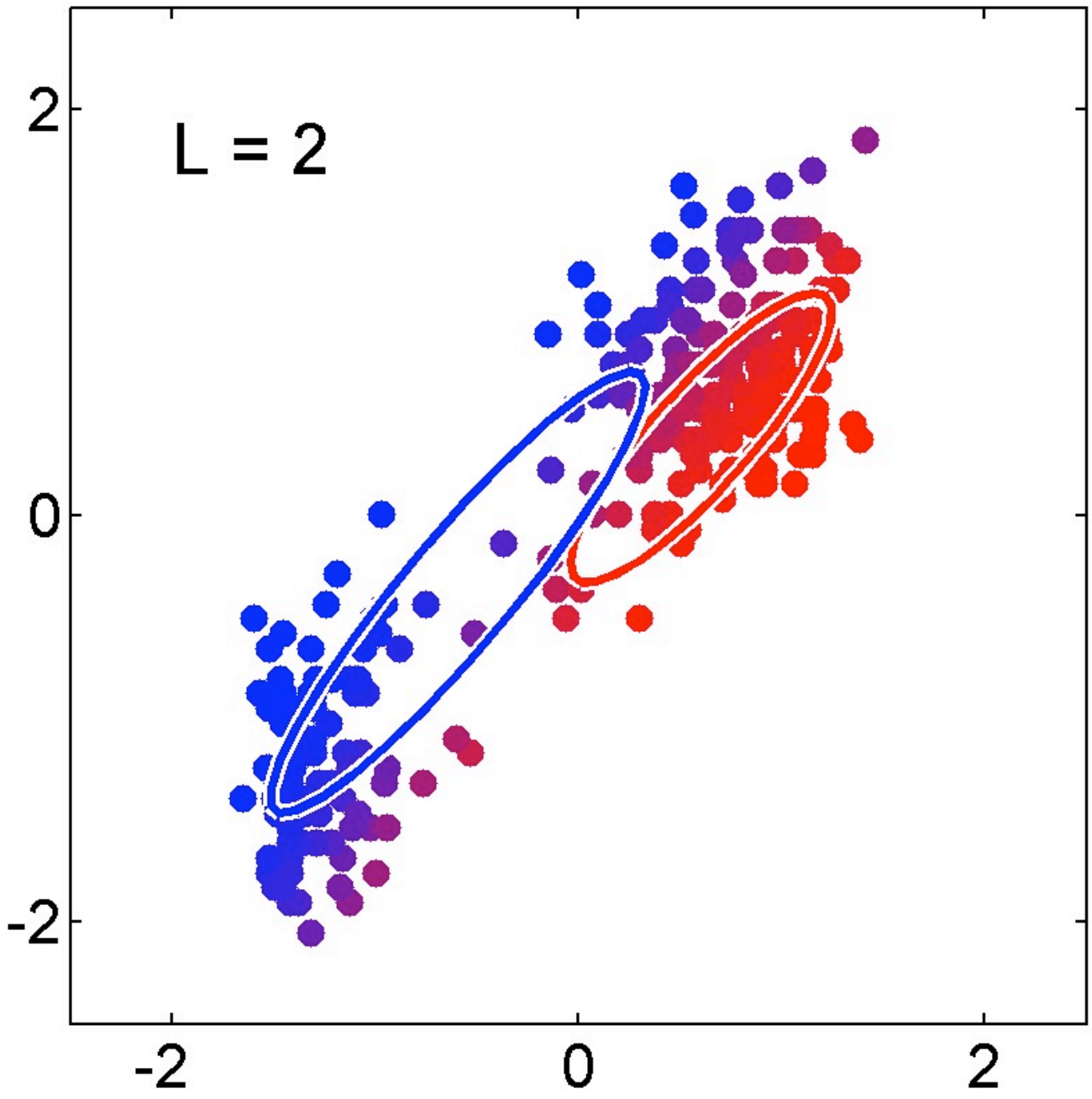


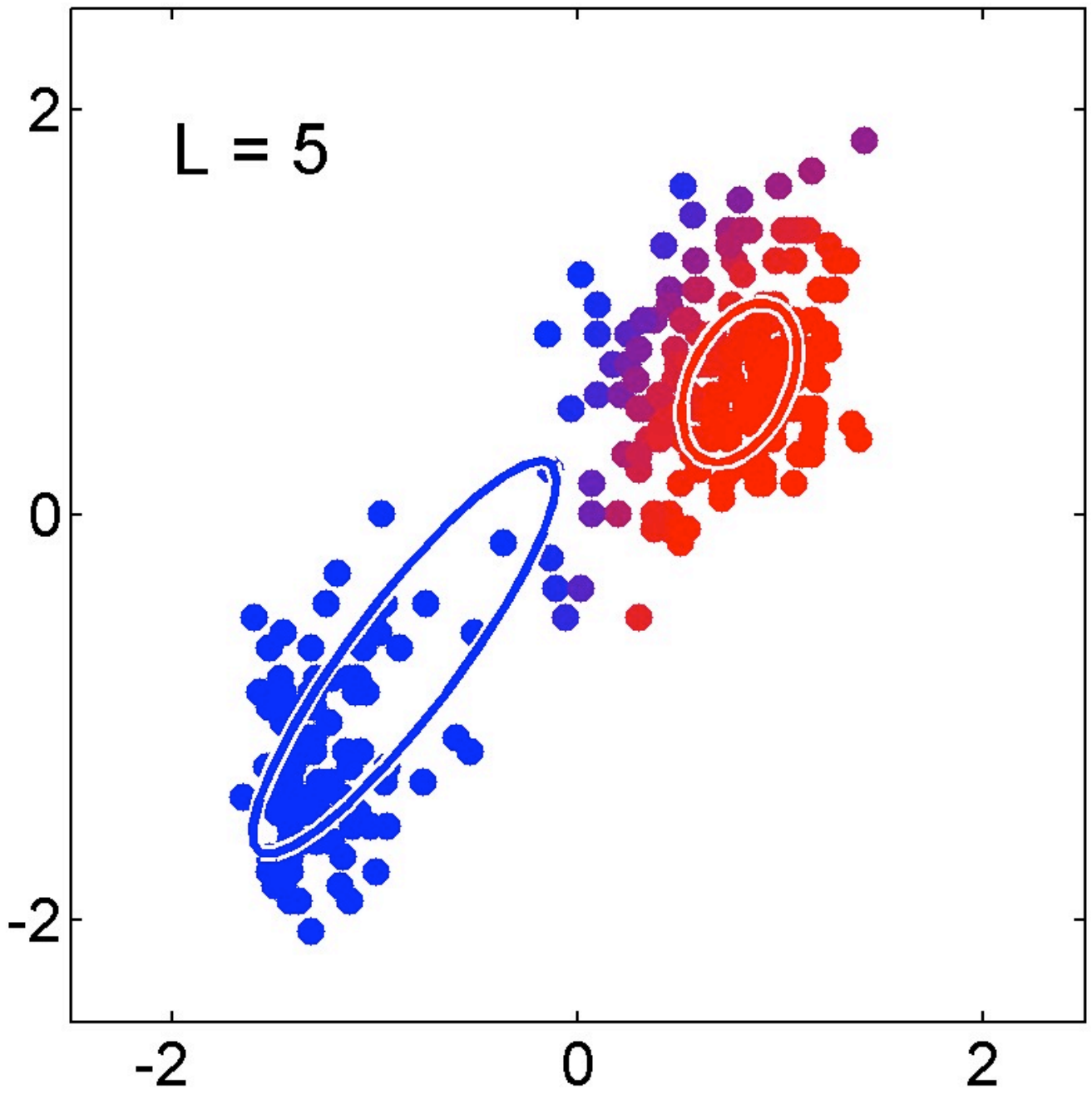
GMM example

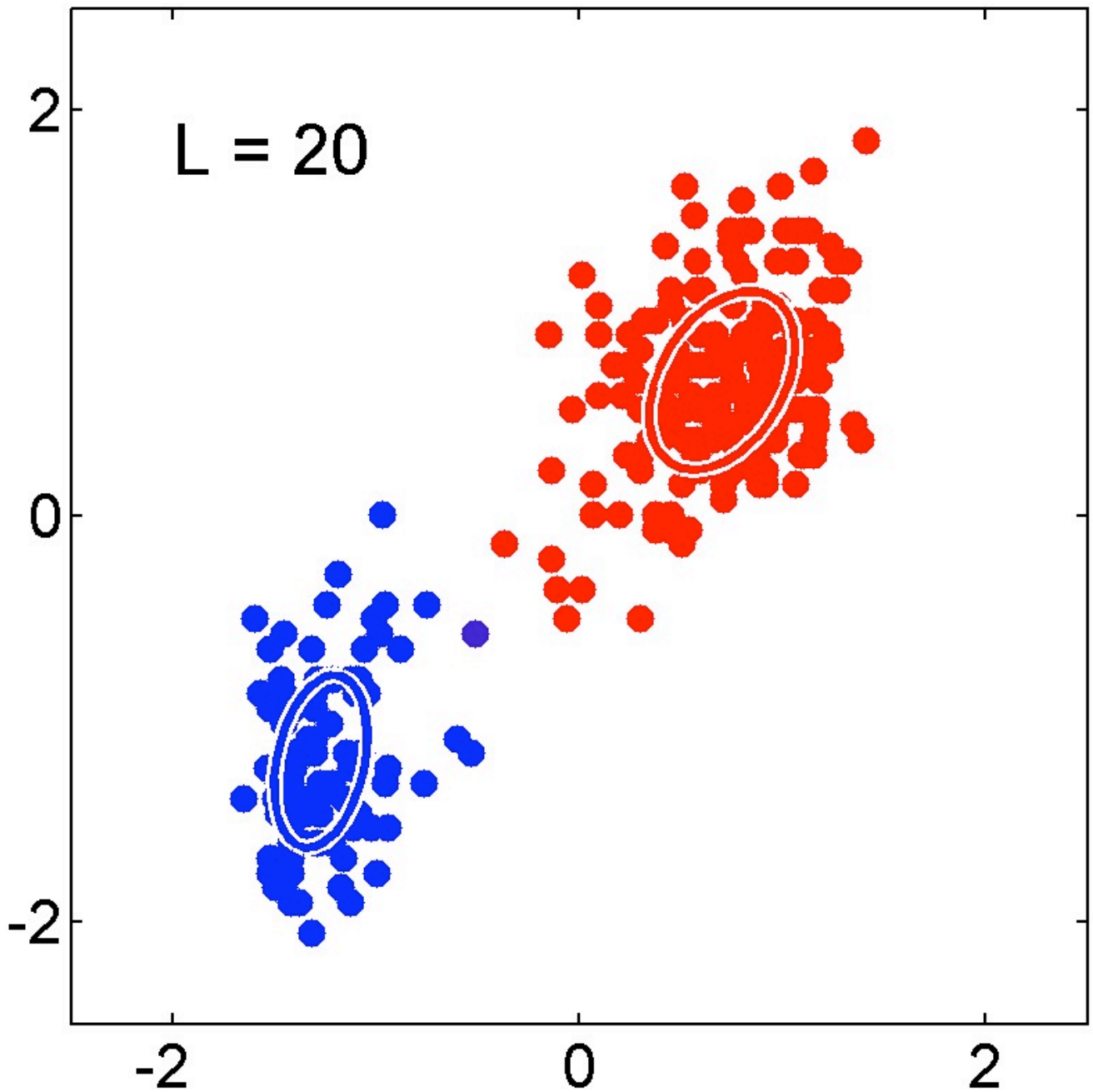

2
0
-2
-2
0
2


2
0
-2
-2
0
2


L = 1
2
0
-2
-2
0
2


L = 2
2
0
-2
-2
0
2


L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

How to learn GMMs?

Score function for GMM

- **Log likelihood** takes the following form (for model M={w,μ,Σ}):

$$
\begin{aligned}
log\, p(D|w,\mu,\Sigma) &= \sum_{i=1}^{N} log\, p(x_n|M) \\
&= \sum_{i=1}^{N} log\, \Big[\sum_{k=1}^{K} p(x_n|k,M)P(k|M)\Big] \\
&= \sum_{i=1}^{N} log\Big[\sum_{k=1}^{K} w_k N(x_n|\mu_k,\Sigma_k)\Big]
\end{aligned}
$$

- Note the sum over components is inside the log
- There is no closed form solution for the MLE

Hidden cluster membership variables

- Consider k cluster indicator variables for example x_n: $\mathbf{z_n} = [z_{n1}, ..., z_{nk}]$ which equals 1 for the cluster that x_n is a member of, and 0 otherwise

- If we knew the values of the hidden cluster membership variables (z) we could easily maximize the complete data log-likelihood, which has a closed form solution:

$$
\begin{aligned}
log\, p(D, \mathbf{z}|w, \mu, \Sigma) &= \sum_{i=1}^{N} log\Big[\sum_{k=1}^{K} z_{nk} \cdot w_k N(x_n|\mu_k, \Sigma_k)\Big] \\
&= \sum_{i=1}^{N} log\, \Big[w_{k'} N(x_n|\mu_{k'}, \Sigma_{k'})\Big] \quad \text{where } z_{nk'} \neq 0 \\
&= \sum_{i=1}^{N} log\, w_{k'} + log\, w_{k'} N(x_n|\mu_{k'}, \Sigma_{k'}) \quad \text{where } z_{nk'} \neq 0
\end{aligned}
$$

- Unfortunately we don't know the values for the hidden variables!

- But, for given set of parameters we can compute the ***expected values*** of the hidden variables (cluster memberships)

Posterior probabilities of cluster membership

- We can think of the mixing coefficients as **prior** probabilities for cluster membership
- Then for a given example x_n, we can evaluate the corresponding **posterior** probabilities of **cluster membership** with Bayes theorem:

$$\gamma_k(x_n) \equiv p(z_{nk} = 1|x_n) = \frac{p(x_n|z_{nk} = 1)p(z_{nk} = 1)}{p(x_n)}$$

$$= \frac{w_k N(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} w_j N(x_n|\mu_j, \Sigma_j)}$$

cluster membership for x

EM for GMM

- Suppose we make a guess for the parameters values
- Use these to evaluate cluster memberships

$$\Gamma(x_n) = [\gamma_1(x_n), ..., \gamma_K(x_n)]$$

E-Step

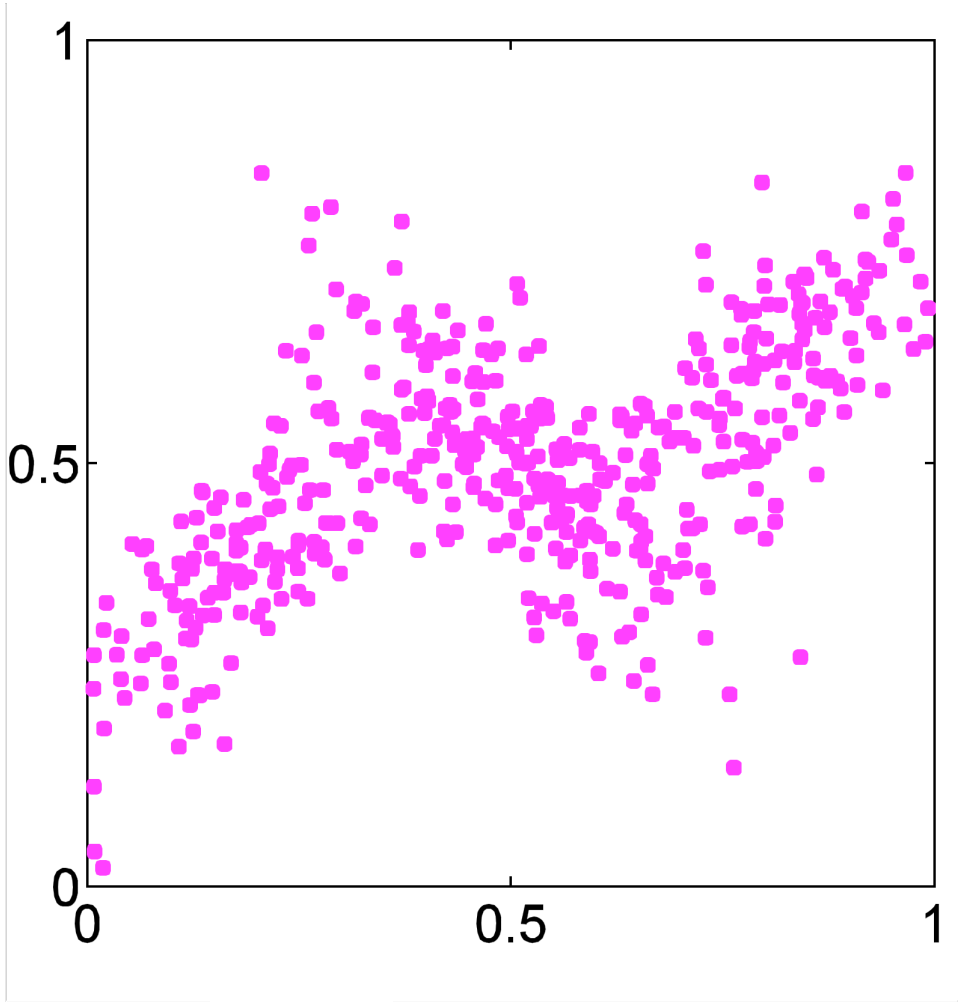
- Now compute the log-likelihood using predicted cluster memberships

$$log\ p(x, z|\theta) = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma_i(x_n) \big[log\ w_k + log\ N(x_n|\mu_k, \Sigma_k) \big]$$

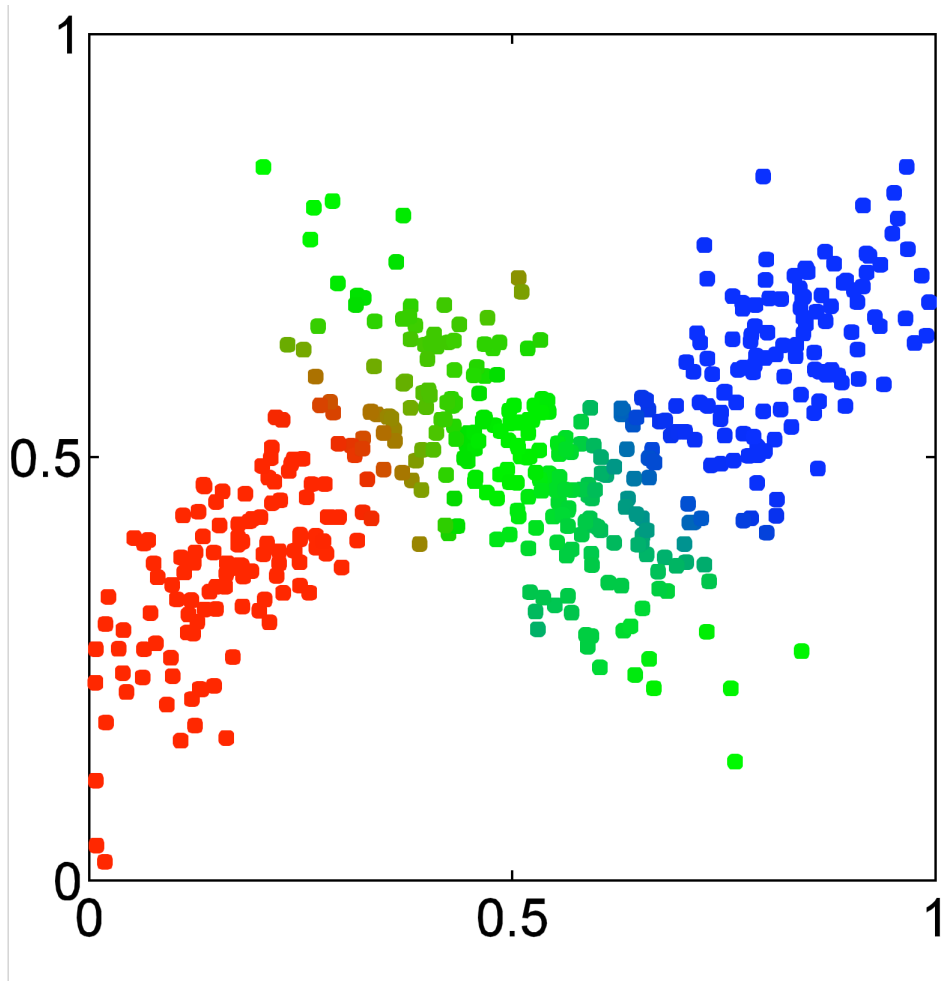
- Use completed likelihood to determine MLE for parameters

M-Step

Unlabeled dataset



Posterior probabilities of cluster membership



More on EM

- Often both the E and the M step can be solved in closed form
- Neither the E step nor the M step can decrease the log-likelihood
- Algorithm is guaranteed to converge to a local maximum of the likelihood
- Must specify initialization and stopping criteria

Probabilistic clustering

- Model provides full distributional description for each component
  - May be able to interpret differences in the distributions
- Soft clustering (compared to k-mean hard clustering)
  - Given the model, each point has a k-component vector of membership probabilities
- Key cost: assumption of parametric model

Mixture models

- Knowledge representation?
  - **Parametric model**
    parameters = mixture coefficient and component parameters
- Score function?
  - **Likelihood**
- Search?
  - **Expectation maximization**
    iteratively find parameters that maximize likelihood and predicts cluster memberships
- Optimal? Exhaustive?

Hierarchical clustering

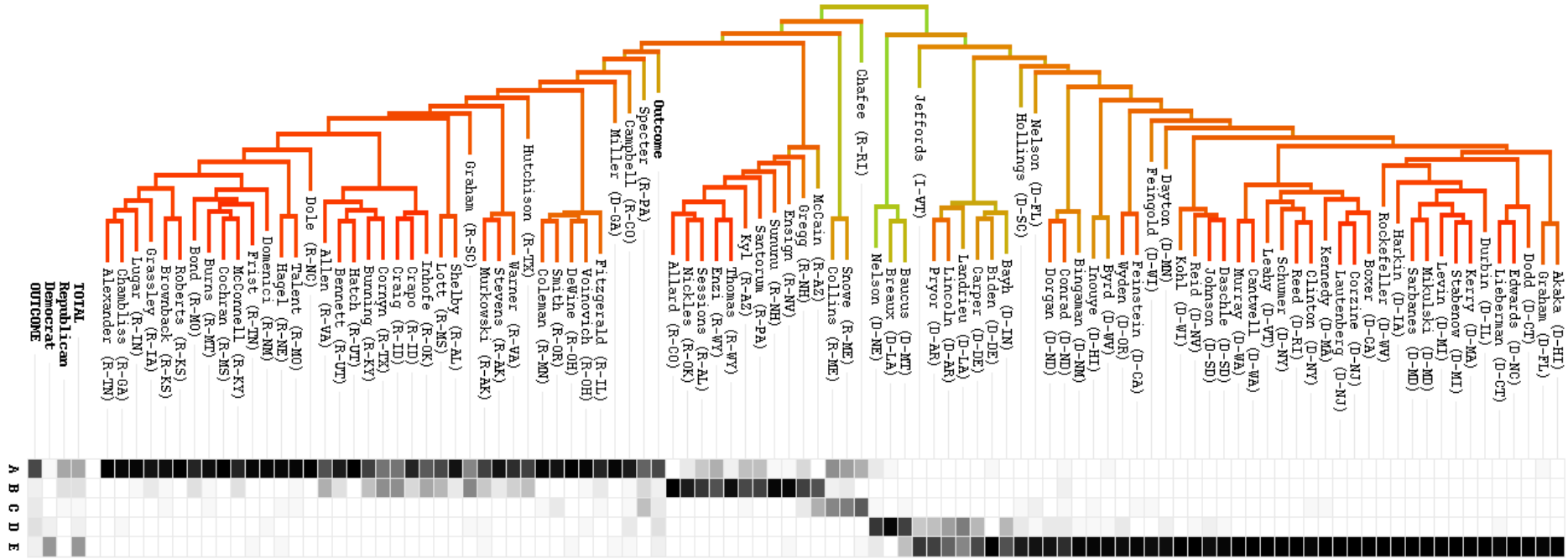
Hierarchical methods

- Construct a hierarchy of nested clusters rather than picking k beforehand
- Approaches:
  - Agglomerative: merge clusters successively
  - Divisive: divided clusters successively
- Dendrogram depicts sequences of merges or splits and height indicates distance

Agglomerative

- For i = 1 to n:
  - Let $C_i = \{x(i)\}$
- While $|C|>1$:
  - Let C_i and C_j be the pair of clusters with min $D(C_i,C_j)$
  - $C_i=C_i \cup C_j$
  - Remove C_j

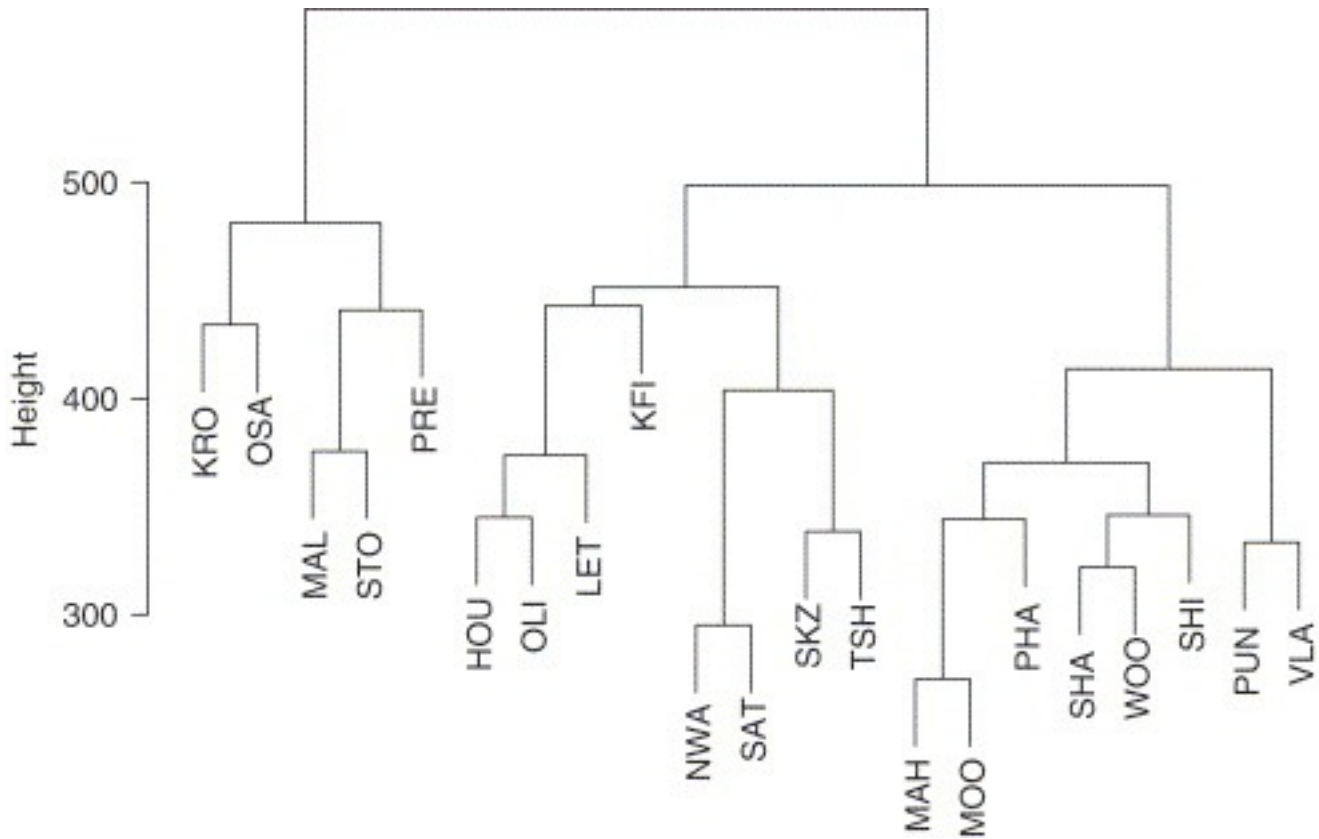
Hierarchical clustering



Clustering represented with dendogram

Example

- Agglomerative clustering results of surface water availability in areas of Kruger National Park, South Africa. Three primary clusters can be distinguished, which correspond to a north, south, and far south spatial division of the KNP.



Distance measures between clusters

- Single-link/nearest neighbor:
  - $D(C_i,C_j) = \mathbf{min}\{ d(x,y) \mid x \in C_i,\ y \in C_j \}$
    $\Rightarrow$ can produce long thin clusters
- Complete-link/furthest neighbor:
  - $D(C_i,C_j) = \mathbf{max}\{ d(x,y) \mid x \in C_i,\ y \in C_j \}$
    $\Rightarrow$ is sensitive to outliers
- Average link:
  - $D(C_i,C_j) = \mathbf{avg}\{ d(x,y) \mid x \in C_i,\ y \in C_j \}$
    $\Rightarrow$ compromise between the two