Clustering

CS294 Practical Machine Learning

Junming Yin

10/09/06

Outline

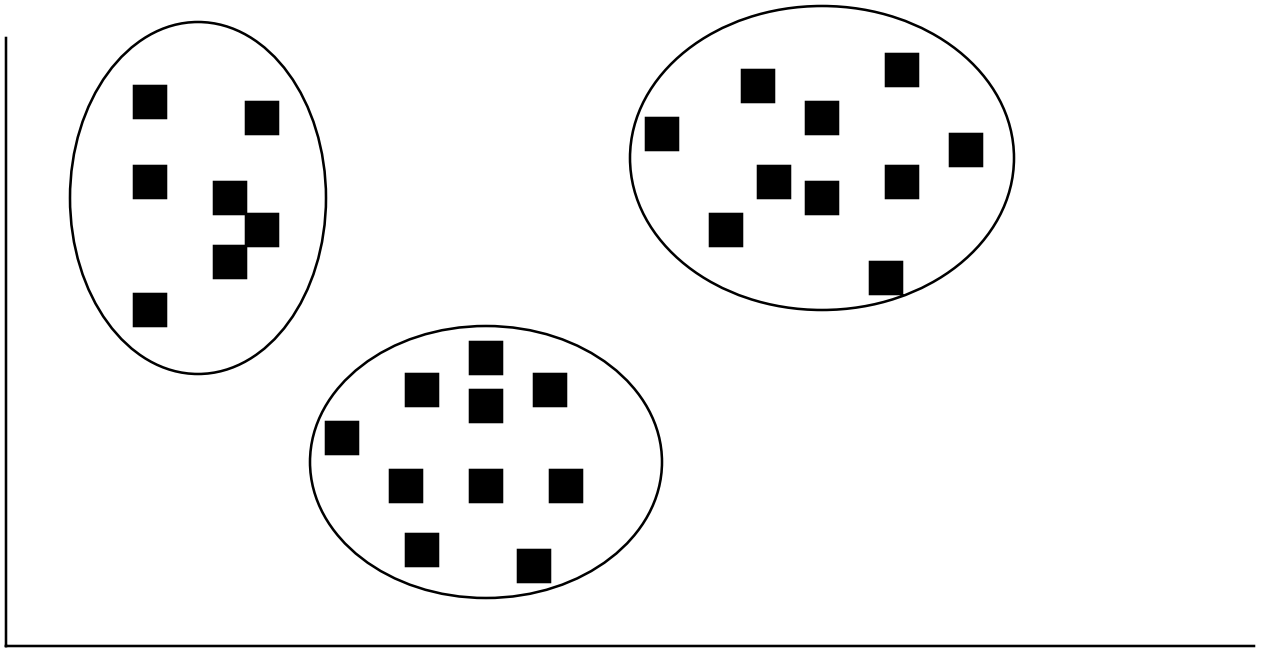
- Introduction
  - Unsupervised learning
  - What is clustering? Application
- Dissimilarity (similarity) of objects
- Clustering algorithm
  - K-means, VQ, K-medoids
  - Gaussian mixture model (GMM), EM
  - Hierarchical clustering
  - Spectral clustering

Unsupervised Learning

- Recall in the setting of classification and regression, the training data are represented as $(x_i, y_i)_{i=1...n}$, the goal is to predict y_{n+1} given a new point x_{n+1}
- They are called supervised learning
- In unsupervised setting, we are only given the unlabelled data $(x_i)_{i=1,...,n}$, the goal is:
  - Estimate density $P(x)$
  - Dimension reduction: PCA, ICA (next week)
  - Clustering, etc

What is Clustering?

- Roughly speaking, clustering analysis yields a data description in terms of clusters or groups of data points that posses strong internal similarity
  - a dissimilarity function between objects
  - an algorithm that operates on the function



What is Clustering?

- Unlike in supervised setting, there is no clear measure of success for clustering algorithms; people usually resort to heuristic argument to judge the quality of the results, e.g. Rand index (see web supplement for more details)

- Nevertheless, clustering methods are widely used to perform exploratory data analysis (EDA) in the early stages of data analysis and gain some insight into the nature or structure of data

Application of Clustering

- Image segmentation: decompose the image into regions with coherent color and texture inside them
- Search result clustering: group the search result set and provide a better user interface (Vivisimo)
- Computational biology: group homologous protein sequences into families; gene expression data analysis
- Signal processing: compress the signal by using codebook derived from vector quantization (VQ)

Outline

- Introduction
  - Unsupervised learning
  - What is clustering? Application
- Dissimilarity (similarity) of objects
- Clustering algorithm
  - K-means, VQ, K-medoids
  - Gaussian mixture model (GMM), EM
  - Hierarchical clustering
  - Spectral clustering

Dissimilarity of objects

- The natural question now is: how should we measure the dissimilarity between objects?
  - fundamental to all clustering methods
  - usually from subject matter consideration
  - not necessarily a metric (i.e. triangle inequality doesn't hold)
  - possible to learn the dissimilarity from data (later)
- Similarities can be turned into dissimilarities by applying any monotonically decreasing transformation

Dissimilarity Based on Attributes

- Most of time, data x_i have measurements x_{ij} on attributes $j = 1, \ldots, p$.
- Define dissimilarities between attribute values
  - common choice: $d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$
- Combine the attribute dissimilarities to the object dissimilarity, using the weighted average

$$D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j d_j(x_{ij}, x_{i'j}) \text{ where } \sum_{j=1}^{p} w_j = 1$$

- The choice of weights is also a subject matter consideration; but possible to learn from data (later)

Dissimilarity Based on Attributes

- Setting all weights equal does not give all attributes equal influence on the overall dissimilarity of objects!
- An attribute's influence depends on its contribution to the average object dissimilarity

$$\bar{D} = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} D(x_i, x_{i'}) = \sum_{j=1}^{p} w_j \bar{d}_j$$

average dissimilarity of *j*th attribute

$$\bar{d}_j = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} d_j(x_{ij}, x_{i'j})$$

- Setting $w_j \propto 1/\bar{d}_j$ gives all attributes equal influence in characterizing overall dissimilarity between objects

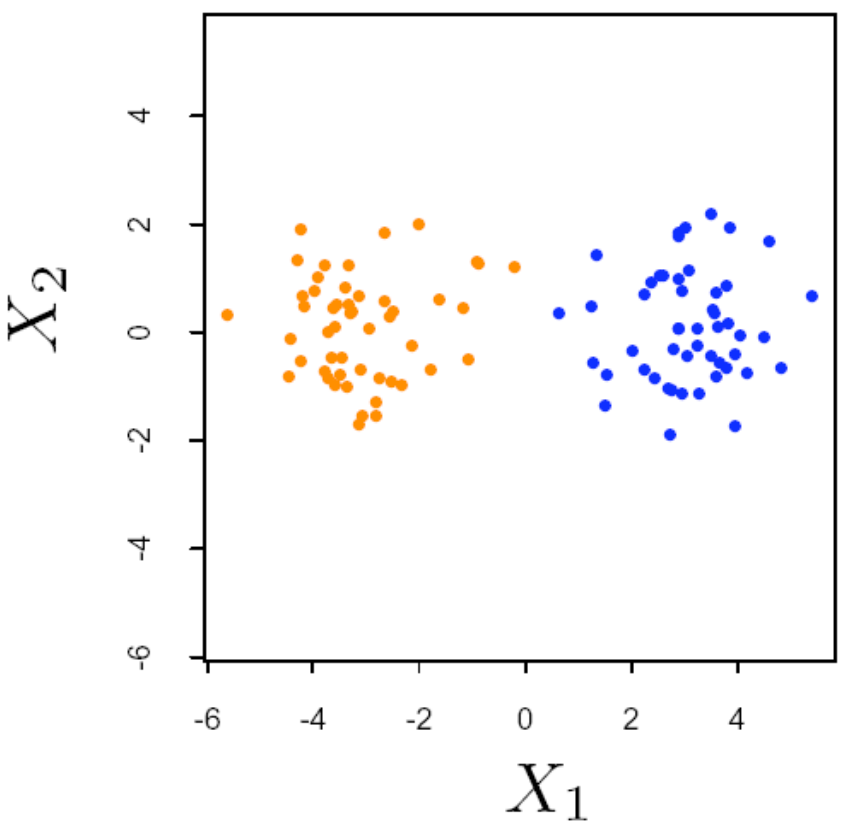
Dissimilarity Based on Attributes

- For instance, for squared error distance, the average dissimilarity of *j*th attribute is twice the sample estimate of the variance

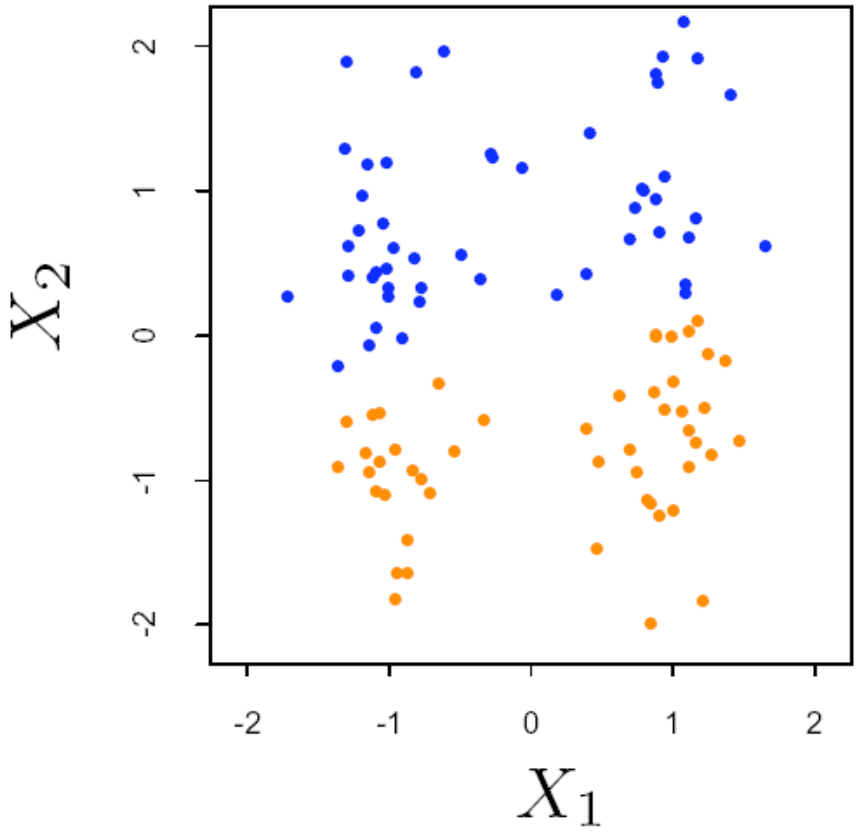
$$\bar{d}_j = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{i'=1}^{n} (x_{ij} - x_{i'j})^2 = 2var_j$$

- The relative importance of each attribute is proportional to its variance over the data set
- Setting $w_j \propto 1/\bar{d}_j$ (equivalent to standardizing the data) is not always helpful since attributes may enter dissimilarity to a different degree

Case Studies



Simulated data, 2-means without standardization



Simulated data, 2-means with standardization

Learning Dissimilarity

- Specifying an appropriate dissimilarity is far more important than choice of clustering algorithm
- Suppose a user indicates certain objects are considered by them to be “similar”:

$$(x_i, x_j) \in \mathcal{S} \text{ if } x_i \text{ and } x_j \text{ are similar}$$

- Consider learning a dissimilarity of form

$$D(x_i, x_j) = ||x_i - x_j||_A = \sqrt{(x_i - x_j)^T A (x_i - x_j)}$$

  - If *A* is diagonal,it corresponds to learn different weights for different attributes
  - Generally, *A* parameterizes a family of Mahalanobis distance
- Leaning such a dissimilarity is equivalent to finding a rescaling of data; replace x by $A^{1/2}x$

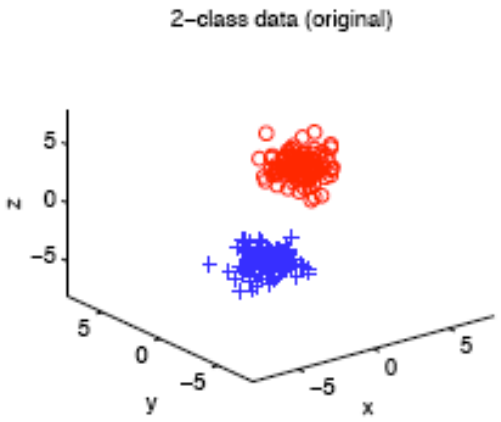
Learning Dissimilarity

- A simple way to define a criterion for the desired dissimilarity:

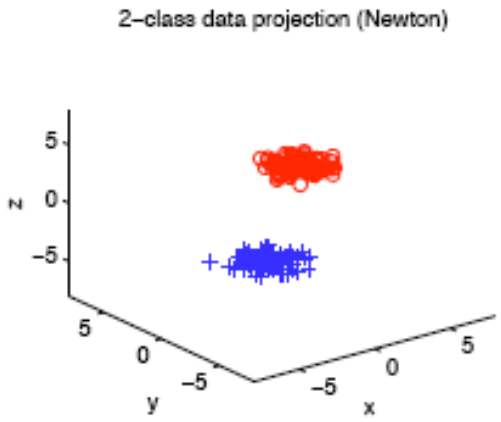
$$\begin{aligned} \min_{A} \quad & \sum_{(x_i,x_j)\in\mathcal{S}} ||x_i - x_j||_A^2 \\ \text{s.t.} \quad & \sum_{(x_i,x_j)\in\mathcal{D}} ||x_i - x_j||_A \geq 1, \\ & A \succeq 0. \end{aligned}$$

- A convex optimization problem, could be solved by gradient descent and iterative projection
- For details, see [Xing, Ng, Jordan, Russell ’03]

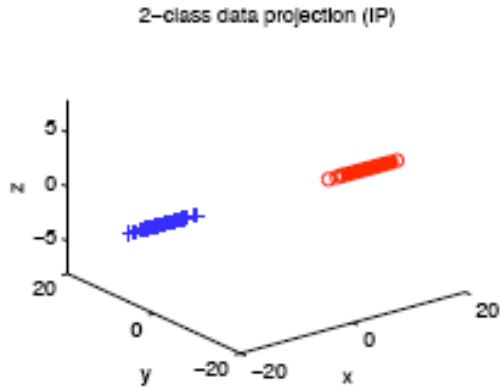
Learning Dissimilarity



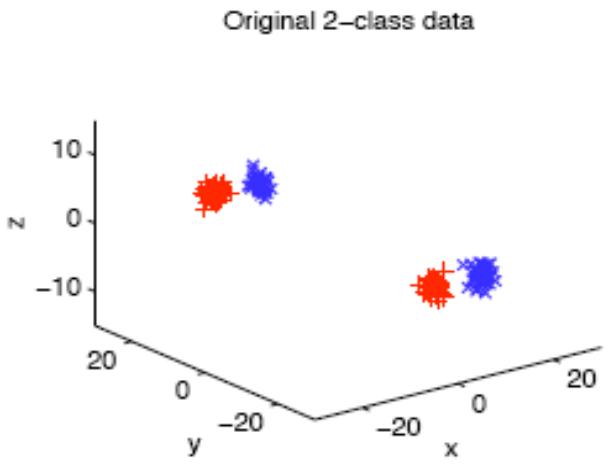
(a)



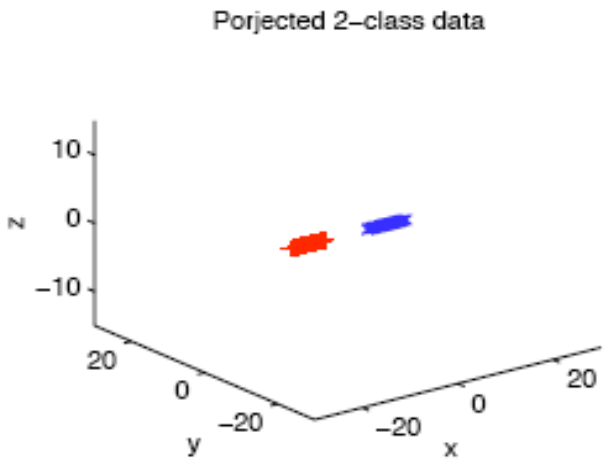
(b)



(c)



(a)



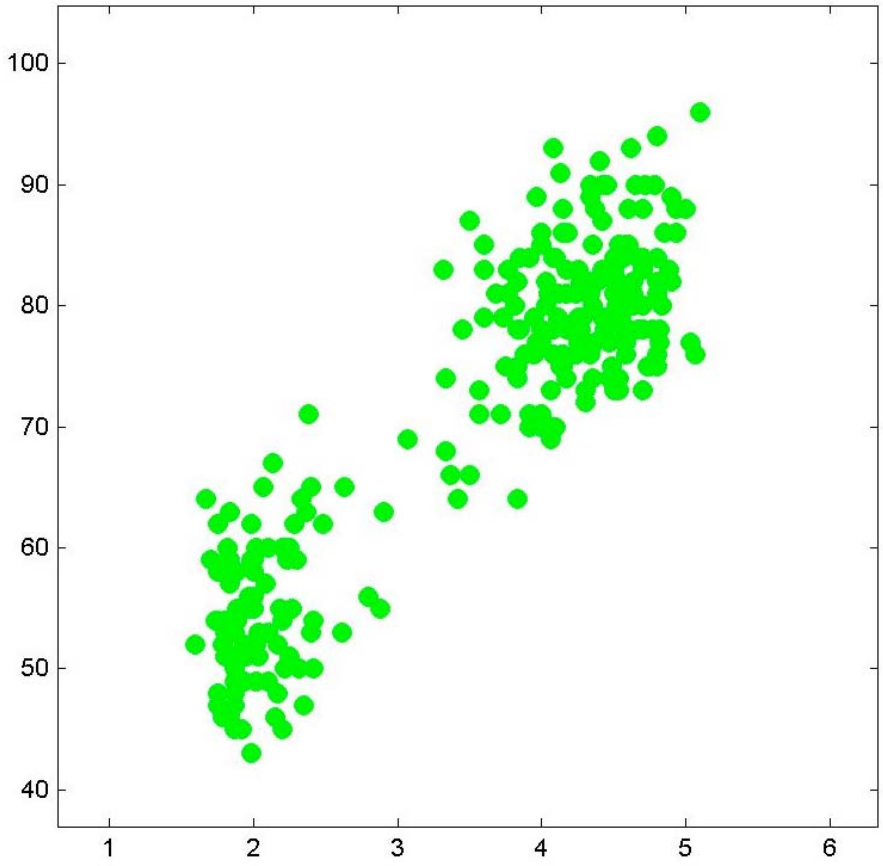
(b)

Outline

- Introduction
  - Unsupervised learning
  - What is clustering? Application
- Dissimilarity (similarity) of objects
- Clustering algorithm
  - K-means, VQ, K-medoids
  - Gaussian mixture model (GMM), EM
  - Hierarchical clustering
  - Spectral clustering

Old Faithful Data Set

Time between eruptions (minutes)



Duration of eruption (minutes)

K-means

- Idea: represent a data set in terms of K clusters, each of which is summarized by a prototype μ_k
  - Usually applied to Euclidean distance (possibly weighted, only need to rescale the data)
- Each data is assigned to one of K clusters
  - Represented by responsibilities $r_{ik} \in \{0, 1\}$ such that $\sum_{k=1}^{K} r_{ik} = 1$ for all data indices *i*

K-means

- Example: 4 data points and 3 clusters

$$(r_{ik}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Cost function:

$$J = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \|x_i - \mu_k\|^2$$

data

responsibilities

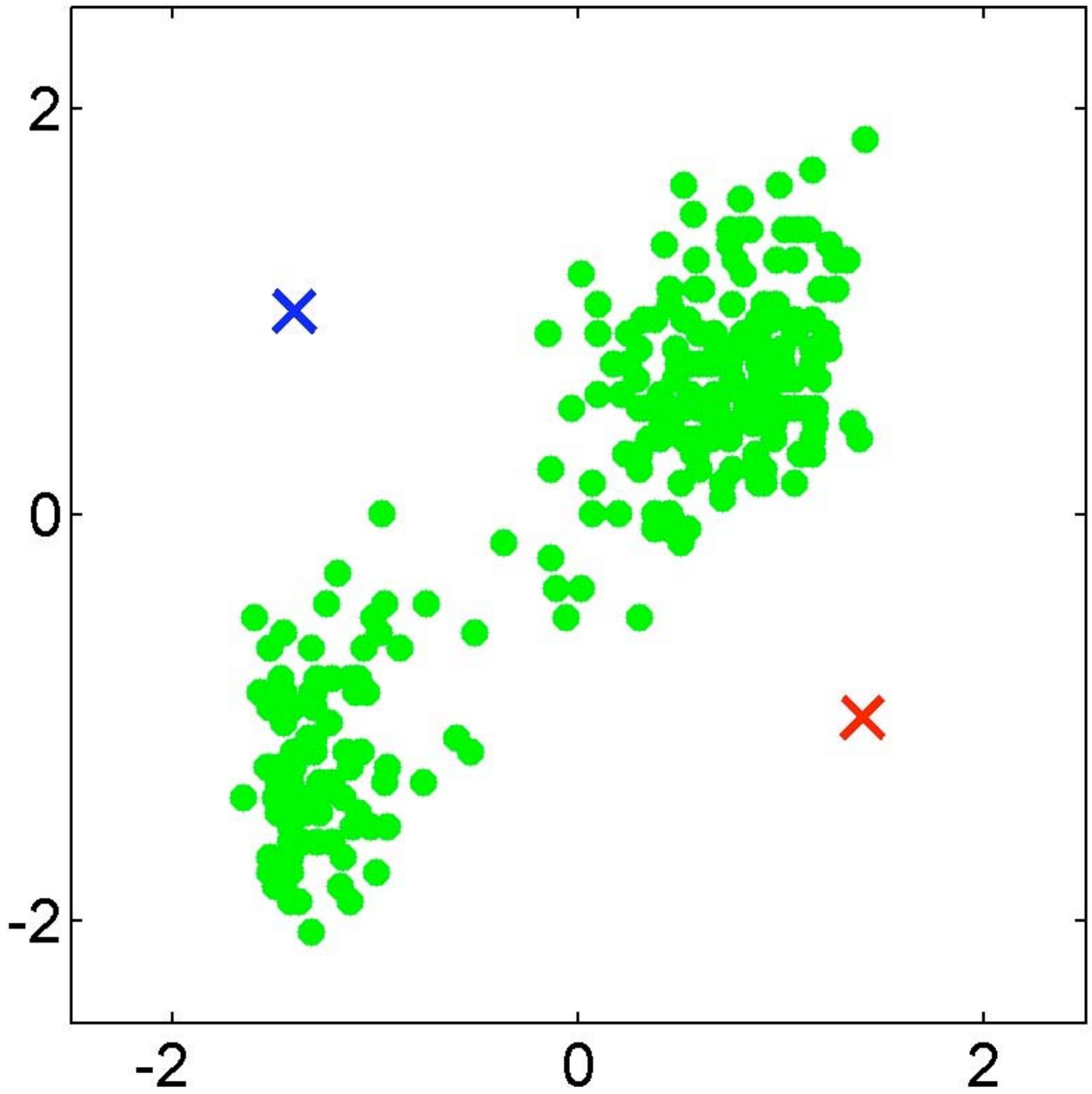
prototypes

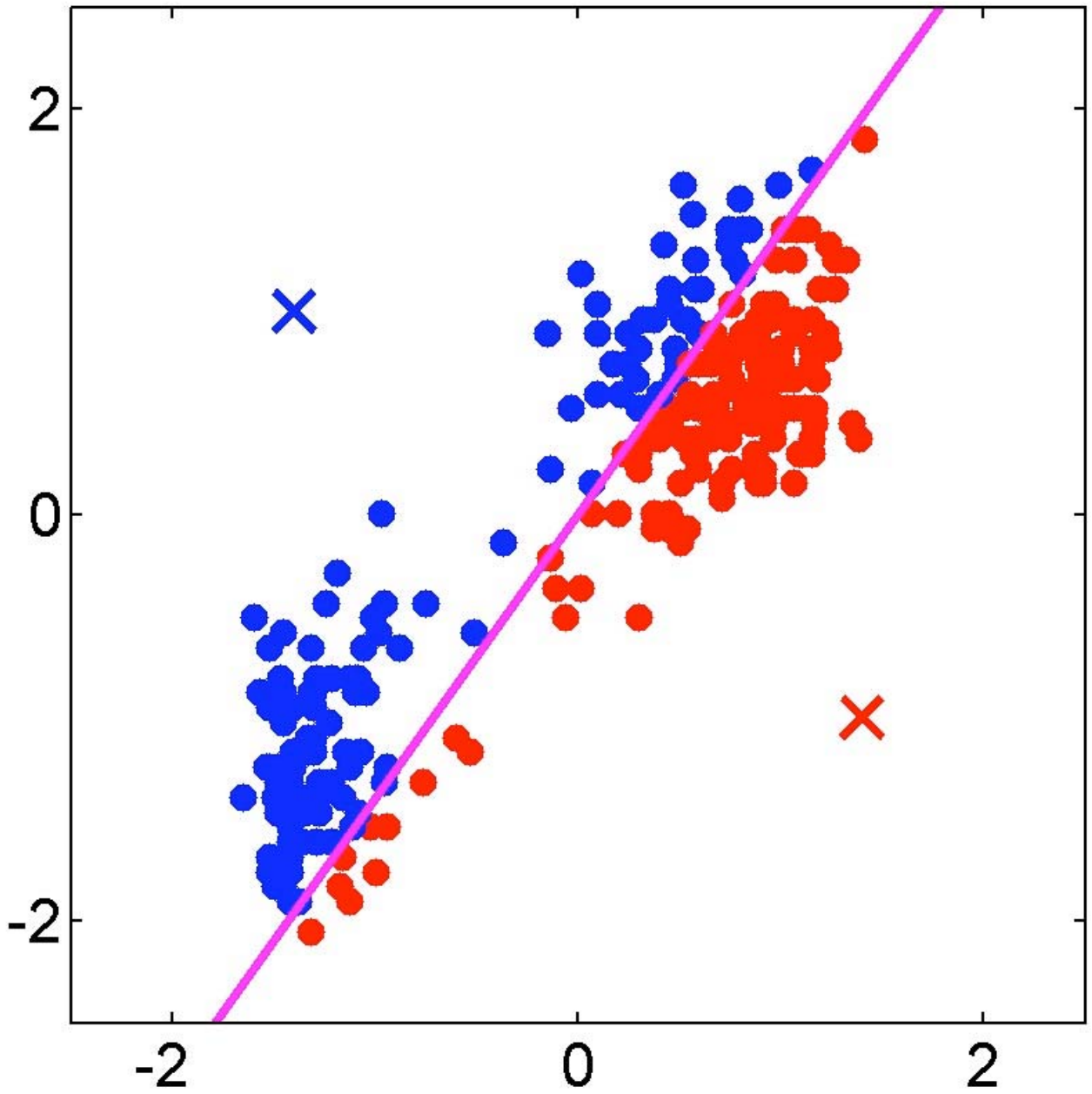
Minimizing the Cost Function

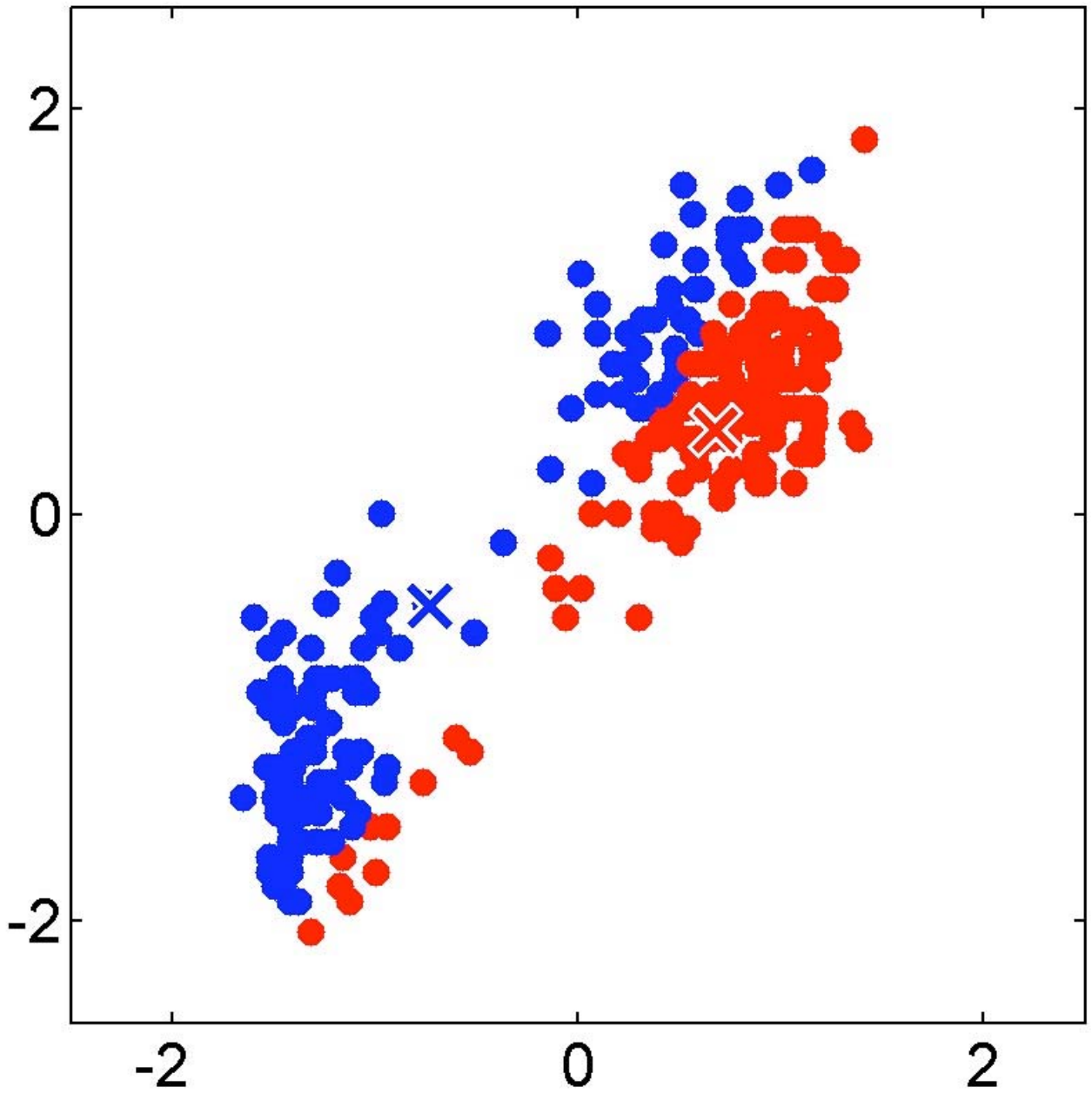
- Chicken and egg problem, have to resort to iterative method
- E-step: minimize J w.r.t. r_{ik}
  - assigns each data point to nearest prototype
- M-step: minimize J w.r.t μ_k
  - gives

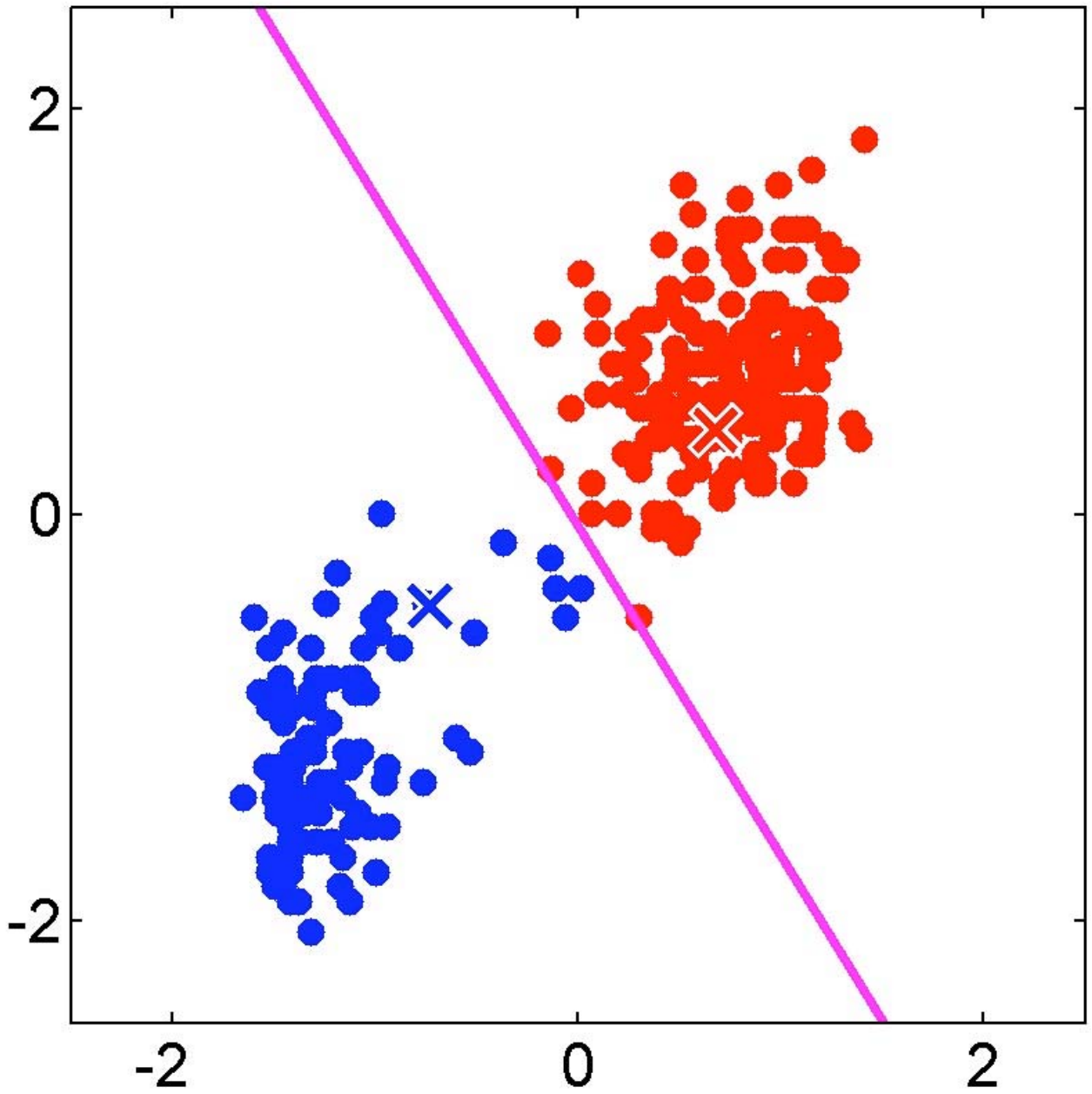
$$\mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$$

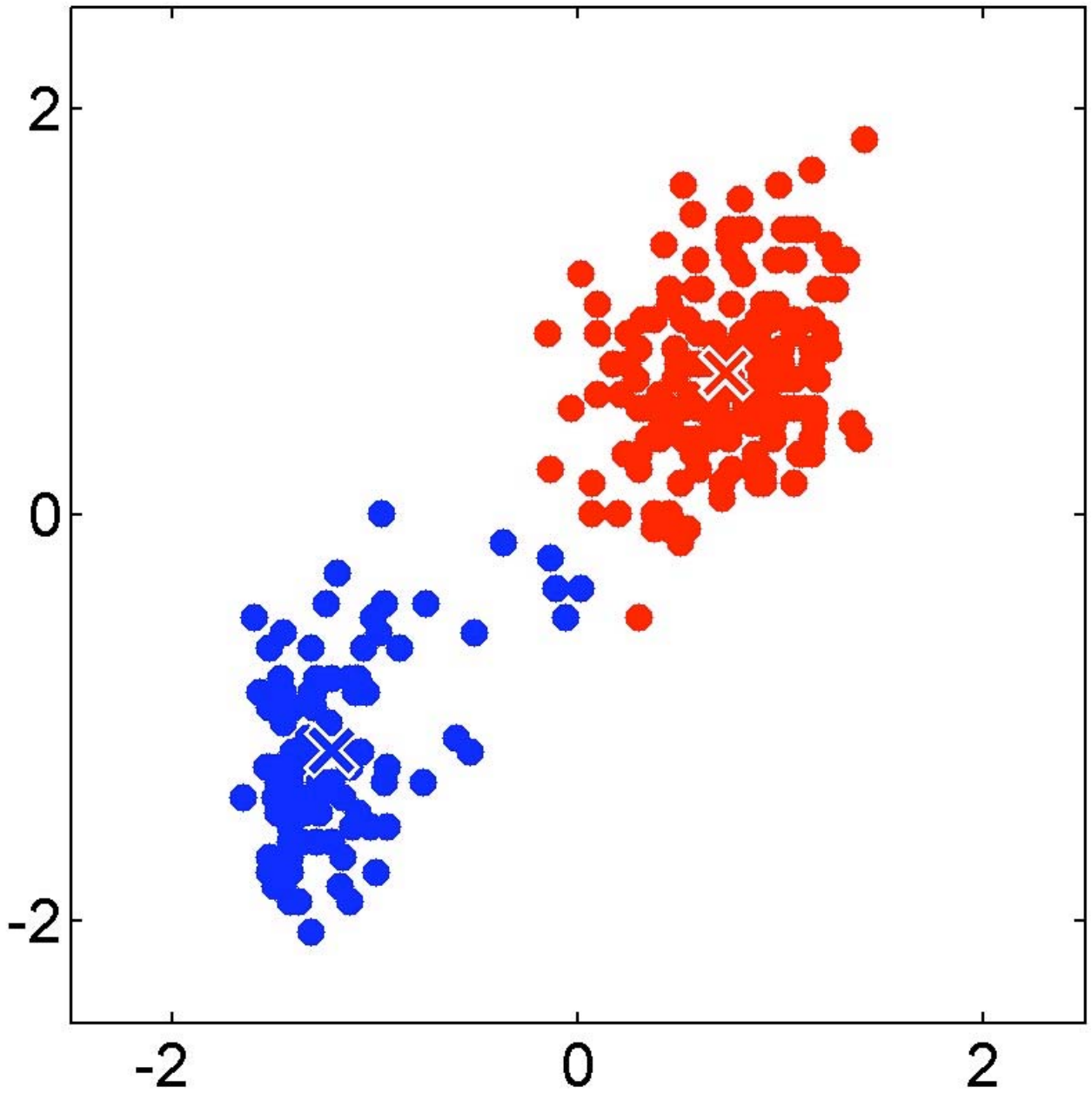
  - each prototype set to the mean of points in that cluster
- Convergence guaranteed since there is a finite number of possible settings for the responsibilities
- only finds local minima, should start the algorithm with many different initial settings

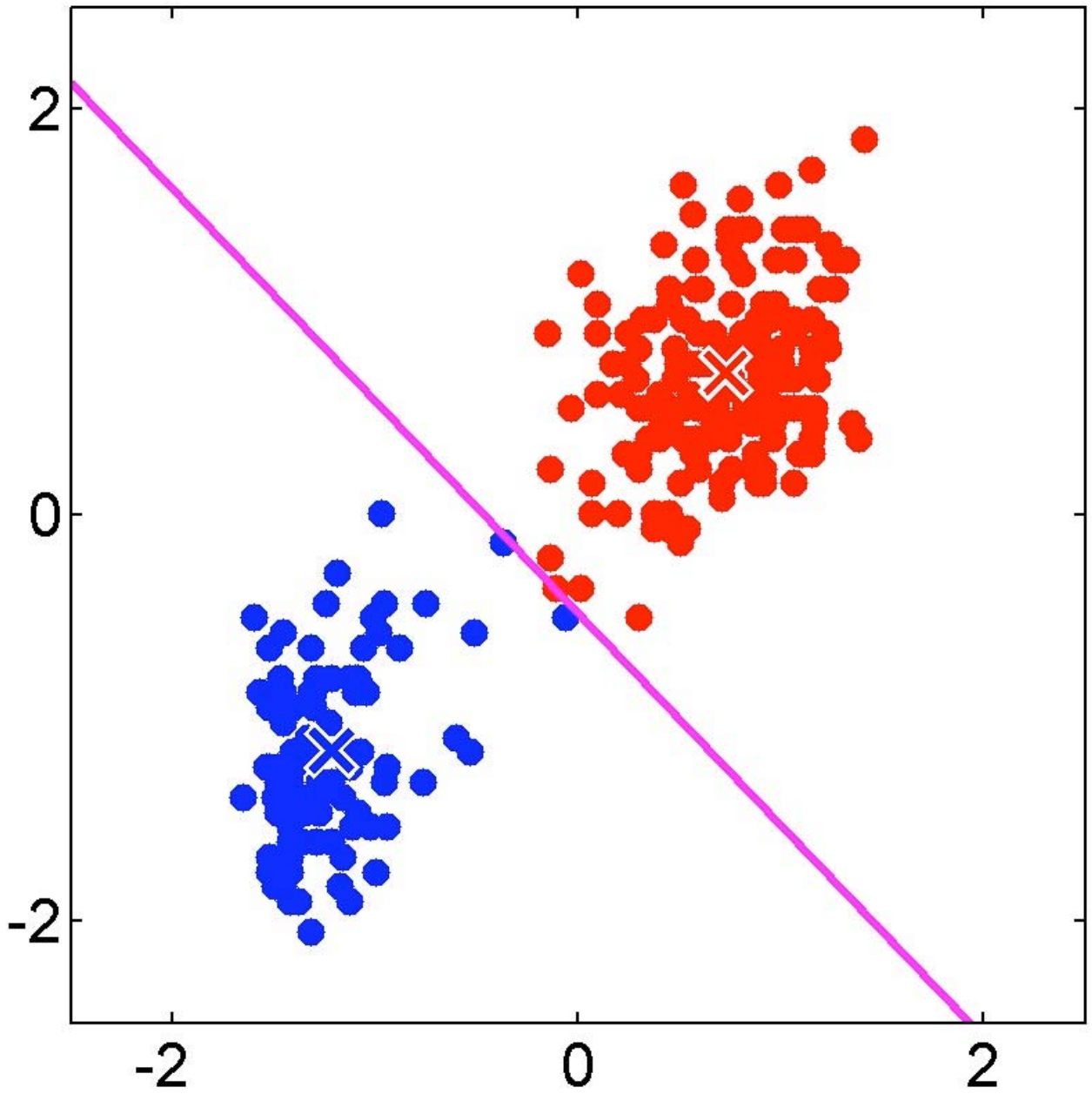

2
0
-2
-2
0
2

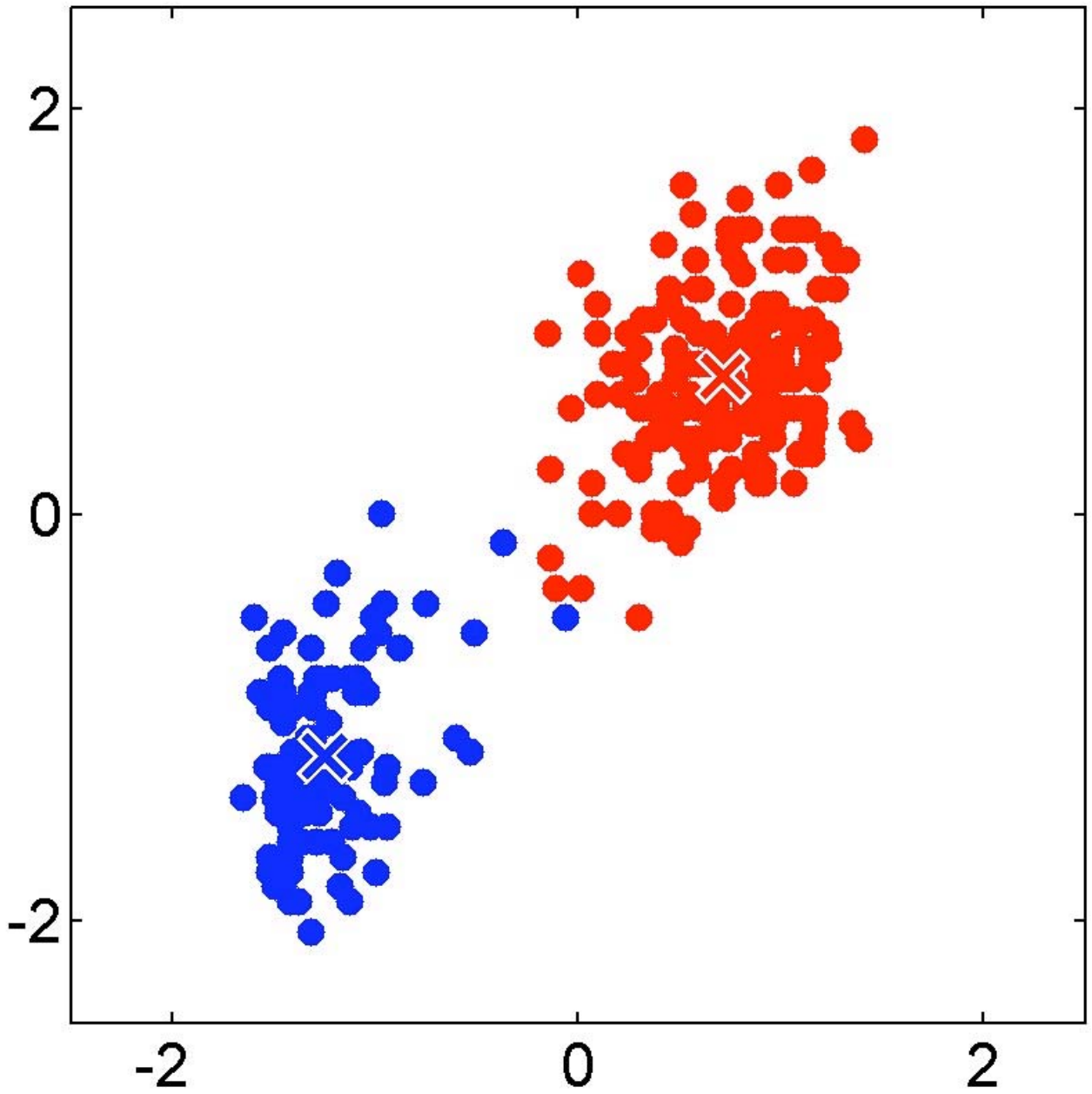

2
0
-2
-2
0
2

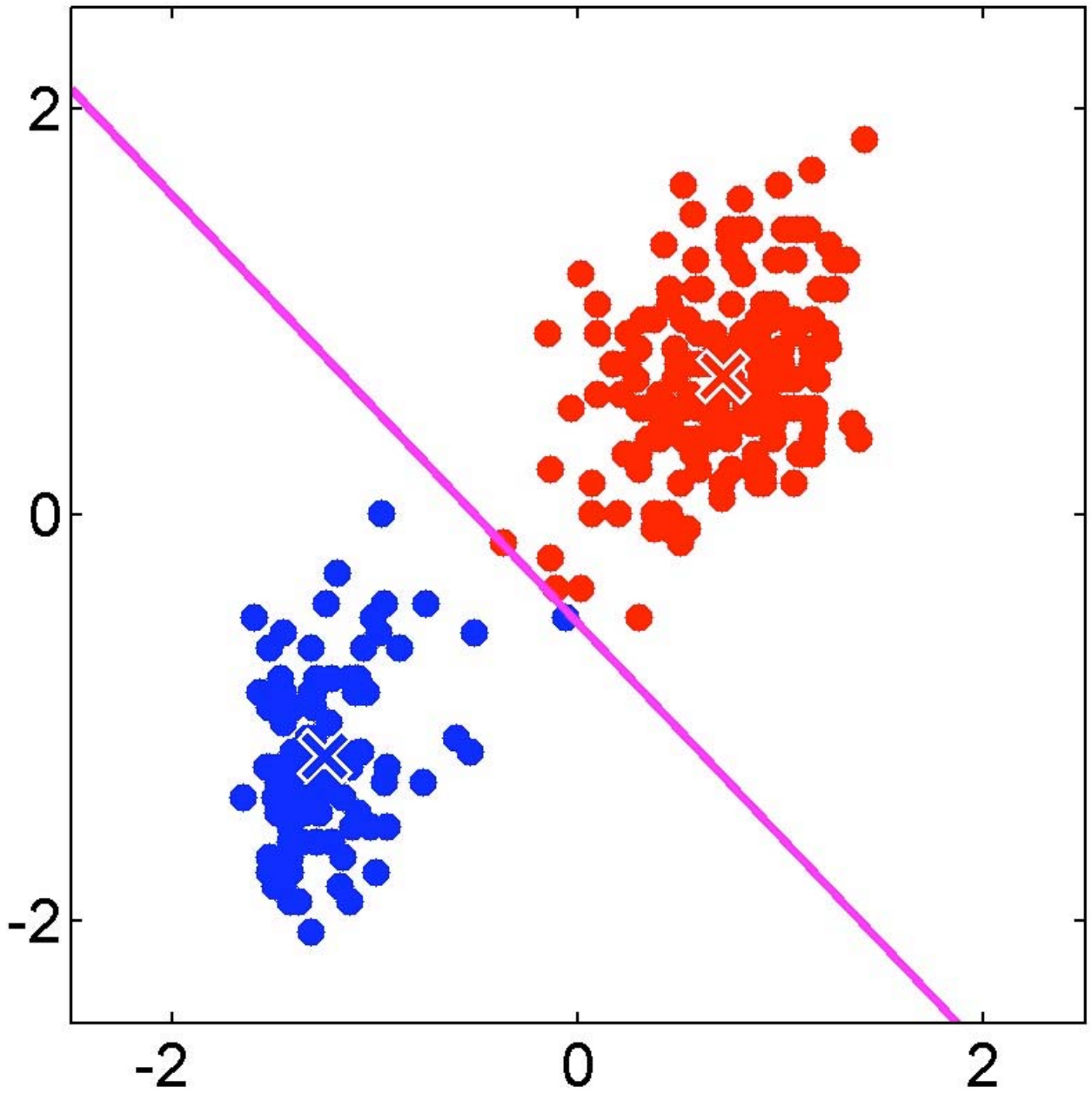

2
0
-2
-2
0
2

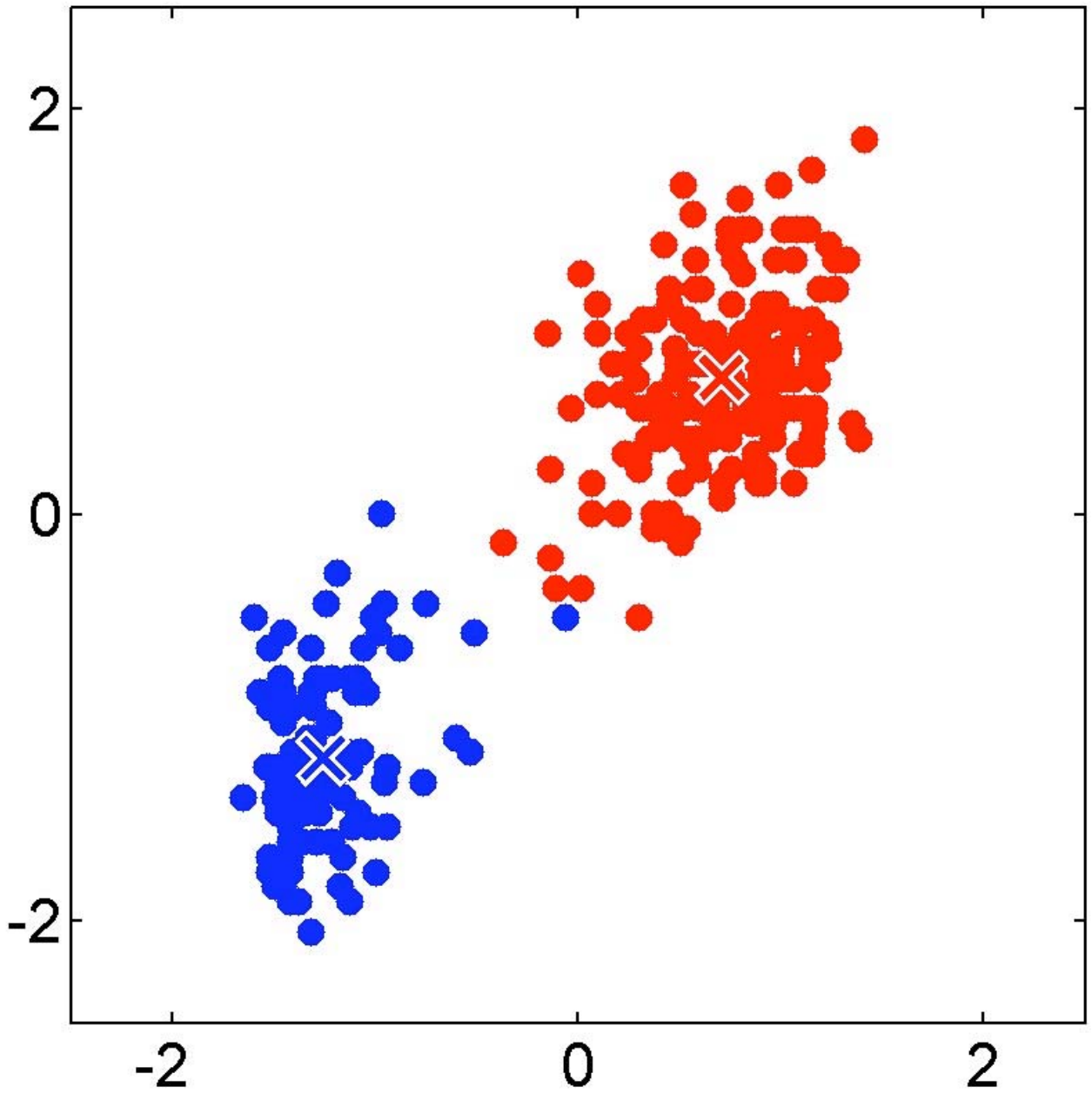

2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

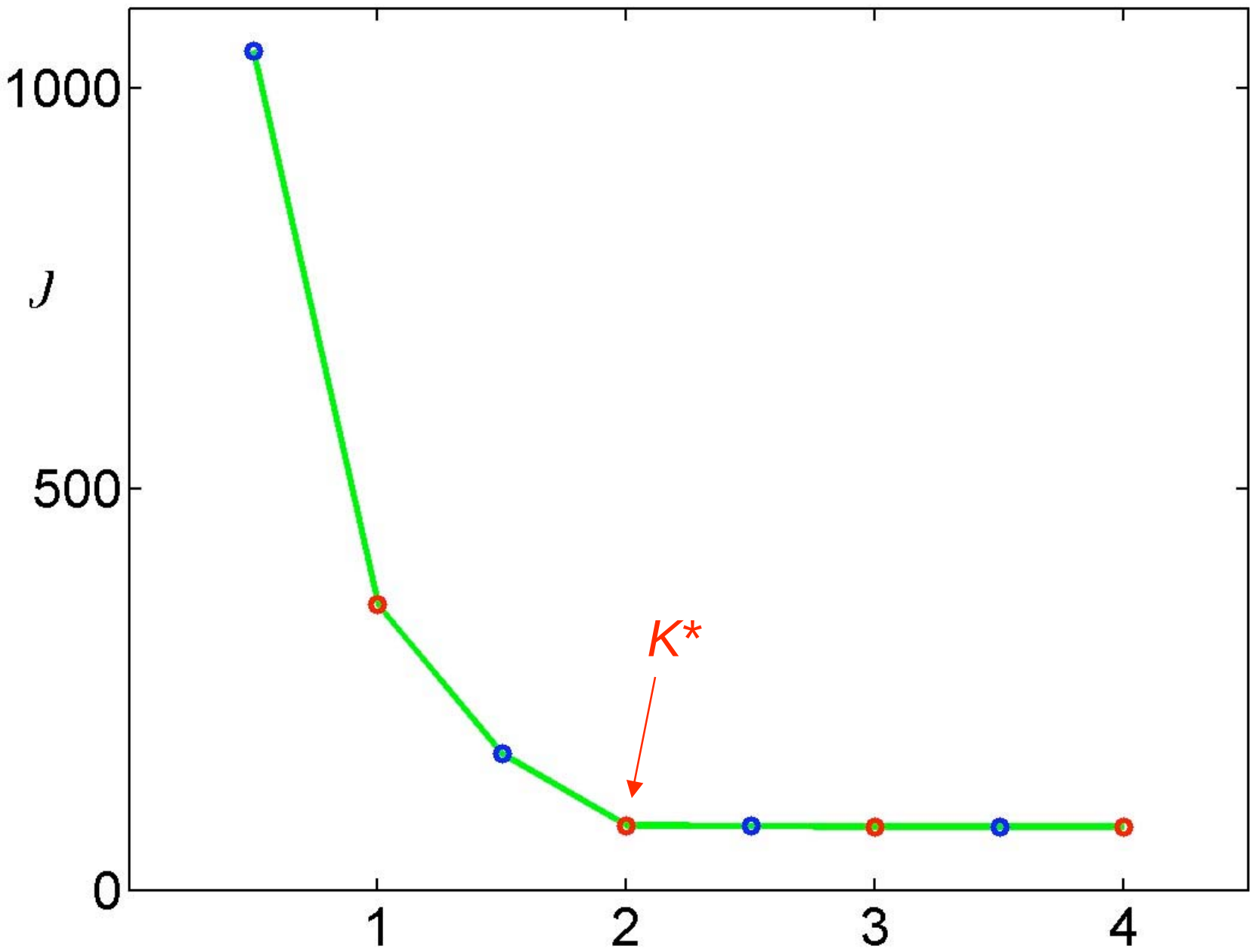

2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

How to Choose *K*?

- In some cases it is known apriori from problem domain
- Generally, it has to be be estimate from data and usually selected by some heuristics in practice
- The cost function *J* generally decrease with increasing *K*
- Idea: Assume that *K** is the right number
  - We assume that for *K*<*K** each estimated cluster contains a subset of true underlying groups
  - For *K*>*K** some natural groups must be split
  - Thus we assume that for *K*<*K** the cost function falls substantially, afterwards not a lot more


1000
J
500
0
1
2
3
4
K*

Vector Quantization

- Application of K-means for compressing signals
- 1024×1024 pixels, 8-bit grayscale
- 1 megabyte in total
- Break image into 2×2 blocks of pixels resulting in 512 ×512 blocks, each represented by a vector in $\mathbb{R}^4$
- Run K-means clustering
  - Known as Lloyd's algorithm
  - Each 512 ×512 block is approximated by its closest cluster centroid, known as codeword
  - Collection of codeword is called the codebook

Sir Ronald A. Fisher (1890-1962)

Vector Quantization

- Application of K-means for compressing signals
- 1024×1024 pixels, 8-bit grayscale
- 1 megabyte in total
- Storage requirement
  - $K\times 4$ real numbers for the codebook (negligible)
  - $\log_2 K$ bits for storing the code for each block (can also use variable length code)
  - The ratio is:

$$\log_2 K /(4\cdot 8)$$

bits per block in compressed image

pixels per block

bits per pixel in uncompressed image

  - K = 200, the ratio is 0.239

K =200

Vector Quantization

- Application of K-means for compressing signals
- 1024×1024 pixels, 8-bit grayscale
- 1 megabyte in total
- Storage requirement
  - $K \times 4$ real numbers for the codebook (negligible)
  - $\log_2 K$ bits for storing the code for each block (can also use variable length code)
  - The ratio is:

$$\log_2 K /(4 \cdot 8)$$

bits per block in compressed image

pixels per block

bits per pixel in uncompressed image

  - K = 4, the ratio is 0.063

K = 4

K-medoids

- K-means algorithm is sensitive to outliers
  - An object with an extremely large distance from others may substantially distort the results, i.e., centroid is not necessarily inside a cluster
- Idea: instead of using mean of data points within the clusters, prototypes of clusters are restricted to be one of the points assigned to the cluster (medoid)
  - given responsibilities (assignments of points to clusters), find one of the point within the cluster that minimizes total dissimilarity to other points in that cluster
- Generally, computation of a cluster prototype increases from n to n^2

Limitations of K-means

- Hard assignments of data points to clusters
  - Small shift of a data point can flip it to a different cluster
  - Solution: replace hard clustering of K-means with soft probabilistic assignments (GMM)
- Hard to choose the value of *K*
  - As K is increased, the cluster memberships can change in an arbitrary way, the resulting clusters are not necessarily nested
  - Solution: hierarchical clustering

The Gaussian Distribution

- Multivariate Gaussian

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right\}$$

mean covariance

- Maximum likelihood estimation

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\hat{\Sigma} = \frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})(x_i-\hat{\mu})^T$$

Gaussian Mixture

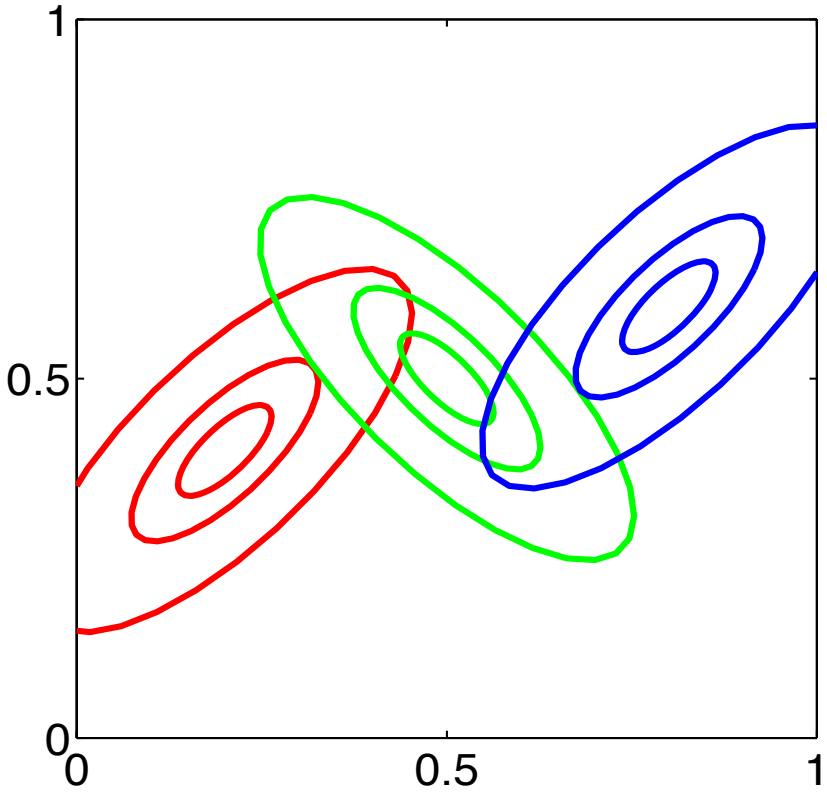
- Linear combination of Gaussians

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \quad \text{where} \quad \sum_{k=1}^{K} \pi_k = 1, \quad 0 \leqslant \pi_k \leqslant 1$$

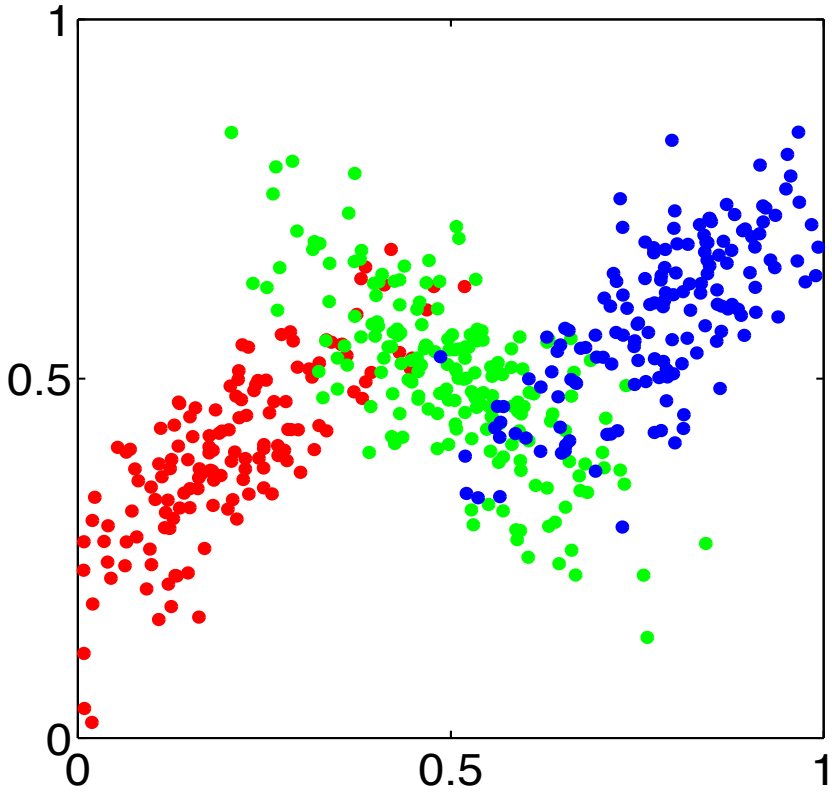
parameters to be estimated

- To generate a data point:
  - first pick one of the components with probability π_k
  - then draw a sample x_i from that component
- Each data is generated by one of *K* Gaussians, a latent variable $z_i = (z_{i1}, \ldots, z_{iK})$ is associated with each x_i , $\sum_{k=1}^{K} z_{ik} = 1$ and $p(z_{ik} = 1) = \pi_k$

Example: Mixture of 3 Gaussians

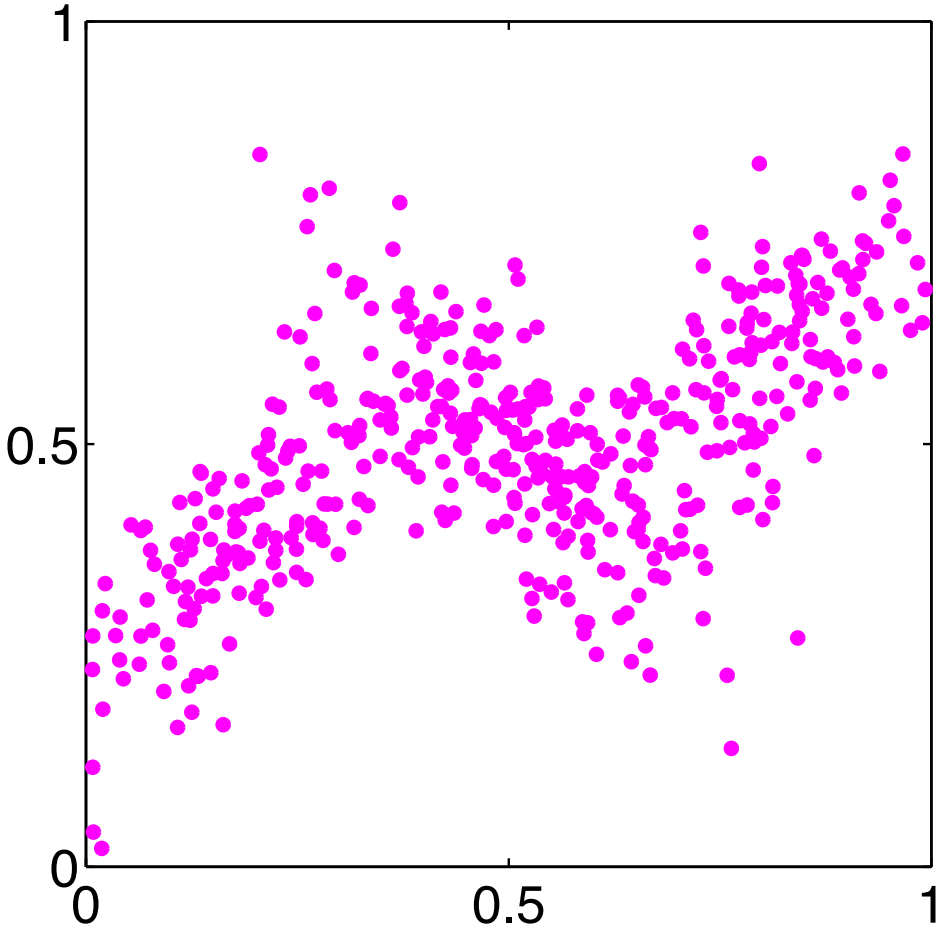


$p(x|k)$



Synthetic Data Set,
the colours are
latent variables

Synthetic Data Set Without Colours



Fitting the Gaussian Mixture

- Given the complete data set $(x, z) = (x_i, z_i)_{i=1,\ldots,n}$
  - the complete log likelihood

$$\ln p(x, z|\pi, \mu, \Sigma) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \{\ln \pi_k + \ln \mathcal{N}(x_i|\mu_k, \Sigma_k)\}$$

  - trivial closed-form solution: fit each component to the corresponding set of data points

$p(x_i|\pi, \mu, \Sigma)$

- Without knowing values of latent variables, we have to maximize the incomplete log likelihood:

$$\ln p(x|\pi, \mu, \Sigma) = \sum_{i=1}^{n} \ln\{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)\}$$

  - Sum over components appears inside the logarithm, no closed-form solution

EM Algorithm

- E-step: for given parameter values we can compute the expected values of the latent variables (responsibilities of data points)

$$
\begin{aligned}
r_{ik} \equiv E(z_{ik}) &= p(z_{ik}=1|x_i,\pi,\mu,\Sigma) \\
&= \frac{p(z_{ik}=1)p(x_i|z_{ik}=1,\pi,\mu,\Sigma)}{\sum_{k=1}^{K} p(z_{ik}=1)p(x_i|z_{ik}=1,\pi,\mu,\Sigma)} \\
&= \frac{\pi_k \mathcal{N}(x_i|u_k,\Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|u_k,\Sigma_k)}
\end{aligned}
$$

Bayes rule

  - Note that $r_{ik} \in [0,1]$ instead of $\{0,1\}$ but we still have $\sum_{k=1}^{K} r_{ik} = 1$ for all i

EM Algorithm

- M-step: maximize the expected complete log likelihood

$$E[\ln p(x, z|\pi, \mu, \Sigma)] = \sum_{i=1}^{n} \sum_{k=1}^{K} r_{ik} \{\ln \pi_k + \ln \mathcal{N}(x_i|\mu_k, \Sigma_k)\}$$

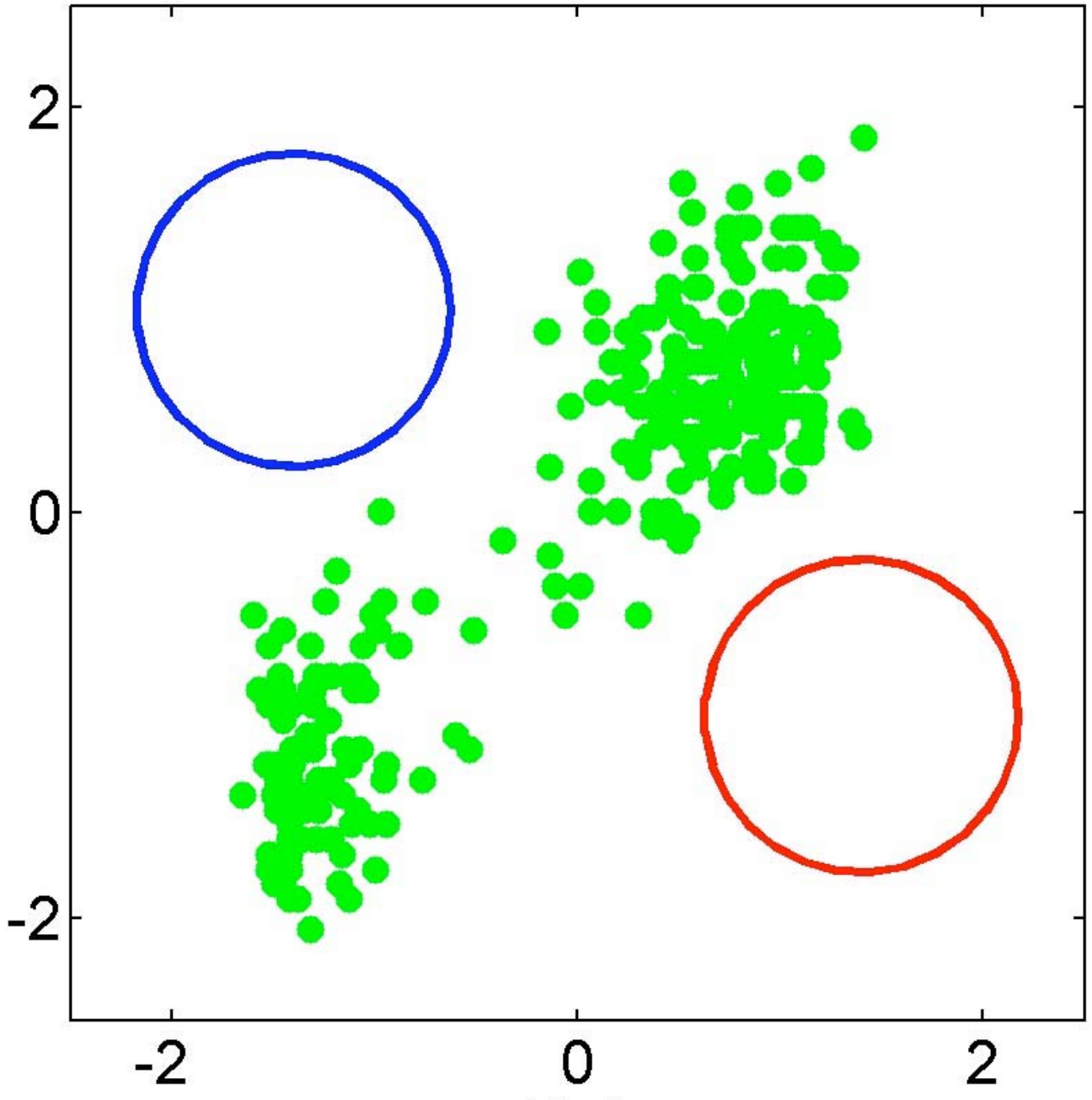
  - update parameters:

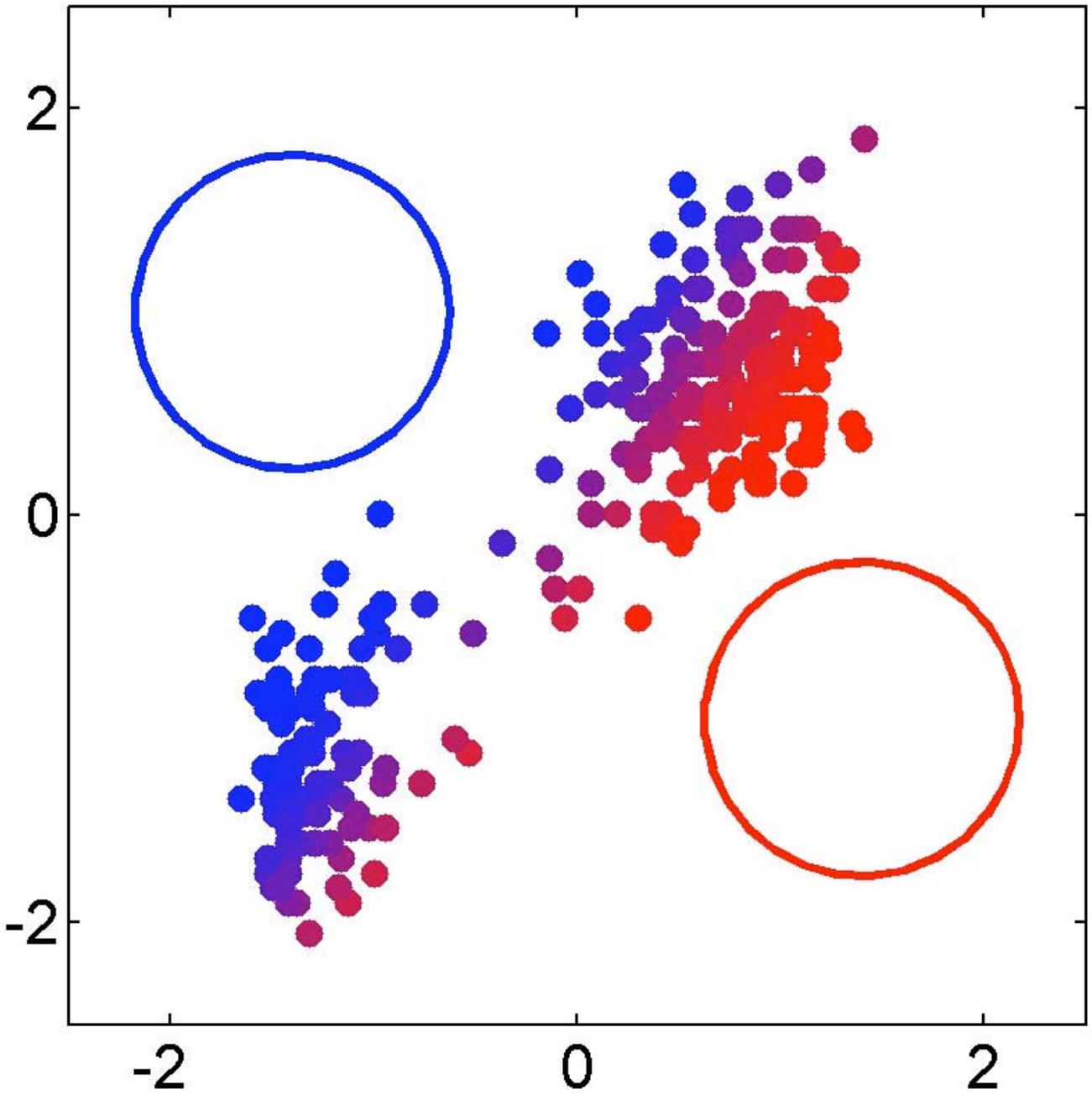
$$\pi_k = \frac{\sum_i r_{ik}}{n} \qquad \mu_k = \frac{\sum_i r_{ik} x_i}{\sum_i r_{ik}}$$

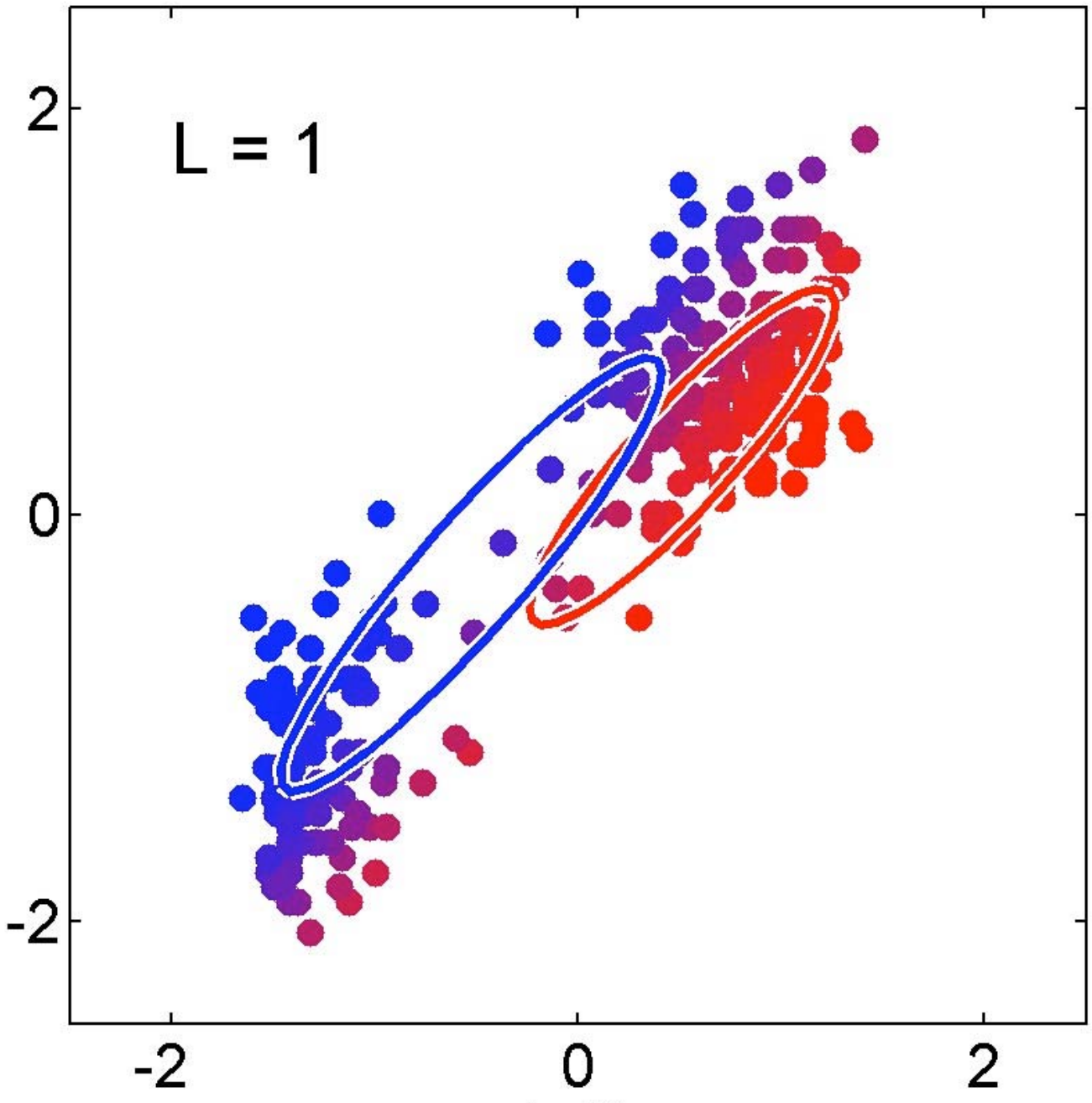
$$\Sigma_k = \frac{\sum_i r_{ik}(x_i - \mu_k)(x_i - \mu_k)^T}{\sum_i r_{ik}}$$

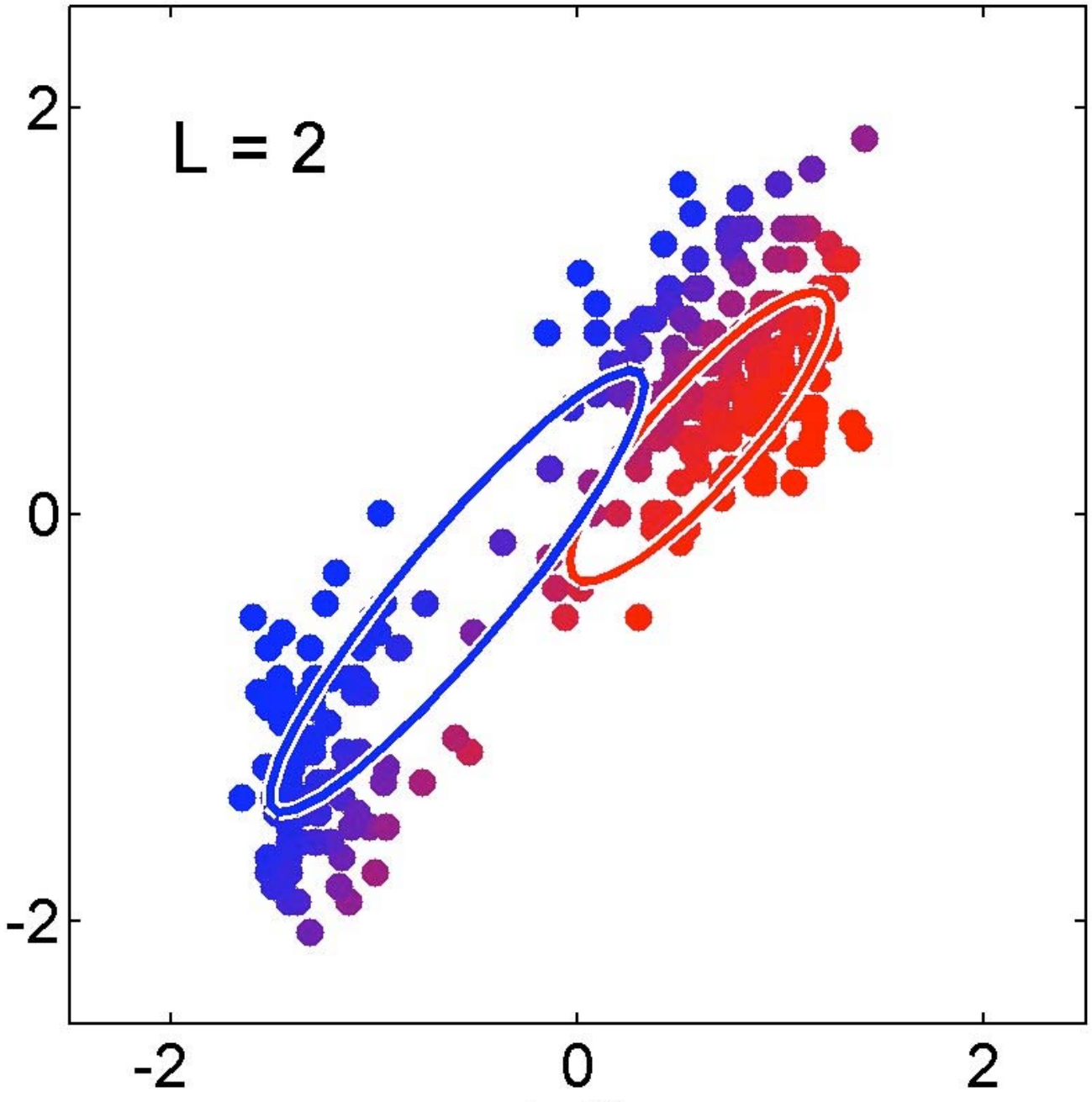
EM Algorithm

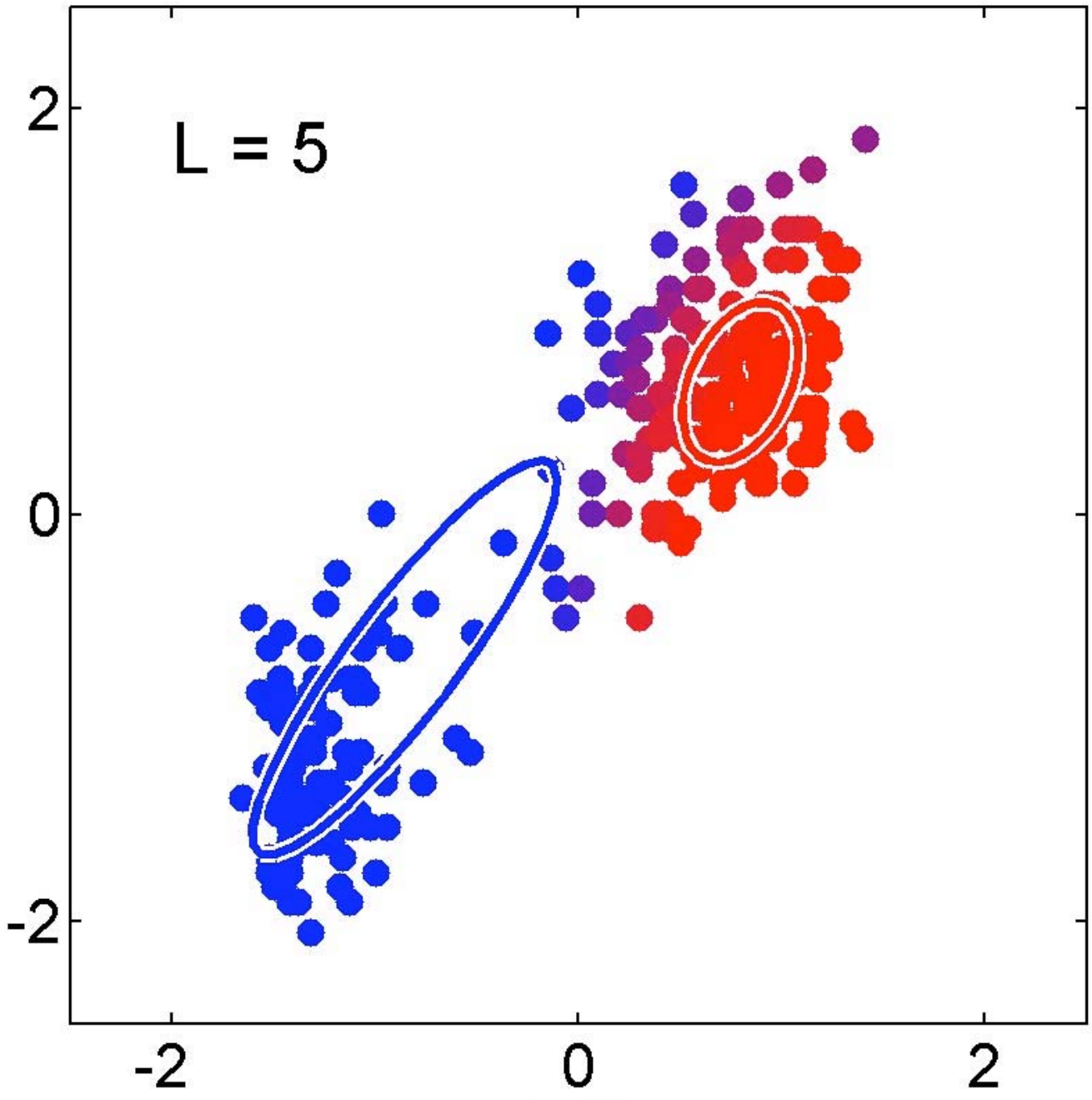
- Iterate E-step and M-step until the log likelihood of data does not increase any more
  - converge to local optima
  - need to restart algorithm with different initial guess of parameters (as in K-means)
- Does maximizing the expected complete log likelihood increases the log likelihood of data?
  - Yes. Coordinate ascent algorithm, see Chapter 8 of Jordan's book
- Relation to K-means
  - Consider GMM with common covariance $\Sigma_k = \delta^2 I$
  - As $\delta^2 \to 0, r_{ik} \to 0 \text{ or } 1$, two methods coincide

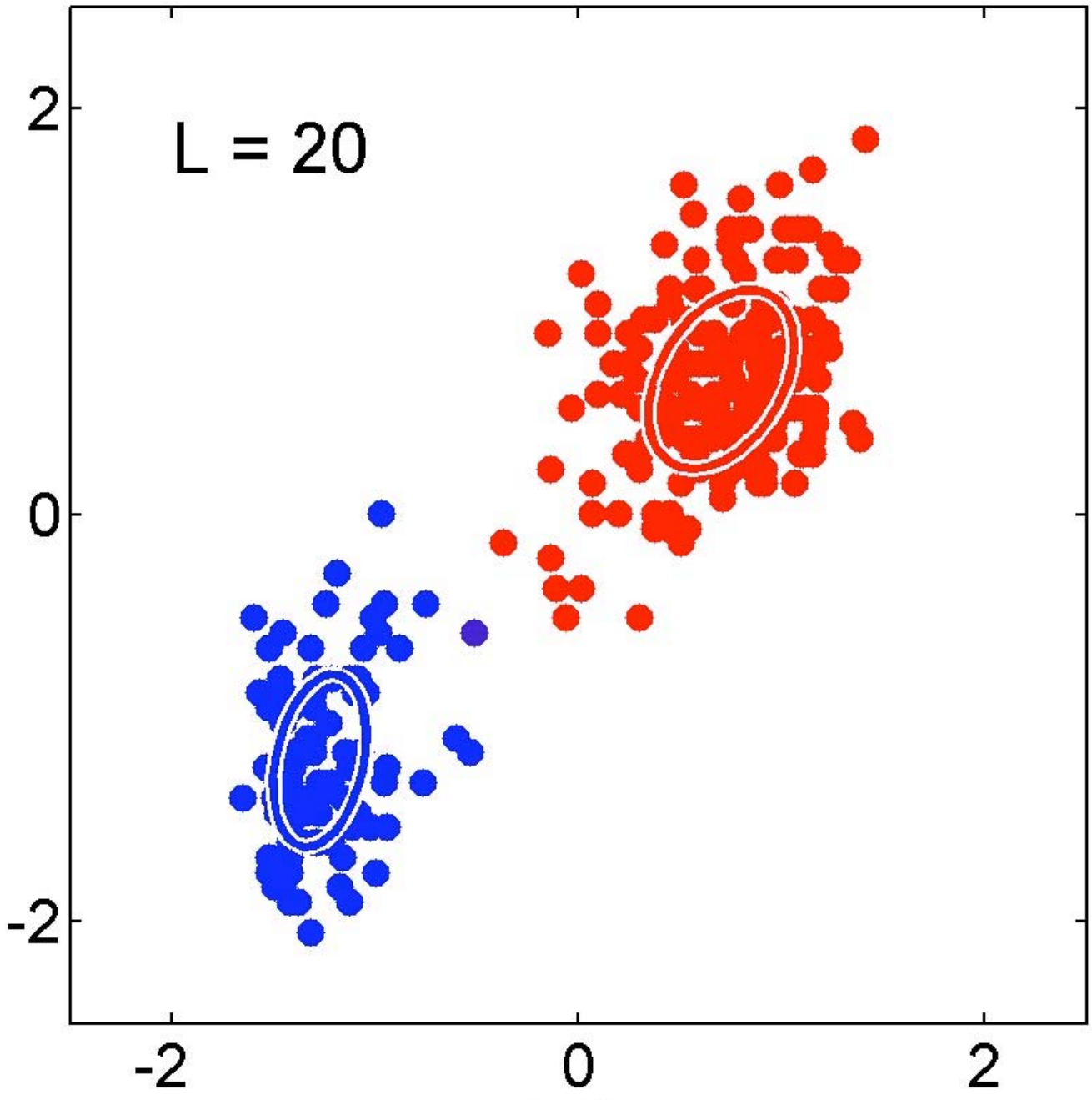

2
0
-2
-2
0
2


2
0
-2
-2
0
2


L = 1
2
0
-2
-2
0
2

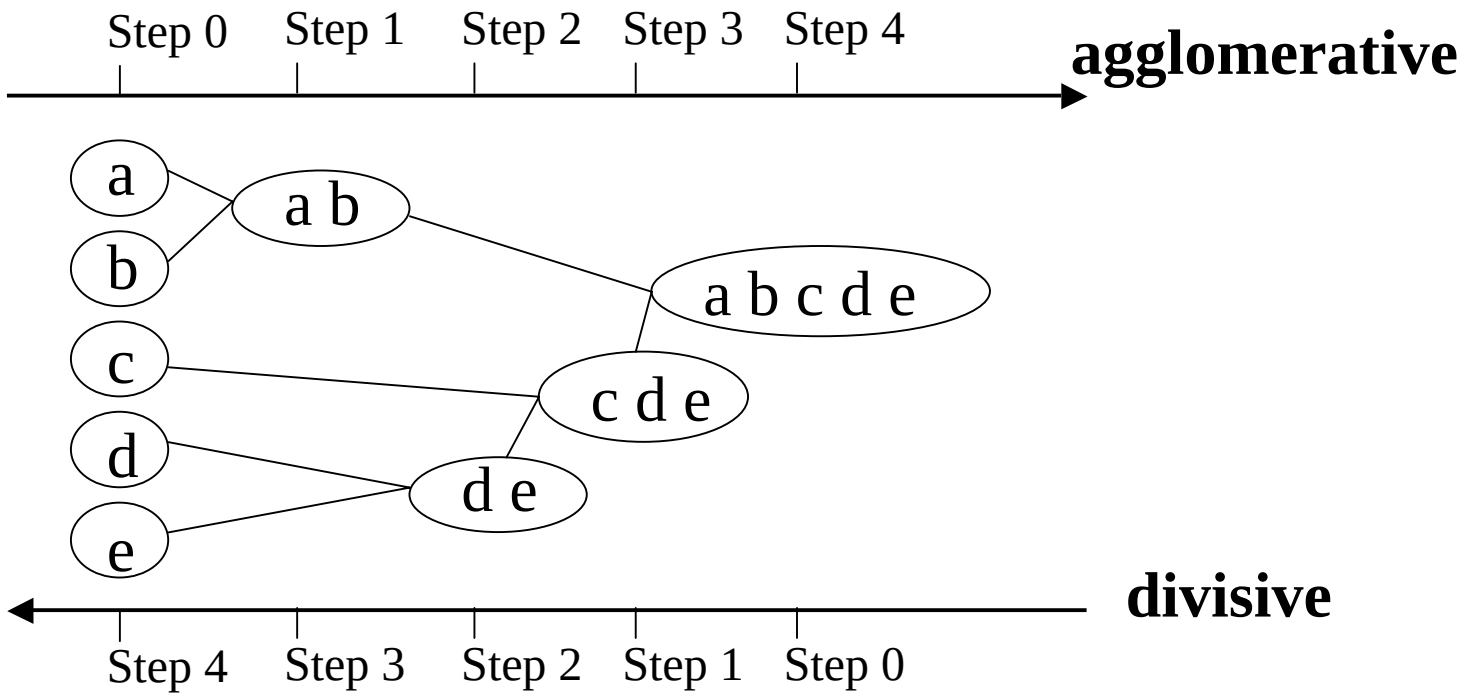

L = 2
2
0
-2
-2
0
2


L = 5
2
0
-2
-2
0
2


L = 20
2
0
-2
-2
0
2

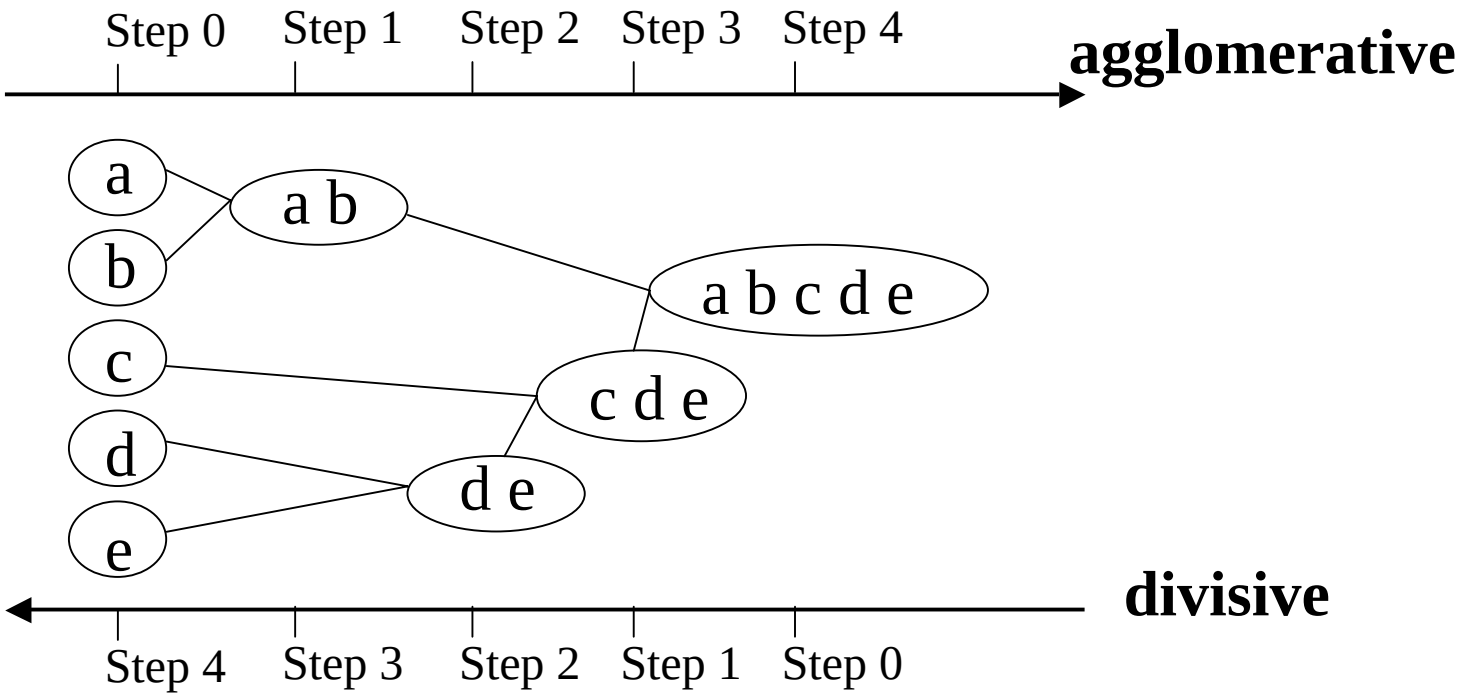
Hierarchical Clustering

- Does not require a preset number of clusters
- Organize the clusters in an hierarchical way
- Produces a rooted (binary) tree (dendrogram)



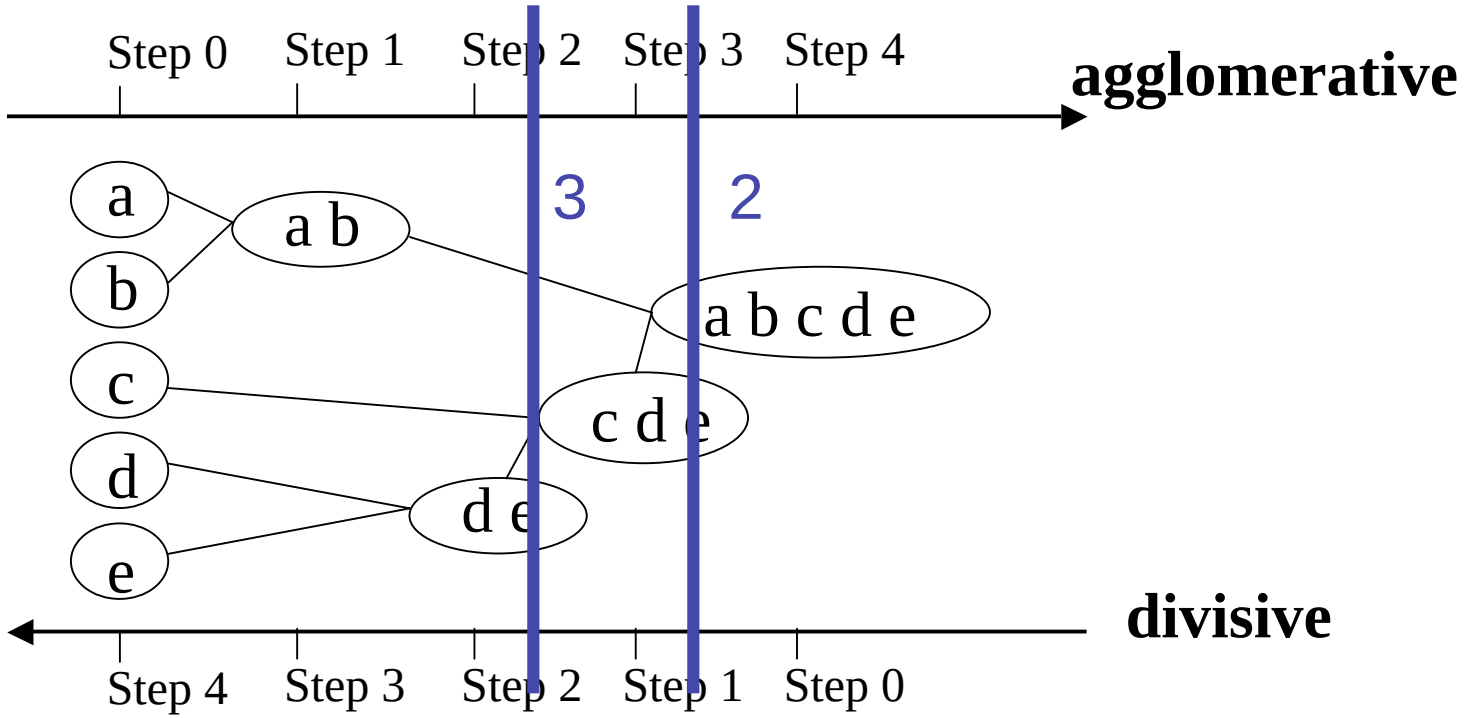
Hierarchical Clustering

- Two kinds of strategy
  - Bottom-up (agglomerative): recursively merge two groups with the smallest between-cluster dissimilarity (defined later on)
  - Top-down (divisive): in each step, split a least coherent cluster (e.g. largest diameter); splitting a cluster is also a clustering problem (usually done in a greedy way); less popular than bottom-up way



Hierarchical Clustering

- User can choose a cut through the hierarchy to represent the most natural division into clusters
  - e.g, choose the cut where intergroup dissimilarity exceeds some threshold



Hierarchical Clustering

- Have to measure the dissimilarity for two disjoint groups *G* and *H*, $D(G, H)$ is computed from pairwise dissimilarities $D(i, j)$ with $i \in G, j \in H$
  - Single Linkage: tends to yield extended clusters

$$D_{SL}(G, H) = \min_{i \in G, j \in H} D(i, j)$$

  - Complete Linkage: tends to yield round clusters

$$D_{CL}(G, H) = \max_{i \in G, j \in H} D(i, j)$$

  - Group Average: tradeoff between them; however, not invariant to monotone transformation of dissimilarity function

$$D_{GA}(G, H) = \frac{1}{n_G n_H} \sum_{i \in G, j \in H} D(i, j)$$

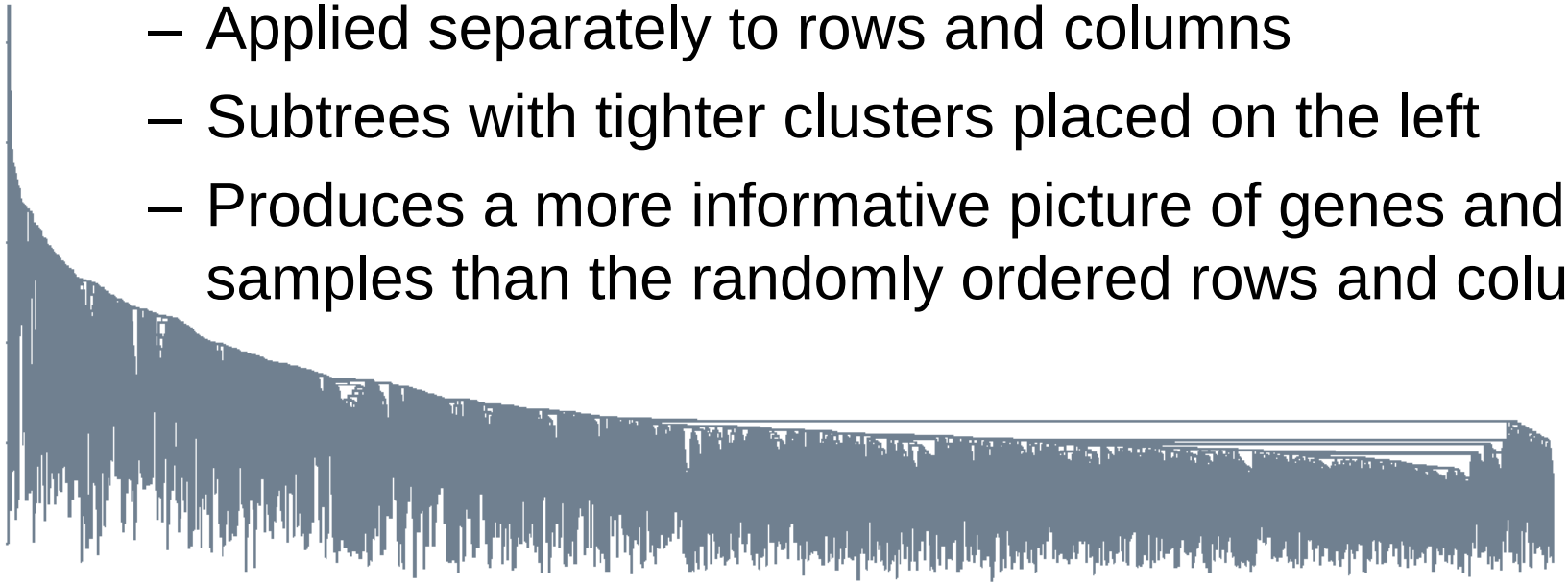
Example: Human Tumor Microarray Data

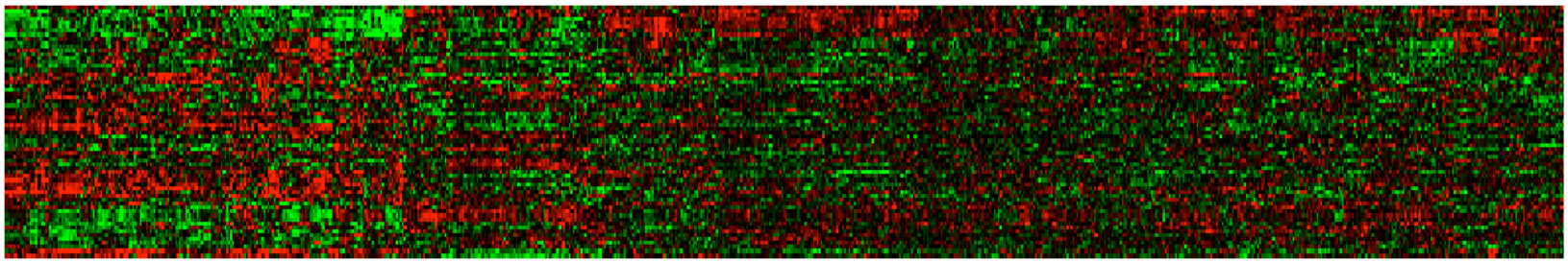
- 6830×64 matrix of real numbers
- Rows correspond to genes, columns to tissue samples
- Cluster rows (genes) can deduce functions of unknown genes from known genes with similar expression profiles
- Cluster columns (samples) can identify disease profiles: tissues with similar disease should yield similar expression profiles

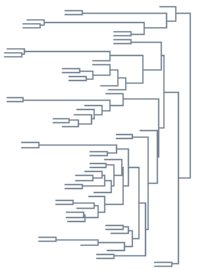
Gene expression matrix

Example: Human Tumor Microarray Data

- 6830×64 matrix of real numbers
- GA clustering of the microarray data
  - Applied separately to rows and columns
  - Subtrees with tighter clusters placed on the left
  - Produces a more informative picture of genes and samples than the randomly ordered rows and columns







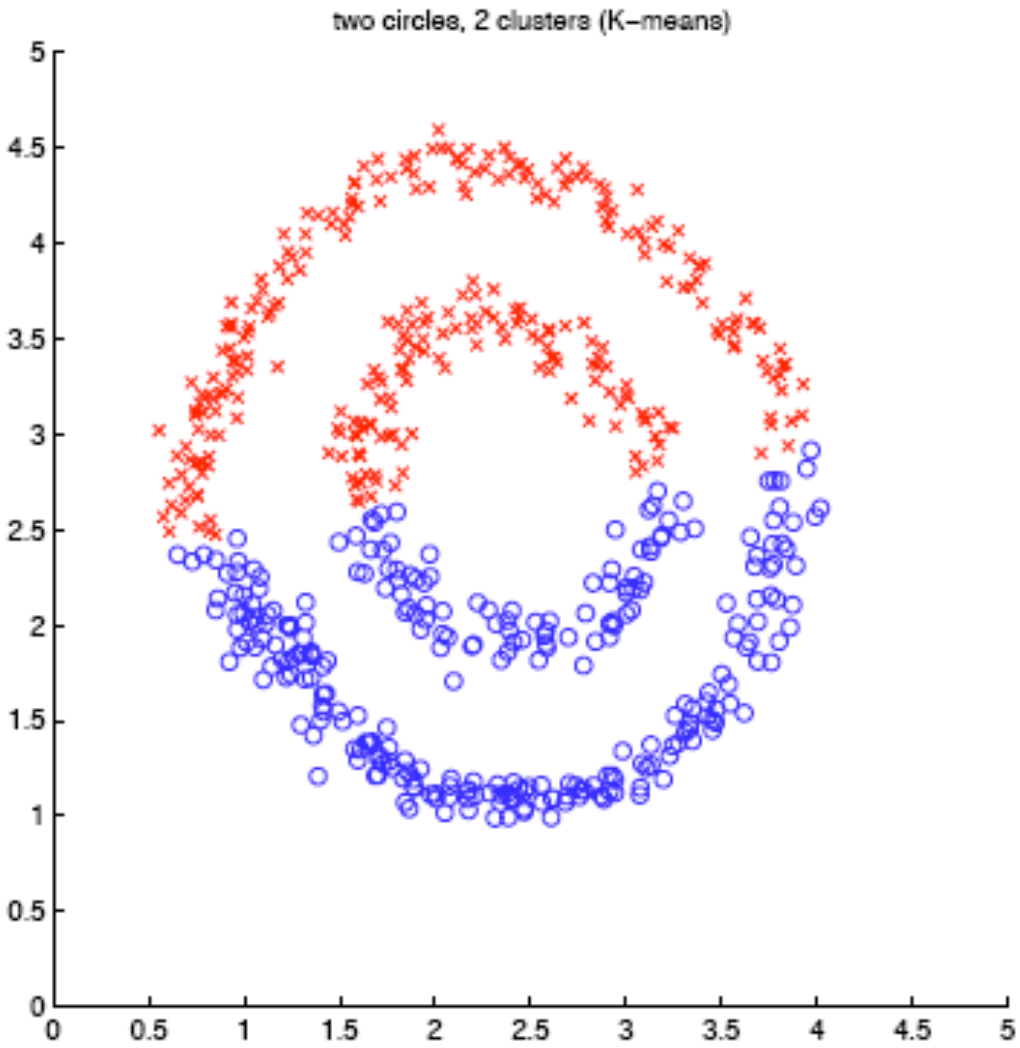
Spectral Clustering

- Idea: use the top eigenvectors of a matrix derived from distance between data points
- Too many versions of spectral clustering algorithms
  - has roots in spectral graph partitioning
  - only look at one version by Ng, Jordan and Weiss
  - see website for more papers and softwares

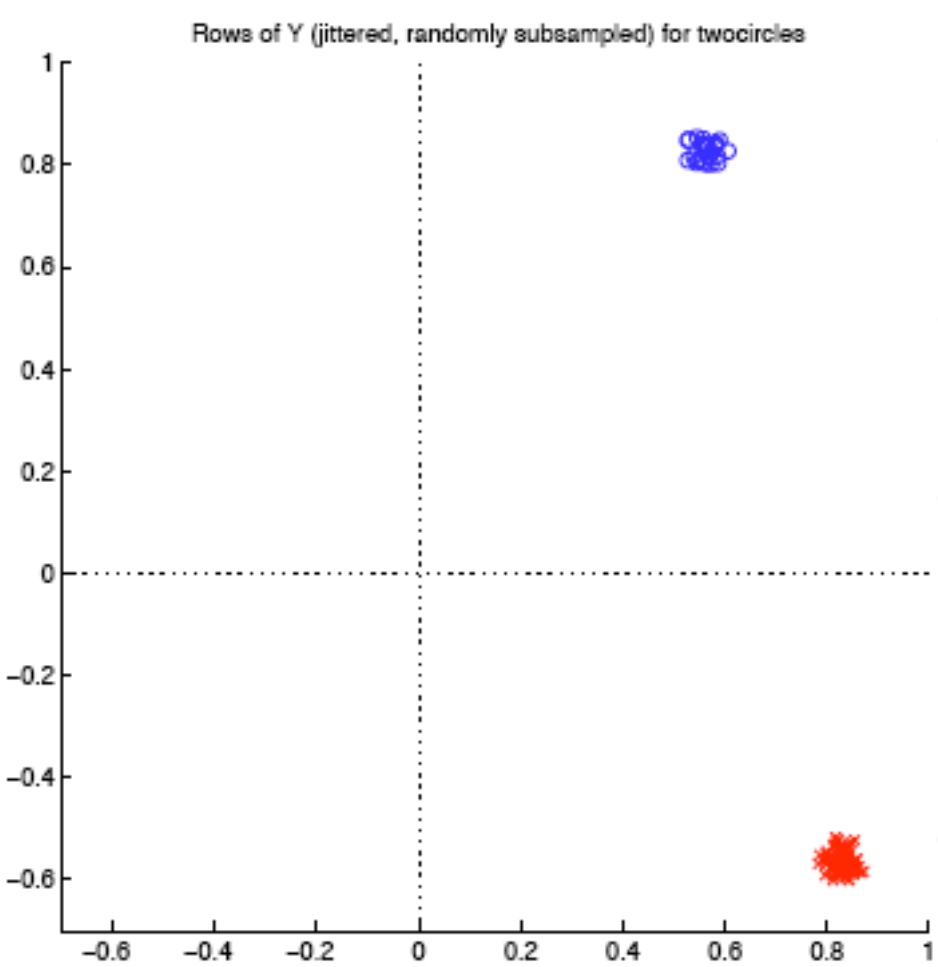
Spectral Clustering

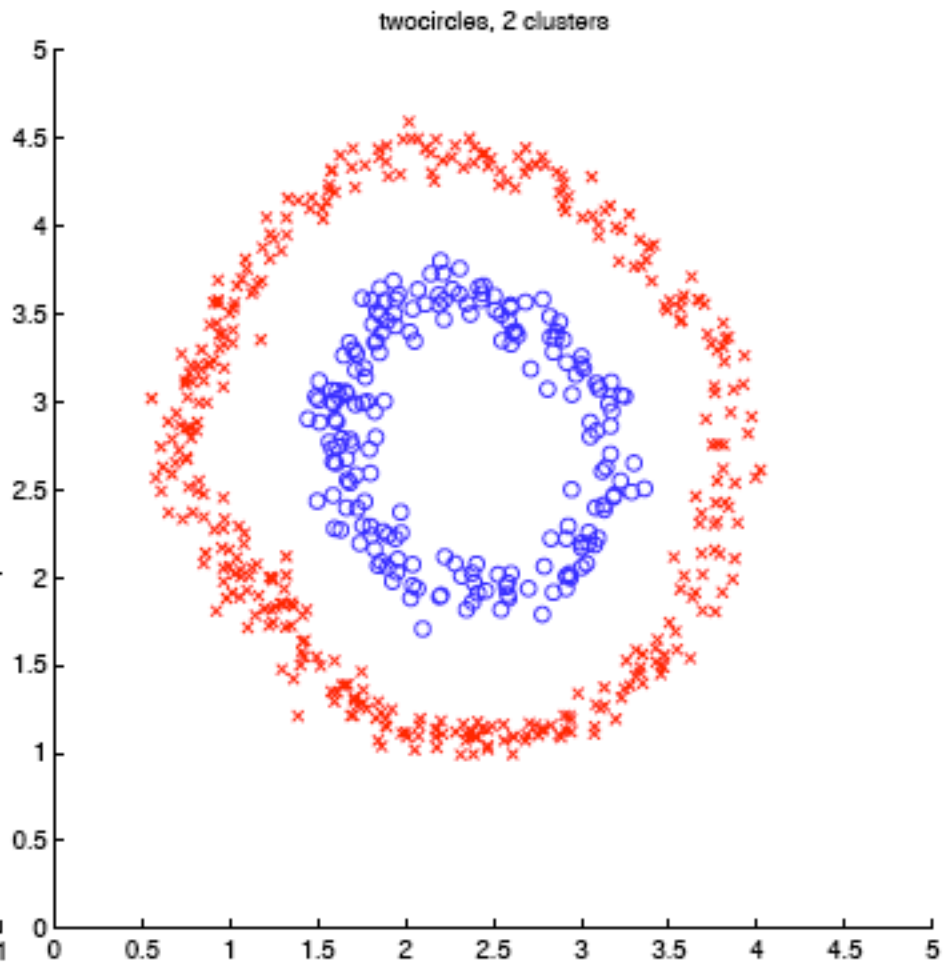
- Given a set of points $s_1, \ldots, s_n \in \mathcal{R}^l$, we'd like to cluster them into *k* clusters
  - Form an affinity matrix $A \in \mathcal{R}^{n \times n}$ where
    $$A_{ij} = exp(-||s_i - s_j||^2 / 2\sigma^2)$$
  - Define $D = diag(W1)$ and $L = D^{-1/2} A D^{-1/2}$
  - Find *k* largest eigenvectors of *L*, concatenate them columnwise to obtain $X = [x_1, x_2, \ldots, x_k] \in \mathcal{R}^{n \times k}$
  - Form the matrix *Y* by normalizing each row of *X* to have unit length
  - Think of *n* rows of *Y* as a new representation of original *n* data points; cluster them into *k* clusters using *K*-means

Example: Two circles



Example: Two circles





Analysis of algorithm (Ideal case)

- In ideal case, say there are 3 clusters that are infinitely far away from each other, then the affinity matrix becomes:

$$A = \begin{bmatrix} A^1 & 0 & 0 \\ 0 & A^2 & 0 \\ 0 & 0 & A^3 \end{bmatrix} \qquad L = \begin{bmatrix} L^1 & 0 & 0 \\ 0 & L^2 & 0 \\ 0 & 0 & L^3 \end{bmatrix}$$

- The eigenvalues and eigenvectors of *L* are the union of eigenvalues and eigenvectors of its block (the latter padded appropriately with zeros)
  - From spectral graph theory, we know that each block L^i has a strictly positive principal eigenvector x_1^i with eigenvalue 1, the next eigenvalue is strictly less than 1

Analysis of algorithm (Ideal case)

- Stack *L*'s eigenvectors in columns to obtain *X* and normalize the rows of *X* to obtain *Y:*

$$X = \begin{bmatrix} x_1^1 & \vec{0} & \vec{0} \\ \vec{0} & x_1^2 & \vec{0} \\ \vec{0} & \vec{0} & x_1^3 \end{bmatrix} \in \mathcal{R}^{n \times 3} \qquad Y = \begin{bmatrix} \vec{1} & \vec{0} & \vec{0} \\ \vec{0} & \vec{1} & \vec{0} \\ \vec{0} & \vec{0} & \vec{1} \end{bmatrix}$$

- The rows of Y corresponds to three orthogonal points lying on a unit sphere. Running K-means will immediately find three clusters
- In general case, have to rely on matrix perturbation theory, see paper for more details
- Also can choose width of Gaussian kernel automatically, see paper for more details