Unsupervised Learning: clustering

AI534

Key Concepts

Distance measures

Hierarchical Clustering: complete, single & average link algorithm

K-means algorithm

Mixture of Gaussians

Expectation maximization

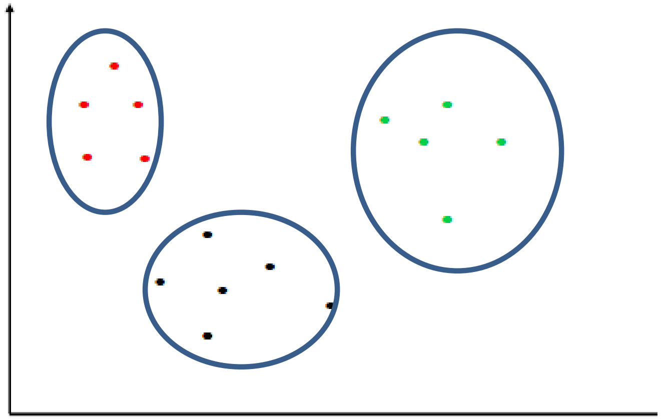
Model selection of # of clusters

Evaluation of clustering

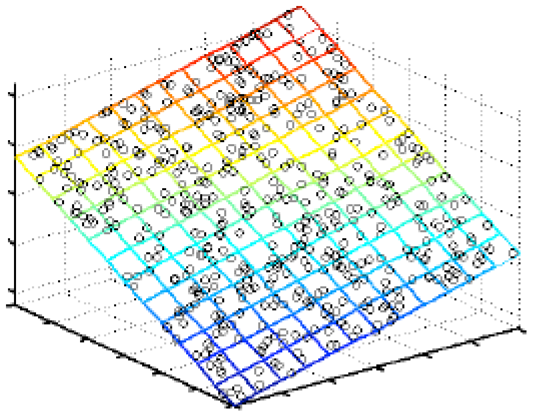
What can we learn from unlabeled data?

- Finding Group of clusters

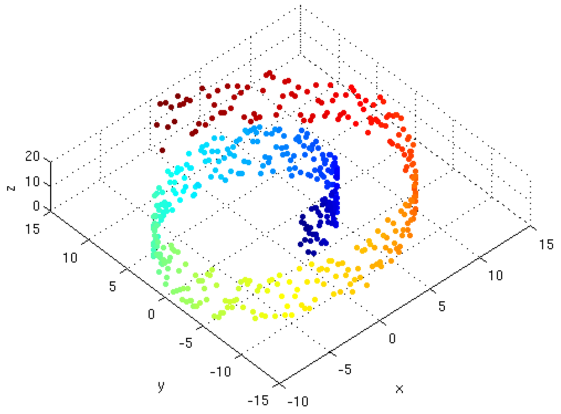
in the data



- Finding low dimensional representation – dimension reduction

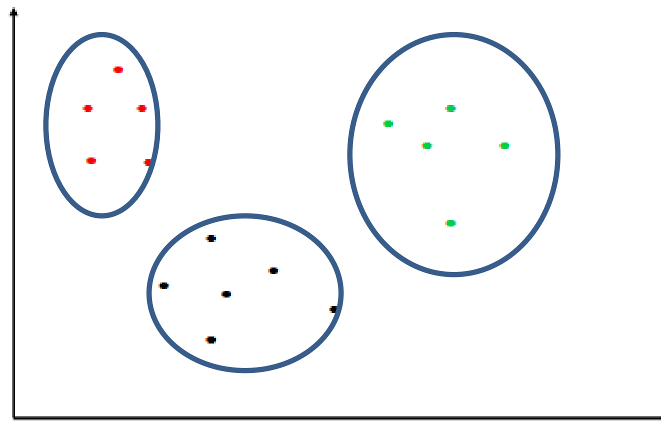


PCA



Nonlinear embedding

Clustering



- Clustering: the process of grouping a set of objects into classes of similar objects
  - high within-class similarity
  - low cross-class similarity
- It is the most common form of unsupervised learning

Example Applications

- Find genes that are similar in their functions
- Group documents based on topics
- Categorize customers based on their purchase history
- Group images based on their contents
- ……

Critical Issues in Clustering

- What makes objects similar?
  - Definition of "similarity/distance“
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
  - Avoid “trivial” clusters - too large or small
- Clustering Algorithms
  - Flat vs hierarchical
  - Hard vs soft

What is similarity

Hard to define but
We know it when we see it

- The real meaning of similarity is a philosophical question. We will take a more pragmatic approach
  – Depends on representation and algorithm. For many rep./alg., easier to think in terms of a distance (rather than similarity) between vectors

What properties should a distance measure have?

- D must be Symmetric
  - $D(A, B) = D(B, A)$
  - Otherwise, we can say A looks like B but B does not look like A
- Positivity, and self-similarity
  - $D(A, B) \geq 0$, and $D(A, B) = 0$ iff $A = B$
  - Otherwise there will be different objects that we cannot tell apart
- Must satisfy triangle inequality
  - $D(A, B) + D(B, C) \geq D(A, C)$
  - Otherwise one can say "A is like B, B is like C, but A is not like C at all"

Distance Measures: Minkowski Metric

- Suppose two object x and y both have d features
  - $x = (x_1, \cdots, x_d), y = (y_1, \cdots, y_d)$
- The Minkowski metric of order r is defined by

$$d(x, y) = \sqrt[r]{\sum_i |x_i - y_i|^r}$$

- Common Minkowski metrics:
  - Euclidean(r=2): $d(x, y) = \sqrt[2]{\sum_i (x_i - y_i)^2}$, also called L_2 distance
  - Manhattan distance(r=1) : $d(x, y) = \sum_i |x_i - y_i|$, also called L_1 distance
  - “Sup” distance(r = $+\infty$): $d(x, y) = \max_i |x_i - y_i|$, also called L_∞ distance

Other Distances

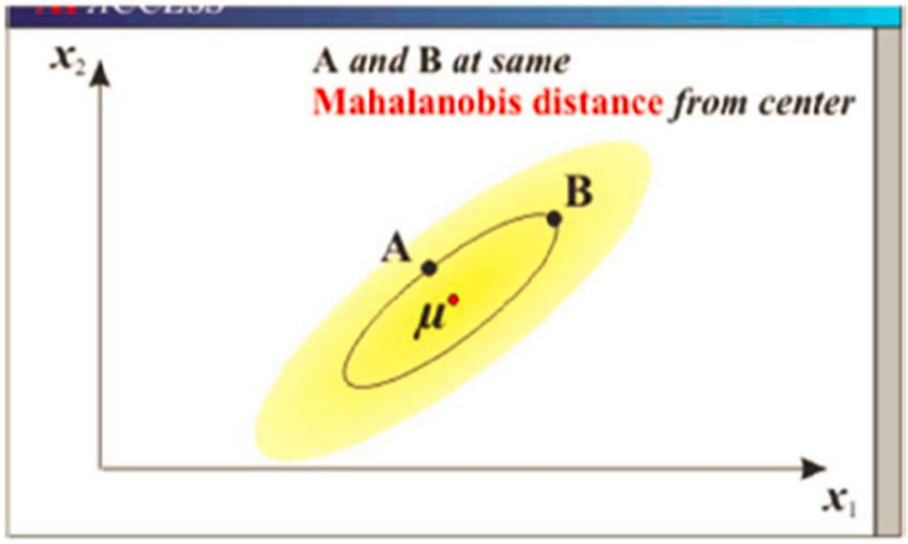
- Hamming distance (Manhattan distance on binary features)
  - # of features that differ
  - e.g.: distance of two sites based on their species composition

	sp1	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9
Site A:	1	0	1	1	0	0	1	0	1
Site B:	0	0	1	0	1	1	1	0	1

D(A, B) = 4

- Mahalanobis distance (assuming $\mathbf{x}, \mathbf{y}$ follows a Gaussian distribution with covariance matrix Σ

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^T \Sigma^{-1} (\mathbf{x}-\mathbf{y})}$$



Similarities

- Cosine similarities – commonly used to measure document similarity

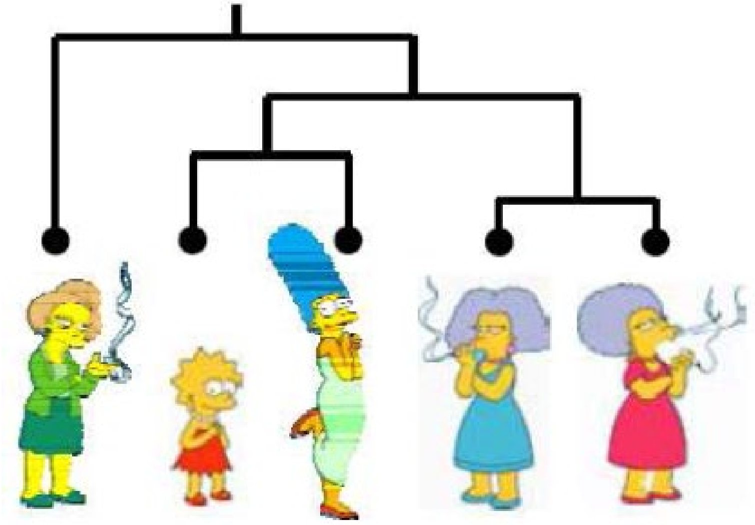
$$cos(\mathrm{x}, \mathrm{x}') = \frac{<\mathrm{x} \cdot \mathrm{x}'>}{|\mathrm{x}| \cdot |\mathrm{x}'|}$$

- Kernels – e.g., RBF (Gaussian) Kernel

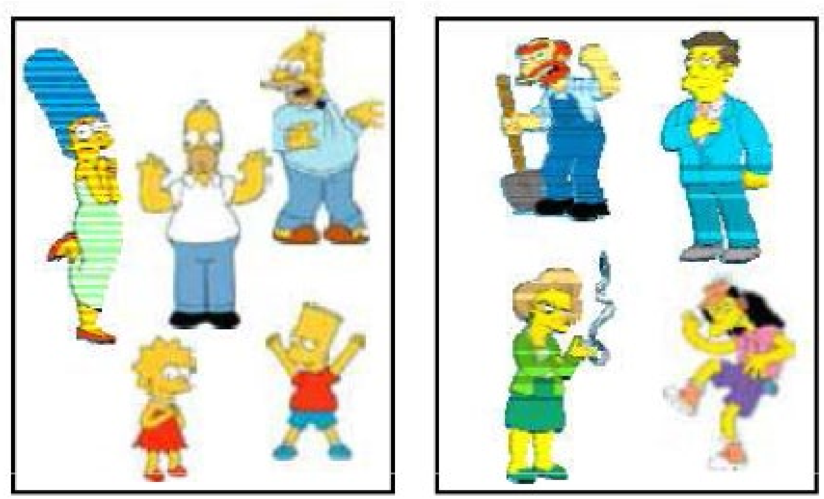
$$S(X, X') = \exp \frac{-|X - X'|^2}{2\sigma^2}$$

Clustering algorithms

- Hierarchical algorithms (not covered)
  - Bottom up – agglomerative
  - Top down – divisive

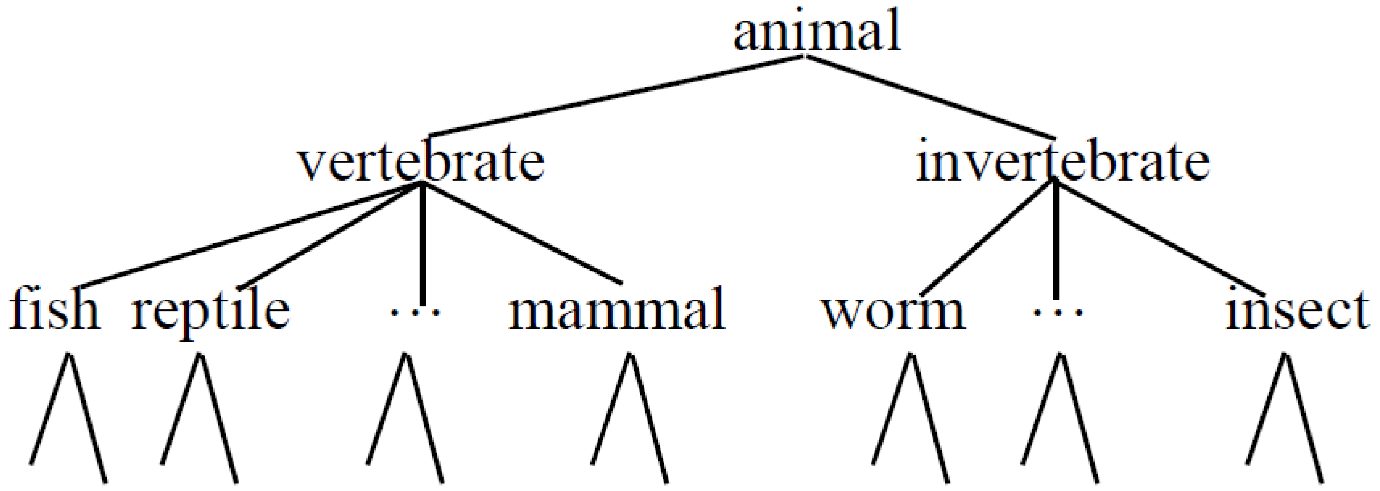


- Flat clustering algorithms
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering (not covered)



Hierarchical Clustering

- Given a set of objects, build a tree-based taxonomy



- Hierarchies are convenient way for organizing information

Hierarchical Agglomerative Clustering (HAC)

- Starts with each object in a separate cluster
- While there are more than 1 cluster:
  - Find the **closest** pair of clusters and join them

The history of merging forms a tree of hierarchy

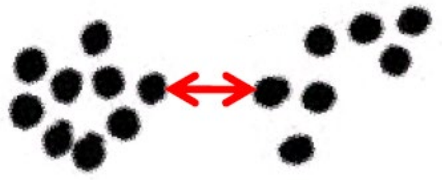
<u>Question</u>: how to measure the **“closeness”** of two clusters?

- Different definition leads to different algorithms

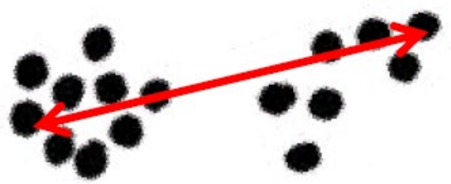
Distance between clusters

The distance between two clusters is defined as follows

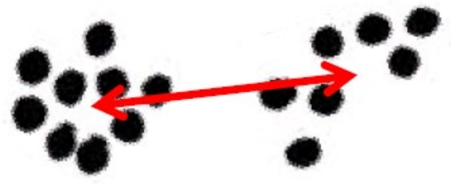
- Single-link
  - Distance between the nearest pair of points across two clusters: $D(C_i, C_j) = \min_{x \in C_i, y \in C_j} d(x, y)$



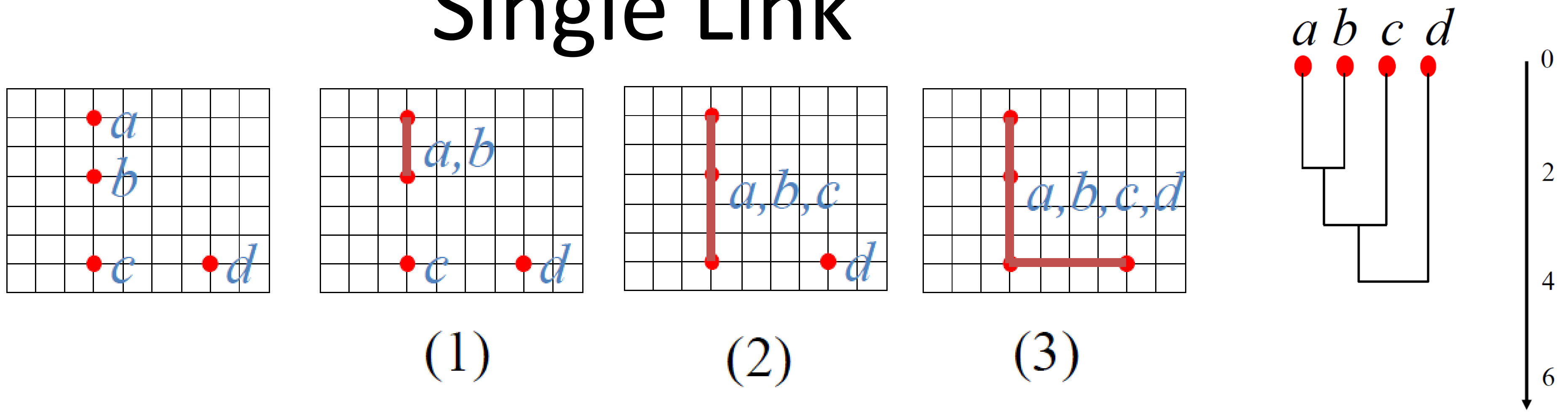
- Complete-link
  - Distance between the furthest pair of points across two clusters: $D(C_i, C_j) = \max_{x \in C_i, y \in C_j} d(x, y)$



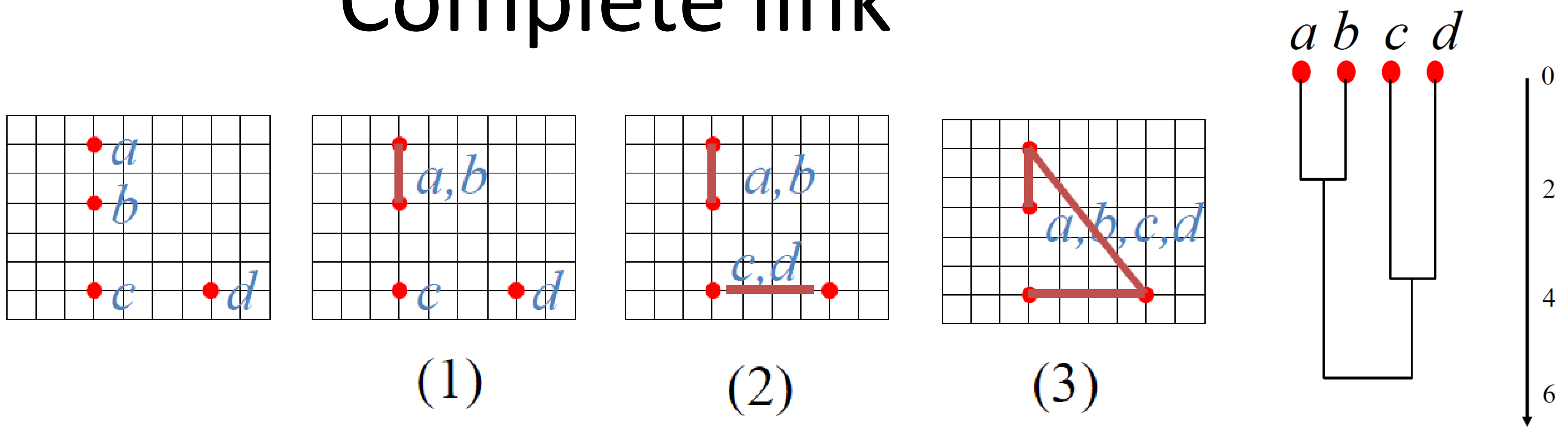
- Average-link
  - Average distance of all cross-cluster pairs
  - $D(C_i, C_j) = \underset{x \in C_i, y \in C_j}{\text{average}} \, d(x, y)$
- Centroid
  - Distance between the means of the two clusters
  - $D(C_i, C_j) = d(\hat{x}, \hat{y})$



Single Link

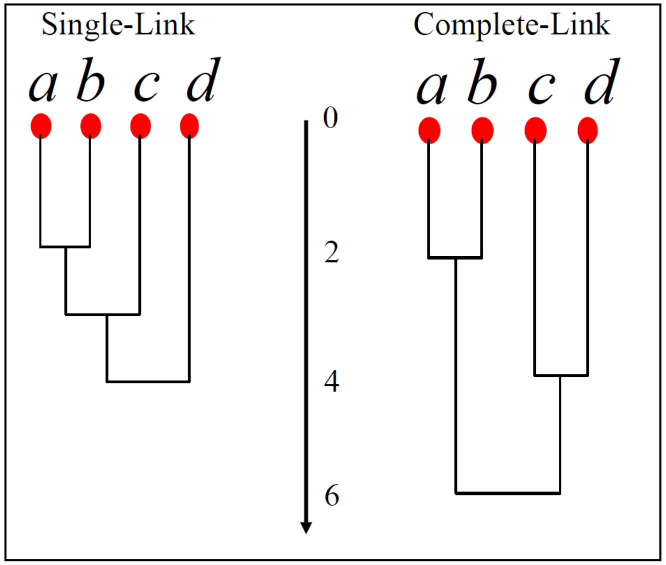


Complete link



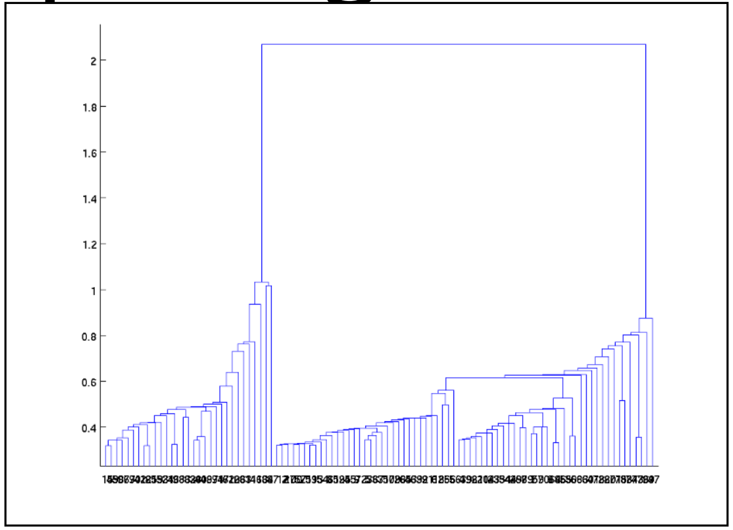
Visualization: Dendrogram

- Height of the joint = the distance between the two merge clusters
- The merge distance monotonically increases as we merge more and more for
  - Single, complete and average linkage methods
  - But not for the centroid method



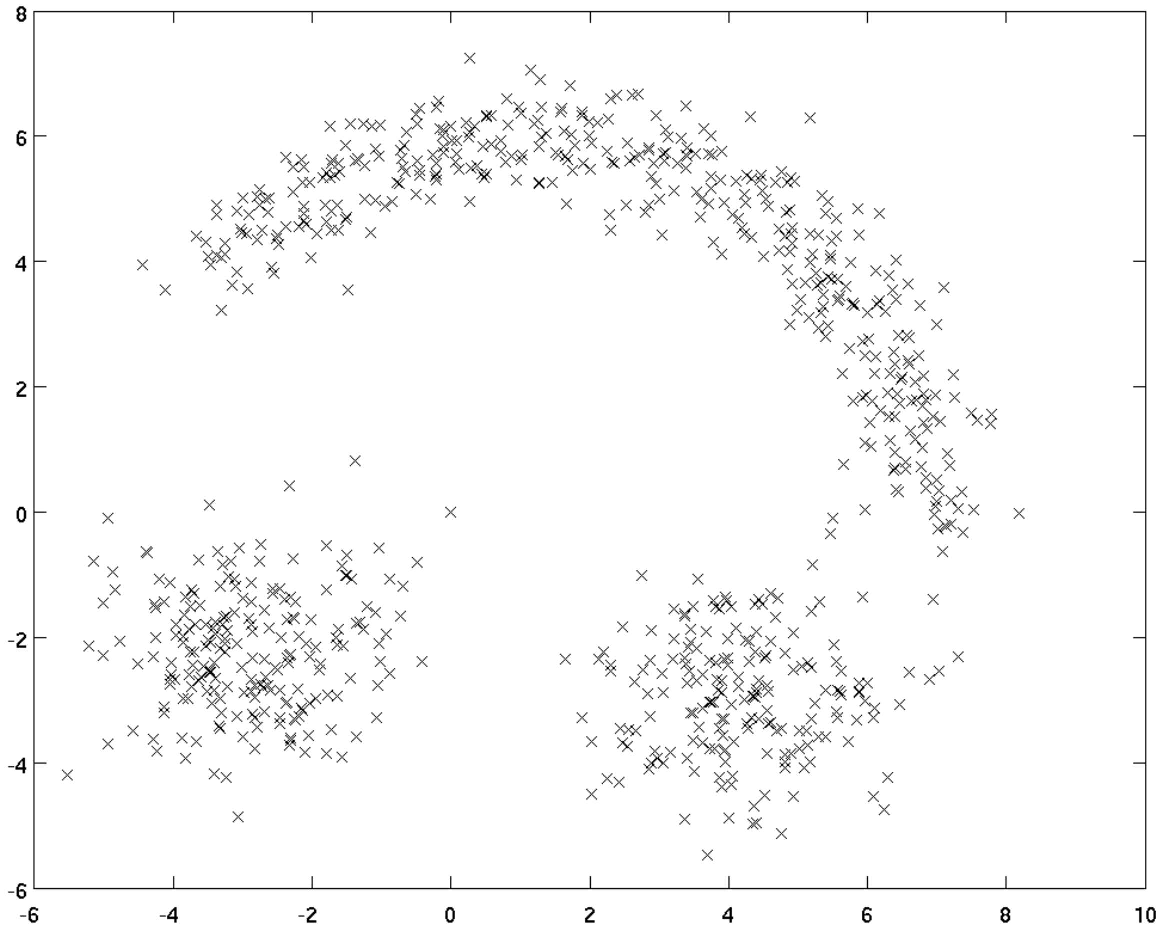
This example is shown upside down.

Interpreting Dendrogram

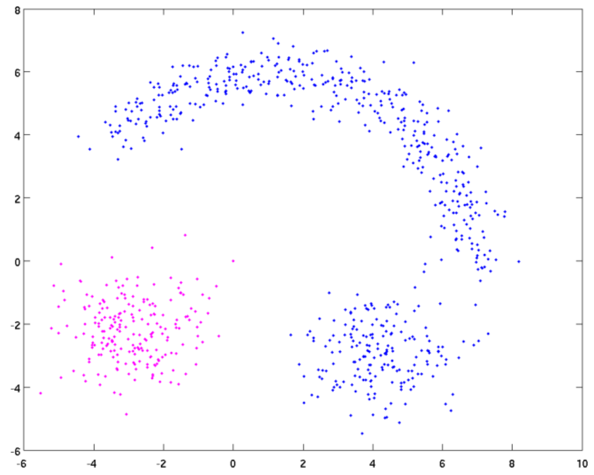


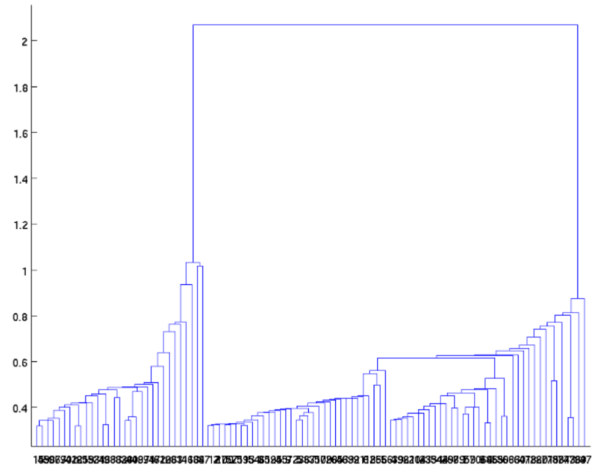
- Dendragram can be used to visually identify
  - the number of clusters in data
    - A horizontal cut creates a partition
    - Moving the cut from root down creates more clusters
    - Large gaps indicate good cutting points
  - well-formed clusters
    - Some clusters are better formed than the other
    - This can be easy to see on a dendrogram

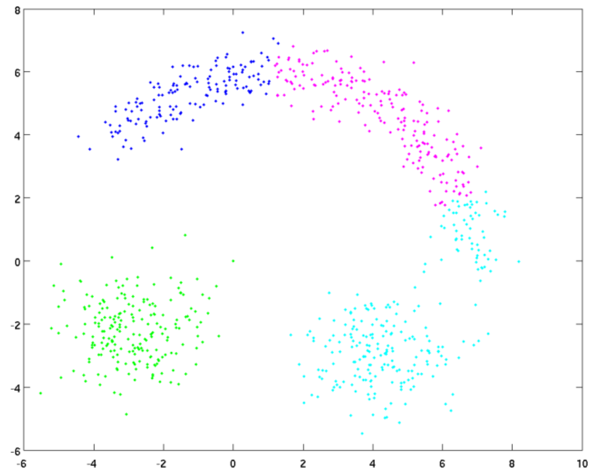
Example

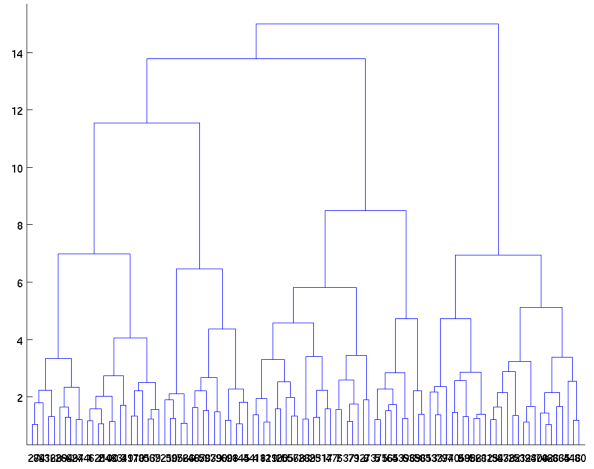


Single Link vs. Complete Link



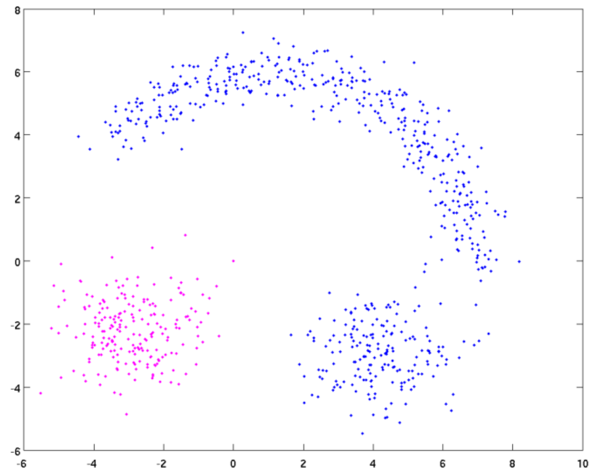
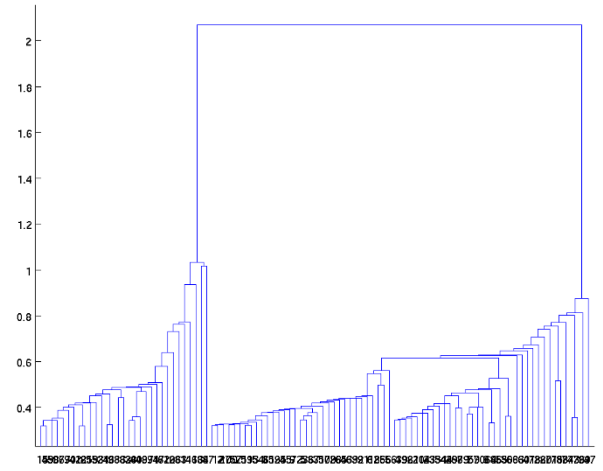




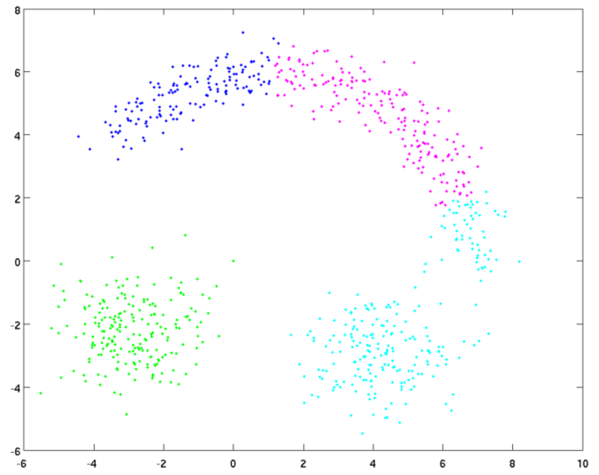
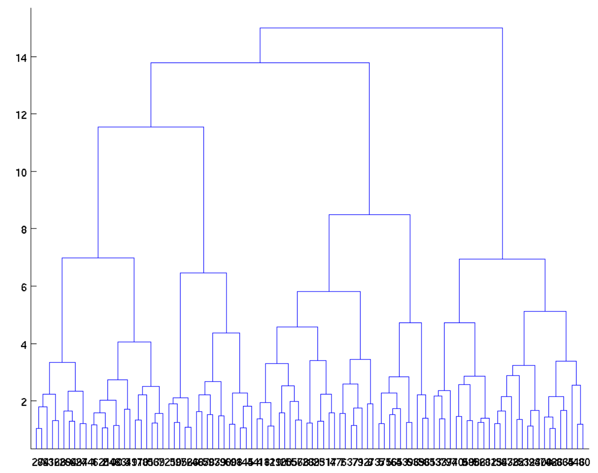


Which is single link? Which is complete link?

Single Link vs. Complete Link

Single

Complete

- Single-link creates straggly clusters due to chaining effect

Flat Clustering

- Given D, a data set of n points, we want to find k clusters
- For each cluster i we would like to represent all points of that cluster with one representative μ_i, and the total representation error should be minimized:

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

where $C = \{C_1, C_2, \dots, C_k\}$ defines a partition of D such that

$$C_i \cap C_j = \emptyset, \forall i \neq j \text{ and } D = \cup_i C_i$$

and μ_i is the representative of C_i

- A combinatorial optimization problem
  - Discrete solution space
  - Exhaustive search for an optimal solution is not feasible

An Iterative Solution

- ***Initialization:*** Start with a random partition of the data, or a random initial $\mu_i's$
- ***Iterative step***: update the cluster assignments and cluster centers to improve the objective
- ***Stopping criterion***: if no improvement can be achieved.

K-Means

Algorithm

Input – N data points, and desired # of clusters: k

Initialize – $\mu_1, \ldots, \mu_k$, the k cluster centers (by randomly selecting k points)

Iterate –

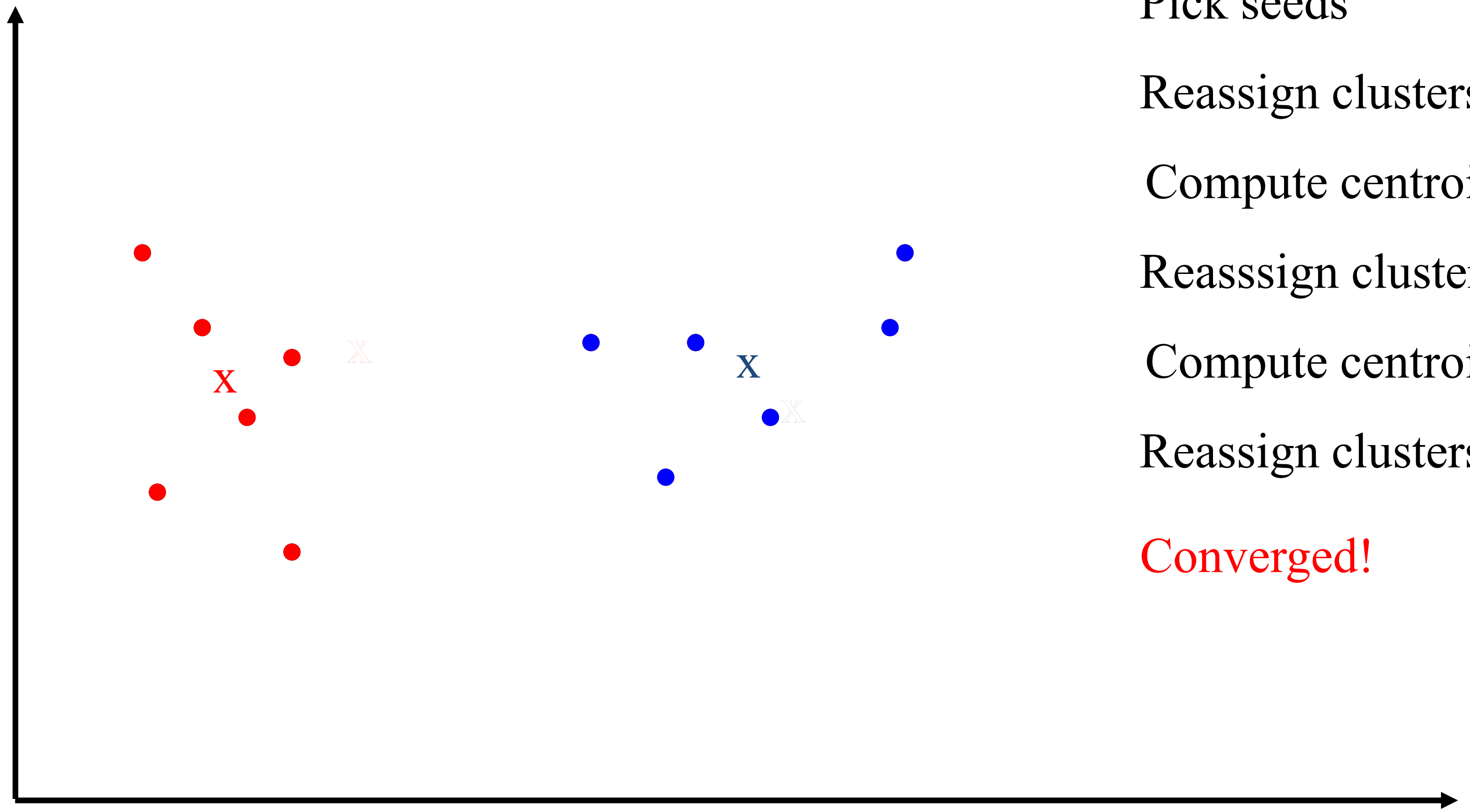
1. Assigning each of the N data points to the closest μ_i
2. Re-estimate the cluster center by assuming that the current assignment is correct

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

Termination –

If none of the data points changed membership in the last iteration, exit. Otherwise, go to 1

Demo: K-Means Example (K=2)



Pick seeds

Reassign clusters

Compute centroids

Reasssign clusters

Compute centroids

Reassign clusters

Converged!

Does KMeans Converge?

Objective

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

1. Fix μ_i's, optimize C, the assignments
   - Assign each point to its closet center

Step 1 of kmeans

2. Fix *C,* optimize μ:

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

   - Take partial derivative of μ_i and set to zero, we have

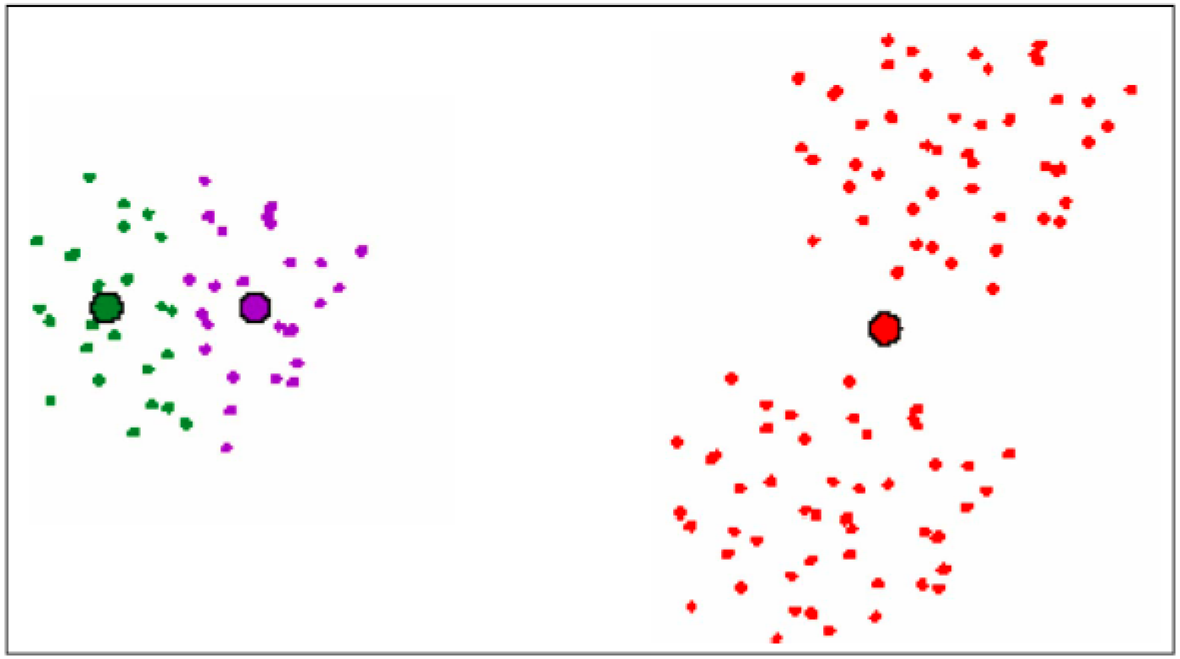
$$2\sum_{\mathbf{x}\in C_i}(\mathbf{x}-\mu_i) = 0 \Rightarrow \mu_i = \frac{1}{|C_i|}\sum_{\mathbf{x}\in C_i}\mathbf{x}$$

Step 2 of kmeans

Each iteration is guaranteed to decrease the objective and there are finite possible C's – thus guaranteed to converge --- but only to local minimum

Impact of Initial Seeds

- Highly sensitive to the initial seeds



- Multiple random trials: choose the one with best sum of squared loss (important!)
- Heuristics for choosing better centers
  - choose initial centers to be far apart – furthest first traversal (kmeans ++)

More Comments

- K-Means is exhaustive:
  - Cluster every data point, no notion of outlier
  - Outliers cause problems
    - Become singular clusters
    - Bias the centroid estimation
- K-medoids methods is more robust to outliers
  - Cluster medoid: must be one of the data points, one that has the minimum sum squared distance to all data points in the cluster
  - More expensive to compute
    - For each pt: sum squared dist with all other pts in cluster $O(|C|^2)$
- Need to specify k: difficult in practice, many heuristic approaches

Hard vs. Soft Clustering

- Hard clustering:
  - Data point is deterministically assigned to one and only one cluster
  - But in reality clusters may overlap
- Soft-clustering:
  - Data points are assigned to clusters with certain probabilities
- Often referred to as model-based clustering for the use of probabilistic model for clusters
  - Assuming each cluster follows a certain probabilistic (e.g., Gaussian) distribution

Side track: Gaussian Bayes Classifier

- We have k classes in our data
- Each class contains data generated from a particular Gaussian distribution $N(\mu_c, \Sigma_c), c = 1, \dots, k$
- The data is generated as follows:
  - randomly draw $y \sim p(y)$
  - Randomly draw $\mathbf{x} \sim N(\mu_y, \Sigma_y)$, i.e., from the Gaussian distribution of class y
  - Repeat n times to generate n points $(\mathbf{x}_i, y_i)$
- In supervised learning, we observe $(\mathbf{x}_i, y_i)$ in pairs and we simply need to estimate $P(y = c)$ and $p(\mathbf{x}|y = c)$ for $c = 1, \dots, k$

MLE Estimates of Mean and Covariance for Gaussian

Given that $p(\mathbf{x}|y = c) \sim N(\mu_c, \Sigma_c)$, and given a set of n_c training examples with $y = c$:

$$\mu_c = \frac{1}{n_c} \sum_{y_i = c} \mathbf{x}_i$$

$$\Sigma_c = \frac{1}{n_c} \sum_{y_i = c} (\mathbf{x}_i - \mu_c)\,(\mathbf{x}_i - \mu_c)^T$$

Back to Unsupervised Learning

- Now assume we know our data is generated in the same way
- But for unsupervised learning, we don't have the y's, and only observe $\mathbf{x}'s$
  - We observe a mixture of Gaussians (or mixture of other distributions)
- How can we learn the correct model from the incomplete data?
- We will still use Maximum likelihood estimation

Gaussian Mixture Model

$$P(\mathbf{x}) = \sum_{i=1}^{k} P(\mathbf{x}, y=i)$$

$$= \sum_{i=1}^{k} P(\mathbf{x} \mid y=i) P(y=i)$$

$$= \sum_{i=1}^{k} \alpha_i P(\mathbf{x}|\theta_i)$$

$\alpha_i = P(y = i)$: class priors
Mixing parameter

$\theta_i = \{\mu_i, \Sigma_i\}$

Goal of learning:

- Given a set of x's, find the values of $\underbrace{\{\alpha_1, \ldots, \alpha_k, \theta_1, \ldots, \theta_k\}}_{\Theta}$ that maximize the likelihood of observing the $x's$

Maximum Marginal Likelihood

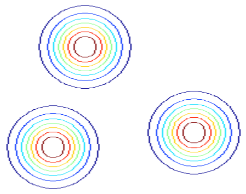
$$
\begin{aligned}
\arg\max_{\theta} \prod_j P(\mathbf{x}^j) &= \arg\max_{\theta} \prod_j \sum_{i=1}^{k} P(\mathbf{x}^j, y^j = i) \\
&= \arg\max_{\theta} \sum_{j=1}^{n} \underbrace{\log \sum_{i=1}^{k} P(\mathbf{x}^j, y^j = i)}_{\boxed{\text{log sum is difficult to optimize !}}}
\end{aligned}
$$

$\boxed{\text{Gradient ascent is doable but very inefficient}}$

Expectation Maximization (EM)

- Commonly used approach for dealing with hidden (missing) data
  - Here the cluster labels are hidden
- Iterative algorithm that starts with an initial guess of the model parameters
- Iteratively performs two linked steps:
  - **Expectation (E-step)**: given current model parameters λ_t, compute the expectation for the hidden (missing) data
  - **Maximization (M-step)**: re-estimate the parameters λ_{t+1} assuming that the expected values computed in the E-step are the true values
- We will first show how it works for mixture of Gaussian

EM – simple case



- A simple case: spherical Gaussians
  - We have unlabeled data $x^1, \cdots, x^m$
  - We know there are K classes
  - We know $\alpha_1 = P(y=1), \ldots \alpha_K = P(y=K)$
  - We don't know $\mu_1 \cdots \mu_K$, but know the common variance σ^2

Start with an initial guess for $\mu_1, \ldots, \mu_K$,

1. If we know $\mu_1, \ldots, \mu_K$, we can easily compute probability that a point x^j belongs to class k:

$$P\left(y = k \middle| x^j\right) \propto \exp\left(-\frac{1}{2\sigma^2}\left|x^j - \mu_k\right|^2\right) p(y = k)$$

Simply evaluate this, then normalize

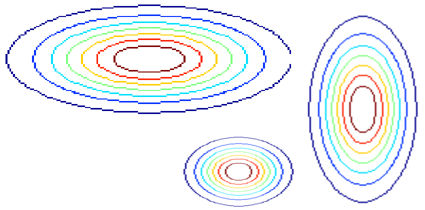
E-step

2. If we know *the* probability that each point belongs to each class, we can estimate the $\mu_1, \ldots, \mu_K$

$$\mu_k = \frac{\sum_{j=1}^{m} P(y=k|x^j)x^j}{\sum_{j=1}^{m} p(y=k|x^j)}$$

M-step

EM – Axis-aligned Gaussian



- We have unlabeled data $x^1, \cdots, x^m$
- We know there are K classes
- We know that the Gaussians are axis aligned

$$\Sigma_k = \begin{bmatrix} \sigma_{k1} & 0 & 0 \\ 0 & \sigma_{kj} & 0 \\ 0 & 0 & \sigma_{kd} \end{bmatrix}$$

Start with an initial guess for $\mu_1, \dots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

1. Given parameters, we can easily compute probability that a point x^j belongs to class i:

$$p(y = k|x^j) \propto p(x^j|\mu_k, \Sigma_k)\, p(y = k)$$

Simply evaluate this, then normalize

E-step

2. Given the probability of each point belonging to each class, re-estimate the $\mu_1, \dots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

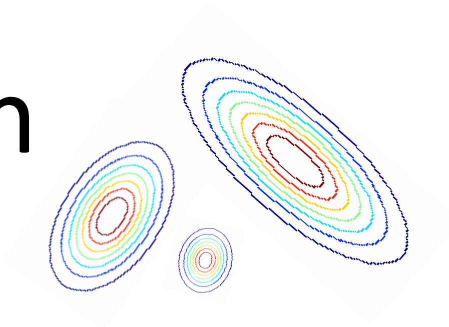
$$\mu_k = \frac{\sum_{j=1}^m P(y=k|x^j)x^j}{\sum_{j=1}^m P(y=k|x^j)} \qquad \alpha_k = \frac{\sum_{j=1}^m P(y=k|x^j)}{m}$$

M-step

$$\sigma_{kl}^2 = \frac{\sum_{j=1}^m P(y=k|x^j)\left(x_l^j - \mu_{kl}\right)^2}{\sum_{j=1}^m P(y=k|x^j)}$$

l-th dimension

EM – General Gaussian



Start with an initial guess for $\mu_1, \dots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

1. If we know the parameters, we can easily compute probability that a point x^j belongs to class k:

$$P(y = k | x^j) \propto p(x^j | \mu_k, \Sigma_k) \, p(y = k)$$

Simply evaluate this, then normalize

E-step

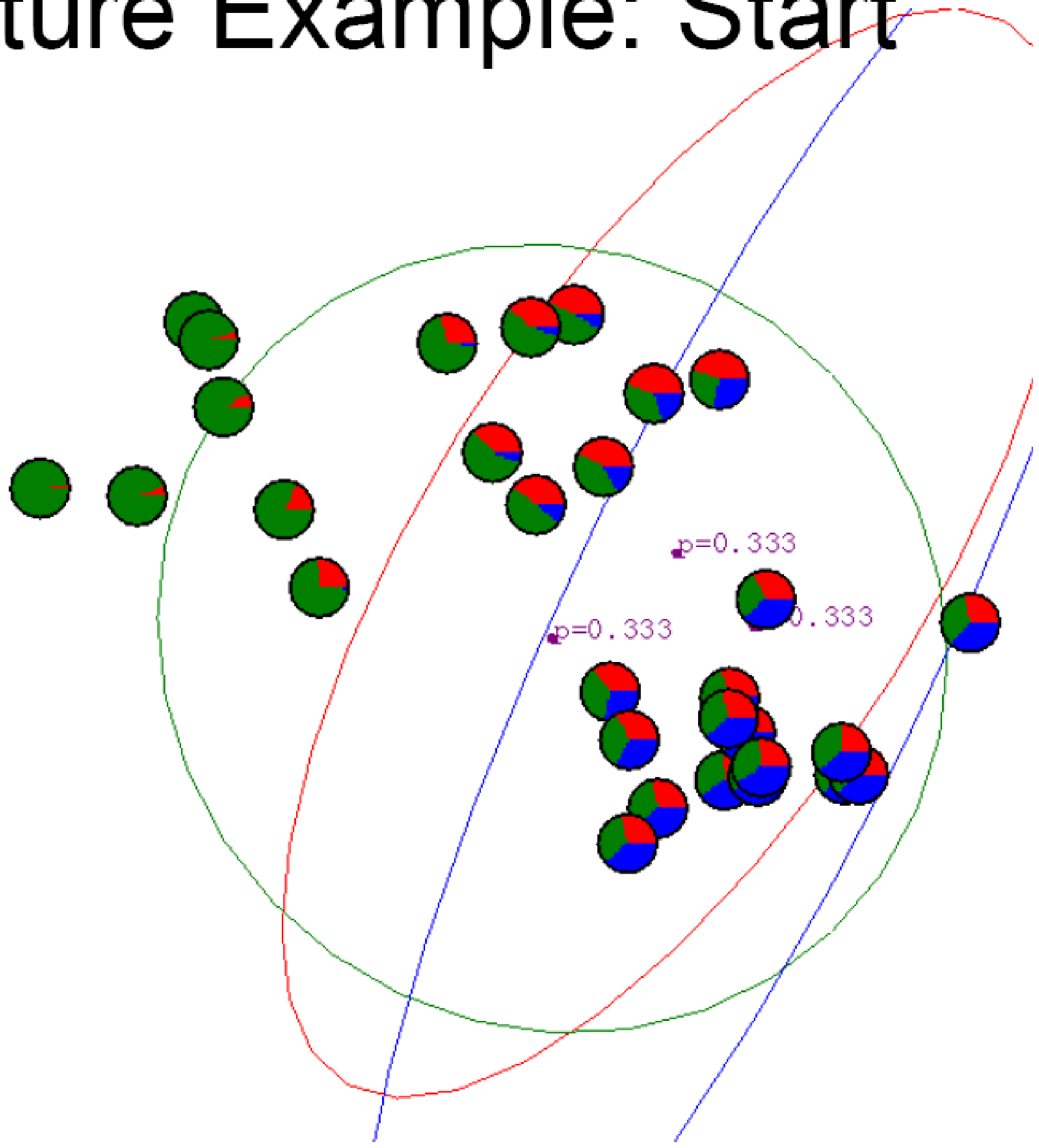
2. If we know *the* probability that each point belongs to each class, we can estimate the $\mu_1, \dots, \mu_K, \Sigma_1, \cdots, \Sigma_K, \alpha_1, \cdots, \alpha_K,$

$$\mu_k = \frac{\sum_{j=1}^{m} P(y=k|x^j) x^j}{\sum_{j=1}^{m} P(y=k|x^j)} \qquad \alpha_k = \frac{\sum_{j=1}^{m} P(y = k | x^j)}{m}$$

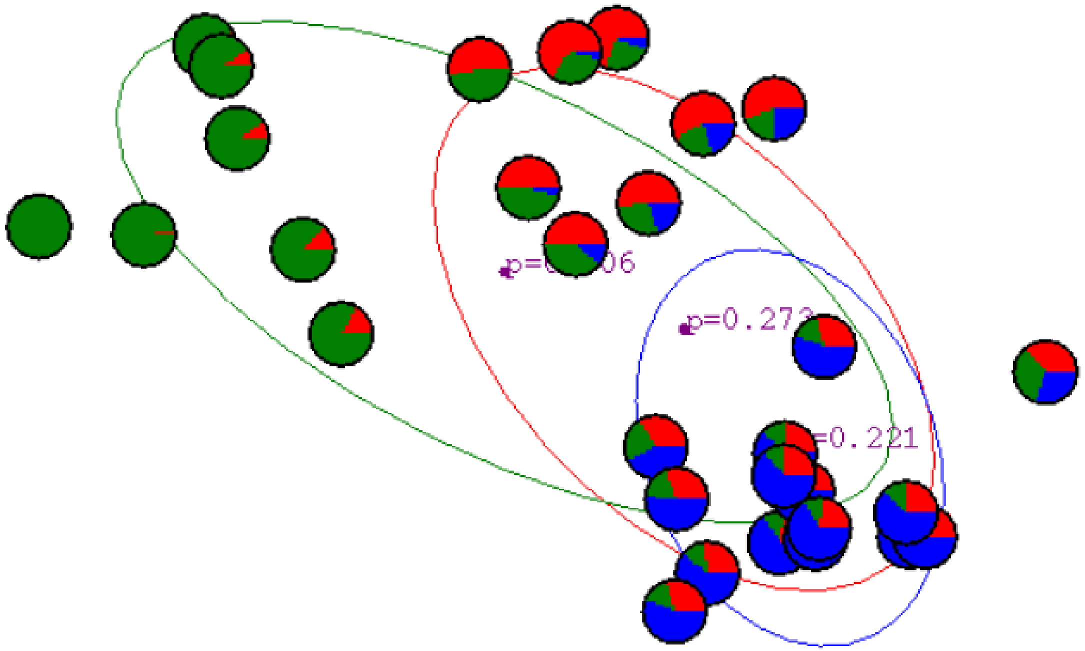
$$\Sigma_k = \frac{\sum_{j=1}^{m} P(y = k | x^j)(x^j - \mu_k)(x^j - \mu_k)^T}{\sum_{j=1}^{m} P(y = k | x^j)}$$

M-step

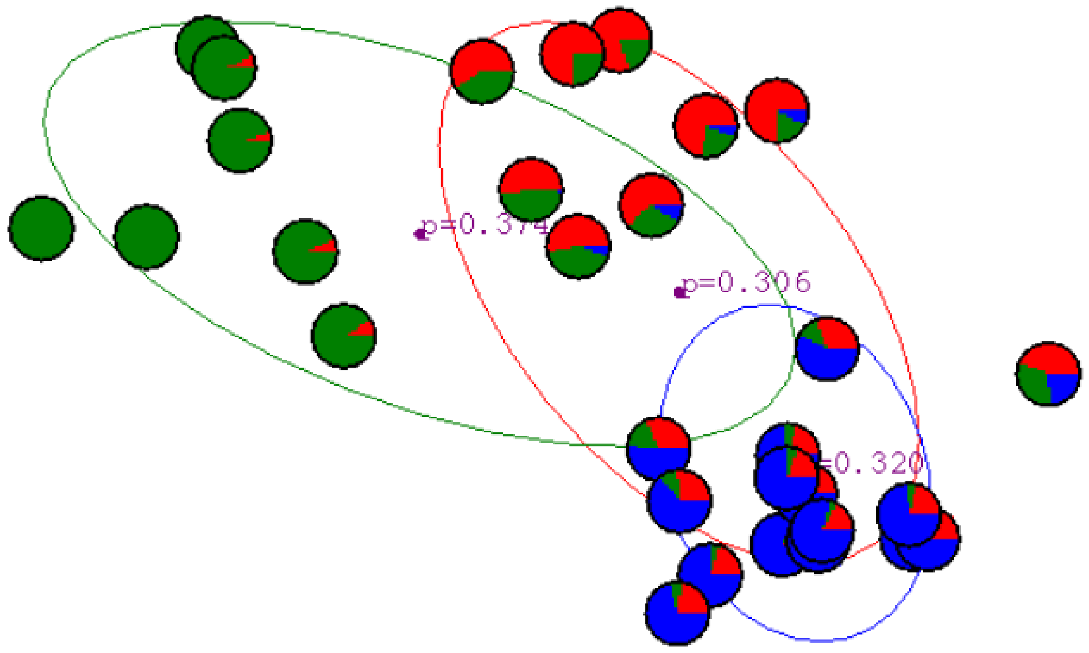
Gaussian Mixture Example: Start



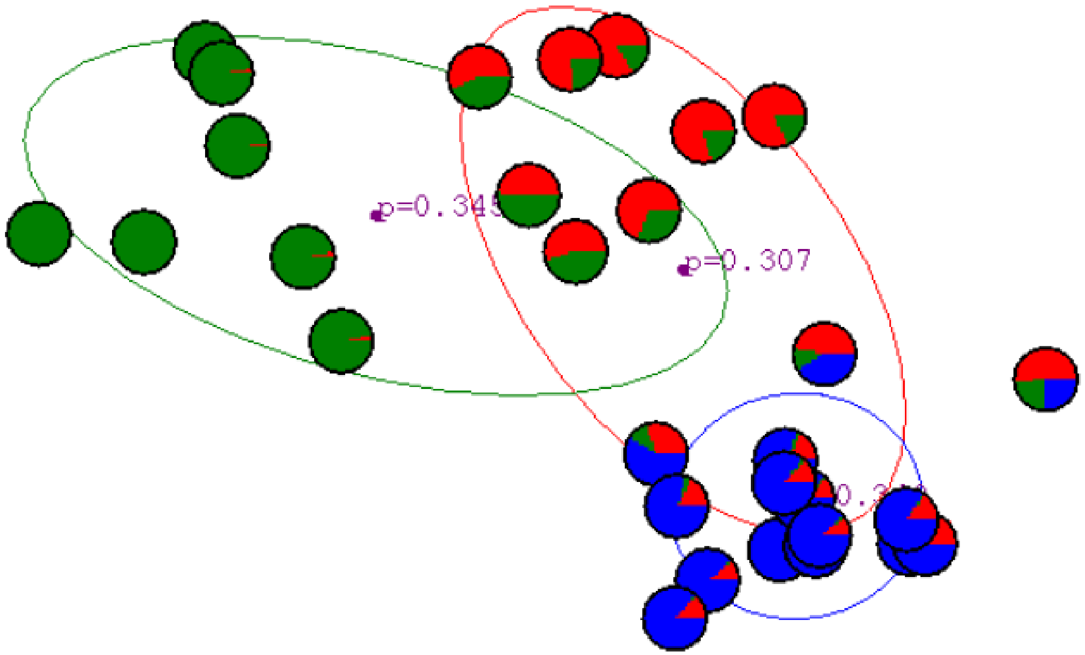
After first iteration



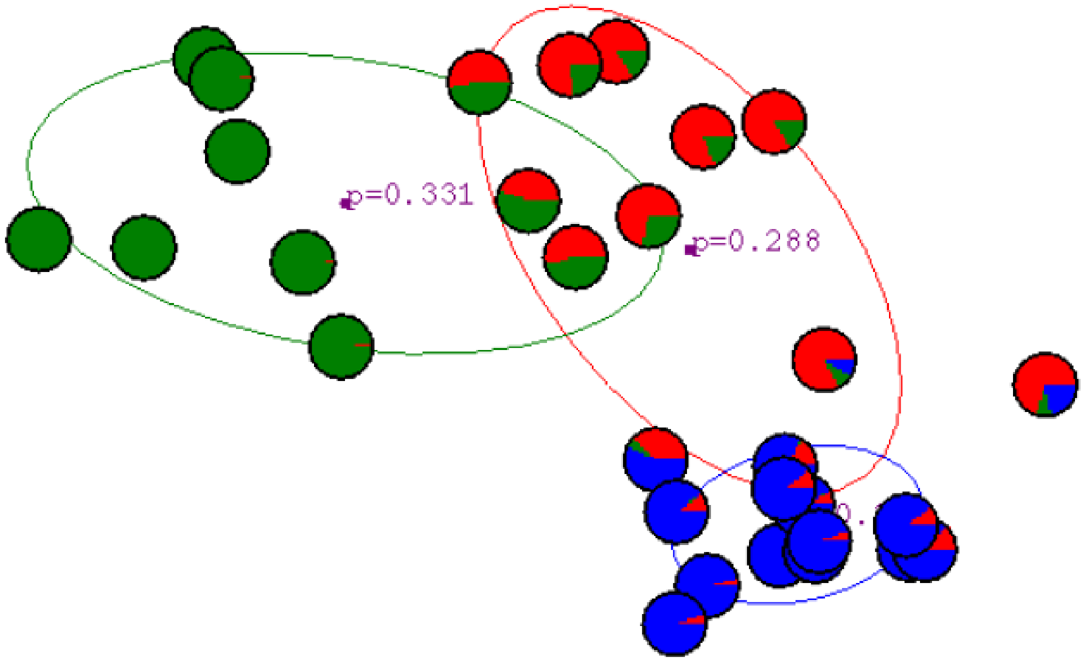
After 2nd iteration



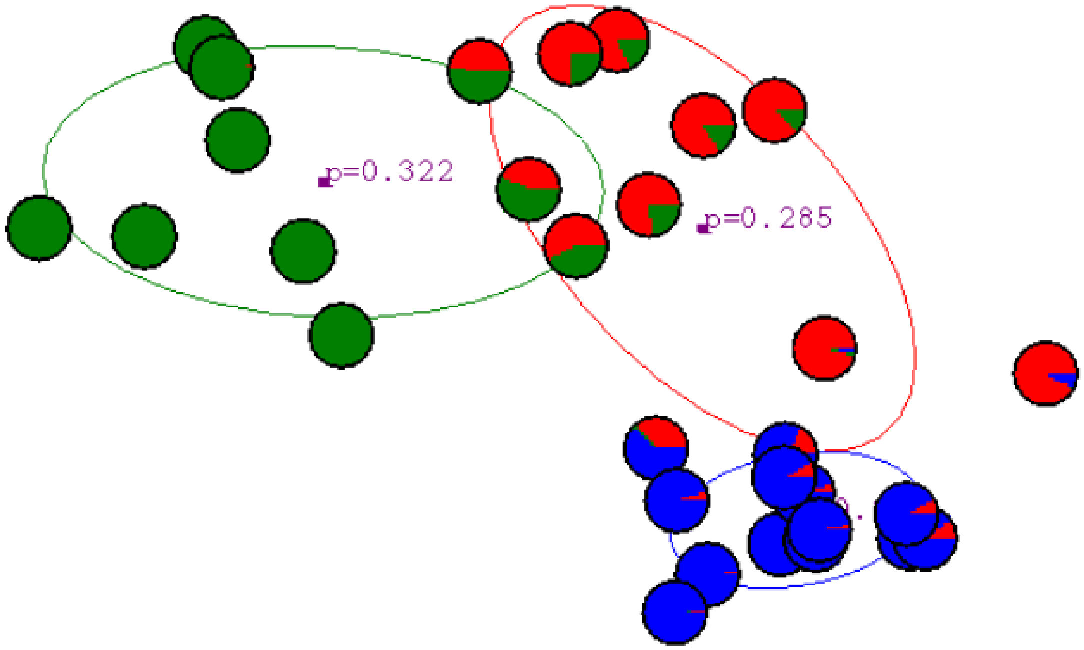
After 3rd iteration



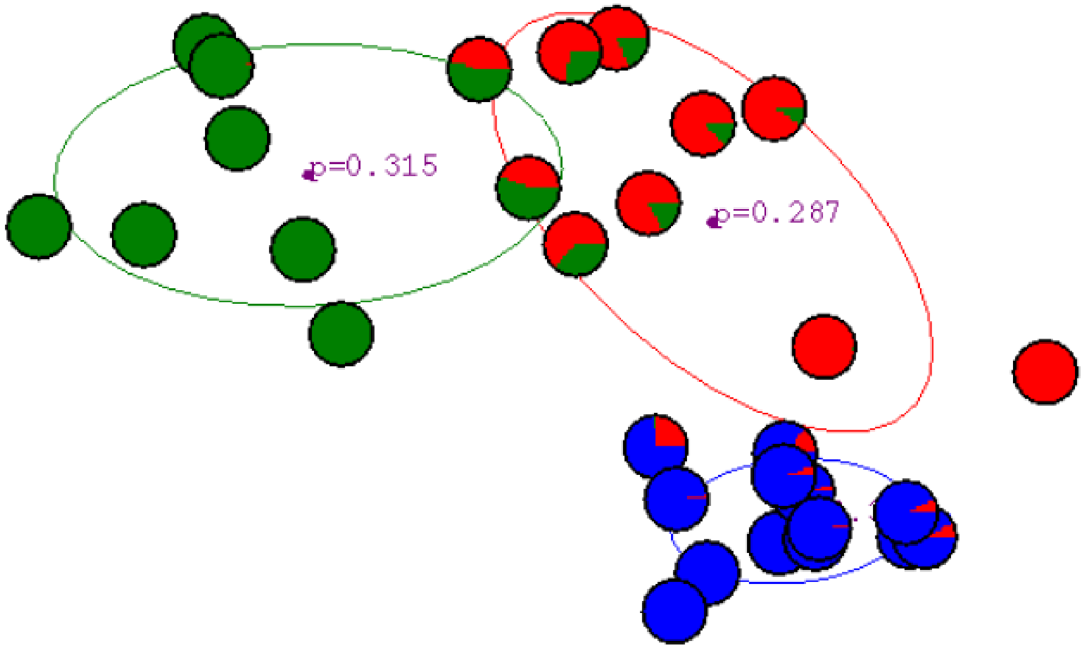
After 4th iteration



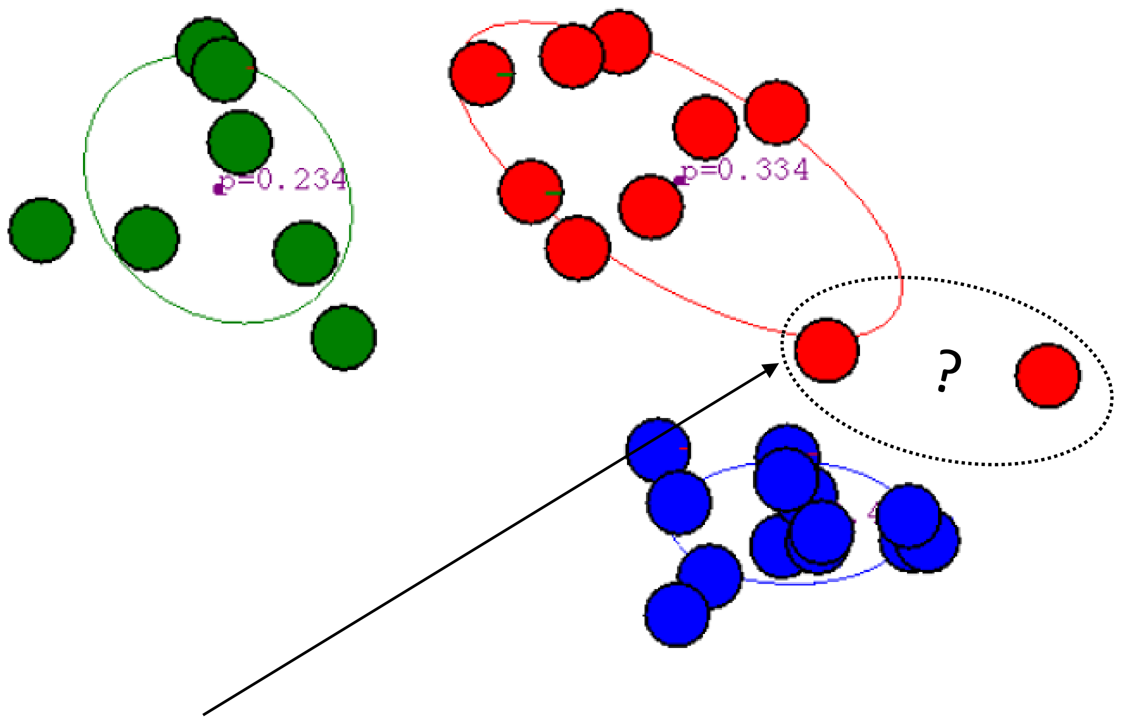
After 5th iteration



After 6th iteration



After 20th iteration



Q: Why are these two points red?

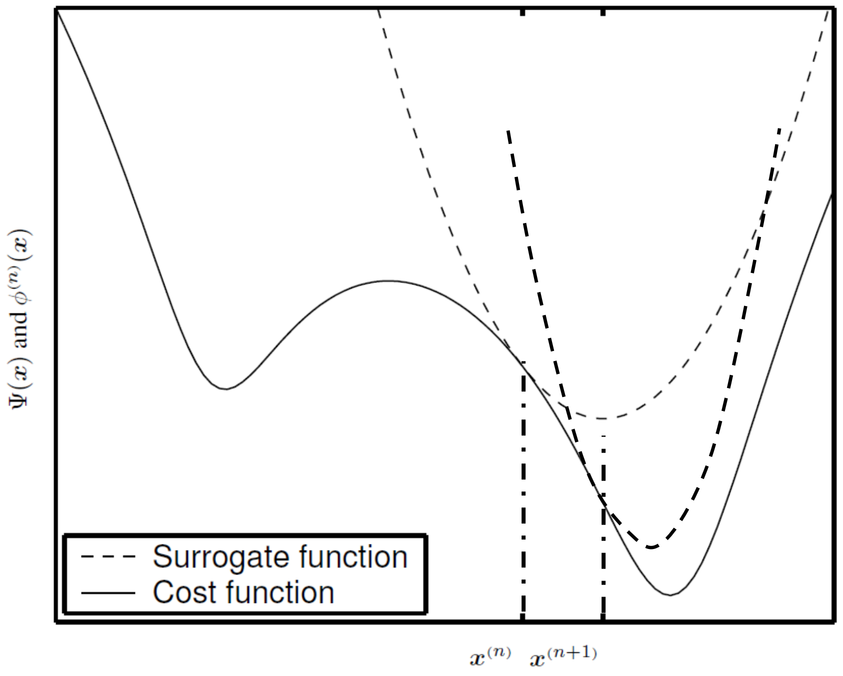
Understanding Expectation Maximization

- EM is a general approach for learning with latent variables
  - Latent: hidden, e.g., the cluster labels
- Consider an estimation problem with a training set $\{x_1, x_2, \dots, x_m\}$
- We wish to fit the parameter θ of a model $p(x, z; \theta)$ to the data without observing z
- Log-likelihood function:

$$l(\theta) = \sum_{i=1}^{m} \log \sum_{z_i} p(x_i, z_i; \theta)$$

- Directly maximizing $l(\theta)$ can be hard (due to the log of sum)
- EM uses a technique called **Optimization Transfer** to solve this optimization problem iteratively

Optimization Transfer (OT)



- Given a complex function Ψ to minimize (shown as the solid line)
- OT works iteratively, minimizes a **surrogate function** ϕ_n (dashed line) at each iteration:
  - $x^{n+1} = \arg\min_x \phi_n(x)$
- In any iteration n, if the surrogate function satisfy the following:
  - $\phi_n(x^n) = \Psi(x^n)$ (match at current pos)
  - $\phi_n(x) \geq \Psi(x)$ (ϕ_n is the upper bound of Ψ)
- We are guaranteed to monotonically improve in each iteration
  - $\Psi(x^{n+1}) \leq \Psi(x^n)$

To apply OT for maximize, we have:

1. $x^{n+1} = \arg\min_x \phi_n(x)$
2. $\phi_n(x^n) = \Psi(x^n)$ (match at current pos) and $\phi_n(x) \leq \Psi(x)$ (lower bound)

Expectation Maximization as Optimization Transfer

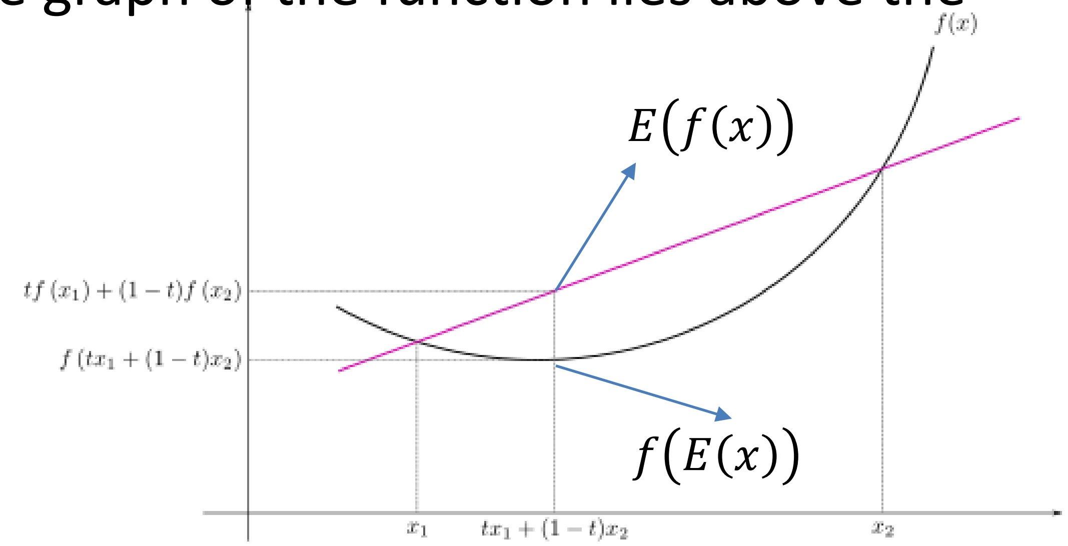
- Expectation maximization uses optimization transfer to maximize the log-likelihood

$$l(\theta) = \sum_{i=1}^{m} \log \sum_{z_i} p(x_i, z_i; \theta)$$

  - Each iteration, it finds an lower bound of the log-likelihood using **Jensen’s inequality**
  - It then maximizes the lower bound

Jensen's Inequality

- Definition: a function is **convex** if the line segment between any two points on the graph of the function lies above the graph



- **Jensen's inequality:**

If f is convex, and let x be a random variable, then:

$$E[f(x)] \geq f(E[x])$$

If f is concave, this is opposite and we have: $E[f(x)] \leq f(E[x])$

Log-likelihood and lower bound

- Objective: Log-likelihood function

$$l(\theta) = \sum_{i=1}^{m} \log \sum_{z_i} p(x_i, z_i; \theta) = \sum_{i=1}^{m} \log \sum_{z_i} \frac{q_i(z_i) p(x_i, z_i; \theta)}{q_i(z_i)}$$

$$= \sum_{i=1}^{m} \log \sum_{z_i} q_i(z_i) [\frac{p(x_i, z_i; \theta)}{q_i(z_i)}] = \sum_{i=1}^{m} \log E_{z_i \sim q_i(z_i)} \left[\frac{p(x_i, z_i; \theta)}{q_i(z_i)}\right]$$

$$\geq \sum_{i=1}^{m} E_{z_i \sim q_i(z_i)} [\log \frac{p(x_i, z_i; \theta)}{q_i(z_i)}]$$

Initial l(t

Log is a concave function
Using Jensen's inequality

- For any distribution $q_i(z_i)$, this gives a lower bound to the log-likelihood
- To be a valid surrogate for optimization transfer, we also need it to match l at current parameter θ^n:

$$\log \sum_{z_i} q_i(z_i) [\frac{p(x_i, z_i; \theta^n)}{q_i(z_i)}] = \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i; \theta^n)}{q_i(z_i)}$$

Further developing the surrogate

- We want to satisfy

$$\log \sum_{z_i} q_i(z_i) \boxed{[\frac{p(x_i, z_i; \theta^n)}{q_i(z_i)}]} = \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i; \theta^n)}{q_i(z_i)}$$

- If the circled part is constant across all possible z_i values then we will have:
  - Left side: $\log \sum_{z_i} C q_i(z_i) = \log C \sum_{z_i} q_i(z_i) = \log C$
  - Right side: $\sum_{z_i} q_i(z_i) \log C = \log C \sum_{z_i} q_i(z_i) = \log C$
- So we have

$$\frac{p(x_i, z_i; \theta^n)}{q_i(z_i)} = C \rightarrow q_i(z_i) = \frac{1}{C} p(x_i, z_i; \theta^n)$$

- Note that q_i must satisfy $\sum_{z_i} q_i(z_i) = 1$

$$\frac{1}{C} \sum_{z_i} p(x_i, z_i; \theta^n) = 1$$

$$C = \sum_{z_i} p(x_i, z_i; \theta^n) = p(x_i; \theta^n)$$

$$q_i(z_i) = \frac{p(x_i, z_i; \theta^n)}{p(x_i; \theta^n)} = p(z_i | x_i; \theta^n)$$

Expectation Maximization

```
Repeat until convergence {
        //$\theta^n$: current parameters in iteration $n$
        (E-step) For each data point $i$, compute posterior of $z_i$:
                $q_i(z_i) = p(z_i | x_i; \theta^n)$
        (M-step) Maximize the expected log-likelihood
            $\theta^{n+1} = \arg\max_\theta \sum_i \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i; \theta)}{q_i(z_i)}$
            $= \arg\max_\theta \sum_i \sum_{z_i} q_i(z_i) \log p(x_i, z_i; \theta)$
        $n \leftarrow n + 1$
}
```

Mixture of Gaussian revisited

- Goal: given $(x_1, \dots, x_m)$, fitting parameters $\alpha_1, \dots, \alpha_k; \mu_1, \dots, \mu_k; \Sigma_1, \dots, \Sigma_k$
- The cluster labels are the latent variables $z_i's$
- E-step:
  - Compute $q_i(z_i) = p(z_i|x_i; \theta^n)$ - posterior of cluster label
- M-step:
  - $\arg\max_\theta \sum_i \sum_{z_i} q_i(z_i) \log p(x_i, z_i|\theta)$ - maximize the expected complete loglikelihood

Behavior of EM

- It is guaranteed to converge
  - Following the principle of Optimization Transfer
    - Each iteration it maximize a lower bound of the likelihood function

$$\theta^{n+1} = \arg\max_{\theta} \sum_{i} \sum_{z_i} p(z_i | x_i; \theta^n) \log p(x_i, z_i | \theta)$$

  - In practice it may converge slowly, one can stop early if the change in the log-likelihood is smaller than a threshold

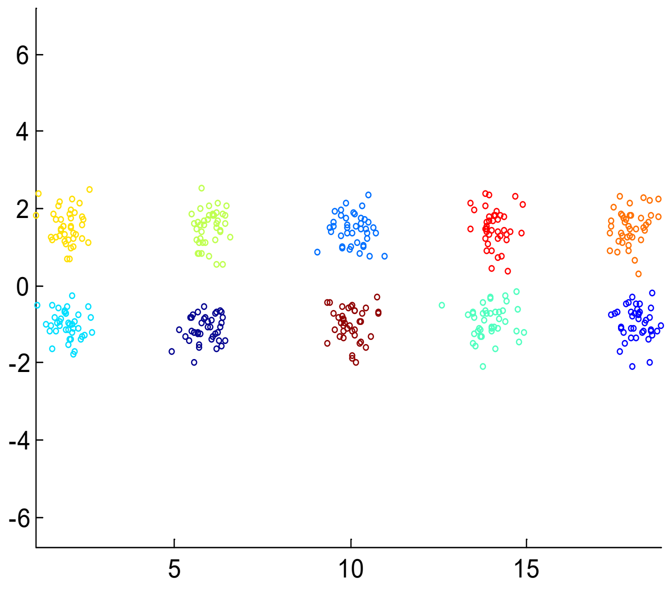
- It converges to a **local optimum**
  - Multiple restart is recommended
  - Choosing the solution with the highest marginal likelihood

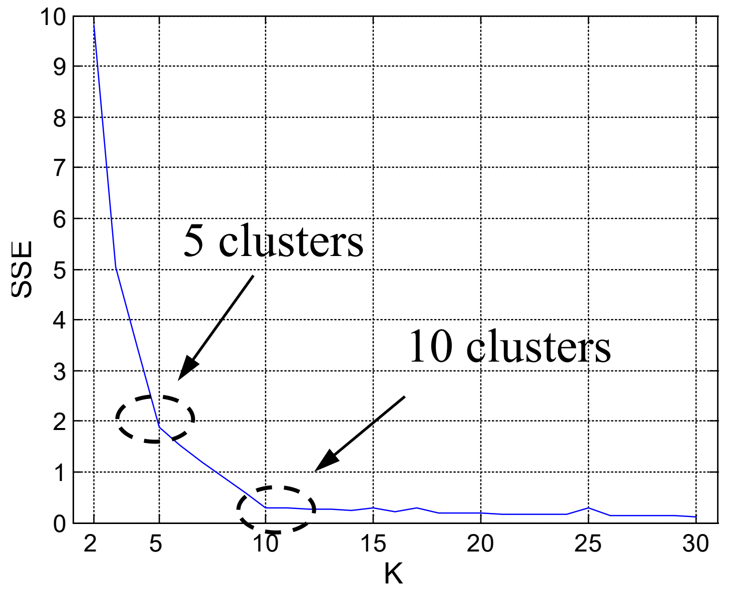
Selecting k: A Model Selection Problem

- Each choice of k corresponds to a different statistical model for the data
- Model selection searches for a model (a choice of k) that gives us the best fit of the training data
  - Penalty method
  - Cross-validation method
  - Model selection methods can also be used to make other model decisions such as choosing among different ways of constraining Σ

Selecting k: heuristic approaches

- For kmeans, plot the sum of squared error for different k values
  - SSE will monotonically decrease as we increase k
  - Knee points on the curve suggest possible candidates for k





Penalty Method: Bayesian Information Criterion

- Based on Bayesian Model Selection
  - Determine the range of k values to consider $1 \leq k \leq K_{max}$
  - Apply EM to learn a maximum likelihood fitting of the Gaussian mixture model for each possible value of *k*
  - Choose *k* that maximizes BIC

$$\underbrace{2l_{\mathcal{M}}(x,\hat{\theta})}_{\boxed{\text{Loglikelihood of the resulting Gaussian Mixture Model}}} - \underset{\substack{\uparrow \\ \boxed{\text{\# of parameters to be estimated in } M}}}{\boxed{m_{\mathcal{M}}}}\log(\overset{\substack{\boxed{\text{\# of data points}} \\ \swarrow}}{n}) \equiv \text{BIC}$$

  - Given two estimated models, the model with higher BIC is preferred
  - Larger k increases the likelihood, but will also cause the second term to increase
  - Often observed to be biased toward less complex model
  - Similar method: $\text{AIC} = 2l_m - 2m_M$, which penalize complex model less severely

Cross-validation Likelihood
(Smyth 1998)

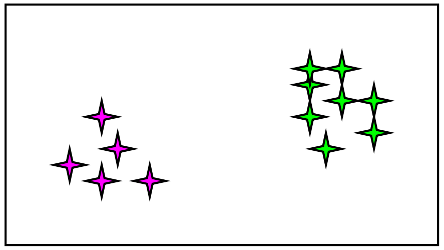
- The likelihood of the training data will always increase as we increase k
  - more clusters, more flexibility leads to better fitting of the data
- Use cross-validation
  - For each fold, learn the GMM model using the training data
  - Compute the log-likelihood of the learned model on the remaining fold as test data

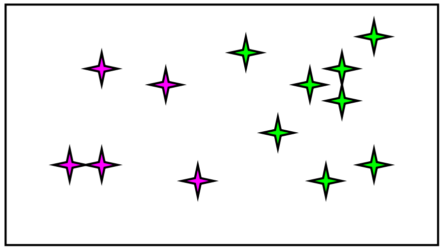
Stability Based Methods

- Stability: repeatedly produce similar clusterings on data originating from the same source. (Levine & Domany, 2001, Tibshirani and Walther, 2005)
- High level of agreement among a set of clusterings $\Rightarrow$ the clustering model (k) is appropriate for the data
- Evaluate multiple models, and select the model resulting in the highest level of stability.

How to Evaluate Clustering?

- By user interpretation
  - does a document cluster seem to correspond to a specific topic?
- Internal criterion – a good clustering will produce high quality clusters:
  - high intra-cluster similarity
  - low inter-cluster similarity





  - The measured quality of a clustering depends on both the object representation and the similarity measure used

External indexes

If true class labels (*ground truth*) are known, the validity of a clustering can be verified by comparing the class labels and clustering labels.

$$\begin{array}{c|c} \boxed{N} & \cdot \\ \hline \cdot & n_{..} \end{array} = \begin{array}{|cccc|c} \hline n_{11} & n_{12} & \cdots & n_{1l} & n_{1.} \\ n_{21} & n_{22} & \cdots & n_{2l} & n_{2.} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ n_{k1} & n_{k2} & \cdots & n_{kl} & n_{k.} \\ \hline n_{.1} & n_{.2} & \cdots & n_{.l} & n_{..} \end{array}$$

n_{ij} = number of objects in class *i* and cluster *j*

Rand Index and Normalized Rand Index

- Given partition (*P*) and ground truth (*G*), measure the number of vector pairs that are:

a:	in the same class <u>both</u> in *P* and *G*.
b:	in the same class in *P*, but different classes in *G*.
c:	in different classes in *P*, but in the same class in *G*.
d:	in different classes <u>both</u> in *P* and *G*.

Rand index 1 is best. It rarely

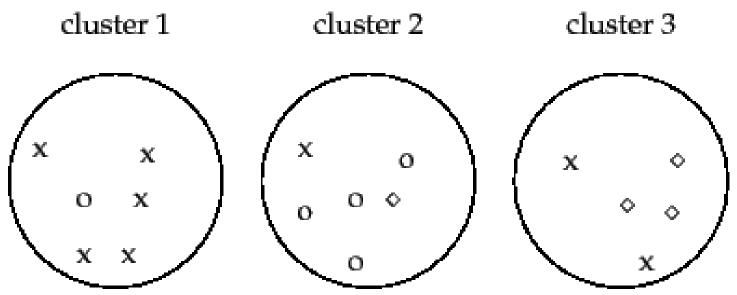
$$R = \frac{a+d}{a+b+c+d}$$

- Adjusted rand index: corrected-for-chance version of rand index
  - Compare to the expectation of the index assuming a random partition of the same cluster sizes

$$ARI = \frac{Index - ExpectedR}{MaxIndex - ExpectedR} = \frac{\sum_{i,j}\binom{n_{ij}}{2} - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{n_{i.}}{2} + \sum_j \binom{n_{.j}}{2}\right] - \left[\sum_i \binom{n_{i.}}{2}\sum_j \binom{n_{.j}}{2}\right] / \binom{n}{2}}$$

Purity and Normalized Mutual Information

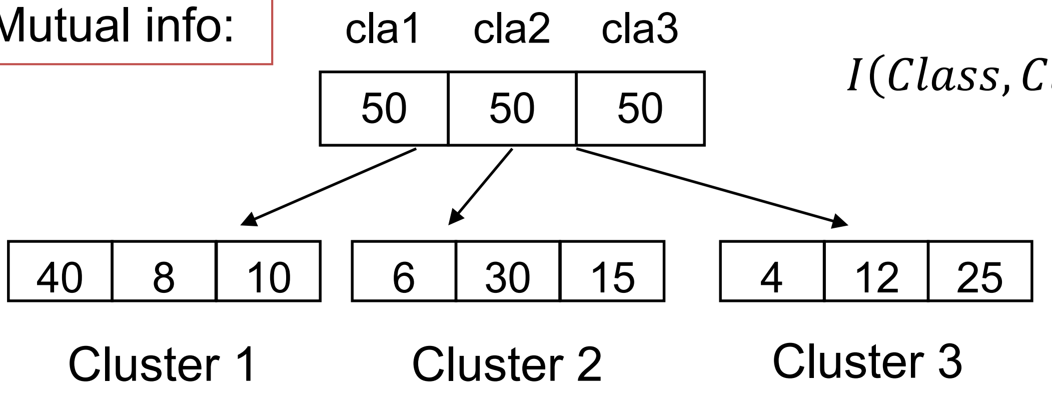
- Purity



► **Figure 16.1** Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ⋄, 3 (cluster 3). Purity is $(1/17) \times (5+4+3) \approx 0.71$.

- Normalized Mutual Information

Mutual info:



$$I(Class, Clust) = H(Class) - H(Class|Clust)$$

$$NMI = \frac{2I(Class, Clust)}{H(Clust) + H(Class)}$$

References for model selection

- Smyth, Padhraic. "Model selection for probabilistic clustering using cross-validated likelihood." *Statistics and Computing* 10.1 (2000): 63-72
- Erel Levine and Eytan Domany. Resampling Method for Unsupervised Estimation of Cluster Validity. *Neural Comput.* 13, 11, 2001, 2573-2593
- Tibshirani, Robert, and Guenther Walther. Cluster validation by prediction strength. *Journal of Computational and Graphical Statistics* 14.3 (2005)