Clustering

STAT5241 Section 2

Statistical Machine Learning

Xiaofei Shi

Task

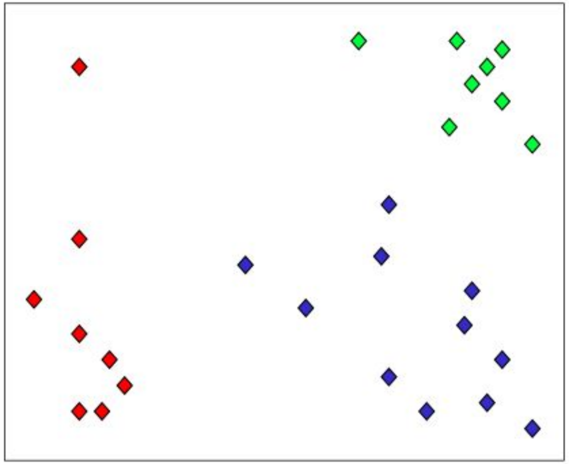
- Supervised learning, or also called prediction:
  Regression - given input, estimate output
  Classification - given input, estimate category
- Unsupervised learning:
  Data only contains inputs, but no "supervision" in data as to the descriptive outputs
  - density estimation
  - clustering
  - dimensionality reduction

Clustering

- Organizing data into groups.
- Unlike classification, there is no label provided!

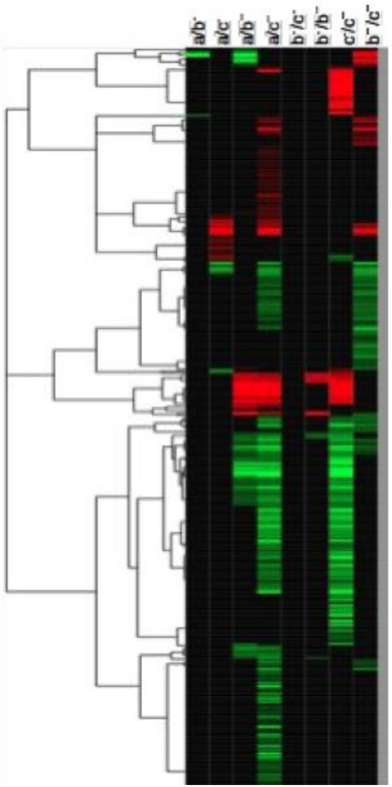
Unsupervised learning

- Informally, finding natural groups in data.
- Why do we want to do that?
- Any real application?



Clustering

- Microarrays measures the activities of all genes in different conditions
- Clustering genes can help determine new functions for unknown genes
- An early “killer application” in this area
  - The most cited (11,591) paper in PNAS!



COLUMBIA UNIVERSITY

Clustering

- Organizing data into clusters provides information about the internal structure of the data

  – Ex. Clusty and clustering genes above

- Sometimes the partitioning is the goal – Ex. Image segmentation
- Knowledge discovery in data

  – Ex. Underlying rules, recurring patterns, topics, etc.

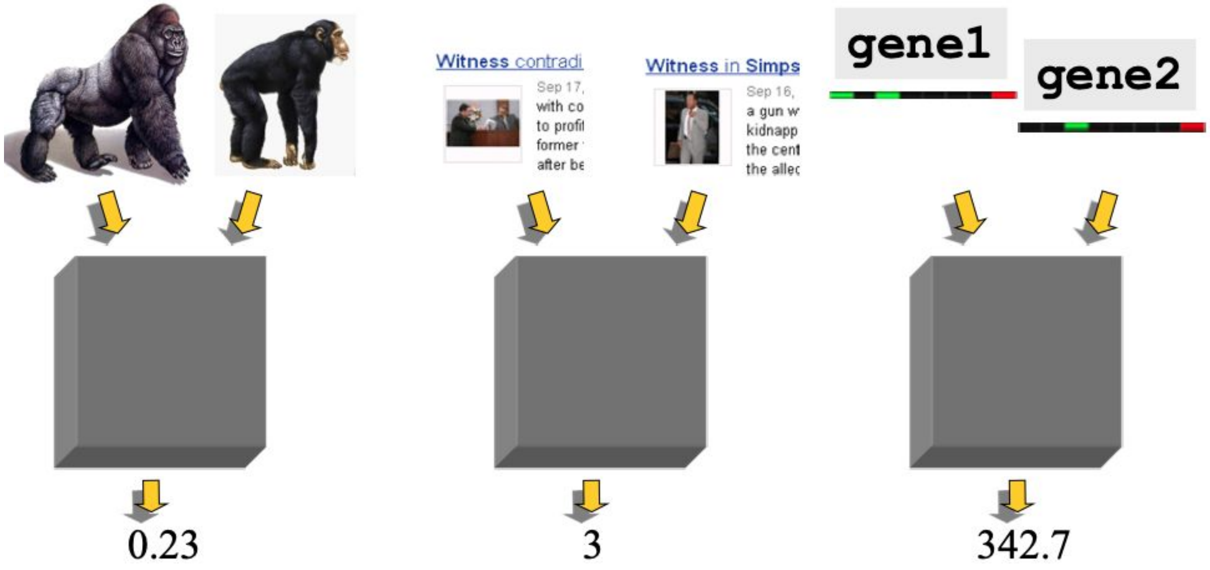
Columbia University

Clustering is subjective!

Similarity

Similarity: distance matrix

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance (dissimilarity) between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$



Similarity: distance matrix

A few examples:

- Euclidian distance

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

- Correlation coefficient

$$s(x,y) = \frac{\sum_i (x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y}$$

- Similarity rather than distance
- Can determine similar trends

COLUMBIA UNIVERSITY

Desirable properties of a clustering algorithm

- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Interpretability and usability
- Incorporation of user-specified constraints

COLUMBIA UNIVERSITY

Similarity: distance matrix

A few examples:

- Euclidian distance

$$d(x,y)=\sqrt{\sum_i (x_i - y_i)^2}$$

- Correlation coefficient

$$s(x,y)=\frac{\sum_i (x_i-\mu_x)(y_i-\mu_y)}{\sigma_x\sigma_y}$$

- Similarity rather than distance
- Can determine similar trends

COLUMBIA UNIVERSITY

Clustering algorithm

- Clustering algorithms:

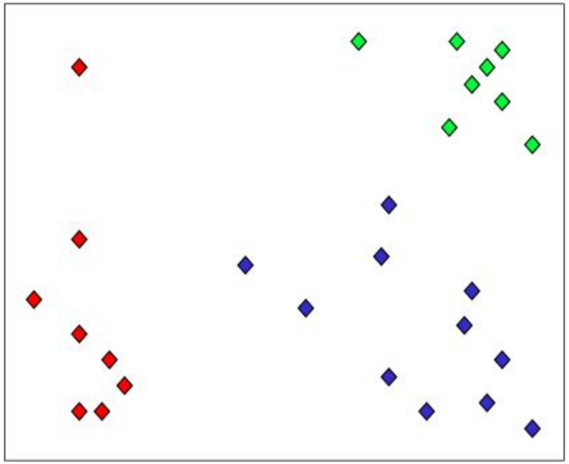
  - k-means

  - mixture methods

  - density based clustering: level sets, trees; modes

  - hierarchical clustering

  - spectral clustering

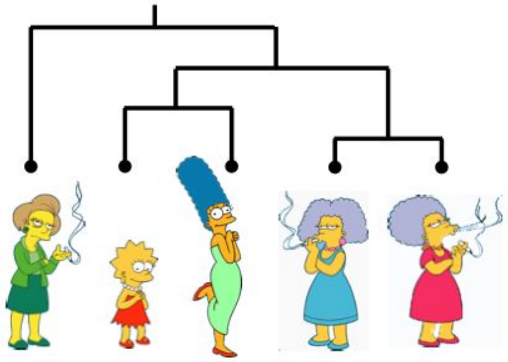


Clustering algorithm

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion (focus of this class)

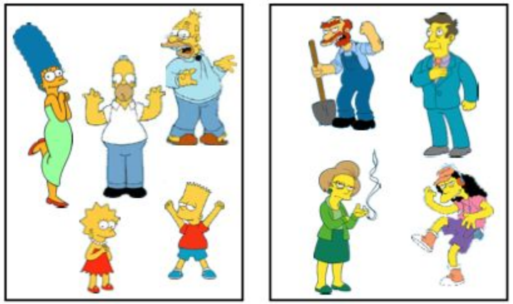
Bottom up or top down

Hierarchical



Top down

Partitional



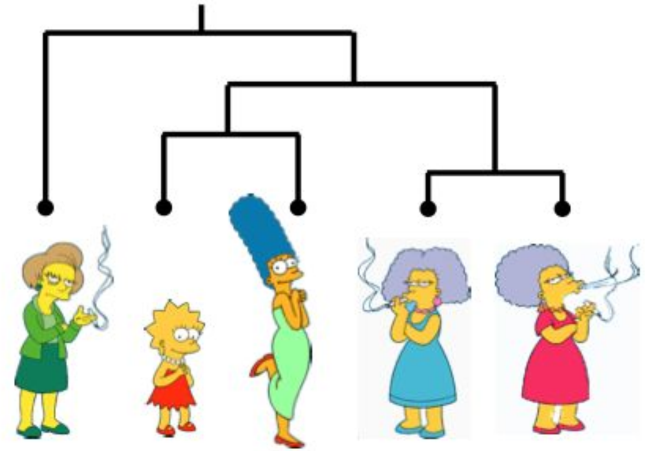
Hierarchical clustering

- Bottom-Up (agglomerative):

  - Starting with each item in its own cluster, find the best pair to merge into a new cluster. - Repeat until all clusters are fused together.

The number of dendrograms with n leafs $= (2n - 3)!/[(2^{(n-2)})\ (n-2)!]$

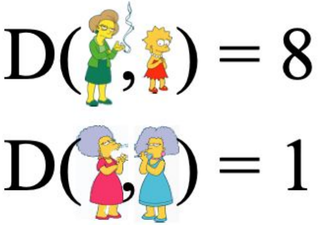
Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	…
10	34,459,425

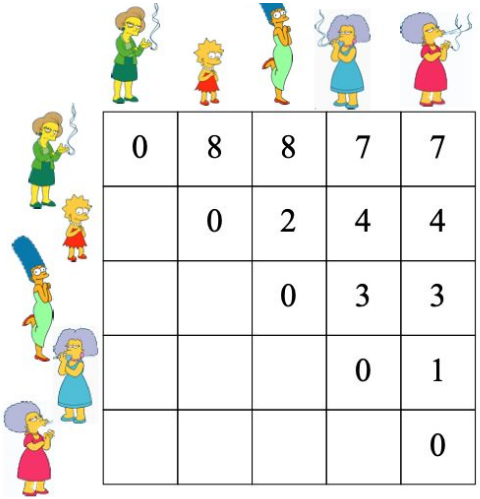


COLUMBIA UNIVERSITY

Hierarchical clustering

- Bottom-Up (agglomerative):
  - Starting with each item in its own cluster, find the best pair to merge into a new cluster. - Repeat until all clusters are fused together.

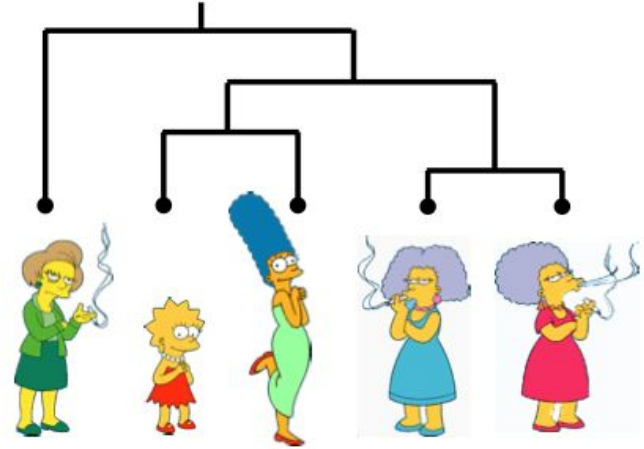




0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

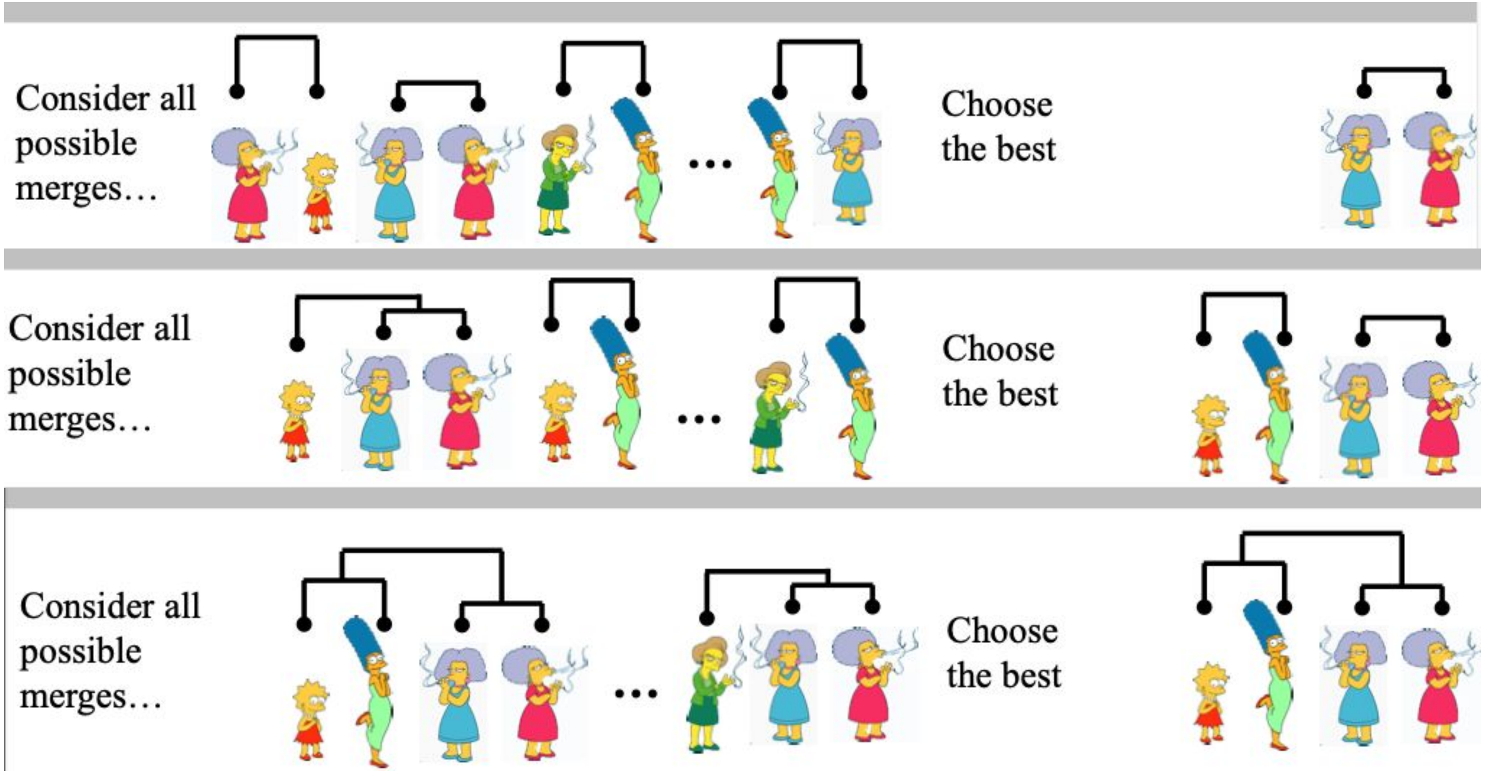
The number of dendrograms with n leafs $= (2n-3)!/[(2^{(n-2)})\,(n-2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	...
10	34,459,425



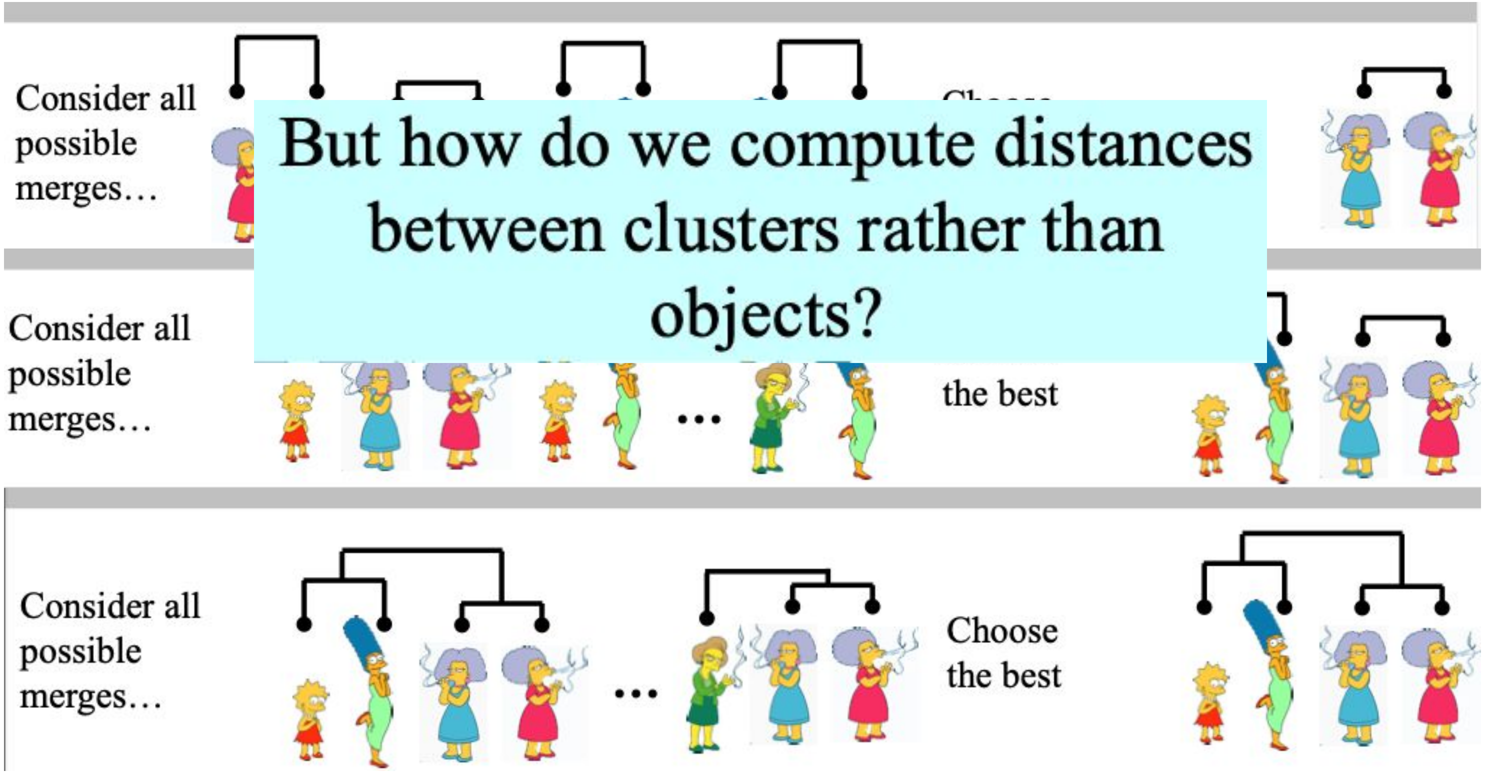
Columbia University

Hierarchical clustering



COLUMBIA UNIVERSITY

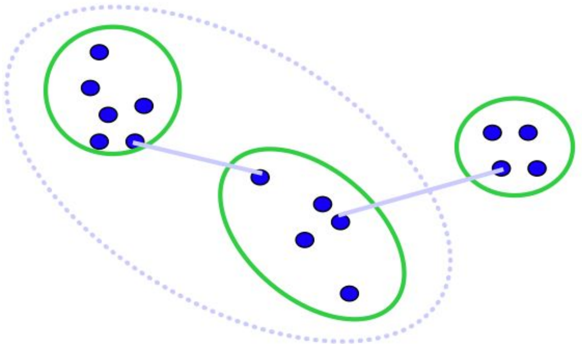
Hierarchical clustering



But how do we compute distances between clusters rather than objects?

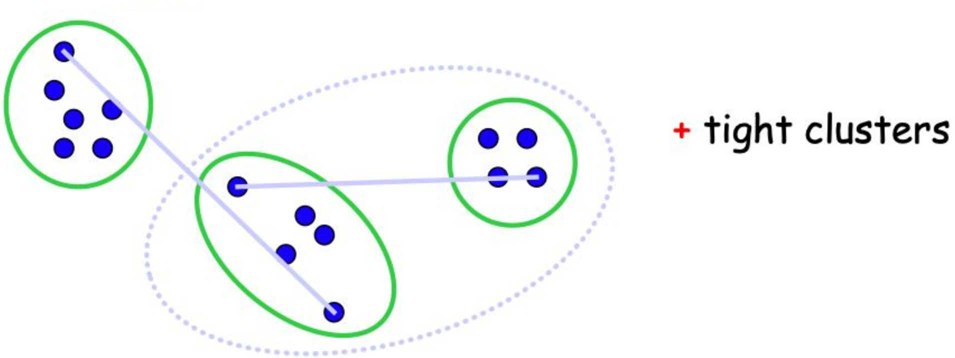
Simple link

- Cluster distance = distance of two closest members in each class



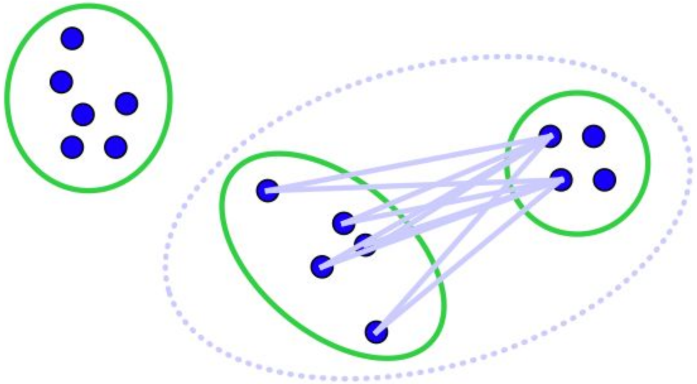
Complete link

- Cluster distance = distance of two farthest members in each class



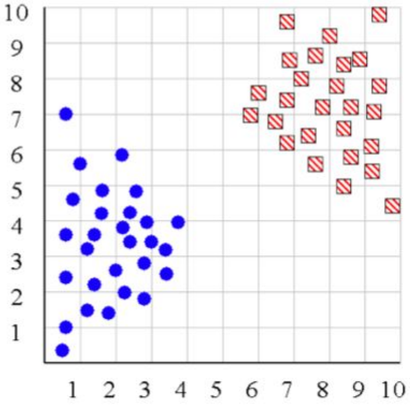
Average link

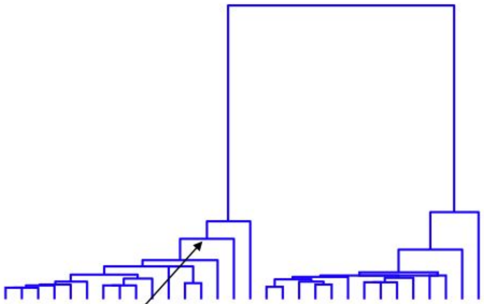
- Cluster distance = average distance of all pairs
- The most widely used measure
- Robust against noise



Columbia University

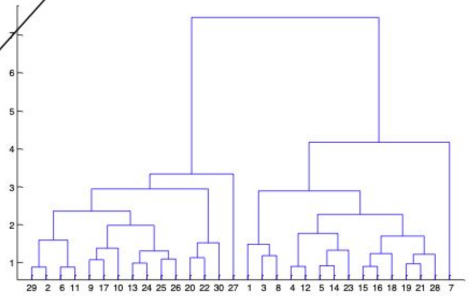
Hierarchical clustering: comparison





Single linkage

Height represents distance between objects / clusters



Average linkage

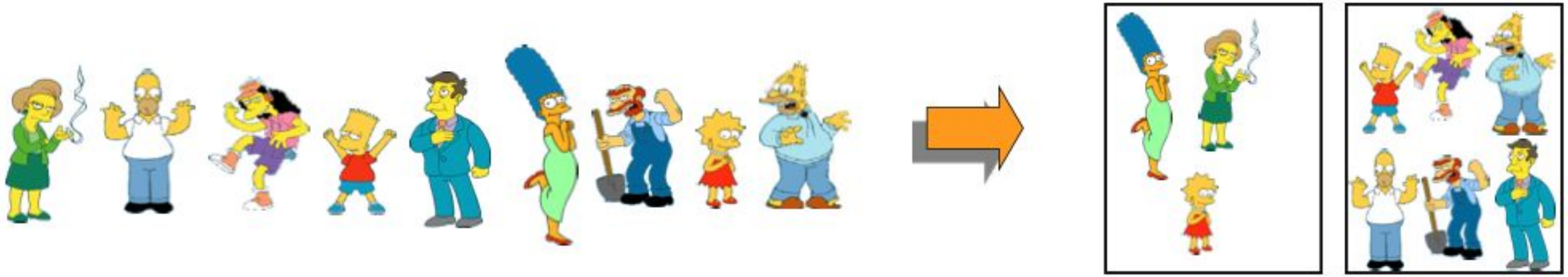
Hierarchical clustering: summary

- No need to specify the number of clusters in advance.
- Hierarchical structure maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least O(n2), where n is the number of total objects.
- Like any heuristic search algorithms, local optima are a problem.
- Interpretation of results is (very) subjective.

Columbia University

Partitional clustering

- Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.
- Since the output is only one set of clusters the user has to specify the desired number of clusters K.



COLUMBIA UNIVERSITY

Partitional clustering

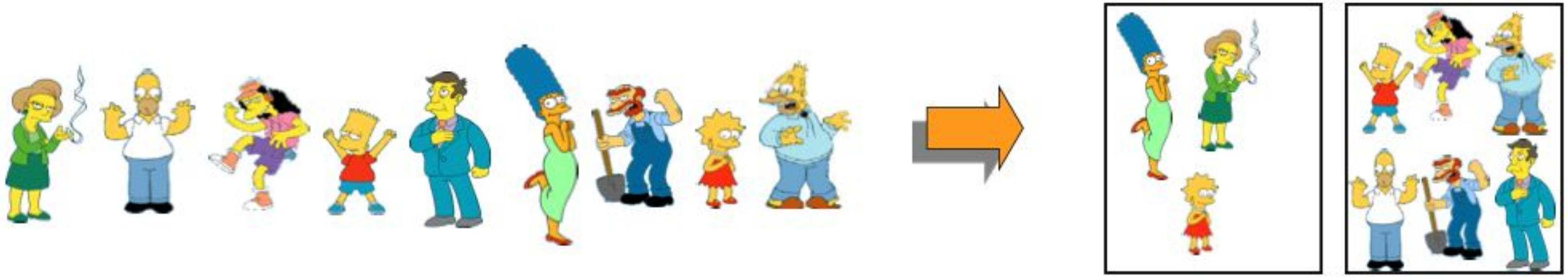
• Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.

• Since the output is only one set of clusters the user has to specify the desired number of clusters K.

• K-means: hard assignment:
each object belongs to only one cluster

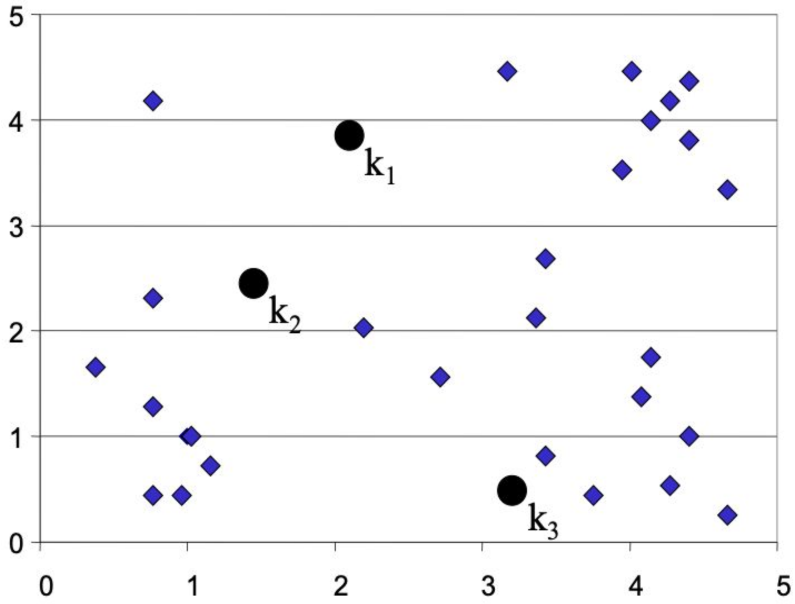
• Mixture modeling: soft assignment
probability that an object belongs to a cluster

Generative approach: think of each cluster as a component distribution, and any data point is drawn from a "mixture" of multiple component distributions



K-means clustering (Llody’s method)

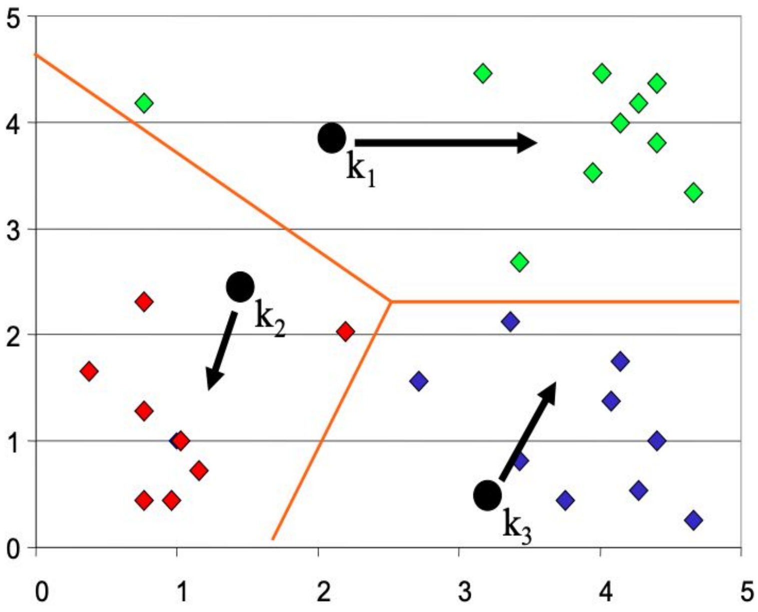
- Decide K and initialize K centers (randomly).
- Assign all objects to the nearest center.
- Move a center to the mean of its members.



COLUMBIA UNIVERSITY

K-means clustering

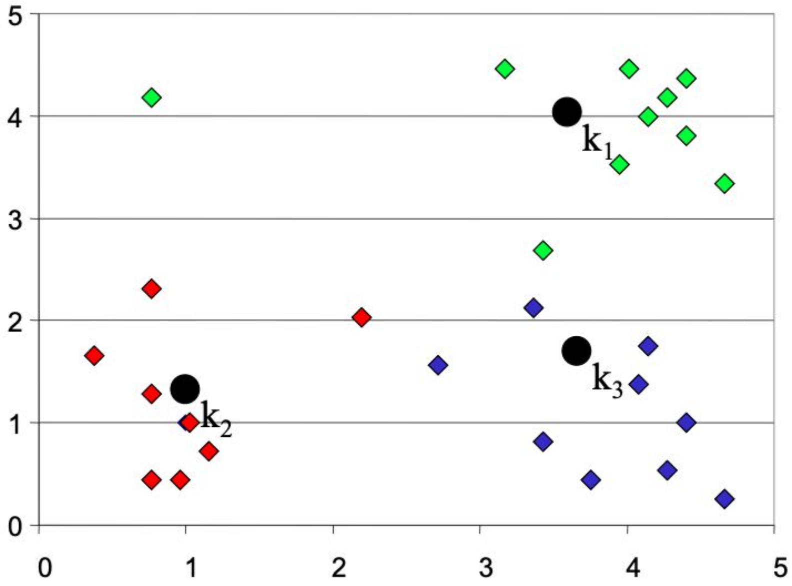
- Decide K and initialize K centers (randomly).
- Assign all objects to the nearest center.
- Move a center to the mean of its members.



COLUMBIA UNIVERSITY

K-means clustering

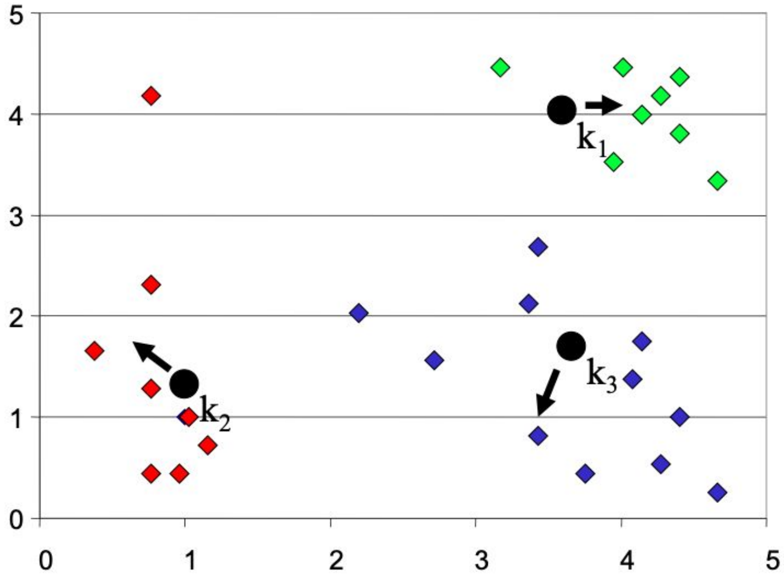
- Decide K and initialize K centers (randomly).
- Assign all objects to the nearest center.
- Move a center to the mean of its members.
- After moving centers, re-assign the objects...



Columbia University

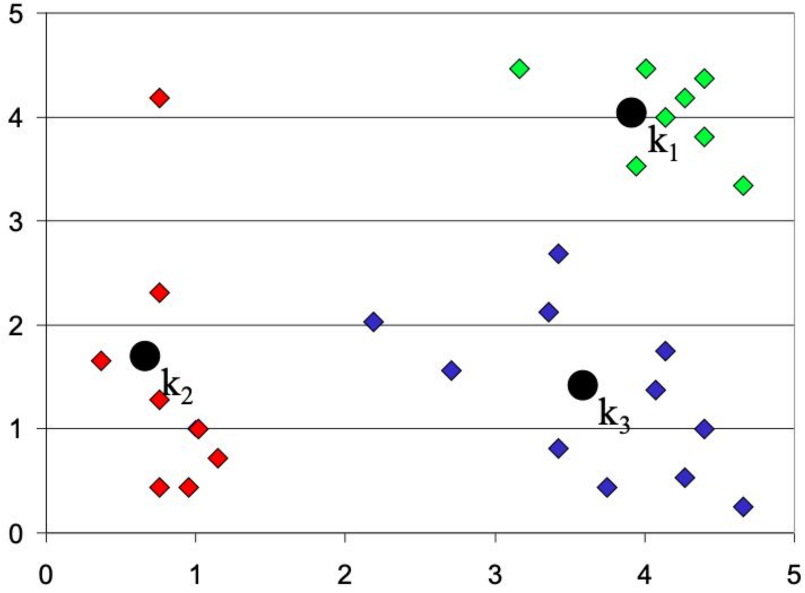
K-means clustering

- Decide K and initialize K centers (randomly).
- Assign all objects to the nearest center.
- Move a center to the mean of its members.
- After moving centers, re-assign the objects.
- Move a center to the mean of its new members.



K-means clustering

- Decide K and initialize K centers (randomly).
- Assign all objects to the nearest center.
- Move a center to the mean of its members.
- After moving centers, re-assign the objects.
- Move a center to the mean of its new members.
- Re-assign and move centers, until ... no objects changed membership.



COLUMBIA UNIVERSITY

K-means algorithm

1. Decide on a value for K, the number of clusters.

2. Initialize the K cluster centers (randomly, if necessary).

3. Decide the class memberships of the N objects by assigning them to the nearest cluster center.

4. Re-estimate the K cluster centers, by assuming the memberships found above are correct.

5. Repeat 3 and 4 until none of the N objects changed membership in the last iteration.

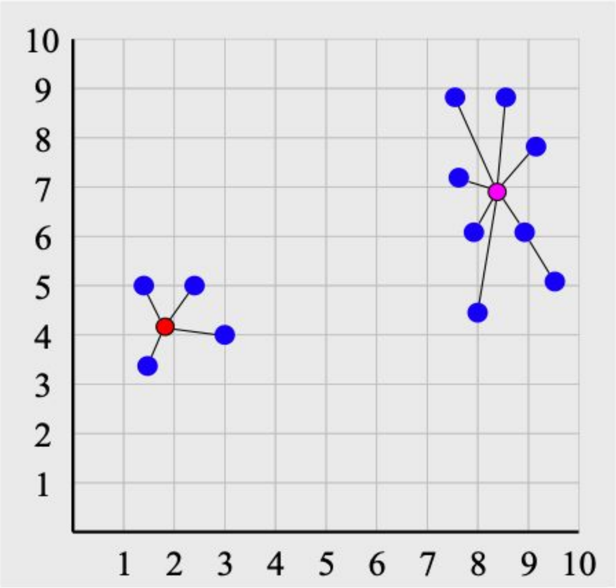
Why K-means works

- What is a good partition?
- High intra-cluster similarity
- K-means optimizes
  - the average distance to members of the same cluster

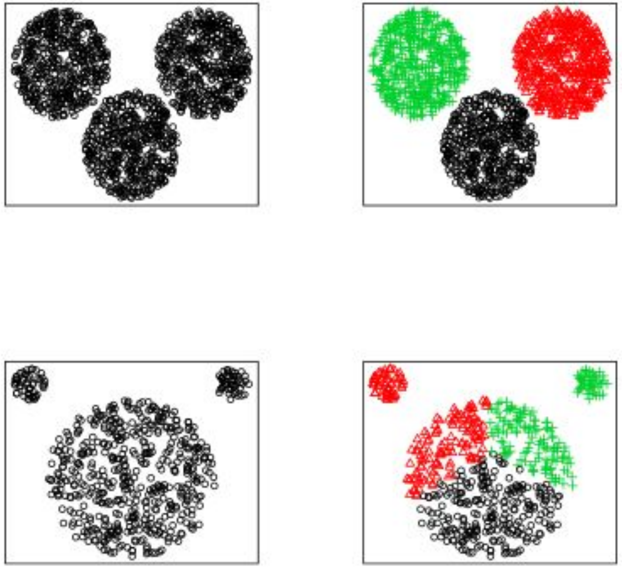
$$\sum_{k=1}^{K} \frac{1}{n_k} \sum_{i=1}^{n_k} \sum_{j=1}^{n_k} \left\| x_{ki} - x_{kj} \right\|^2$$

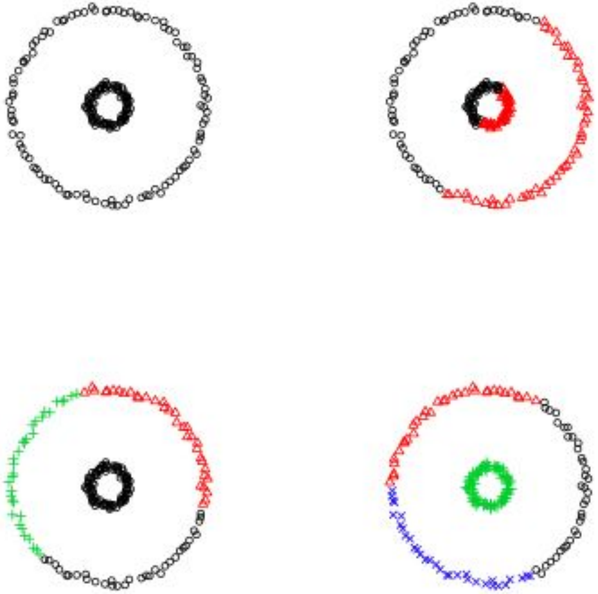
  - which is twice the total distance to centers, also called squared error

$$se = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left\| x_{ki} - \mu_k \right\|^2$$



But sometimes...





K-means: summary

- Pros:

  - simple to implement (and debug)

  - intuitive objective function: optimizes intra-cluster similarity

  - relatively efficient: O(tkn), where normally, k,t<<n

- Cons:

  - Often terminates at a local optimum. Initialization is important

  - K is a hyperparameter, needs to be specified

  - Unable to handle noisy data and outliers

  - Not suitable to discover clusters with non-convex shapes

- Summary:

  – Assign members based on current centers

  – Re-estimate centers based on current assignment

COLUMBIA UNIVERSITY

Partitional clustering

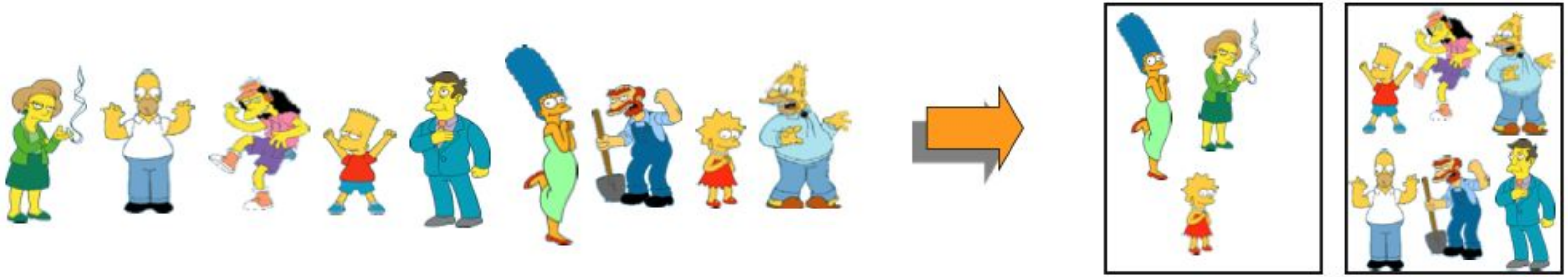
• Nonhierarchical, each instance is placed in exactly one of K non-overlapping clusters.

• Since the output is only one set of clusters the user has to specify the desired number of clusters K.

• K-means: hard assignment:
each object belongs to only one cluster

• Mixture modeling: soft assignment
probability that an object belongs to a cluster

Generative approach: think of each cluster as a component distribution, and any data point is drawn from a "mixture" of multiple component distributions



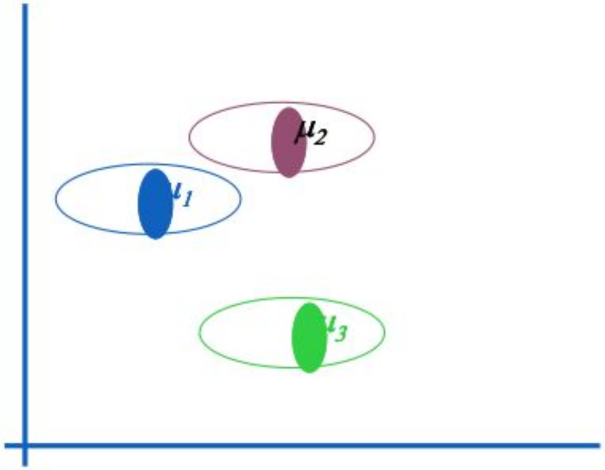
Columbia University

Gaussian mixture modeling

Mixture of K Gaussian distributions: (Multi-modal distribution)

$$p(x|y=i) \sim \mathrm{N}(\mu_i, \sigma^2 I)$$

$$p(x) = \sum_i \underbrace{p(x|y=i)}_{\text{Mixture component}} \underbrace{P(y=i)}_{\text{Mixture proportion}}$$

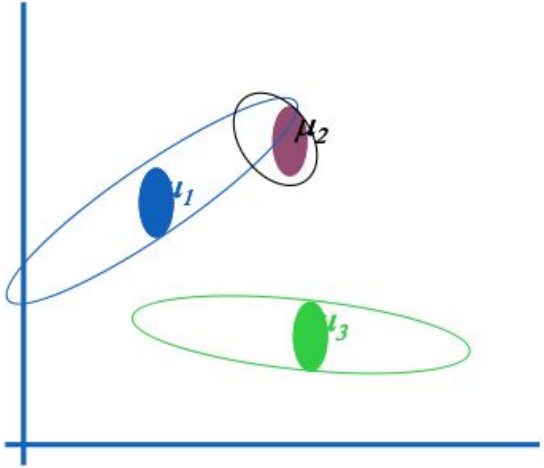


Gaussian mixture modeling

GMM – Gaussian Mixture Model (Multi-modal distribution)

$$p(x|y=i) \sim \mathrm{N}(\mu_i, \Sigma_i)$$

$$p(x) = \sum_i \underbrace{p(x|y=i)}_{\text{Mixture component}} \; \underbrace{P(y=i)}_{\text{Mixture proportion}}$$



Gaussian mixture modeling

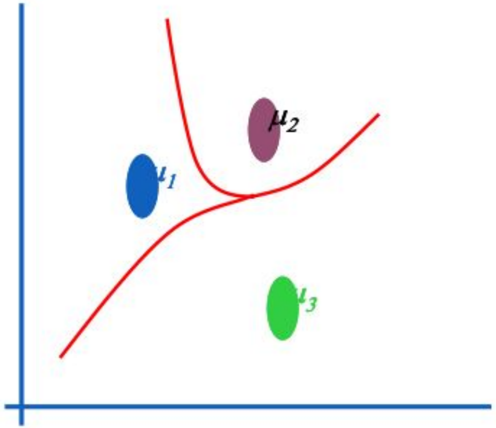
GMM – Gaussian Mixture Model (Multi-modal distribution)

$p(x|y=i) \sim \mathrm{N}(\mu_i, \Sigma_i)$

Gaussian Bayes Classifier:

$$\log \frac{P(y=i\,|\,x)}{P(y=j\,|\,x)}$$

$$= \log \frac{p(x\,|\,y=i)P(y=i)}{p(x\,|\,y=j)P(y=j)}$$

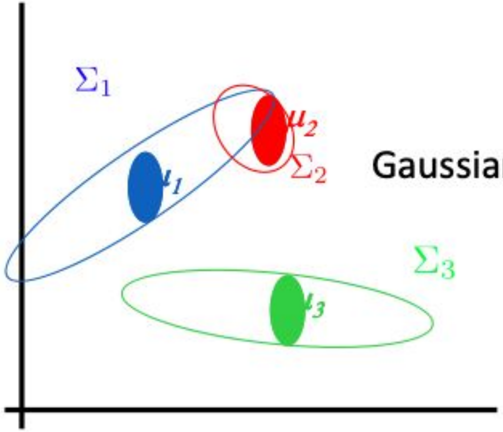


Learning General GMM

$$x_1, \ldots, x_m \sim p(x) = \sum_{i=1}^{k} p(x|Y=i)P(Y=i)$$

Mixture component

Mixture proportion, p_i



Gaussian mixture model

$$p(x|Y=i) \sim \mathcal{N}(\mu_i, \Sigma_i)$$

Parameters: $\{p_i, \mu_i, \Sigma_i\}_{i=1}^{K}$

- How to estimate parameters? Maximum Likelihood
  But don't know labels Y (recall Gaussian Bayes classifier)

Columbia University

Learning General GMM

Maximize marginal likelihood:

$$\arg\max \prod_j P(x_j) = \arg\max \prod_j \sum_{i=1}^{K} P(y_j=i,x_j) \qquad \text{... marginalizing } y_j$$

$$= \arg\max \prod_j \sum_{i=1}^{K} P(y_j=i)p(x_j|y_j=i)$$

$P(y_j=i) = P(y=i)$ Mixture component i is chosen with prob $P(y = i)$

$$= \arg\max \prod_{j=1}^{m} \sum_{i=1}^{k} P(y=i) \frac{1}{\sqrt{\det(\Sigma_i)}} \exp\left[-\frac{1}{2}(x_j-\mu_i)^T \Sigma_i (x_j-\mu_i)\right]$$

How do we find the $\boldsymbol{\mu}_i$'s and P(y=i)s which give max. marginal likelihood?

* Set $\frac{\partial}{\partial \mu_i} \log \text{Prob}(....) = 0$ and solve for μ_i's. Non-linear non-analytically solvable

* Use gradient descent: Doable, but often slow

COLUMBIA UNIVERSITY

Connection to K-means

Maximize marginal likelihood:

$$\arg\max \prod_j P(x_j) = \arg\max \prod_j \sum_{i=1}^{K} P(y_j=i, x_j)$$
$$= \arg\max \prod_j \sum_{i=1}^{K} P(y_j=i)p(x_j|y_j=i)$$

- What happens if we assume **Hard assignment**?

$$P(y_j = i) = 1 \text{ if } i = C(j)$$
$$= 0 \text{ otherwise}$$

Same as k-means (if we assume equal covariance matrix)!

$$\arg\max \prod_j P(x_j) = \arg\max \prod_j p(x_j|y_j=C(j))$$
$$= \arg\max \prod_{j=1}^{n} \exp\left(\frac{-1}{2\sigma^2}\|x_j - \mu_{C(j)}\|^2\right)$$
$$= \arg\min \sum_{j=1}^{n} \|x_j - \mu_{C(j)}\|^2) = \arg\min_{\mu,C} F(\mu, C)$$

Columbia University

Expectation-Maximization (EM Algorithm)

A general algorithm to deal with hidden data, but we will study it in the context of unsupervised learning (hidden labels) first

- No need to choose step size as in Gradient methods.
- EM is an Iterative algorithm with two linked steps:

  E-step: fill-in hidden data (Y) using inference

  M-step: apply standard MLE/MAP method to estimate parameters

  $\{p_i, \mu_i, \Sigma_i\}^k_{i=1}$
- We will see that this procedure monotonically improves the likelihood (or leaves it unchanged). Thus it always converges to a local optimum of the likelihood.

COLUMBIA UNIVERSITY

EM Algorithm with known variance

E-step

Compute "expected" classes of all datapoints for each class

$$P(y=i \mid x_j, \mu_1 \ldots \mu_k) \propto \exp\left(-\frac{1}{2\sigma^2}\|x_j - \mu_i\|^2\right) P(y=i)$$

In K-means "E-step"
we do hard assignment

EM does soft assignment

M-step

Compute MLE for $\boldsymbol{p_i}$, **μ and Σ** given our data's class membership distributions (weights)

Similar to K-means, but with weighted data

Iterate.

COLUMBIA UNIVERSITY

General EM Algorithm

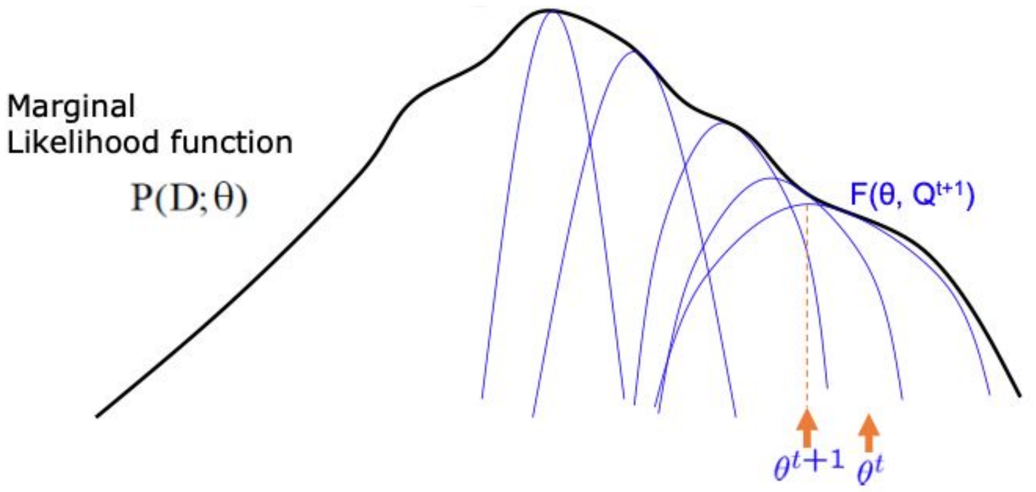
Marginal likelihood – **x** is observed, **z** is missing:

$$\begin{aligned} \log \mathrm{P}(\mathrm{D};\theta) &= \log \prod_{j=1}^{m} P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log P(\mathbf{x}_j \mid \theta) \\ &= \sum_{j=1}^{m} \log \sum_{\mathbf{z}} P(\mathbf{x}_j, \mathbf{z} \mid \theta) \end{aligned}$$

$$\mathrm{D} = \{\mathrm{x}_j\}_{j=1}^{m}$$

θ - model parameter(s)

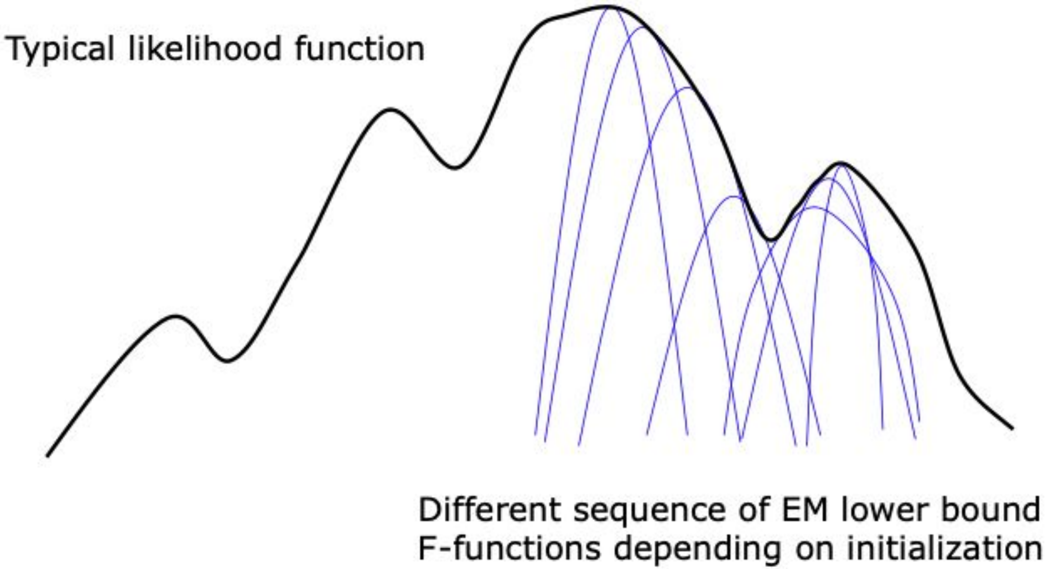
General EM Algorithm



Sequence of EM lower bound F-functions

EM monotonically converges to a local maximum of likelihood !

General EM Algorithm



Use multiple, randomized initializations in practice

EM Algorithm

- A way of maximizing likelihood function for hidden variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some “missing” or “unobserved” data from observed data and current parameters.
  2. Using this “complete” data, find the maximum likelihood parameter estimates.
- Alternate between filling in the latent variables using the best guess (posterior) and updating the parameters based on this guess:
  1. E-step: $Q^{t+1} = \arg\max_{Q} F(\theta^t, Q)$
  2. M-step: $\theta^{t+1} = \arg\max_{\theta} F(\theta, Q^{t+1})$
- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.
- EM performs coordinate ascent on F, but can get stuck in local minima.
- Extremely popular and useful in practice.

Columbia University

K-means vs GMM

1. Decide on a value for *K*, the number of clusters.

2. Initialize the *K* cluster centers / parameters (randomly).

K-means

3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *K* cluster centers, by assuming the memberships found above are correct.

GMM

3. E-step: assign *probabilistic* membership

4. M-step: re-estimate parameters based on *probabilistic* membership

5. Repeat 3 and 4 until parameters do not change.

COLUMBIA UNIVERSITY

Comparison

	Hierarchical	K-means	GMM
Running time	naively, $O(N^3)$	fastest (each iteration is linear)	fast (each iteration is linear)
Assumptions	requires a similarity / distance measure	strong assumptions	strongest assumptions
Input parameters	none	K (number of clusters)	K (number of clusters)
Clusters	subjective (only a tree is returned)	exactly K clusters	exactly K clusters

Columbia University

References

- Christopher Bishop: Pattern Recognition and Machine Learning, Chapter 9
- Trevor Hastie, Robert Tibshirani, Jerome Friedman: The Elements of Statistical Learning: Data Mining, Inference and Prediction, Chapter 14
- Ziv Bar-Joseph, Tom Mitchell, Pradeep Ravikumar and Aarti Singh: CMU 10-701

COLUMBIA UNIVERSITY