Unsupervised learning: latent space analysis and clustering

Yifeng Tao

School of Computer Science

Carnegie Mellon University

Slides adapted from Tom Mitchell, David Sontag, Ziv Bar-Joseph

Outline

- Dimension reduction/latent space analysis
  - PCA
  - ICA
  - t-SNE
- Clustering
  - K-means
  - GMM
  - Hierarchical/agglomerative clustering

Unsupervised mapping to lower dimension

- Instead of choosing subset of features, create new features (dimensions) defined as functions over all features
- Don't consider class labels, just the data points

$$\mathbf{x}_1, \ldots, \mathbf{x}_N \qquad \mathbf{y}_1, \ldots, \mathbf{y}_N \text{ (with } \mathbf{y}_i \in \mathbb{R}^d\text{)}$$

Principle Components Analysis

- Given data points in *d*-dimensional space, project into lower dimensional space while preserving as much information as possible
  - E.g., find best planar approximation to 3D data
  - E.g., find best planar approximation to 10^4D data
- In particular, choose projection that minimizes the squared error in reconstructing original data

[Slide from Tom Mitchell]

PCA: Find Projections to Minimize Reconstruction Error

- Assume data is set of d-dimensional vectors, where *n*-th vector is

$$\mathbf{x}^n = \langle x_1^n \dots x_d^n \rangle$$

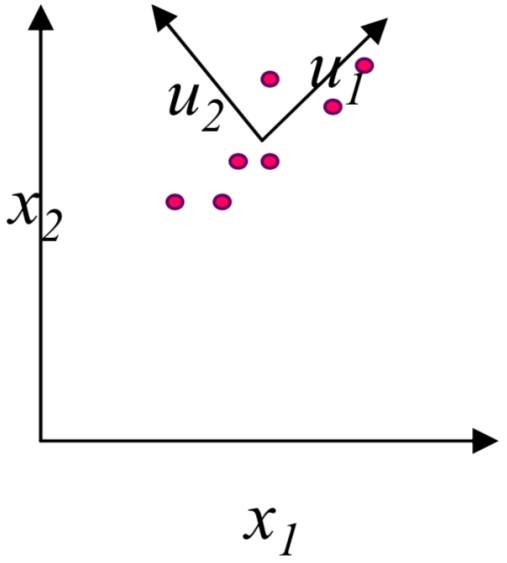
- We can represent these in terms of any *d* orthogonal basis vectors

$$\mathbf{x}^n = \sum_{i=1}^{d} z_i^n \mathbf{u}_i; \quad \mathbf{u}_i^T \mathbf{u}_j = \delta_{ij}$$

PCA: given M<d. Find $\langle \mathbf{u}_1 \dots \mathbf{u}_M \rangle$

that minimizes $E_M \equiv \sum_{n=1}^{N} ||\mathbf{x}^n - \hat{\mathbf{x}}^n||^2$

where $\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^{M} z_i^n \mathbf{u}_i$



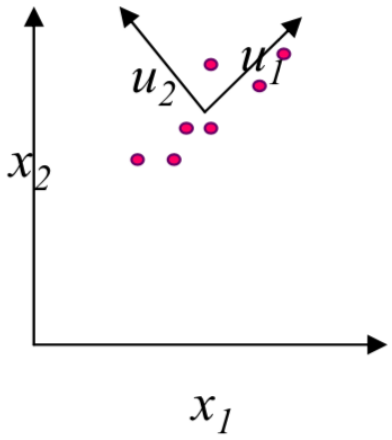
[Slide from Tom Mitchell]

PCA

PCA: given M<d. Find $\langle \mathbf{u}_1 \dots \mathbf{u}_M \rangle$

that minimizes $E_M \equiv \sum_{n=1}^{N} ||\mathbf{x}^n - \hat{\mathbf{x}}^n||^2$

where $\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^{M} z_i^n \mathbf{u}_i$



- Note we get zero error if M=d, so all error is due to missing components.

Therefore, $E_M = \sum_{i=M+1}^{d} \sum_{n=1}^{N} [\mathbf{u}_i^T (\mathbf{x}^n - \bar{\mathbf{x}})]^2$

$= \sum_{i=M+1}^{d} \mathbf{u}_i^T \Sigma \, \mathbf{u}_i$

This minimized when $\boldsymbol{u}_i$ is eigenvector of Σ, the covariance matrix of X. i.e., minimized when:

$$\Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$$

Covariance matrix: $\Sigma = \sum_n (\mathbf{x}^n - \bar{\mathbf{x}})(\mathbf{x}^n - \bar{\mathbf{x}})^T$

[Slide from Tom Mitchell]

PCA

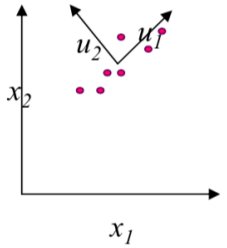
Minimize $E_M = \sum_{i=M+1}^{d} \mathbf{u}_i^T \Sigma \, \mathbf{u}_i$

$\rightarrow \quad \Sigma \mathbf{u}_i = \lambda_i \mathbf{u}_i$

Eigenvector of Σ

Eigenvalue (scalar)

$\rightarrow \quad E_M = \sum_{i=M+1}^{d} \lambda_i$



PCA algorithm 1:

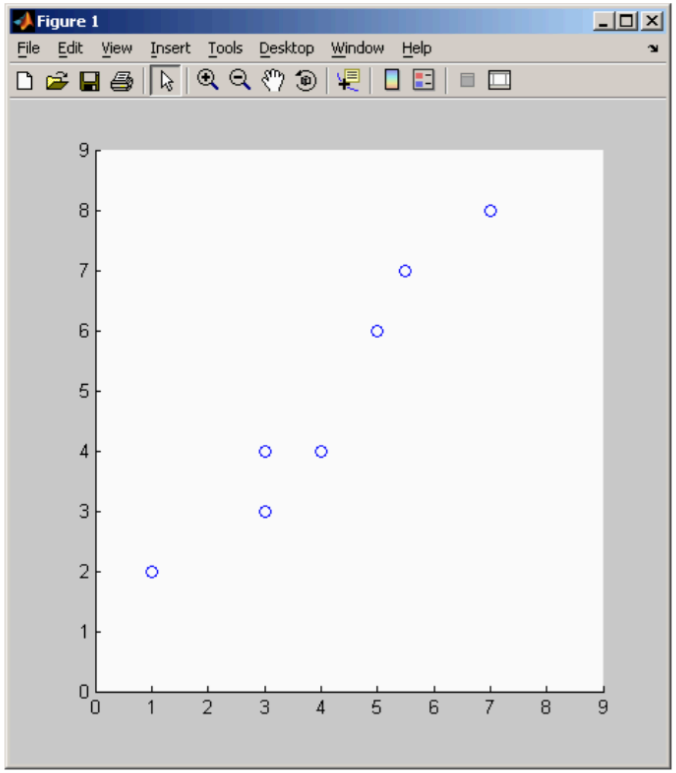
1. X ← Create N x d data matrix, with one row vector x^n per data point
2. X ← subtract mean $\bar{x}$ from each row vector x^n in X
3. Σ ← covariance matrix of X
4. Find eigenvectors and eigenvalues of Σ
5. PC's ← the M eigenvectors with largest eigenvalues

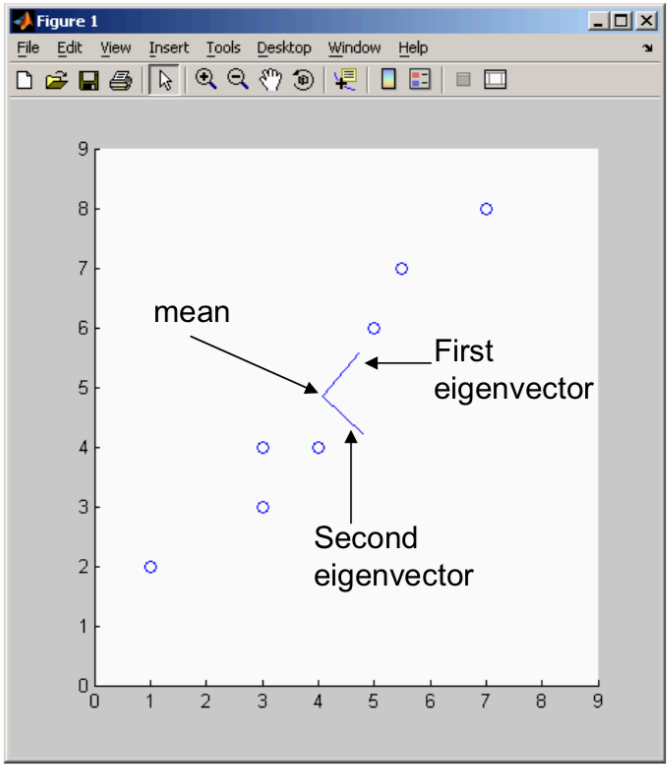
- More strict derivation in Bishop book.

[Slide from Tom Mitchell]

PCA Example

$$\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^{M} z_i^n \mathbf{u}_i$$



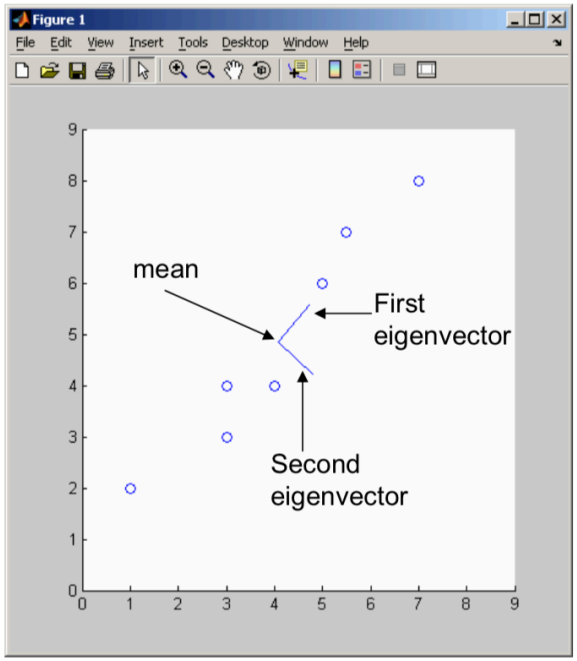


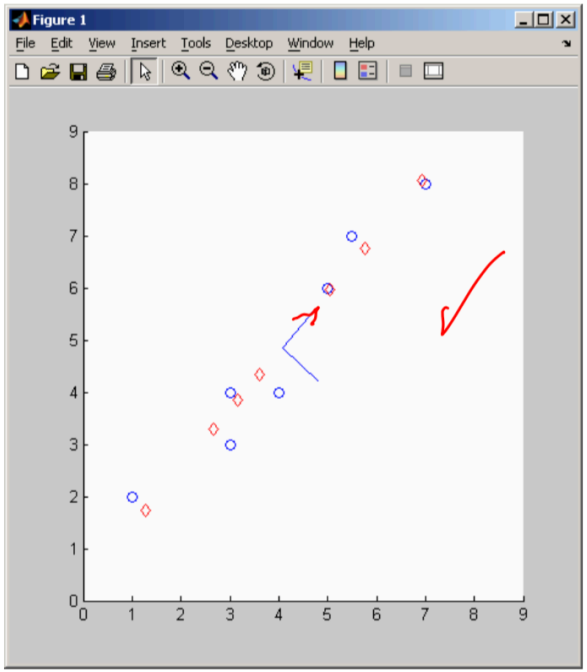
[Slide from Tom Mitchell]

PCA Example

$$\hat{\mathbf{x}}^n = \bar{\mathbf{x}} + \sum_{i=1}^{M} z_i^n \mathbf{u}_i$$

Reconstructed data using only first eigenvector (M=1)





[Slide from Tom Mitchell]

PCA Example

faces

Eigenfaces

Thanks to Christopher DeCoro
see http://www.cs.princeton.edu/~cdecoro/eigenfaces/

[Slide from Tom Mitchell]

Reconstructing a face from the first N components (eigenfaces)

In this next image, we show a similar picture, but with each additional face representing an additional 8 principle components. You can see that it takes a rather large number of images before the picture looks totally correct.

[Slide from Tom Mitchell]

Independent Components Analysis

- PCA seeks directions <Y_1 ... Y_M> in feature space X that minimize reconstruction error
- ICA seeks directions <Y_1 ... Y_M> that are most *statistically independent*. I.e., that minimize $I(Y)$, the mutual information between the Y_j:

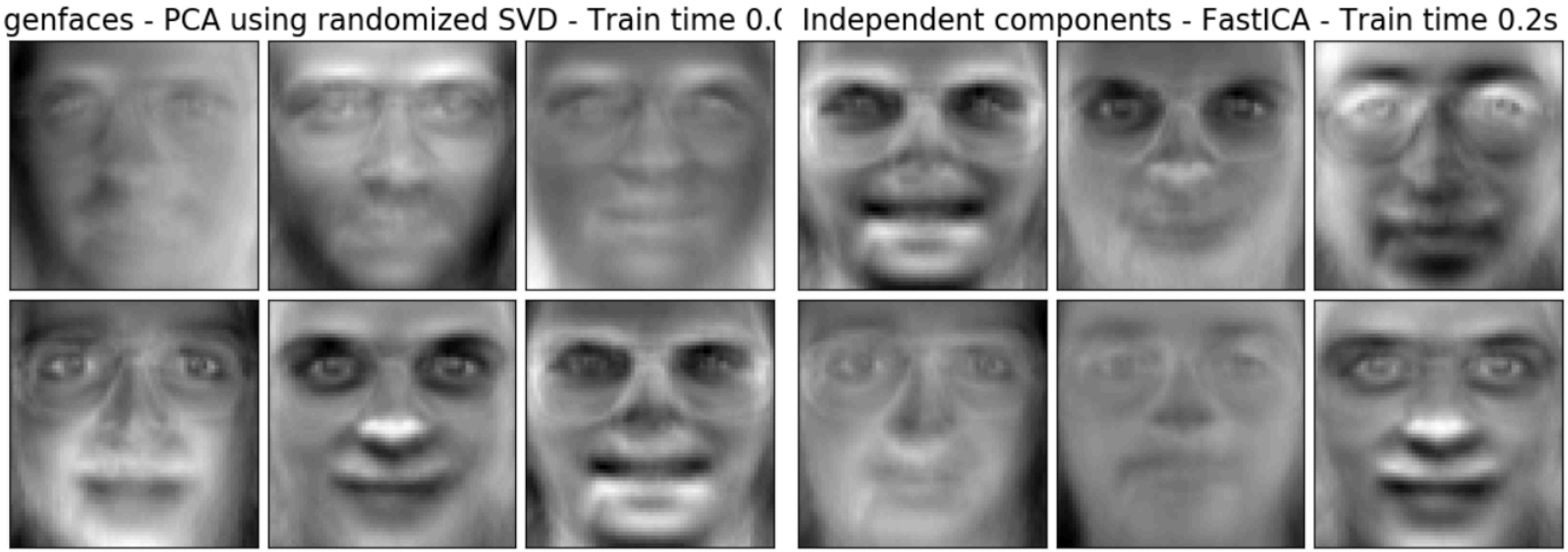
$$I(Y) = \left[\sum_{j=1}^{J} H(Y_j)\right] - H(Y)$$

  where $H(Y)$ is the entropy of Y

- Widely used in signal processing

[Slide from Tom Mitchell]

ICA example

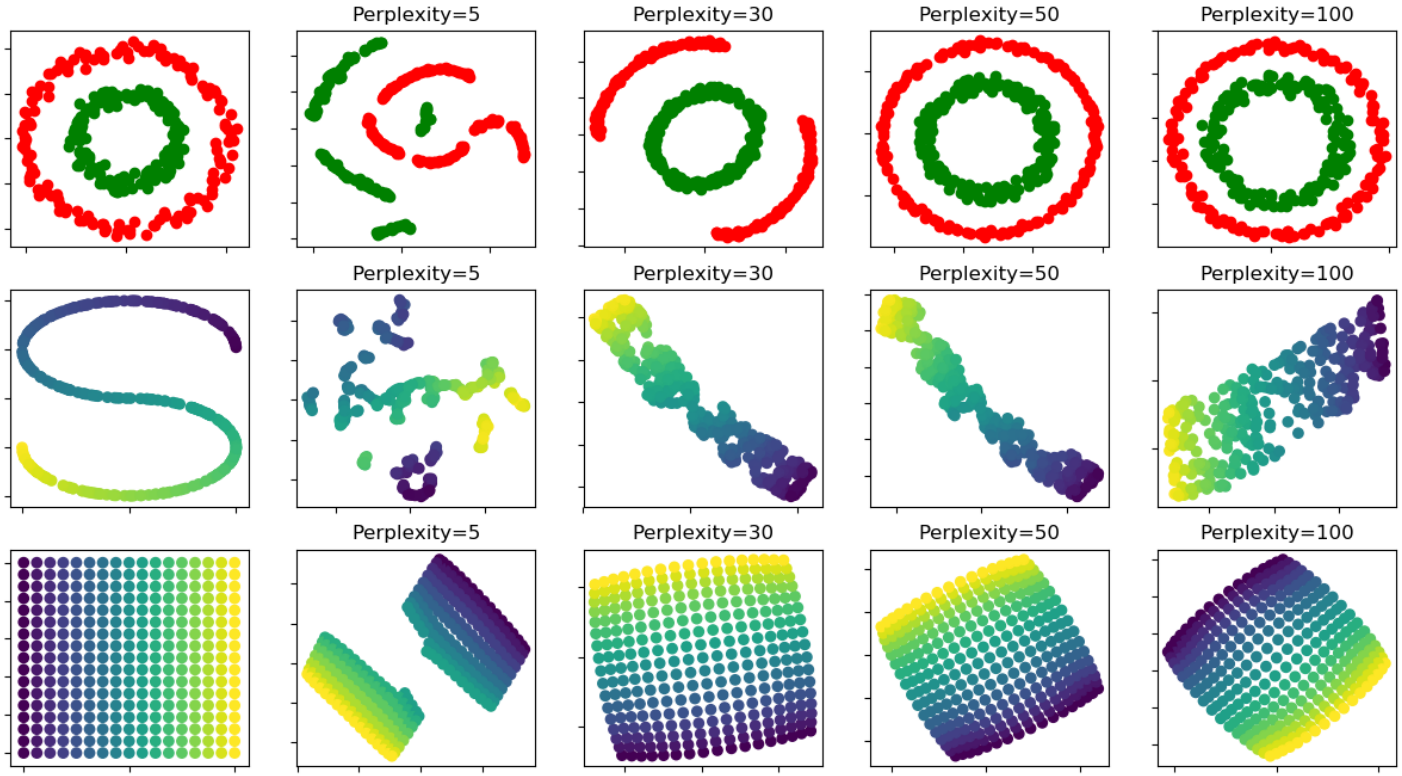


- Both PCA and ICA try to find a set of vectors, a basis, for the data. So you can write any point (vector) in your data as a linear combination of the basis.
- In PCA the basis you want to find is the one that best explains the variability of your data.
- In ICA the basis you want to find is the one in which each vector is an independent component of your data.

[Slide from https://www.quora.com/What-is-the-difference-between-PCA-and-ICA]

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- Nonlinear dimensionality reduction technique
- Manifold learning



[Figure from https://scikit-learn.org/stable/auto_examples/manifold/plot_t_sne_perplexity.html#sphx-glr-auto-examples-manifold-plot-t-sne-perplexity-py]

t-SNE

- $\mathbf{x}_1, \ldots, \mathbf{x}_N \rightarrow \mathbf{y}_1, \ldots, \mathbf{y}_N$ (with $\mathbf{y}_i \in \mathbb{R}^d$)
- Two stages:
  - First, t-SNE constructs a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar points have an extremely small probability of being picked.

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)} \qquad p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

  - Second, t-SNE defines a similar probability distribution over the points in the low-dimensional map, and it minimizes the Kullback-Leibler divergence between the two distributions with respect to the locations of the points in the map.

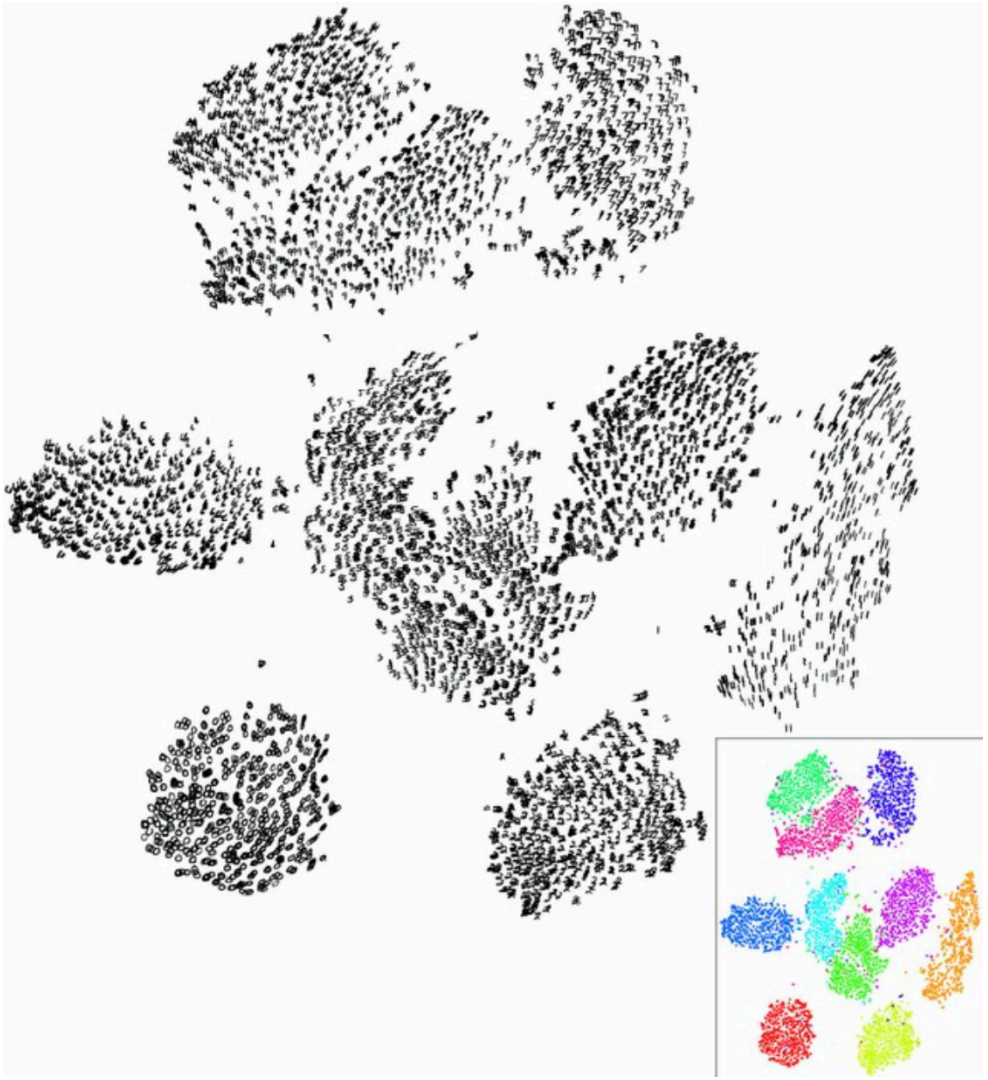
$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}} \qquad KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- Minimized using gradient descent

[Slide from https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding]

t-SNE example

- Visualizing MNIST



[Figure from https://lvdmaaten.github.io/tsne/]

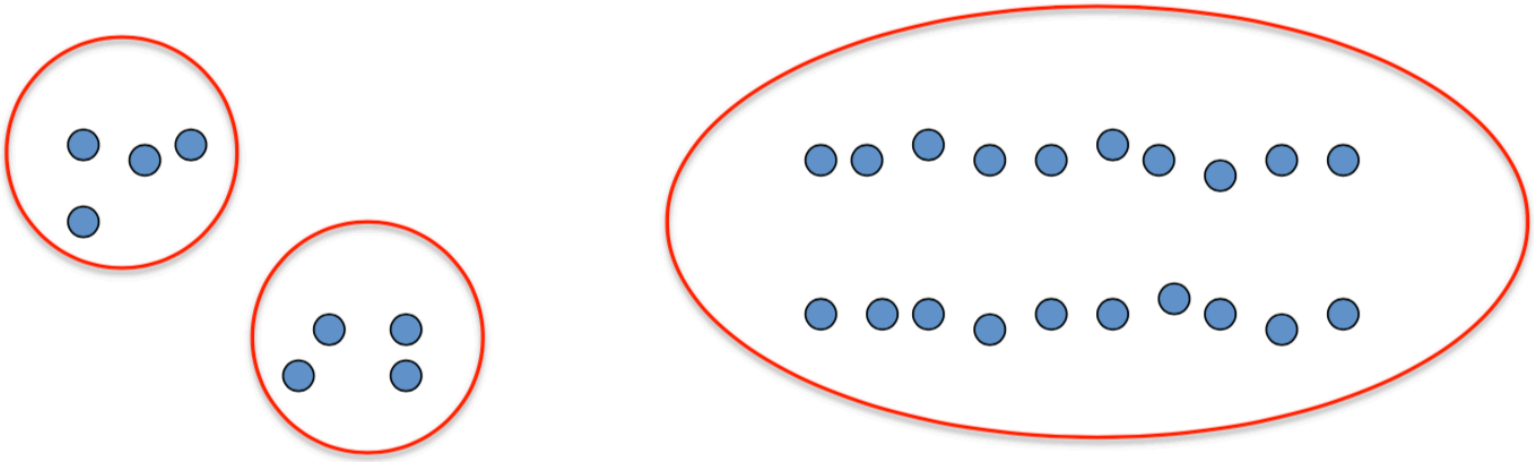
Clustering

- Unsupervised learning
  - Requires data, but no labels
- Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
- Useful when don't know what you're looking for

[Slide from David Sontag]

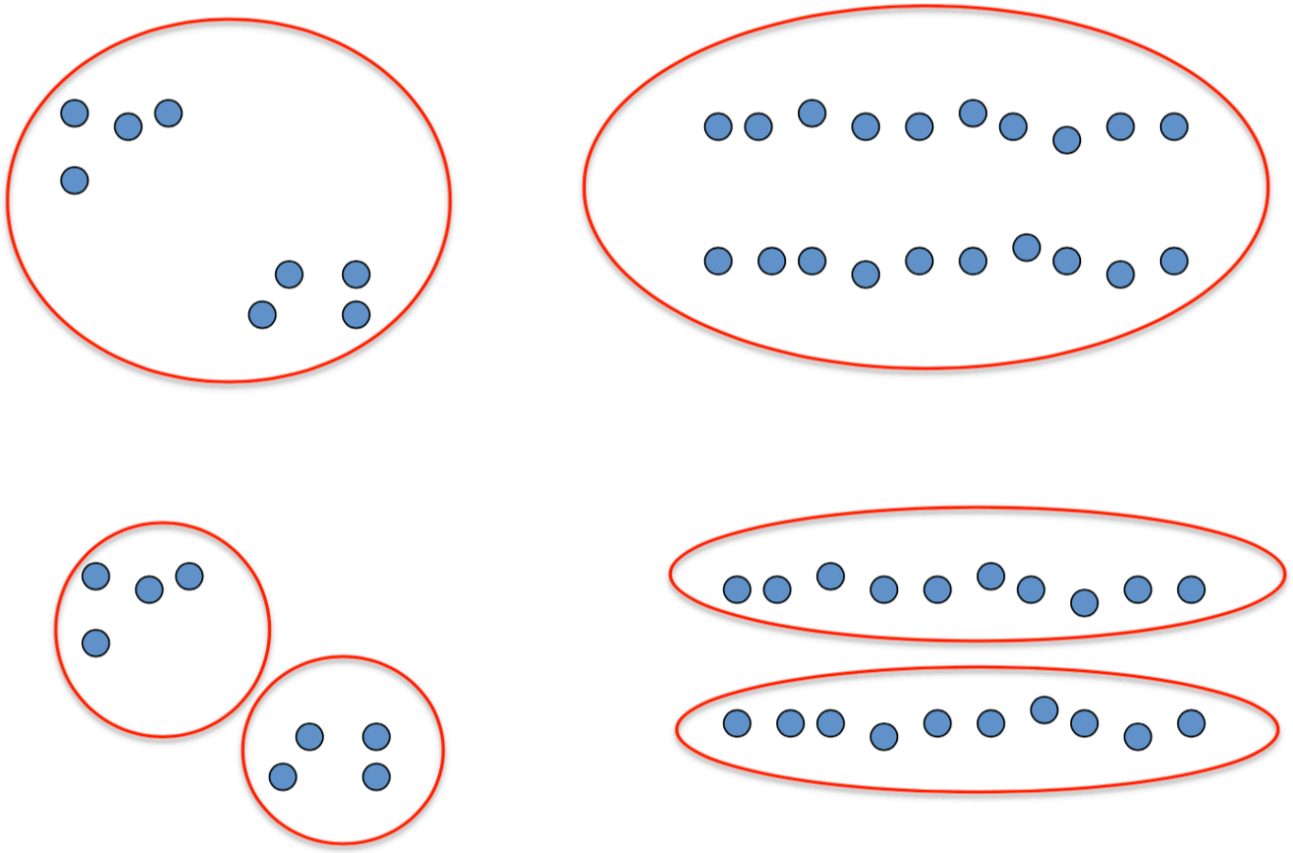
Clustering

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



[Slide from David Sontag]

- The clustering result can be quite different based on different rules.



[Slide from David Sontag]

Distance measure

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

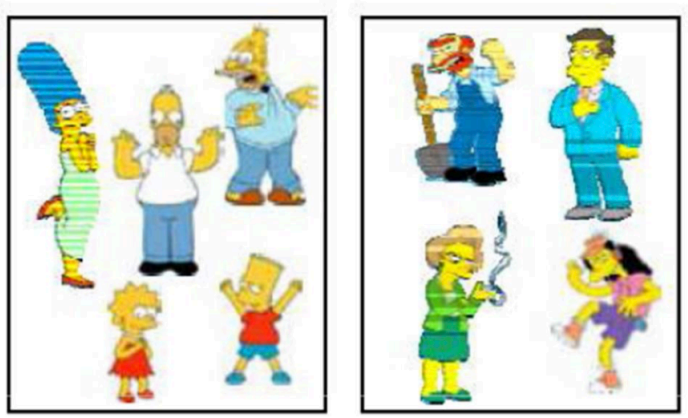
$$\text{dist}(\vec{x}, \vec{y}) = ||\vec{x} - \vec{y}||_2^2$$

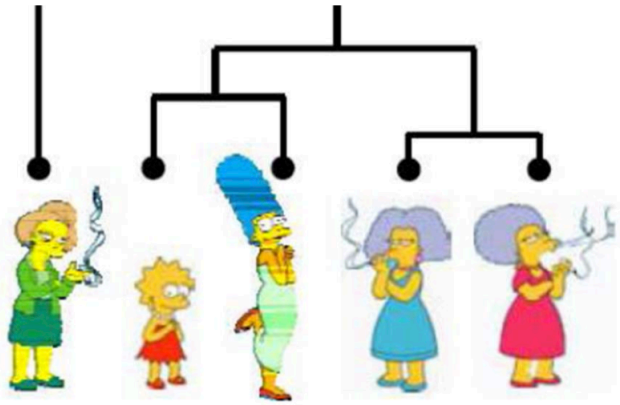
  - Clustering results are crucially dependent on the measure of similarity (or distance) between "points" to be clustered
- What properties should a distance measure have?
  - Symmetric
    - D(A,B)=D(B,A)
    - Otherwise, we can say A looks like B but B does not look like A
  - Positivity, and self-similarity
    - D(A, B) >= 0, and D(A, B)=0 iff A=B
    - Otherwise there will different objects that we can not tell apart
  - Triangle inequality
    - D(A, B) + D(B, C) >= D(A, C)
    - Otherwise one can say "A is like B, B is like C, but A is not like C at all"

[Slide from David Sontag]

Clustering algorithms

- Partition algorithms
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering (in graph, not discussed in this lecture.)
- Hierarchical algorithms
  - Bottom up - agglomerative
  - Top down – divisive (not discussed in this lecture.)





[Slide from David Sontag]

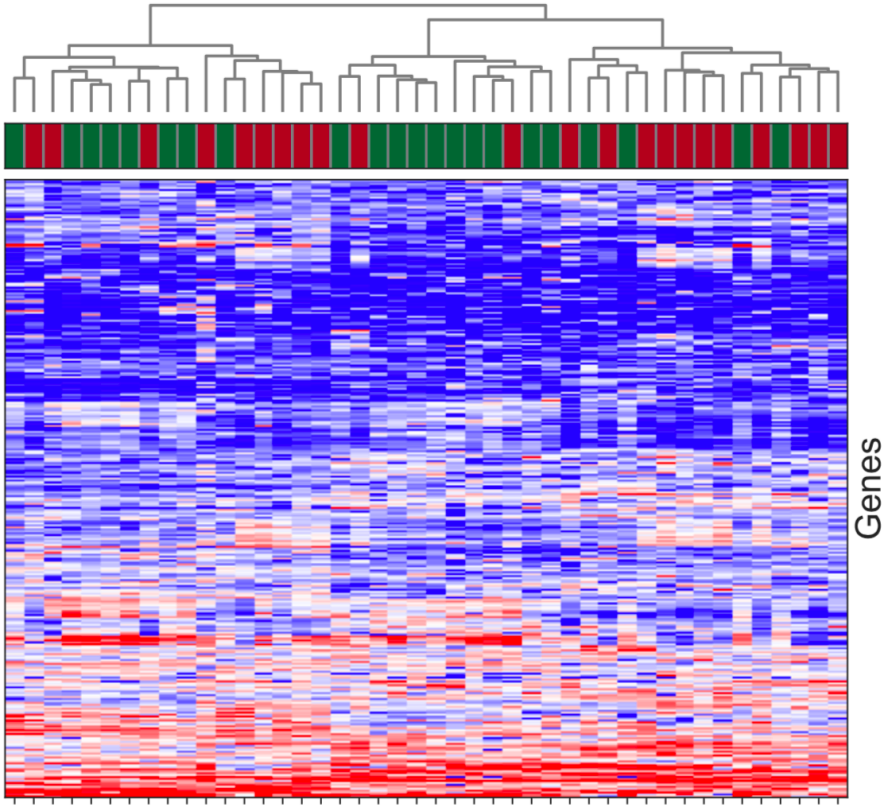
Clustering examples

- **Image segmentation**
- Goal: Break up the image into meaningful or perceptually similar regions

[Slide from David Sontag]

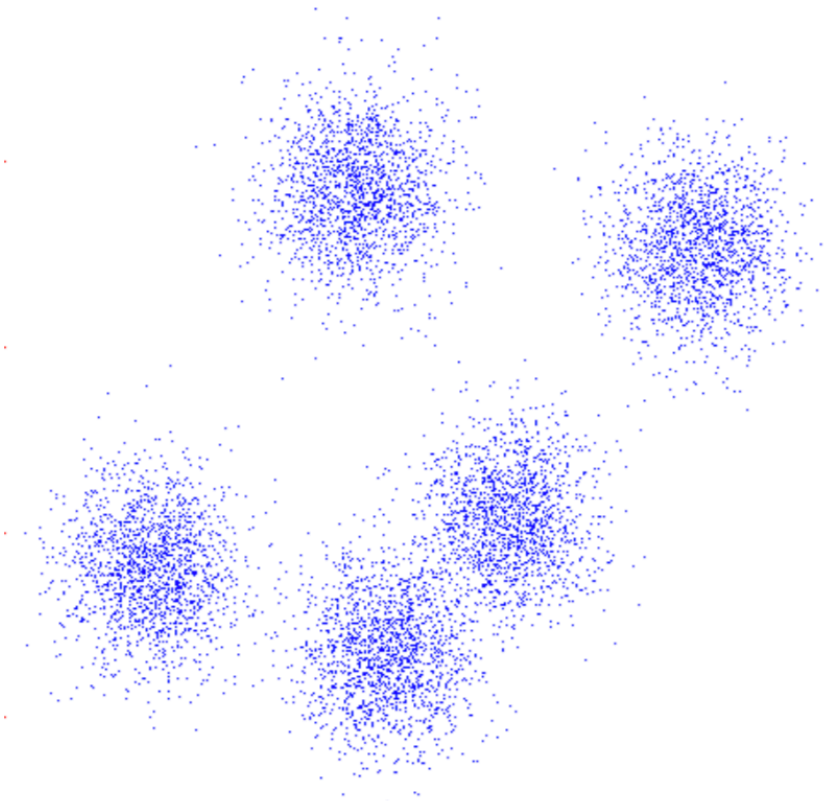
Clustering examples

- **Clustering gene expression data**



K-Means

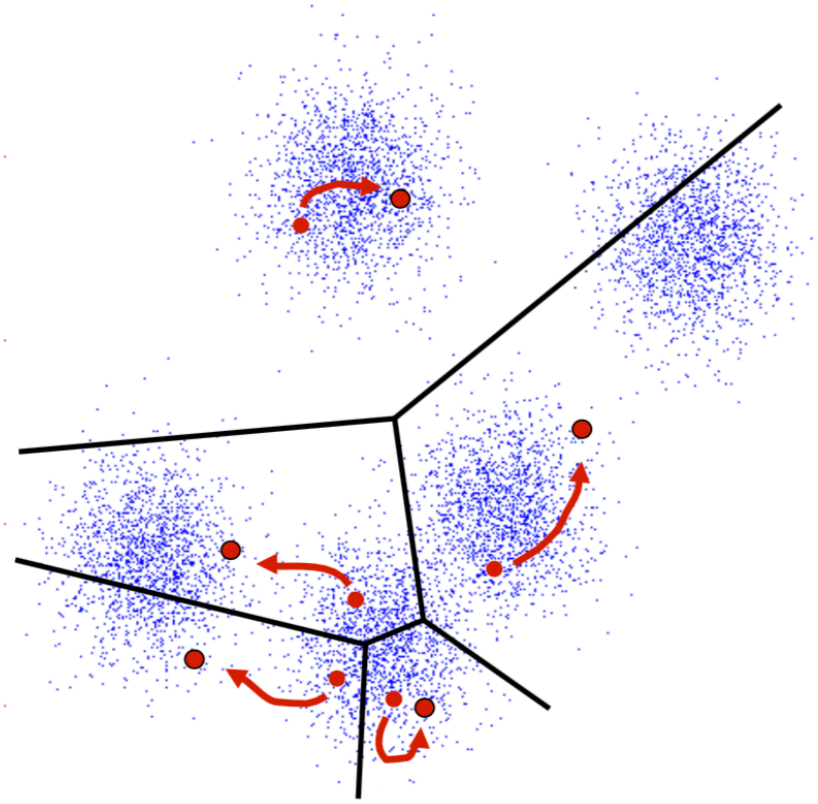
- An iterative clustering algorithm
- Initialize: Pick *K* random points as cluster centers
- Alternate:
  - Assign data points to closest cluster center
  - Change the cluster center to the average of its assigned points
- Stop when no points' assignments change



[Slide from David Sontag]

K-Means

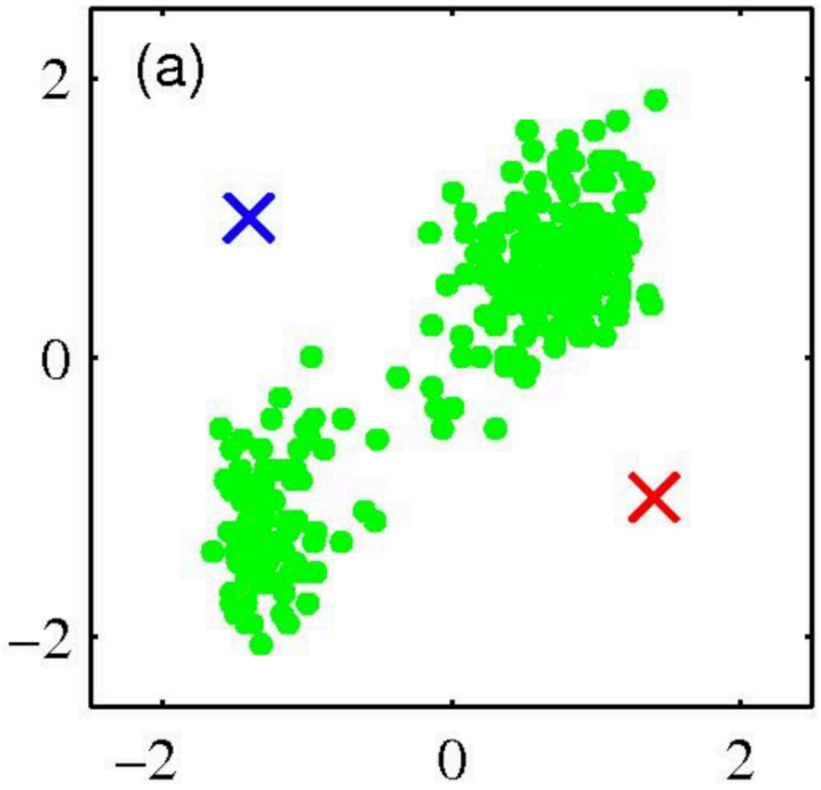
- An iterative clustering algorithm
- Initialize: Pick *K* random points as cluster centers
- Alternate:
  - Assign data points to closest cluster center
  - Change the cluster center to the average of its assigned points
- Stop when no points' assignments change



[Slide from David Sontag]

K-means clustering: Example

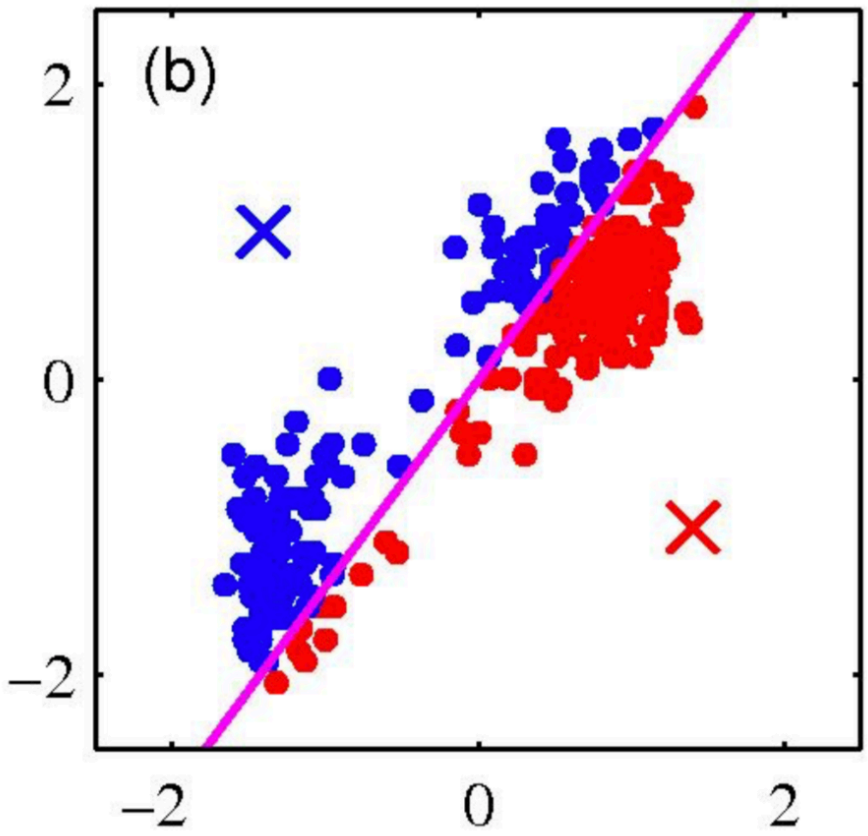
- Pick *K* random points as cluster centers (means)
- Shown here for *K*=2



[Slide from David Sontag]

K-means clustering: Example

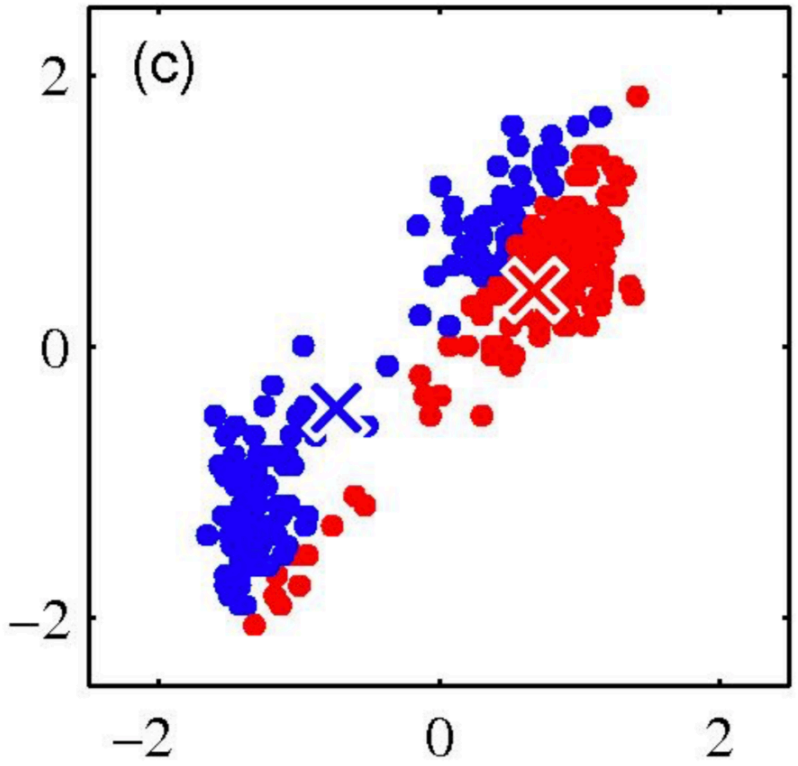
- Iterative Step 1
  - Assign data points to closest cluster center



[Slide from David Sontag]

K-means clustering: Example

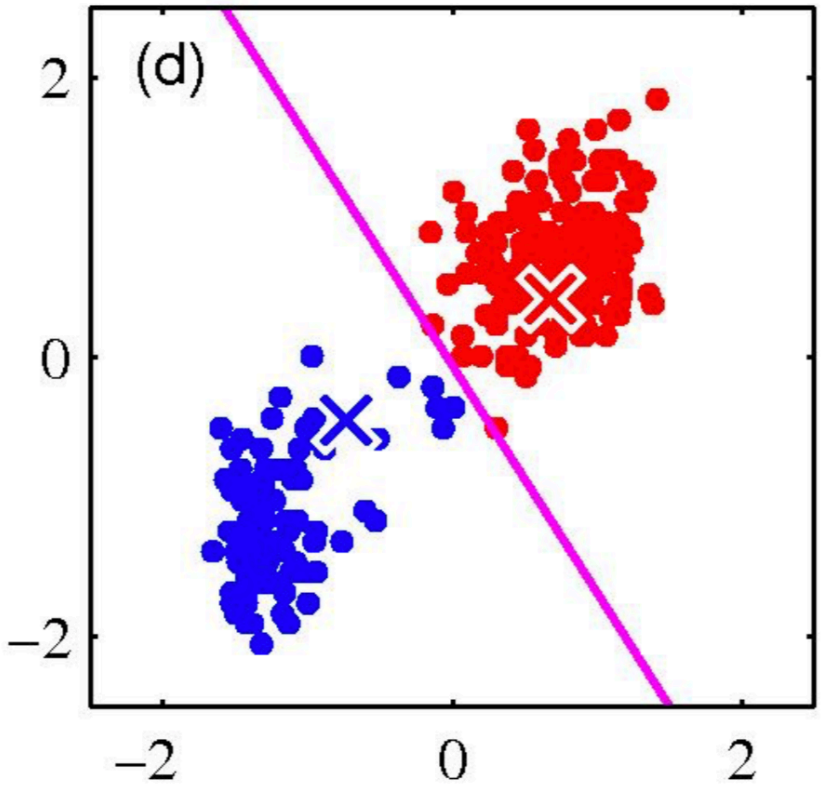
- Iterative Step 2
  - Change the cluster center to the average of the assigned points



[Slide from David Sontag]

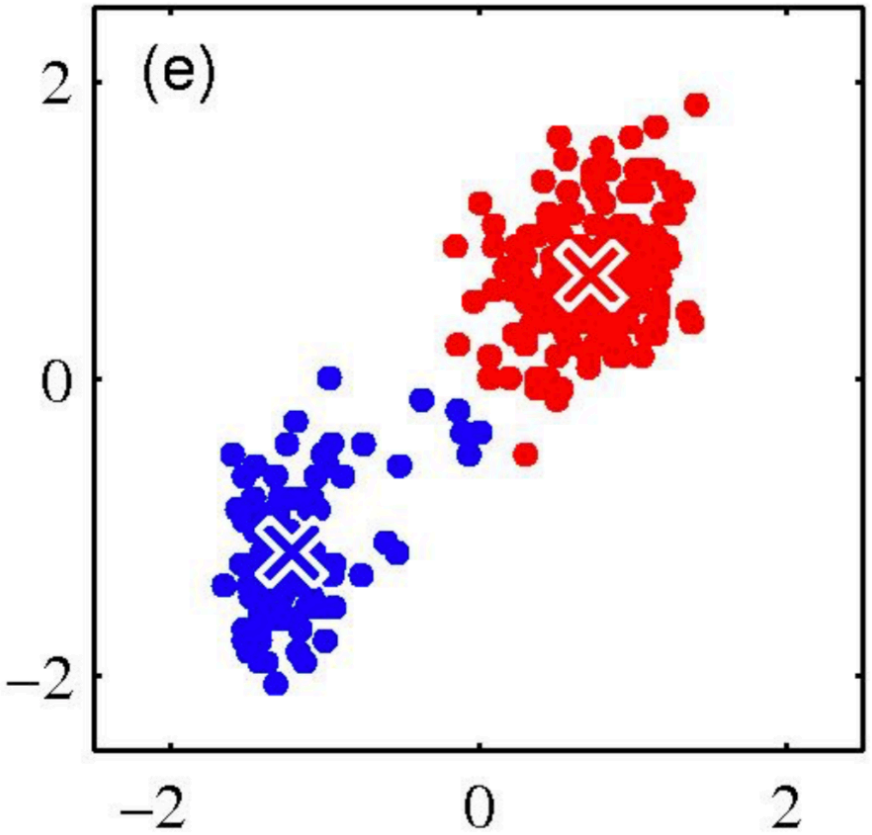
K-means clustering: Example

- Repeat until convergence

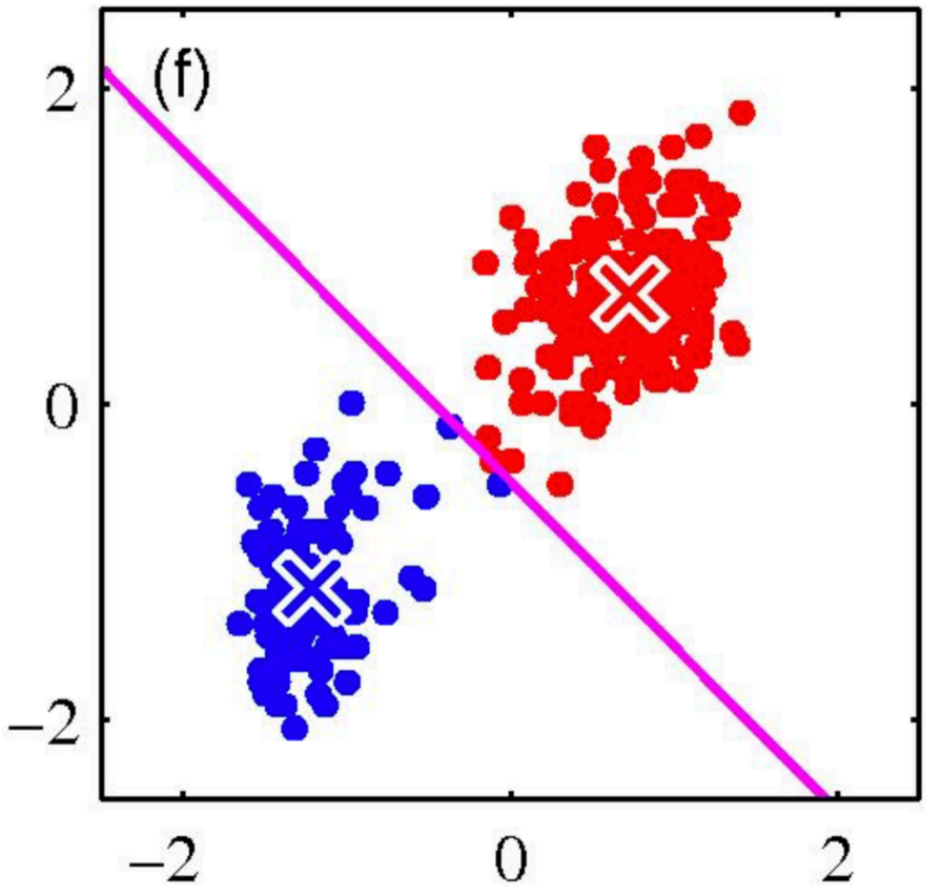


[Slide from David Sontag]

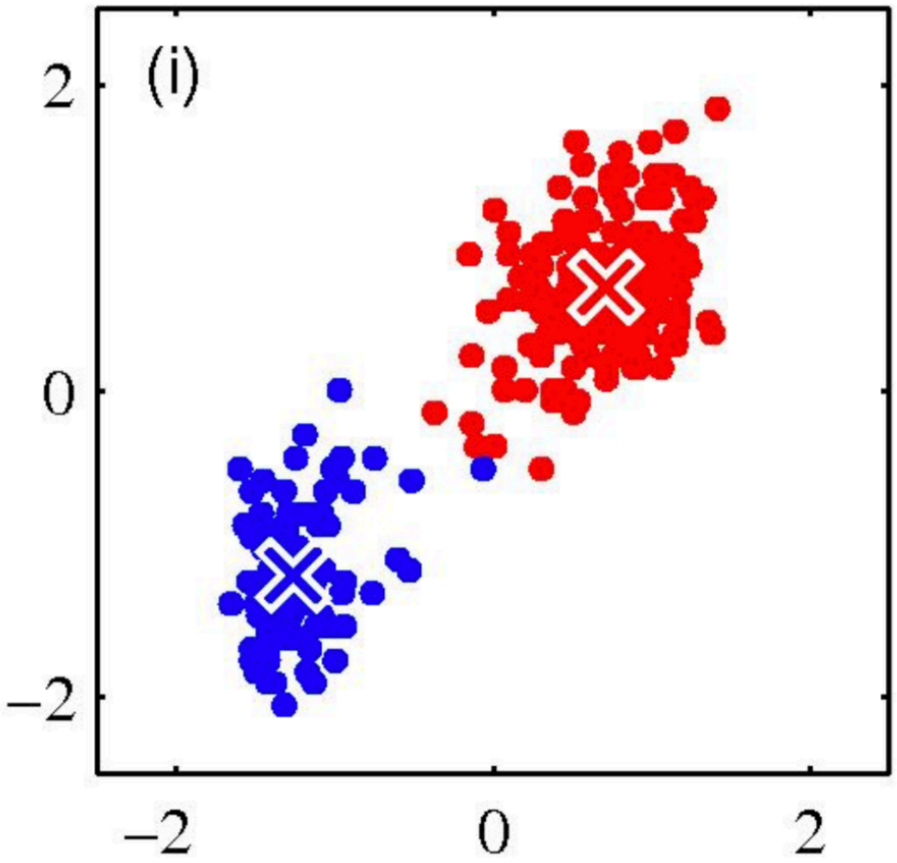
K-means clustering: Example



[Slide from David Sontag]



[Slide from David Sontag]



[Slide from David Sontag]

Properties of K-means algorithm

- Guaranteed to converge in a finite number of iterations
- Running time per iteration:
  - Assign data points to closest cluster center
  - O(KN) time

  - Change the cluster center to the average of its assigned points
  - O(N)

[Slide from David Sontag]

K-means convergence

Objective

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

1. Fix μ, optimize C:

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

Step 1 of kmeans

2. Fix C, optimize μ:

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

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

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

Step 2 of kmeans

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

[Slide from David Sontag]

Example: K-Means for Segmentation

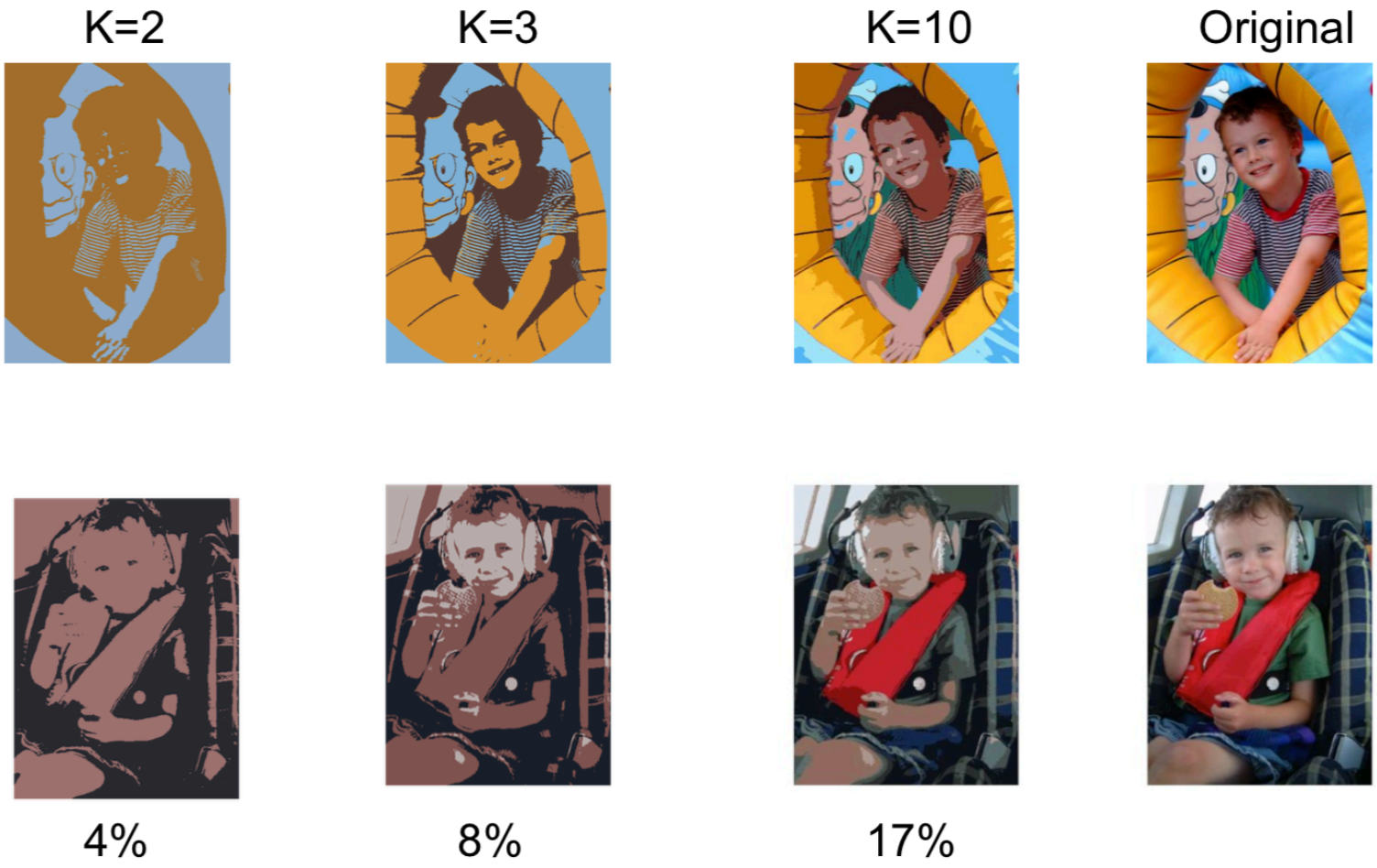
K=2

Goal of Segmentation is to partition an image into regions each of which has reasonably homogenous visual appearance.

Original

[Slide from David Sontag]

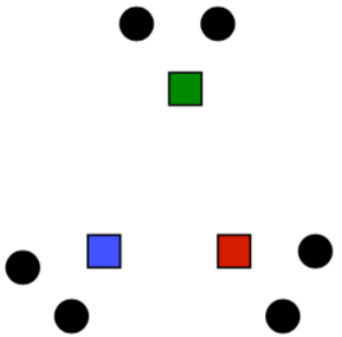
Example: K-Means for Segmentation

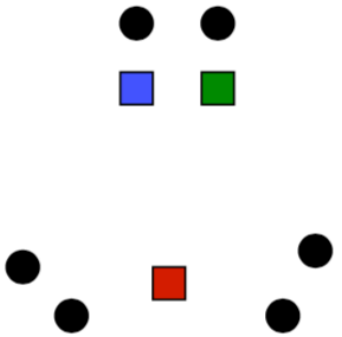


[Slide from David Sontag]

Initialization

- K-means algorithm is a heuristic
- Requires initial means
- It does matter what you pick!
- What can go wrong?
- Various schemes for preventing this kind of thin variance-based split / merge, initialization heuristics
  - E.g., multiple initialization, k-means++

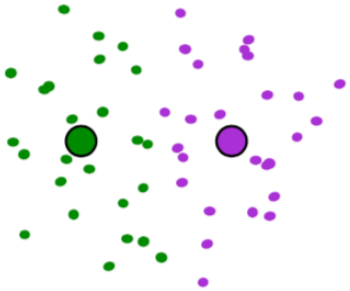




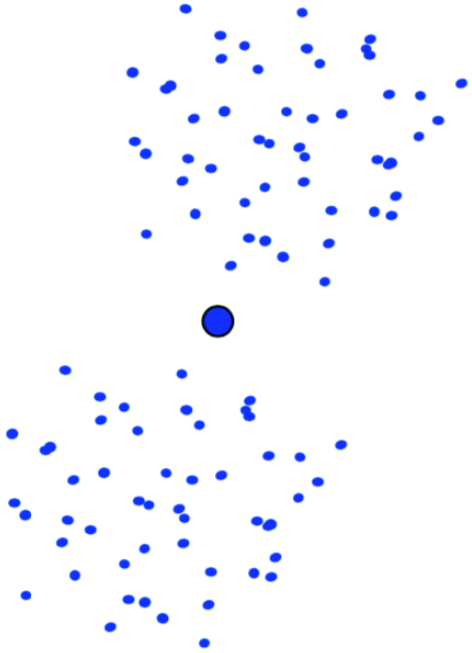
[Slide from David Sontag]

K-Means Getting Stuck

- A local optimum:



Would be better to have one cluster here

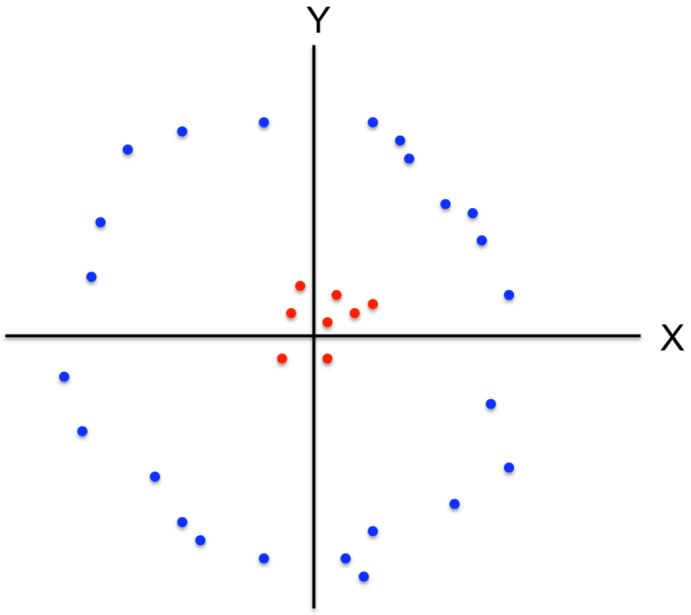


… and two clusters here

[Slide from David Sontag]

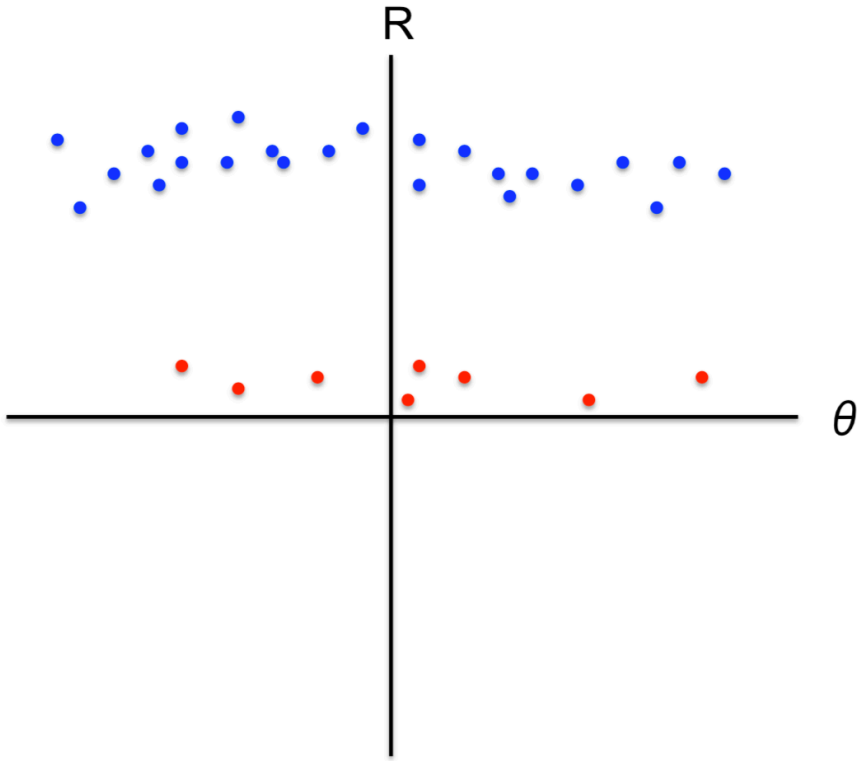
K-means not able to properly cluster

- Spectral clustering will help in this case.



[Slide from David Sontag]

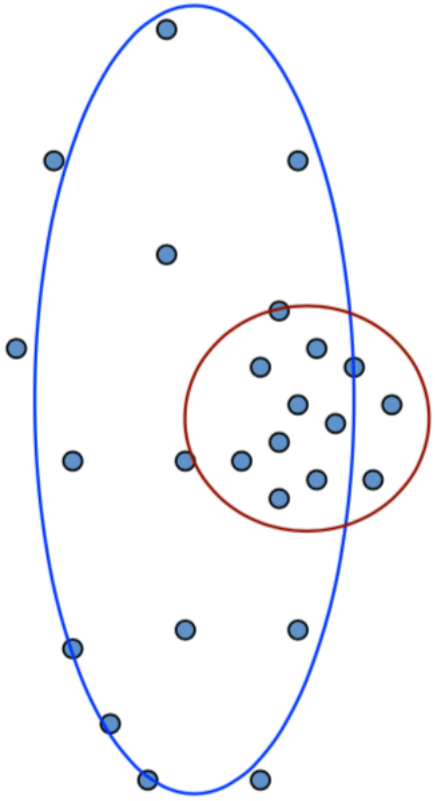
Changing the features (distance function) can help



[Slide from David Sontag]

Reconsidering "**hard assignments**"?

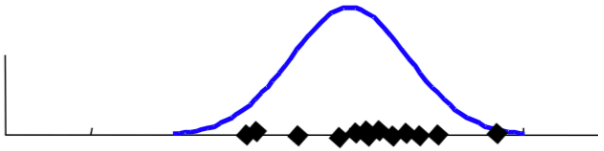
- Clusters may overlap
- Some clusters may be "wider" than others
- Distances can be deceiving



[Slide from Ziv Bar-Joseph]

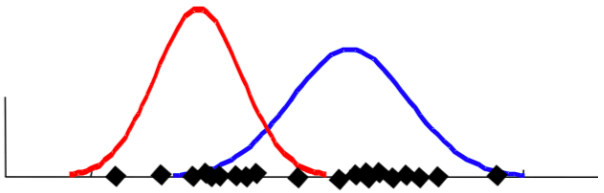
Gaussian Mixture Models

- Gaussian $P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\eta)^2}{2\sigma^2}}$
  - ex. height of one population



- Gaussian Mixture: Generative modeling framework

$$P(C=i) = w_i \quad , \quad P(x \mid C=i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$$



$$P(x \mid \Theta) = \sum_i P(C=i, x \mid \Theta) = \sum_i P(x \mid C=i, \Theta) P(C=i \mid \Theta) =$$

$$\sum_i w_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$$

Likelihood of a data point given the model

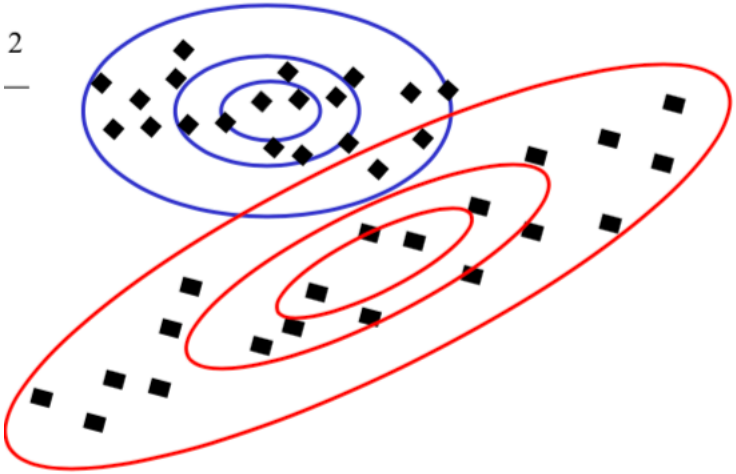
[Slide from Ziv Bar-Joseph]

Gaussian Mixture Models

- Mixture of Multivariate Gaussian

$$P(C=i)=w_i \qquad P(x\mid C=i)=\frac{1}{\sqrt{2\pi\sigma_i^2}}e^{-\frac{(x-\eta_i)^2}{2\sigma_i^2}}$$

- ex. y-axis is blood pressure and x-axis is age



[Slide from Ziv Bar-Joseph]

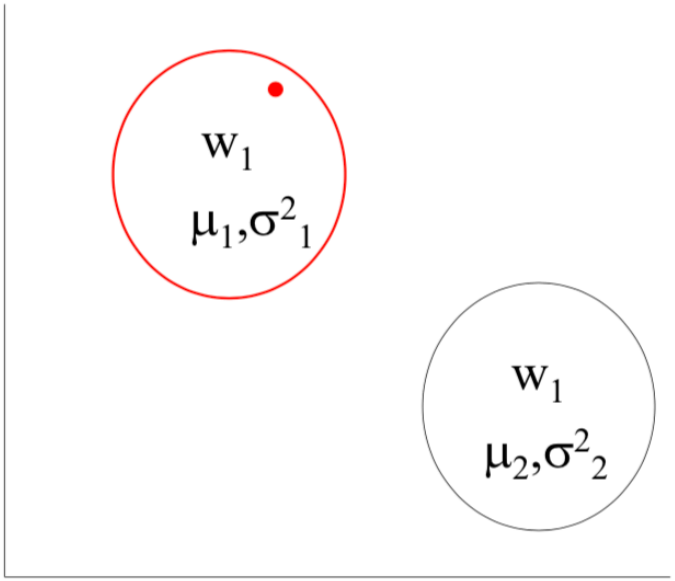
GMM: A generative model

$$\sum_i w_i = 1$$

Assuming we know the number of components (k), their weights (w_i) and parameters (μ_i, $\sum_i$) we can generate new instances from a GMM in the following way:

- Pick one component at random with probability w_i for each component

- Sample a point x from $N(\mu_i, \sum_i)$



[Slide from Ziv Bar-Joseph]

Estimating model parameters

- We have a weight, mean and covariance parameters for each class
- As usual we can write the likelihood function for our model

$$p(x_1 \cdots x_n \mid \theta) = \prod_{j=1}^{n} \left(\sum_{i=1}^{k} p(x_j \mid C = i) w_i \right)$$

[Slide from Ziv Bar-Joseph]

EM algorithm

- Auxiliary function:

$$
\begin{aligned}
l(\theta; x) &= \log p(x \mid \theta) \\
&= \log \sum_z p(x, z \mid \theta) \\
&= \log \sum_z q(z \mid x) \frac{p(x, z \mid \theta)}{q(z \mid x)} \\
&\geq \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)} \\
&\triangleq \mathcal{L}(q, \theta),
\end{aligned}
$$

- Instead of directly optimizing the log-likelihood, EM algorithm maximizes the auxiliary function.

[Slide from Michael I. Jordan, An Introduction to Probabilistic Graphical Models. Chapter 11]

EM algorithm

- Expectation-maximization (EM) algorithm is an iterative method to find maximum likelihood or maximum a posteriori (MAP) estimates of parameters in models, where the model depends on unobserved latent variables
- Alternates between performing:
  - Expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters
  - Maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the E step

$$(\mathbf{E\ step}) \qquad q^{(t+1)} = \arg\max_{q} \mathcal{L}(q, \theta^{(t)})$$

$$(\mathbf{M\ step}) \qquad \theta^{(t+1)} = \arg\max_{\theta} \mathcal{L}(q^{(t+1)}, \theta).$$

[Slide from Michael I. Jordan, An Introduction to Probabilistic Graphical Models. Chapter 11]

GMM+EM = "Soft K-means"

- Decide the number of clusters, *K*
- Initialize parameters (randomly)
- E-step: assign probabilistic membership to all input samples *j*

One for each cluster

$$p_{i,j} = p(C = i \mid x_j) = \frac{p(x_j \mid C = i)p(C = i)}{\sum_k p(x_j \mid C = k)p(C = k)}$$

$$p_i = \sum_j p_{i,j}$$

- M-step: re-estimate parameters based on probabilistic membership

$$\mu_i \leftarrow \sum_j \frac{p_{i,j}\mathrm{x}_j}{p_i}$$

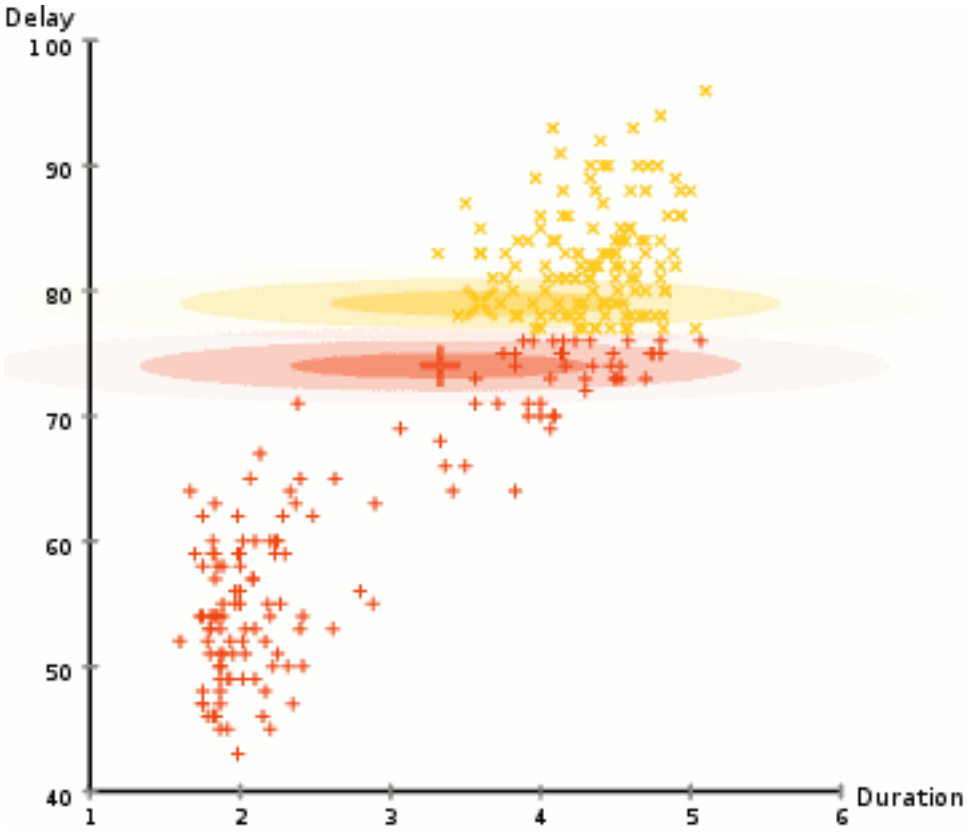
$$\Sigma_i \leftarrow \sum_j \frac{p_{i,j}\mathrm{x}_j\mathrm{x}_j^{\mathrm{T}}}{p_i}$$

$$w_i = \frac{p_i}{\sum_j p_j}$$

- Repeat until change in parameters are smaller than a threshold

[Slide from Ziv Bar-Joseph]

Example of EM+GMM



[Figure from https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm]

Strength of Gaussian Mixture Models

- Interpretability: learns a generative model of each cluster
  - you can generate new data based on the learned model
- Relatively efficient: *O(tkn)*, where n is # objects, k is # clusters, and t is # iterations. Normally, $k, t << n$.
- Intuitive objective function: optimizes data likelihood

[Slide from Ziv Bar-Joseph]

Weakness of Gaussian Mixture Models

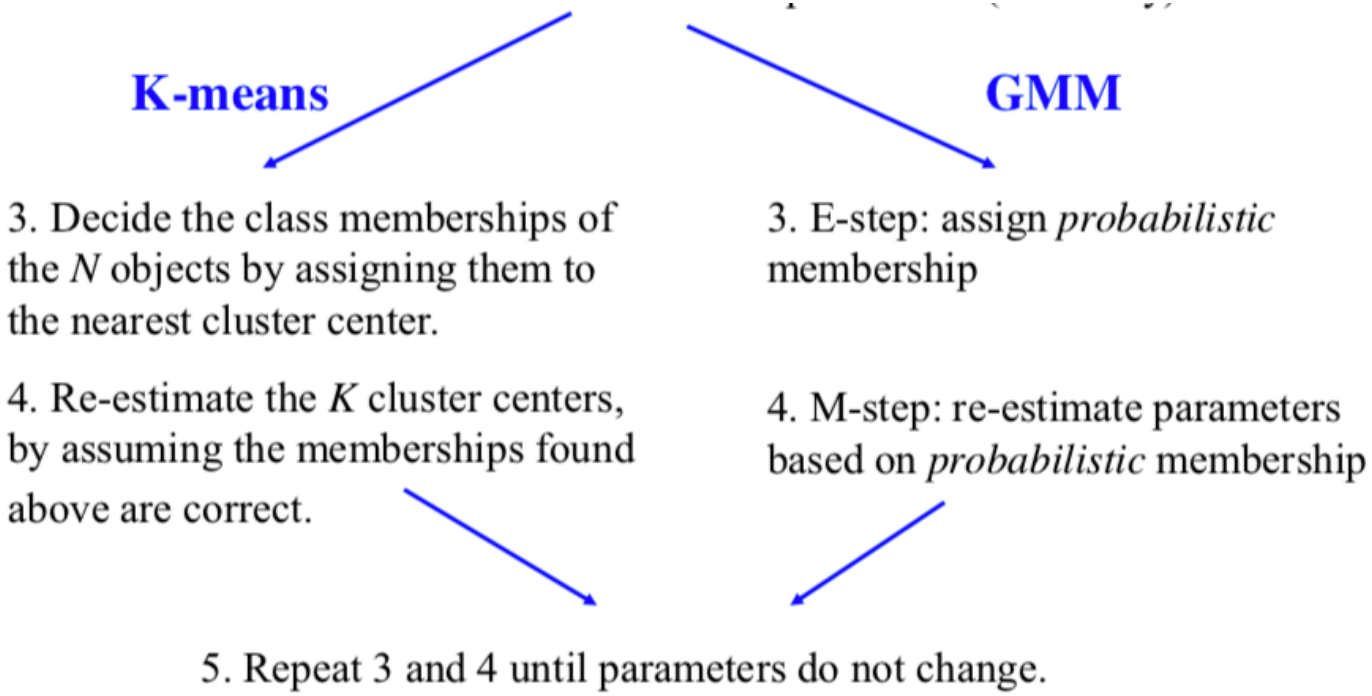
- Often terminates at a local optimum. Initialization is important.
- Need to specify *K*, the number of clusters, in advance
- Not suitable to discover clusters with non-convex shapes

- Summary
  - To learn Gaussian mixture, assign probabilistic membership based on current parameters, and re-estimate parameters based on current membership

[Slide from Ziv Bar-Joseph]

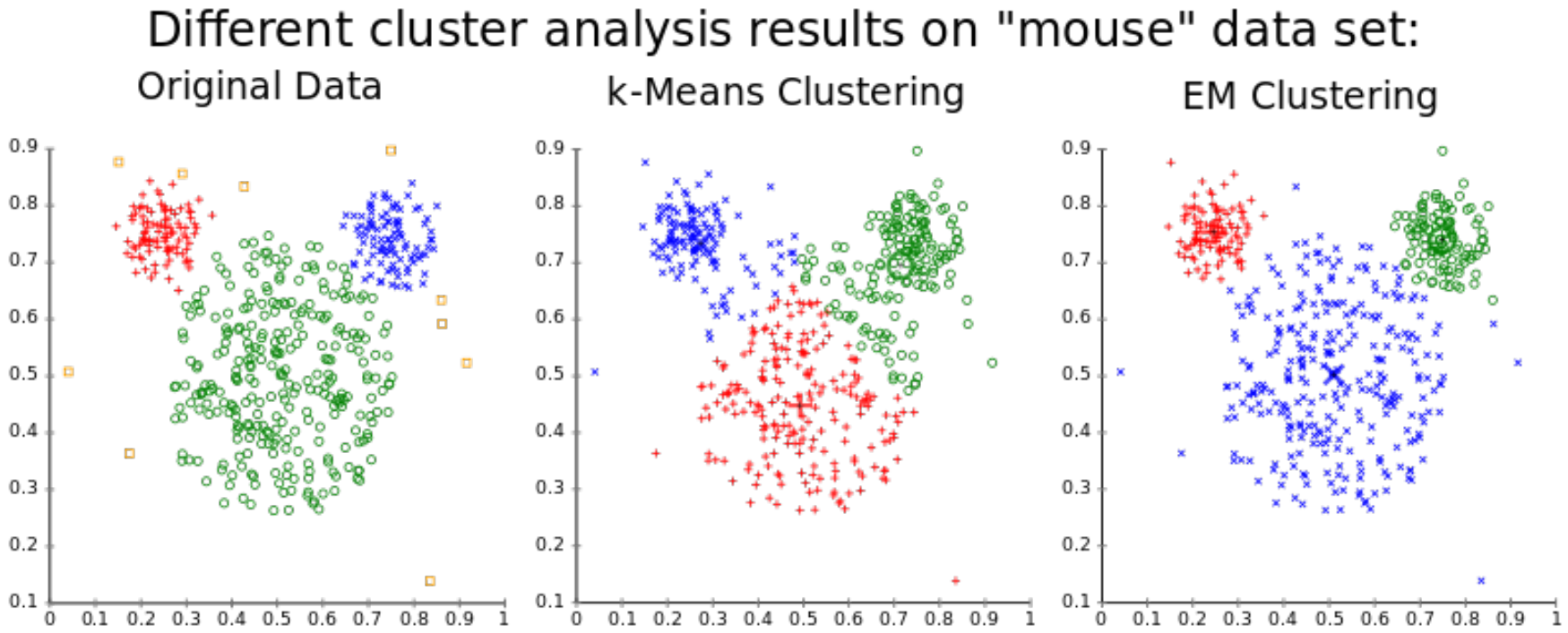
Algorithm: K-means and GMM

- Decide on a value for *K*, the number of clusters.
- Initialize the *K* cluster centers / parameters (randomly).



[Slide from Ziv Bar-Joseph]

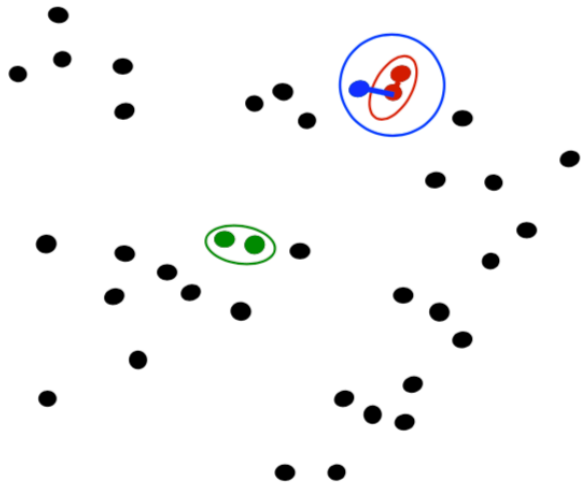
K-means vs GMM

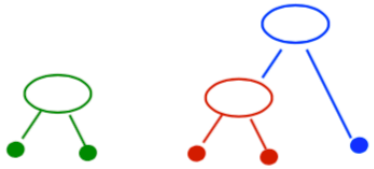


[Slide from David Sontag]

Agglomerative Clustering

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

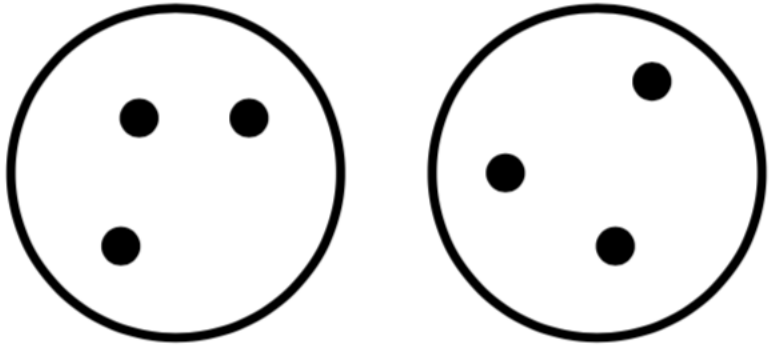




[Slide from David Sontag]

Agglomerative Clustering

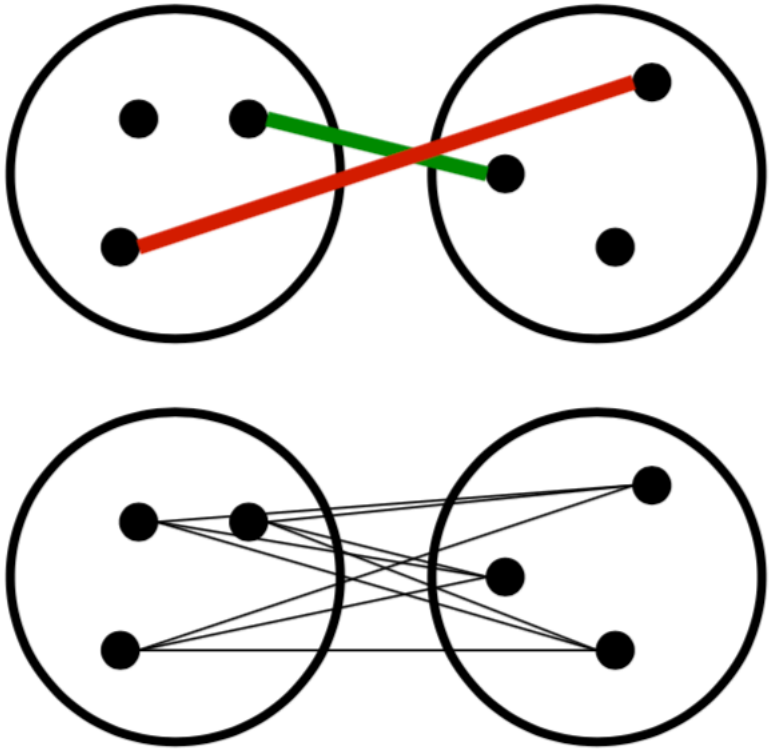
- How should we define “closest” for clusters with multiple elements?



[Slide from David Sontag]

Agglomerative Clustering

- How should we define "closest" for clusters with multiple elements?
- Many options:
  - Closest pair (single-link clustering)
  - Farthest pair (complete-link clustering)
  - Average of all pairs
- Different choices create different clustering behaviors

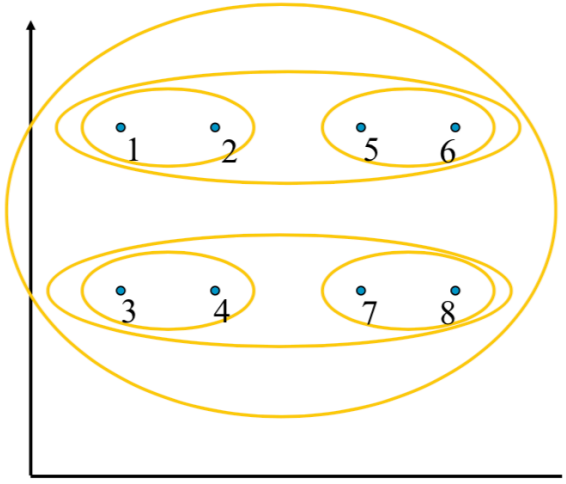


[Slide from David Sontag]

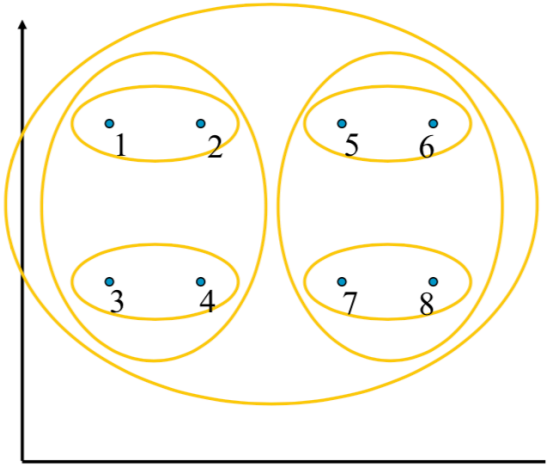
Agglomerative Clustering

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

Closest pair
(single-link clustering)

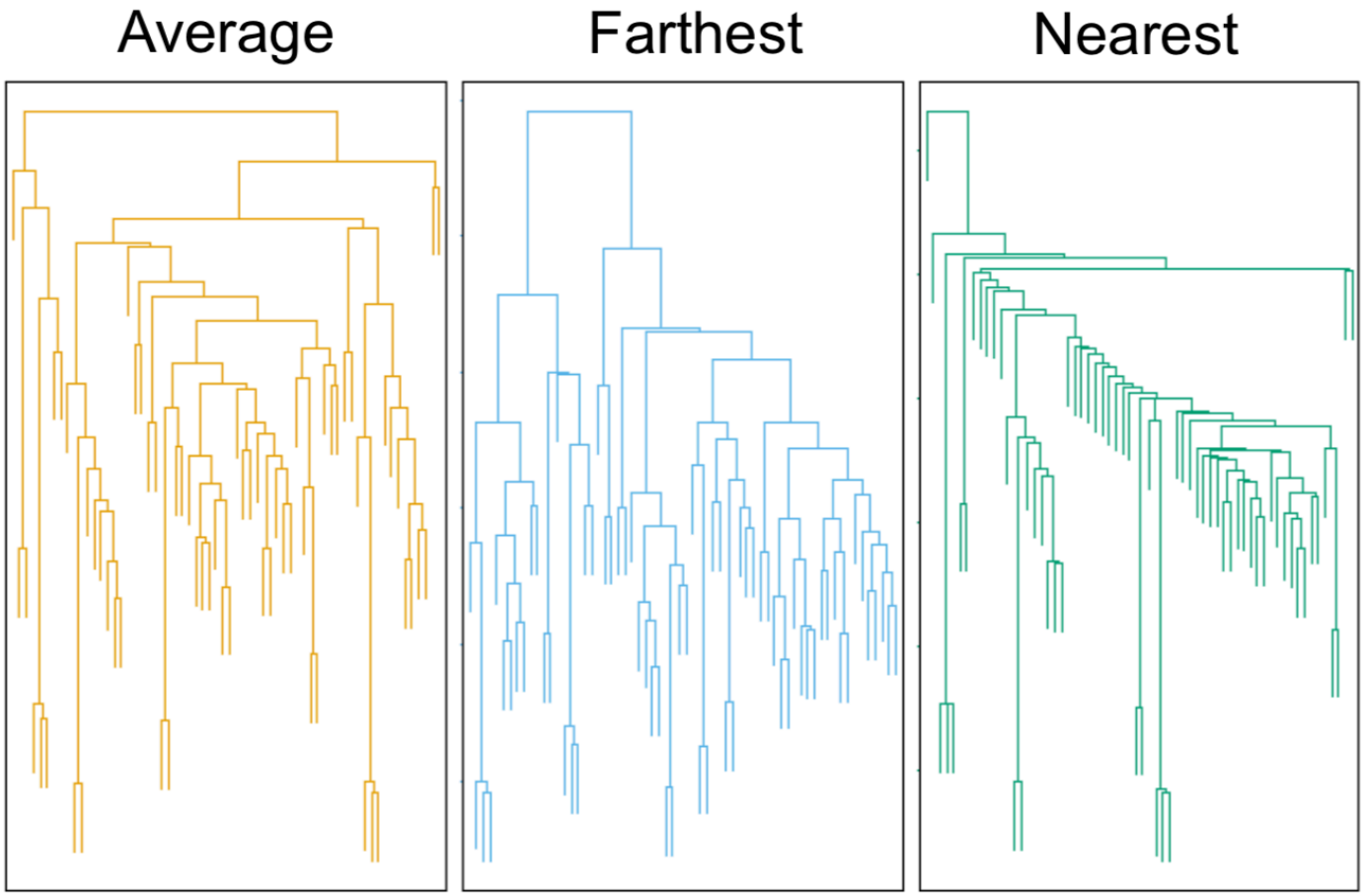


Farthest pair
(complete-link clustering)



[Slide from David Sontag]

Clustering Behavior



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

[Slide from David Sontag]

Agglomerative Clustering Questions

- Will agglomerative clustering converge?
  - To a global optimum?
- Will it always find the true patterns in the data?
- Do people ever use it?
- How many clusters to pick?

[Slide from David Sontag]

Programming

- Most models mentioned in this lectures have been implemented in Python package scikit-learn.

Take home message

- PCA, ICA and t-SNE
  - PCA aims at finding bases that best explain the variance of data
  - ICA aims at finding independent signals that explain the data
  - t-SNE is a useful tool for data visualization
- K-means and (GMM+EM)
  - K-means tries to minimize the distances of points to the cluster center
  - K-means is guaranteed to converge
  - GMM uses EM to maximize the log-likelihood
  - K-means is a hard-version of GMM
- Hierarchical clustering results can be very different depending on the similarity metrics

References

- Eric Xing, Tom Mitchell. 10701 Introduction to Machine Learning: http://www.cs.cmu.edu/~epxing/Class/10701-06f/
- Eric Xing, Ziv Bar-Joseph. 10701 Introduction to Machine Learning: http://www.cs.cmu.edu/~epxing/Class/10701/
- David Sontag. Introduction To Machine Learning. https://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture14.pdf