DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

November 15 2018

Review

- Neural Network Architectures
  - Feed-Forward Neural Networks
  - Convolutional Neural Networks (CNNs)
  - Recurrent Neural Networks (RNNs)
- Training with backpropagation
  - Mini-batch Gradient Descent
- Unsupervised learning
  - No labels available in training data
  - Discover hidden patterns in data
  - Not standard metrics to evaluate

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)
x is data, y is label

Goal: Learn a *function* to map x -> y

Examples: Classification, regression, object detection, semantic segmentation, image captioning, etc.

Standard metrics for evaluation

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Difficult to evaluate

Unsupervised Learning

- Different learning tasks
- Dimensionality reduction
  - Project the data to lower dimensional space
  - Example: PCA (Principal Component Analysis)
- Feature learning
  - Find feature representations
  - Example: Autoencoders
- Clustering
  - Group similar data points into clusters
  - Example: k-means, hierarchical clustering

PCA Algorithm

- Given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, compute covariance matrix $\mathbf{\Sigma}$
  - X is the n x d data matrix
  - Compute data mean (average over all rows of X)
  - Subtract mean from each row of X (centering the data)
  - Compute covariance matrix $\mathbf{\Sigma} = \mathrm{X}^{\mathsf{T}}\mathrm{X}$ ($\mathbf{\Sigma}$ is d x d)
- **PCA** basis vectors are given by the eigenvectors of $\mathbf{\Sigma}$
  - $\mathrm{Q}, \Lambda =$ numpy.linalg.eig(Σ)
  - $\{\mathbf{q}_i, \lambda_i\}_{i=1..n}$ are the eigenvectors/eigenvalues of $\mathbf{\Sigma}$
    ... $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$
- Larger eigenvalue $\Rightarrow$ more important eigenvectors

PCA

- We can apply these formulas to get the new representation for each instance $\mathbf{x}$

$$X = \begin{array}{cl} \left[\begin{array}{c} 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1 \ldots \\ 1\ 1\ 0\ 1\ 1\ 1\ 0\ 0 \ldots \\ 0\ 0\ 1\ 1\ 1\ 0\ 0\ 0 \ldots \\ \vdots \\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0 \ldots \end{array}\right] & \begin{array}{l} \\ \\ \mathbf{x}_3 \\ \\ \\ \end{array} \end{array} \qquad \hat{Q} = \begin{bmatrix} 0.34 & 0.23 \\ 0.04 & 0.13 \\ -0.64 & 0.93 \\ \vdots & \vdots \\ -0.20 & -0.83 \end{bmatrix}$$

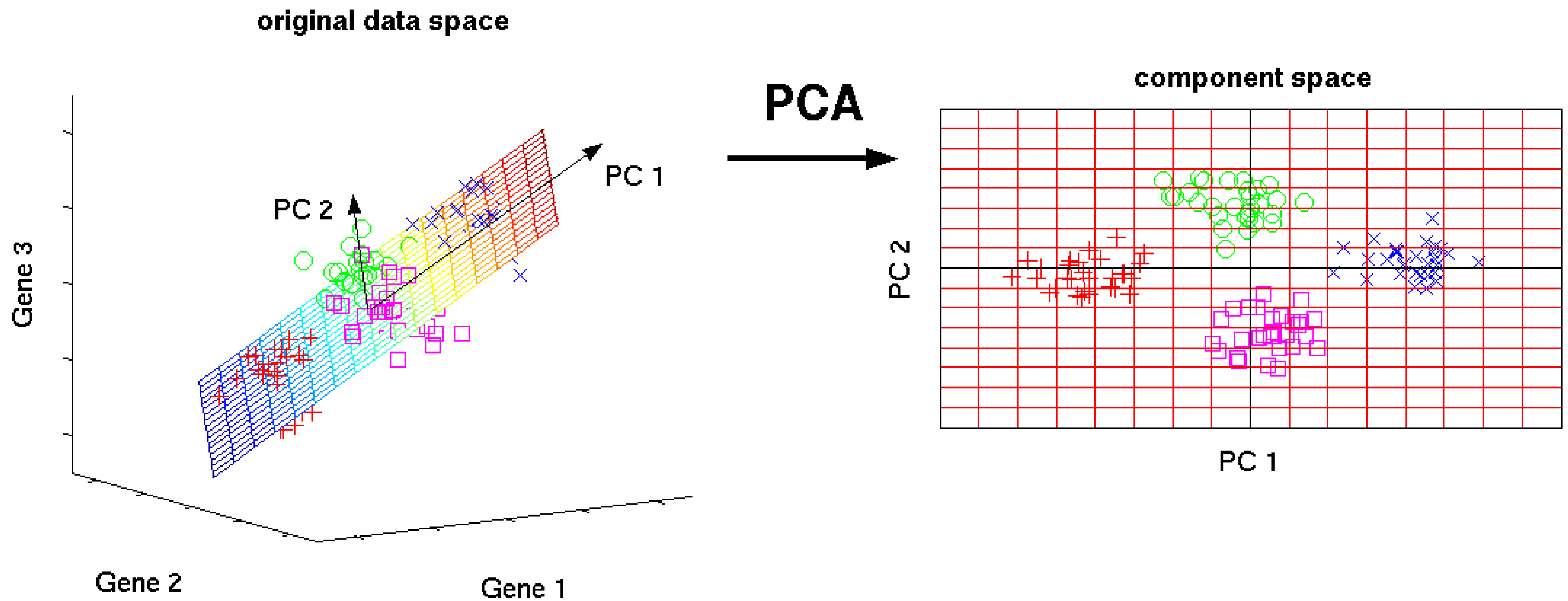
- The new 2D representation for $\mathbf{x}_3$ is given by:

$$\hat{x}_{31} = 0.34(0) + 0.04(0) - 0.64(1) + \ldots$$

$$\hat{x}_{32} = 0.23(0) + 0.13(0) + 0.93(1) + \ldots$$

- The re-projected data matrix is given by $\hat{\mathrm{X}} = \mathrm{X}\hat{\mathrm{Q}}$

Visualizing data



PCA for image compression




d=1 **d=2** **d=4** **d=8**

d=16 **d=32** **d=64** **d=100** **Original Image**

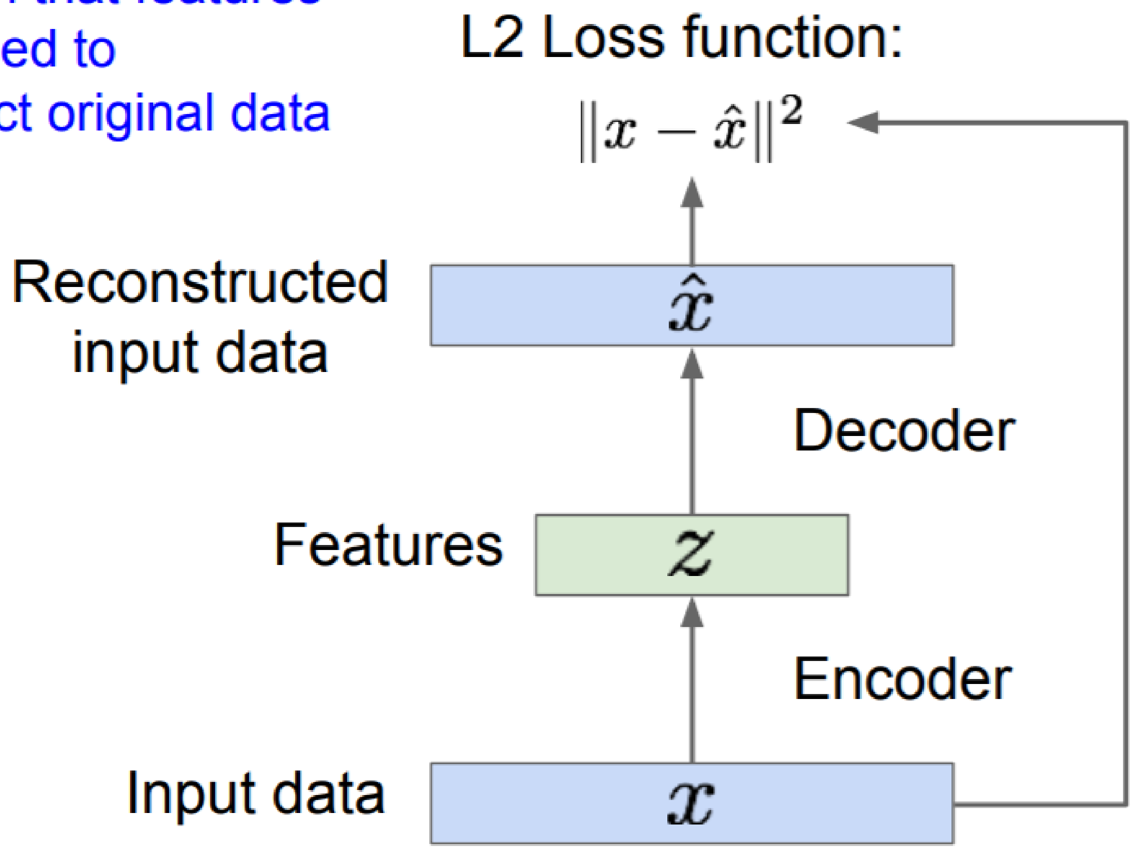




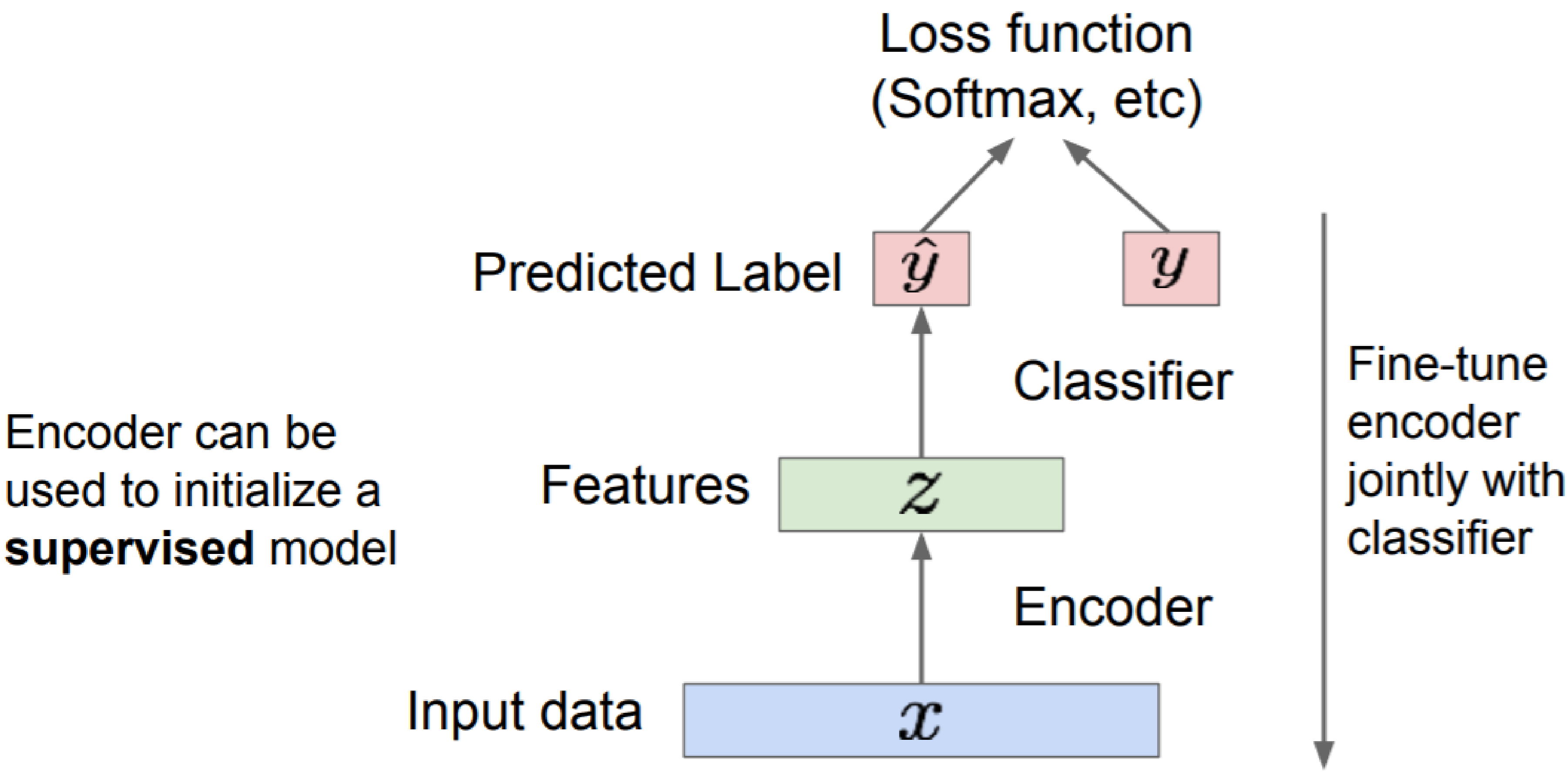
Training Autoencoders

Doesn't use labels!

Train such that features can be used to reconstruct original data



Using Features for Classification



Clustering

- Goal: Automatically segment data into groups of similar points
- Question: When and why would we want to do this?
- Useful for:
  - Automatically organizing data
  - Understanding hidden structure in data and data distribution
  - Detect similar points in data and generate representative samples

Clustering Examples

- Social networks
  - Facebook user group according to their interests and profiles
- Image search
  - Retrieve similar images to input image
- NLP
  - Topic discovery in articles
- Medicine
  - Patients with similar disease and symptoms
- Cyber security
  - Machine with same malware infection
  - New attack has no label

Setup

Our data are

$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}.$$

Each data point is d dimensional, i.e.,

$$\mathbf{x}_n = \; < x_{n,1}, \ldots, x_{n,d} >$$

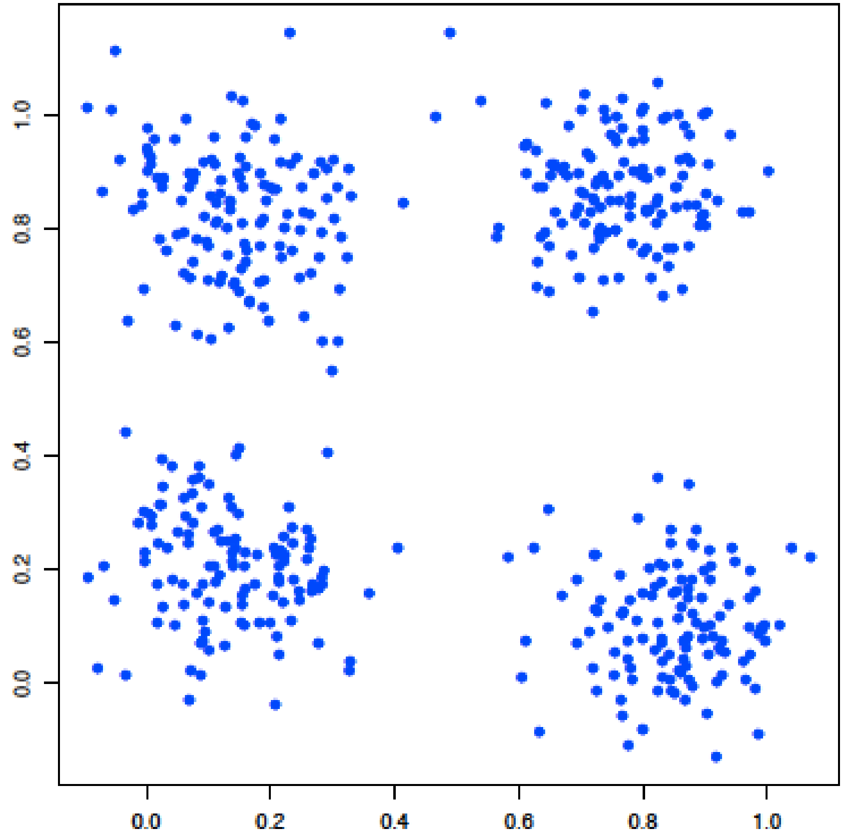
Define a *distance function* between data, $d(\mathbf{x}_n, \mathbf{x}_m)$.

Goal: segment the data into k groups

$$\{z_1, \ldots, z_N\} \quad \text{where} \quad z_i \in \{1, \ldots, K\}.$$

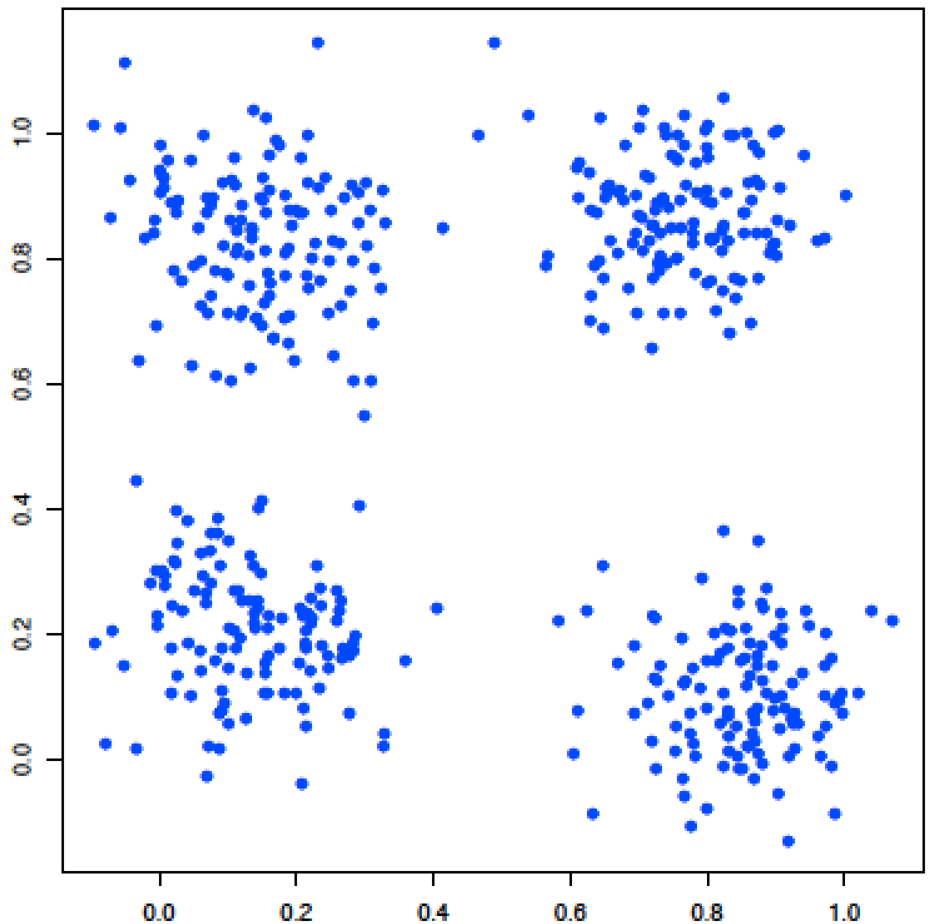
Assignment from each point to cluster index

Example



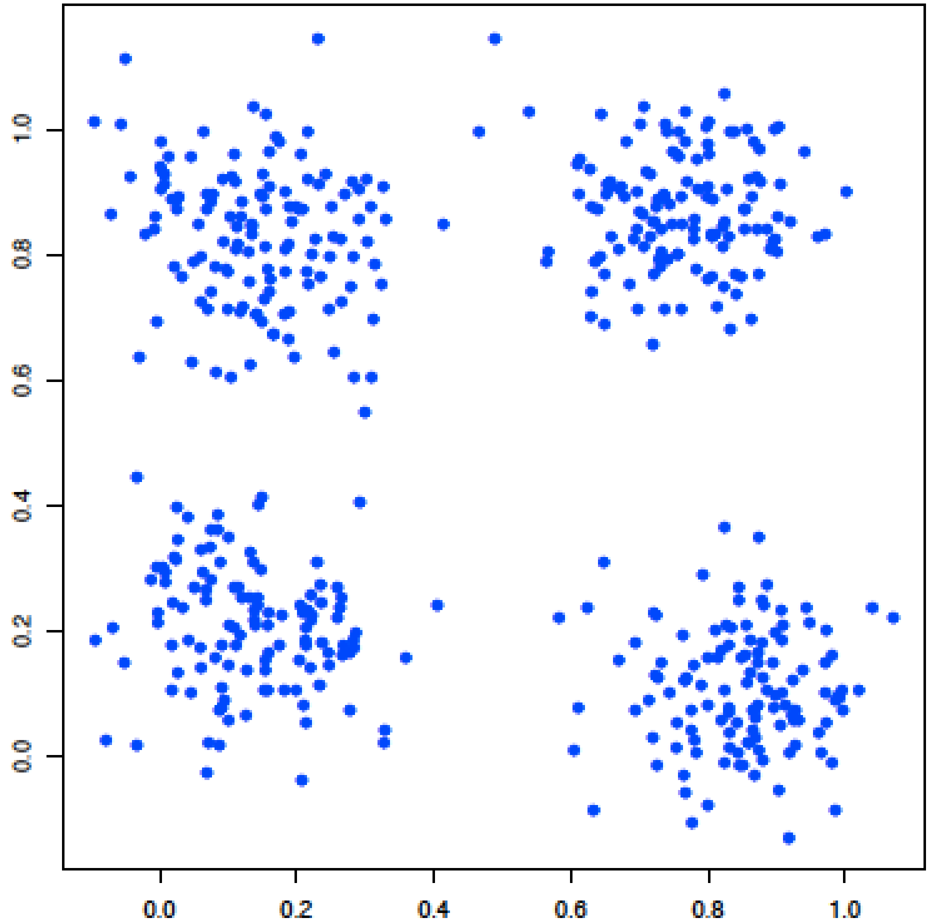
500 2-dimensional data points: $\mathbf{x}_n = \langle x_{n,1}, x_{n,2} \rangle$

Partition this data into k groups



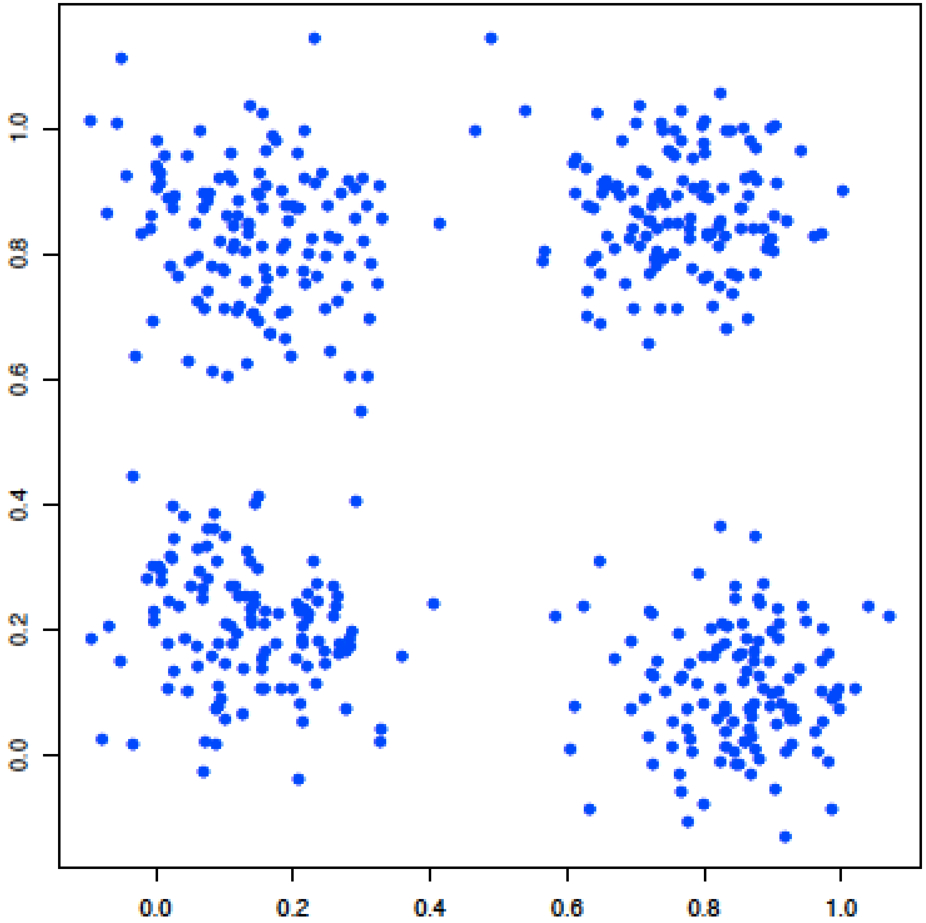
What is a good distance function?

Distance



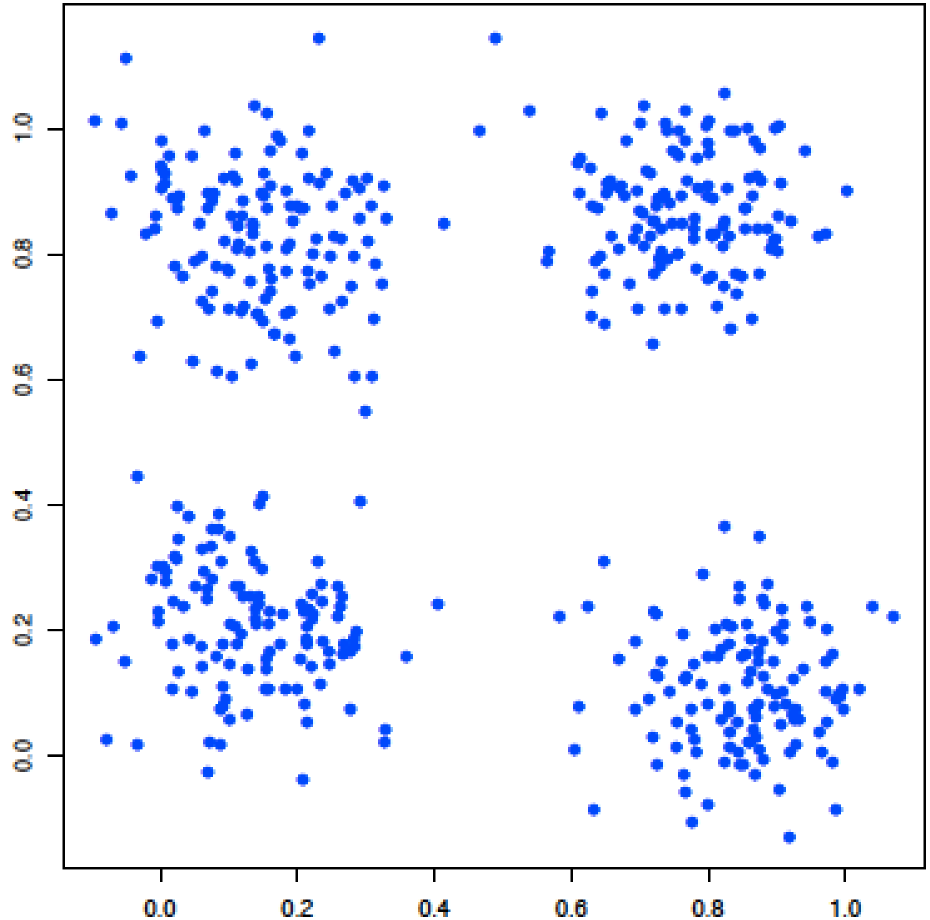
Euclidean distance: $d(x,y) = \sqrt{\sum_{j=1}^{d} (x_j - y_j)^2}$

Choice of k



What should k be?

Choice of k



For example, k is 4

K means Algorithm

- Fix a number of desired clusters k
- Key insight: describe each cluster by its mean value (called cluster representative)

- Algorithm
  - Select k cluster means at random
  - Assign points to “closest cluster”
  - Re-compute cluster means based on new assignment
  - Refine assignment iteratively until convergence

K means Algorithm

1. Initialization
   - Data are $\mathbf{x}_{1:N}$
   - Choose initial cluster means $\mathbf{m}_{1:k}$ (same dimension as data).
2. Repeat
   1. Assign each data point to its closest mean

$$z_n = \arg \min_{i \in \{1,\ldots,k\}} d(\mathbf{x}_n, \mathbf{m}_i)$$

   2. Compute each cluster mean to be the coordinate-wise average over data points assigned to that cluster,

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{\{n : z_n = k\}} \mathbf{x}_n$$

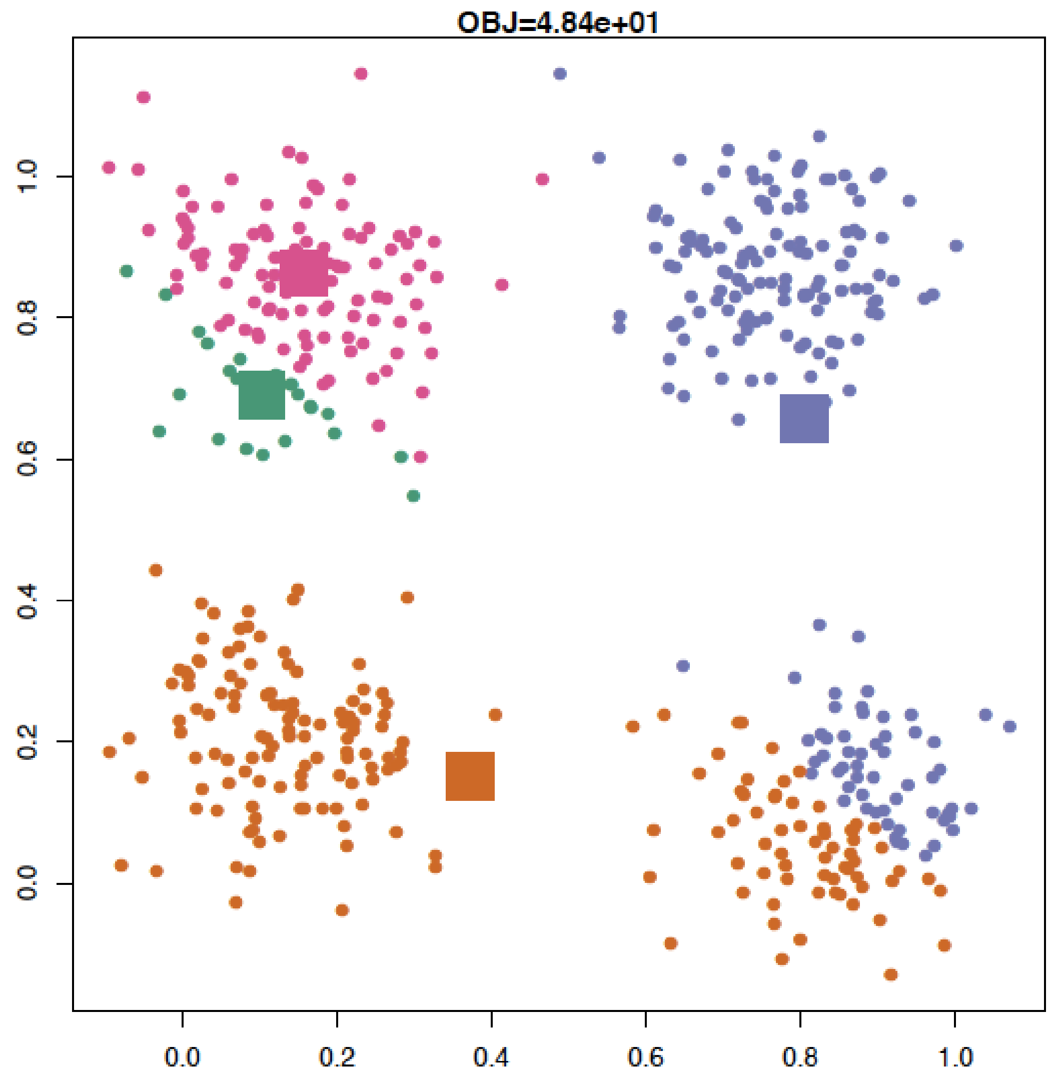
3. Until assignments $\mathbf{z}_{1:N}$ do not change

Objective Function

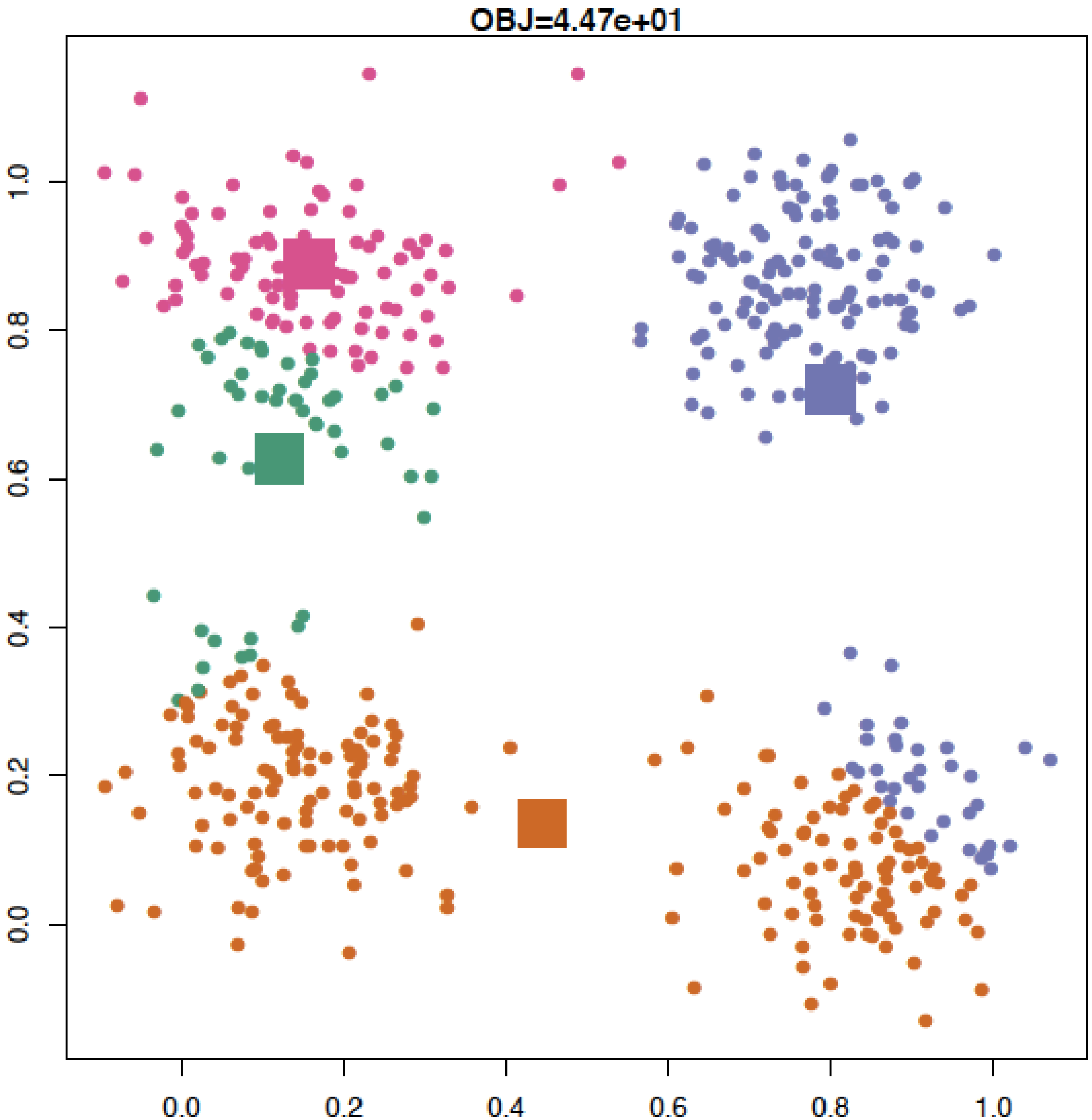
- How can we measure how well our algorithm is doing?
- The k-means objective function is the sum of the squared distances of each point to each assigned mean

$$F(z_{1:N}, \mathbf{m}_{1:k}) = \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{m}_{z_n}||^2$$

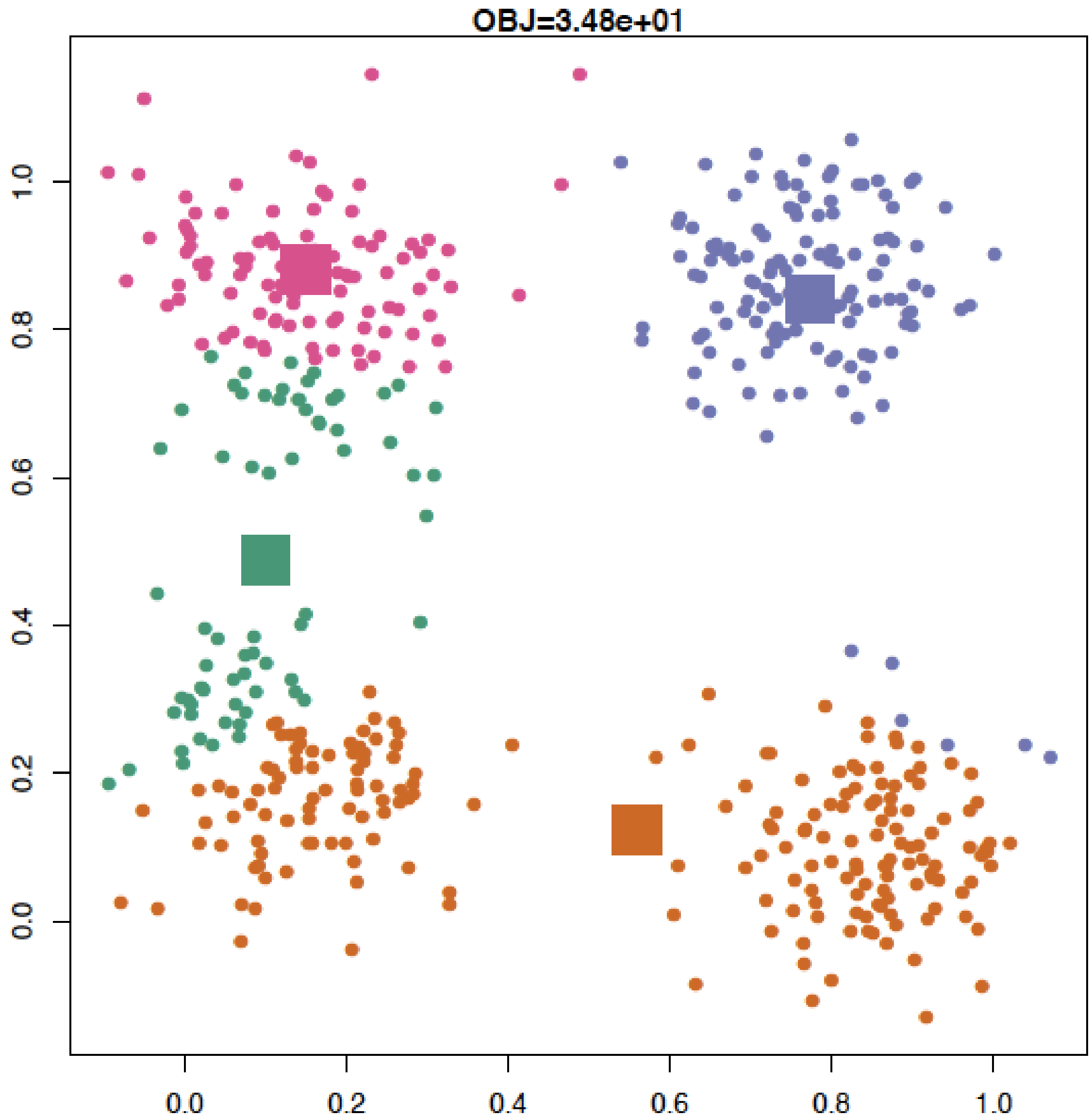
Example: Start



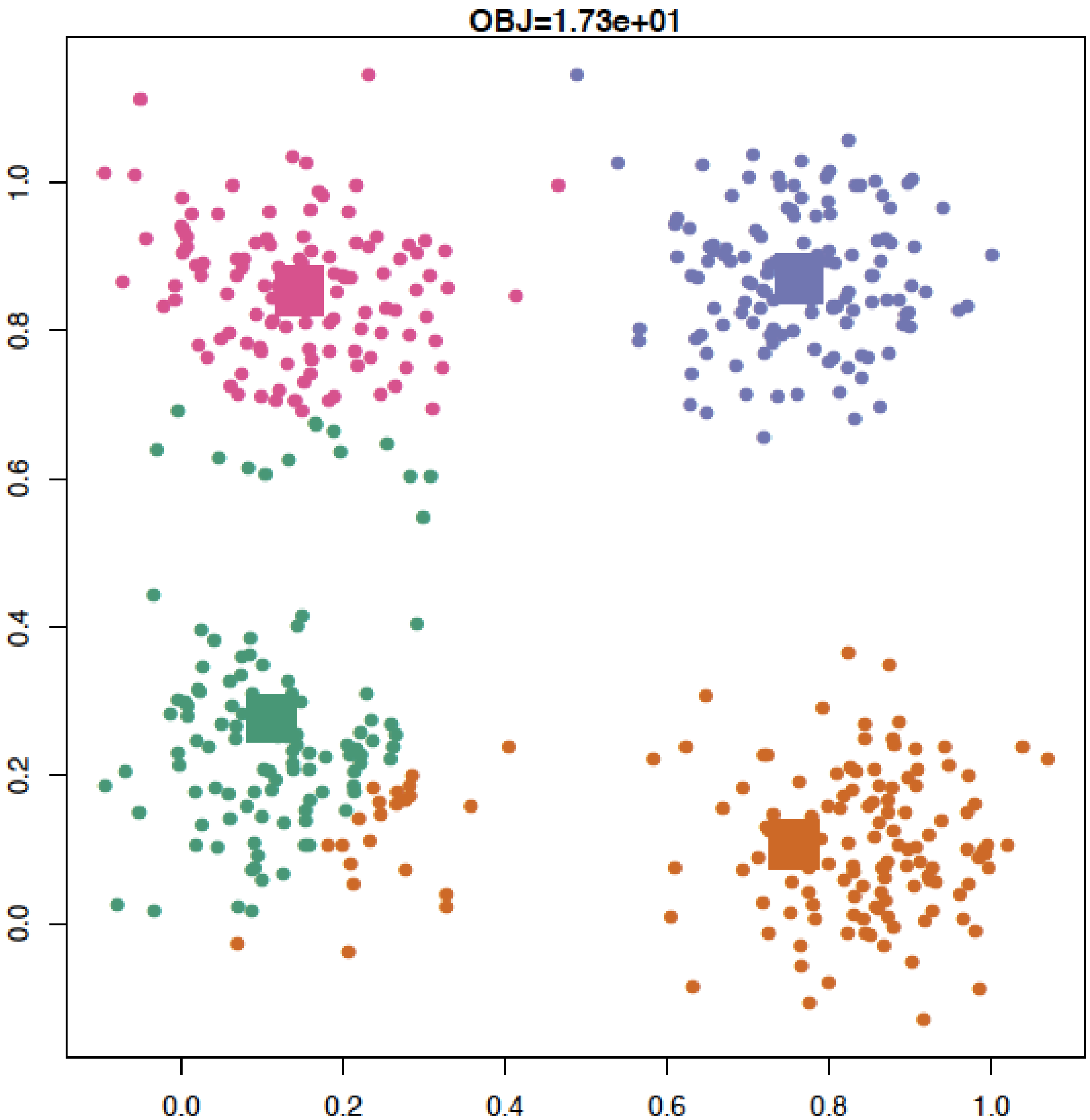
Example: Iteration 1



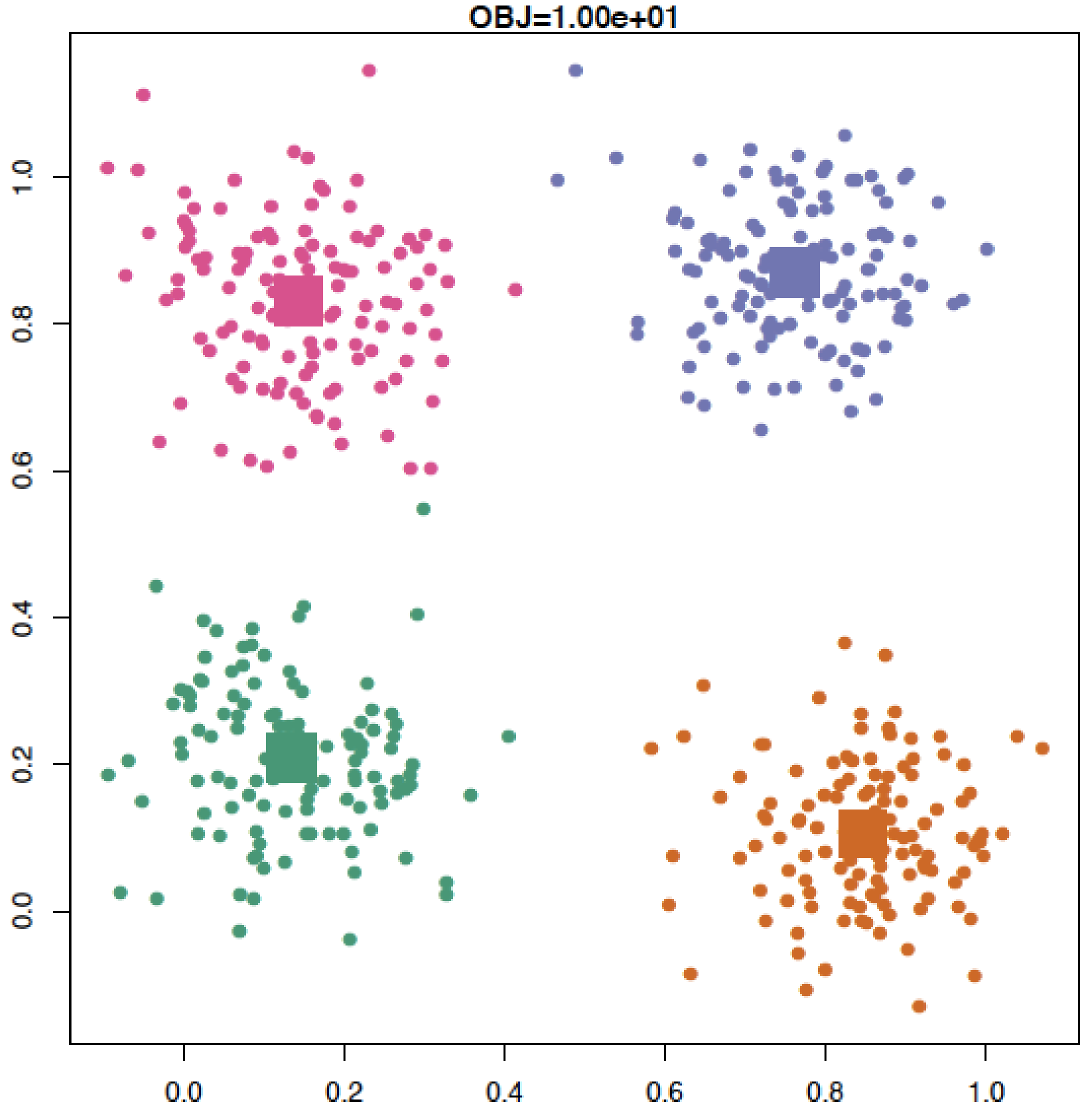
Example: Iteration 2



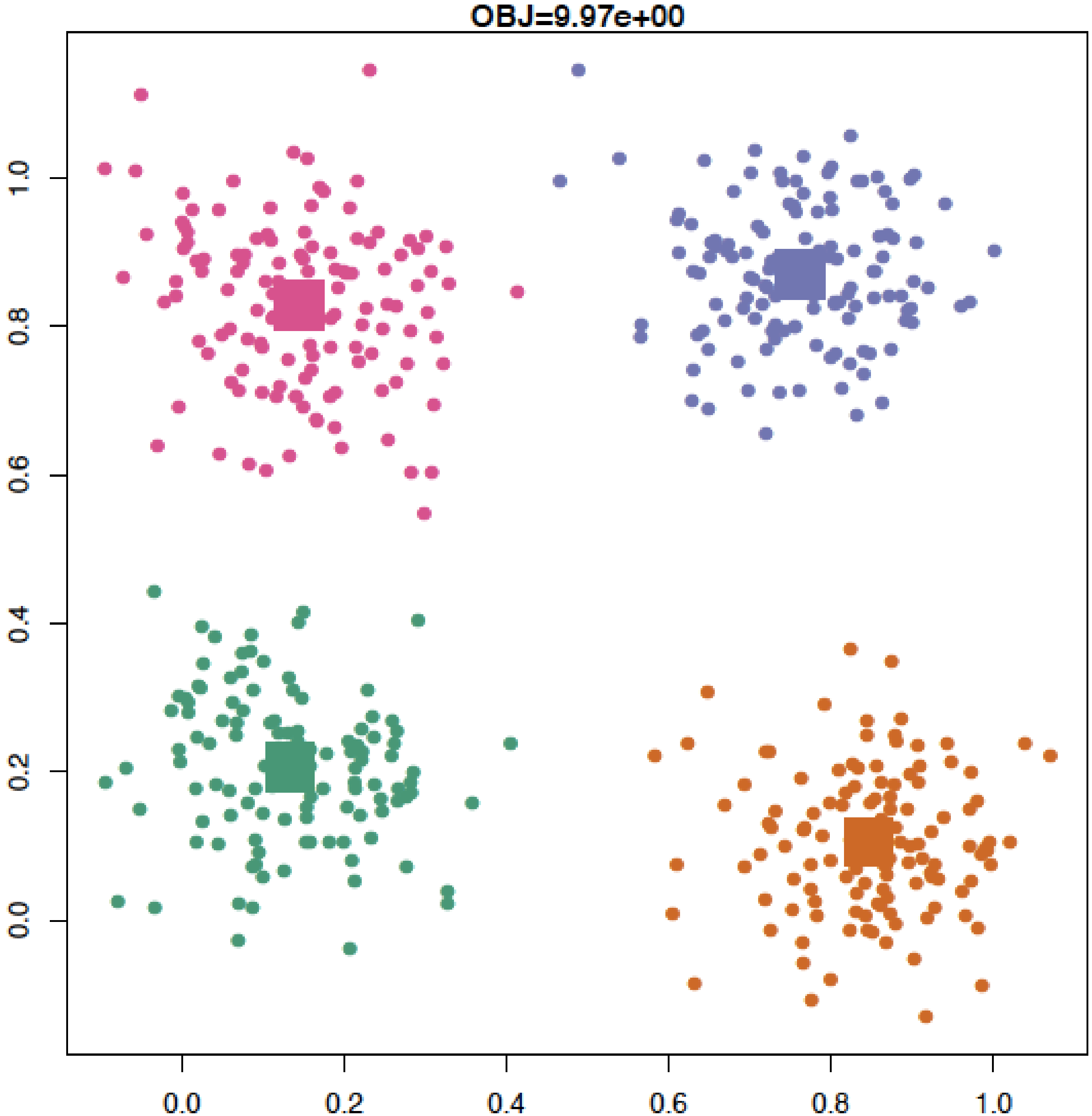
Example: Iteration 3



Example: Iteration 4



Example: Iteration 5



Coordinate descent

Coordinate descent is an optimization procedure for a multi-variate function that optimizes one direction at the time

$$F(z_{1:N}, \mathbf{m}_{1:k}) = \frac{1}{2} \sum_{n=1}^{N} ||\mathbf{x}_n - \mathbf{m}_{z_n}||^2$$

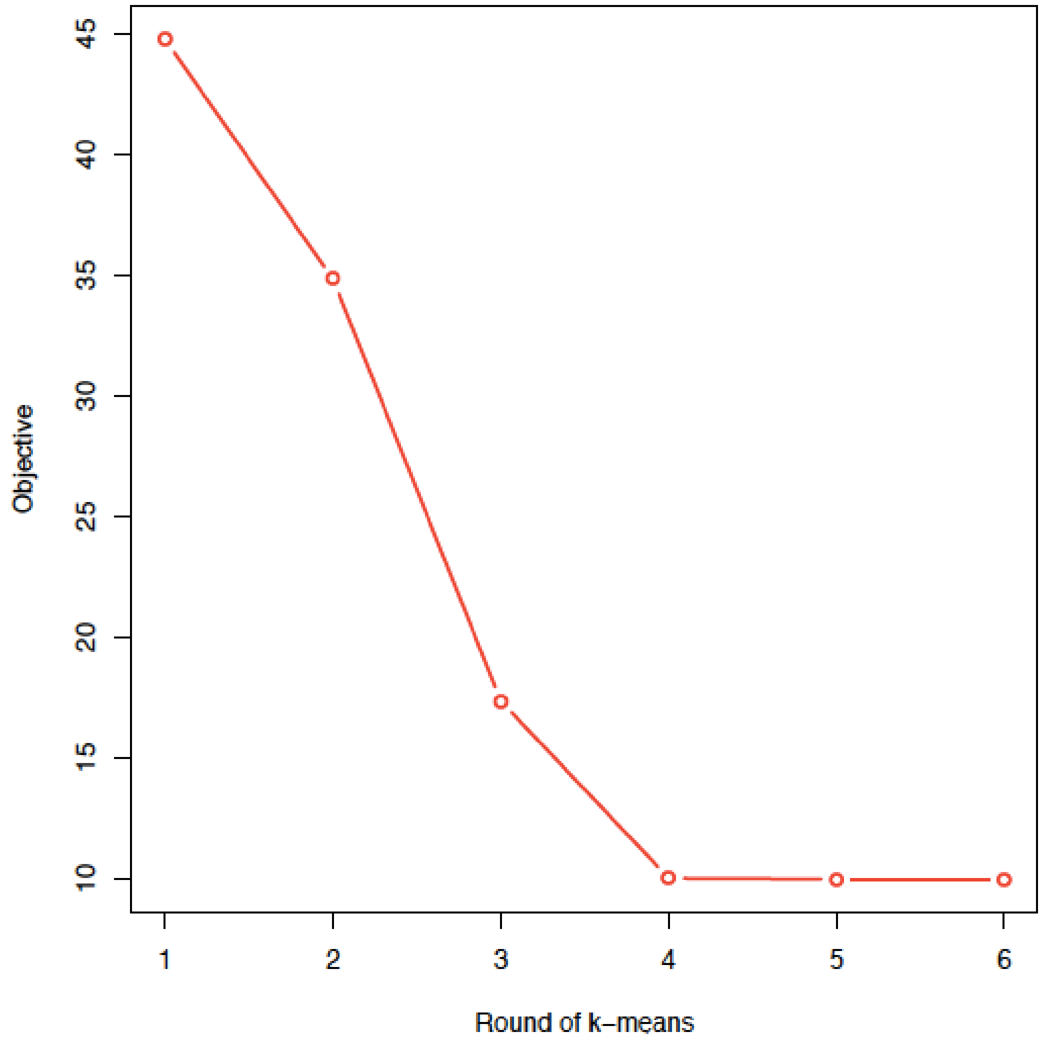
Holding the means fixed, assigning each point to its closest mean minimizes F with respect to $z_{1:N}$.

Holding the assignments fixed, computing the centroids of each cluster minimizes F with respect to $\mathbf{m}_{1:k}$.

Thus, k-means is a *coordinate descent* algorithm.

- However, it finds a local minimum
- Multiple restarts are often necessary

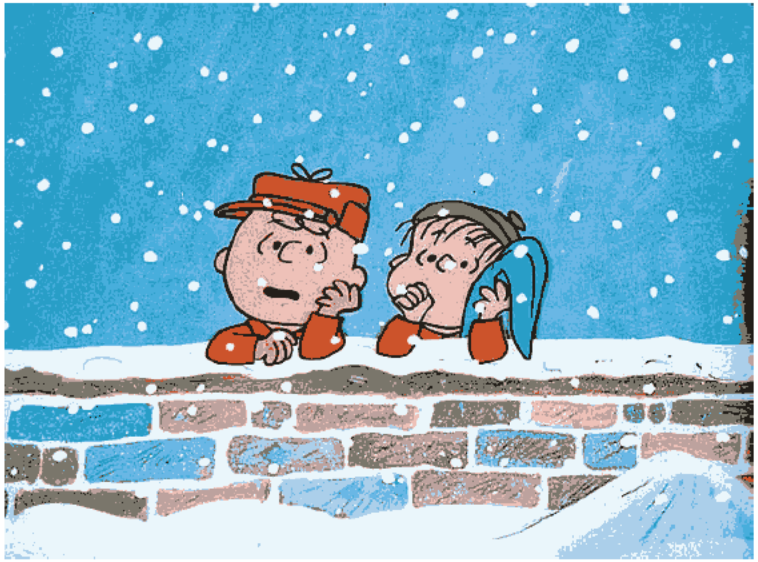
Objective for the example data



Compressing Images

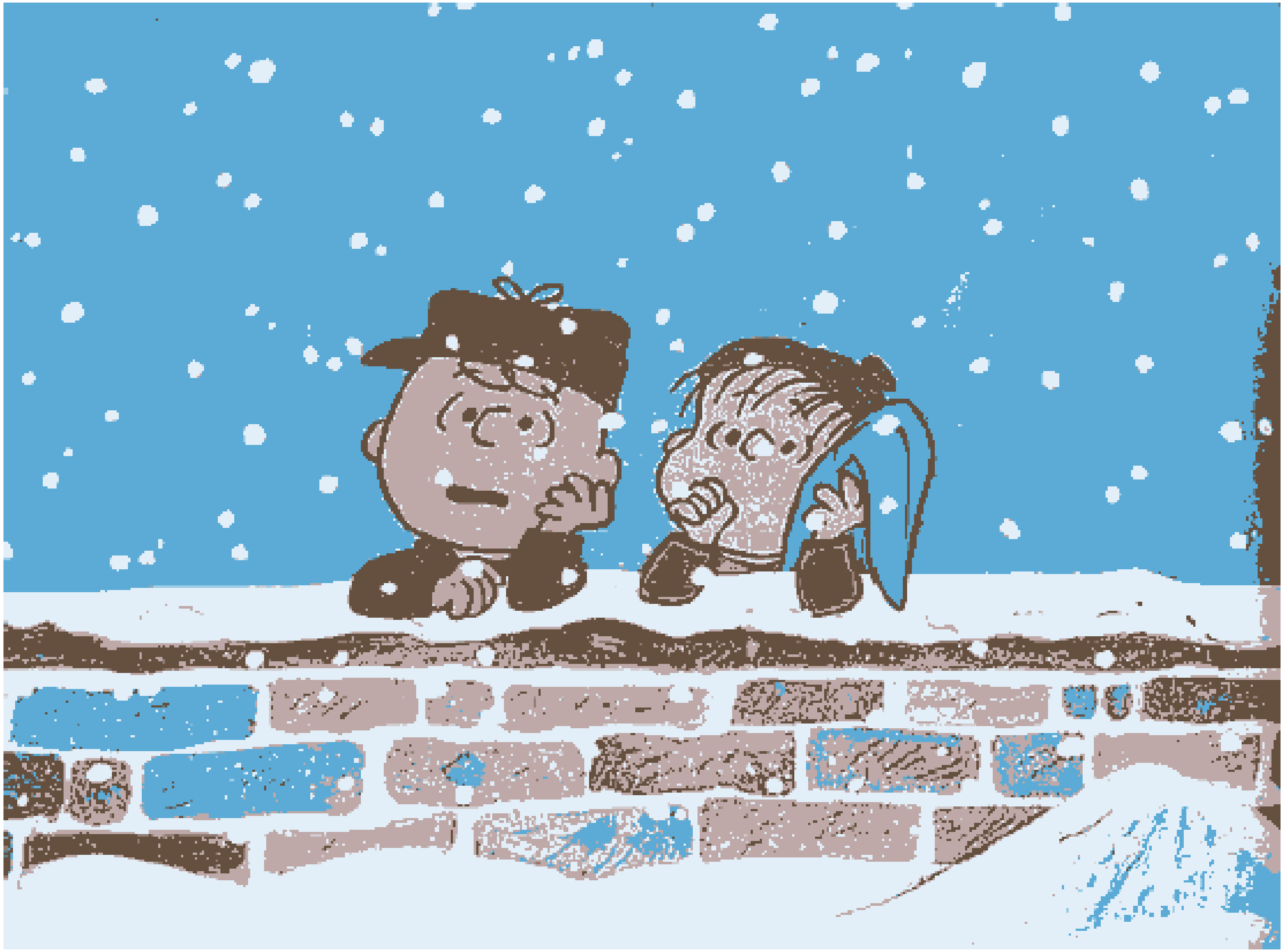
- Each pixel is associated with a red, green, and blue value
- A 1024×1024 image is a collection of 1048576 values $\langle x_1, x_2, x_3 \rangle$, which requires 3M of storage
- How can we use k-means to compress this image?

Vector Quantization

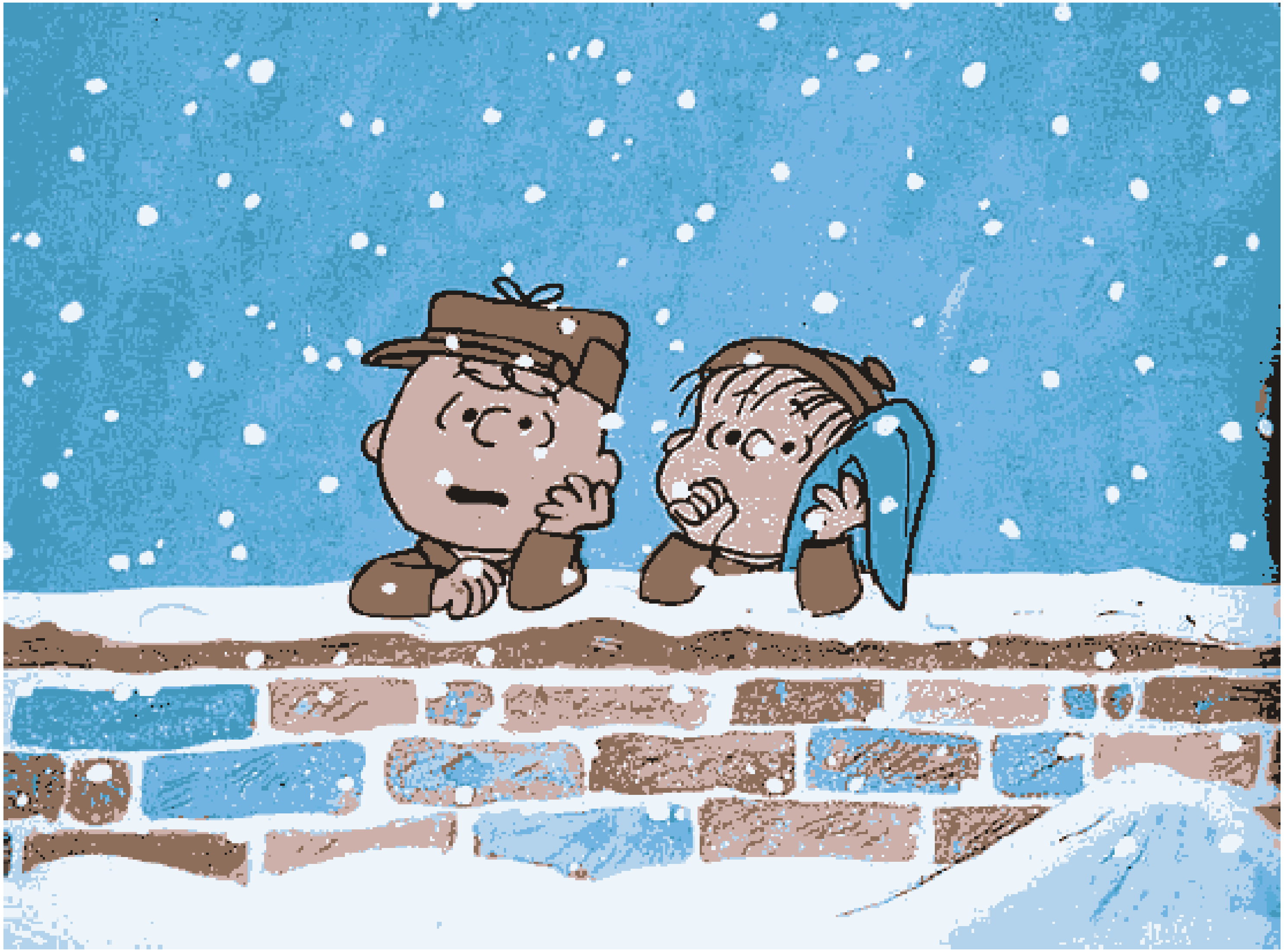


- Replace each pixel $\mathbf{x}_n$ with its assignment $\mathbf{m}_{z_n}$ ("paint by numbers").
- The k means are called the *codebook*.
- With $k = 100$, we need 7 bits per pixel plus 100×3 bits $\approx$ 897K.

2 means

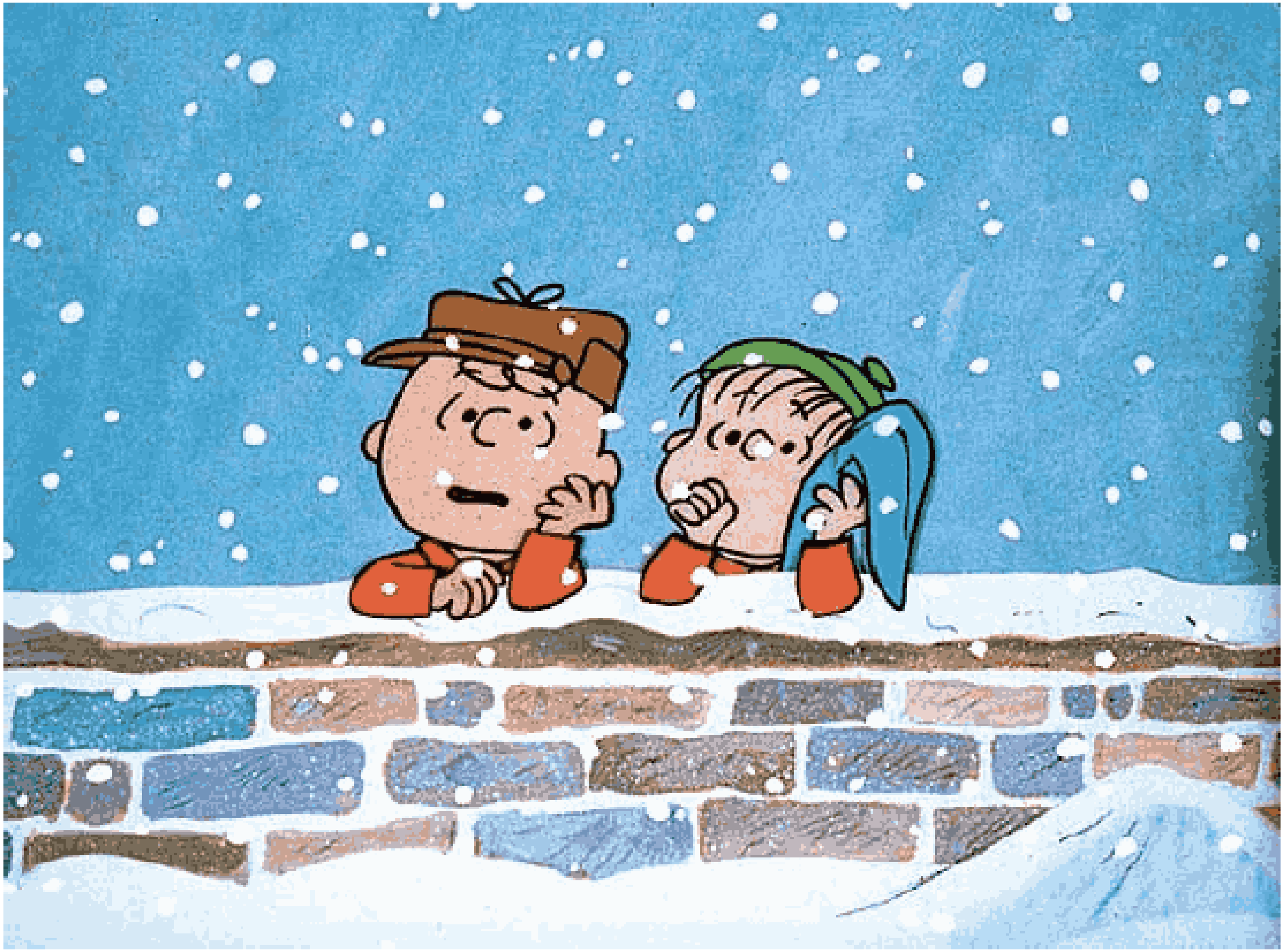


4 means



8 means

16 means



32 means

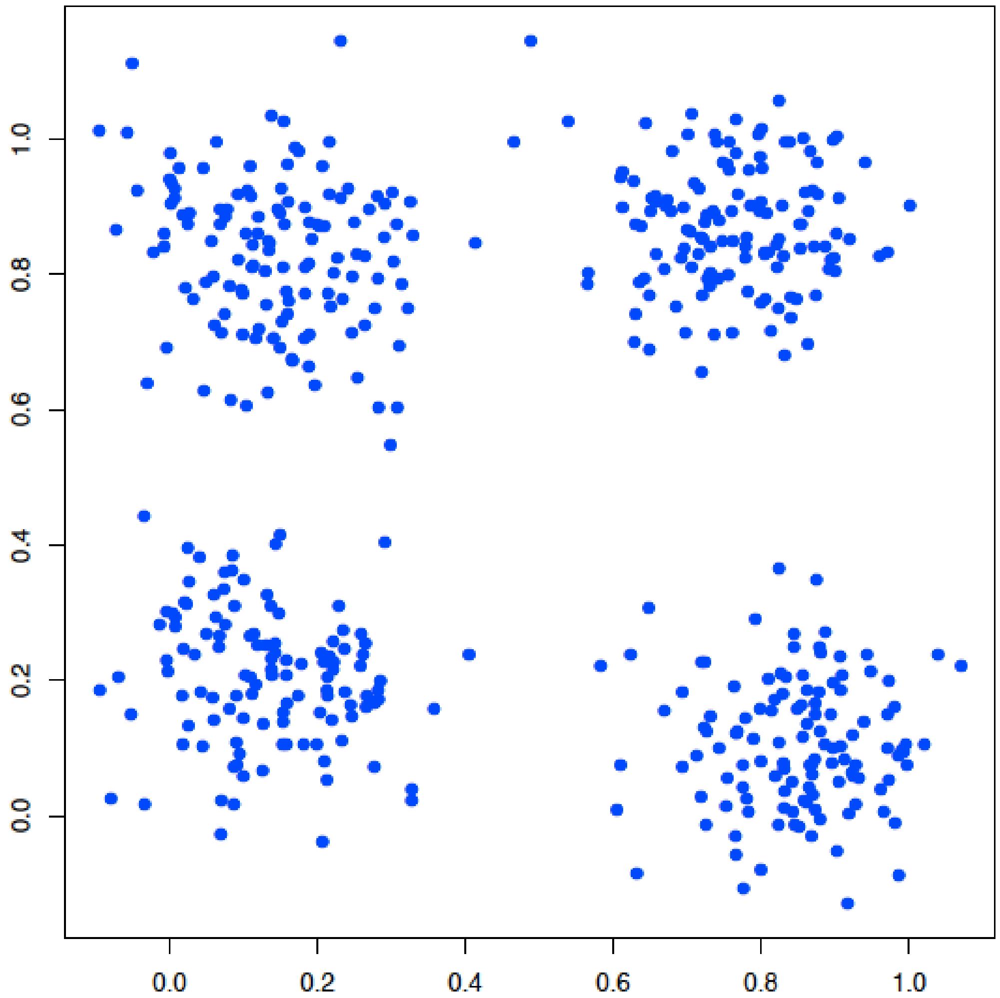
128 means

256 means

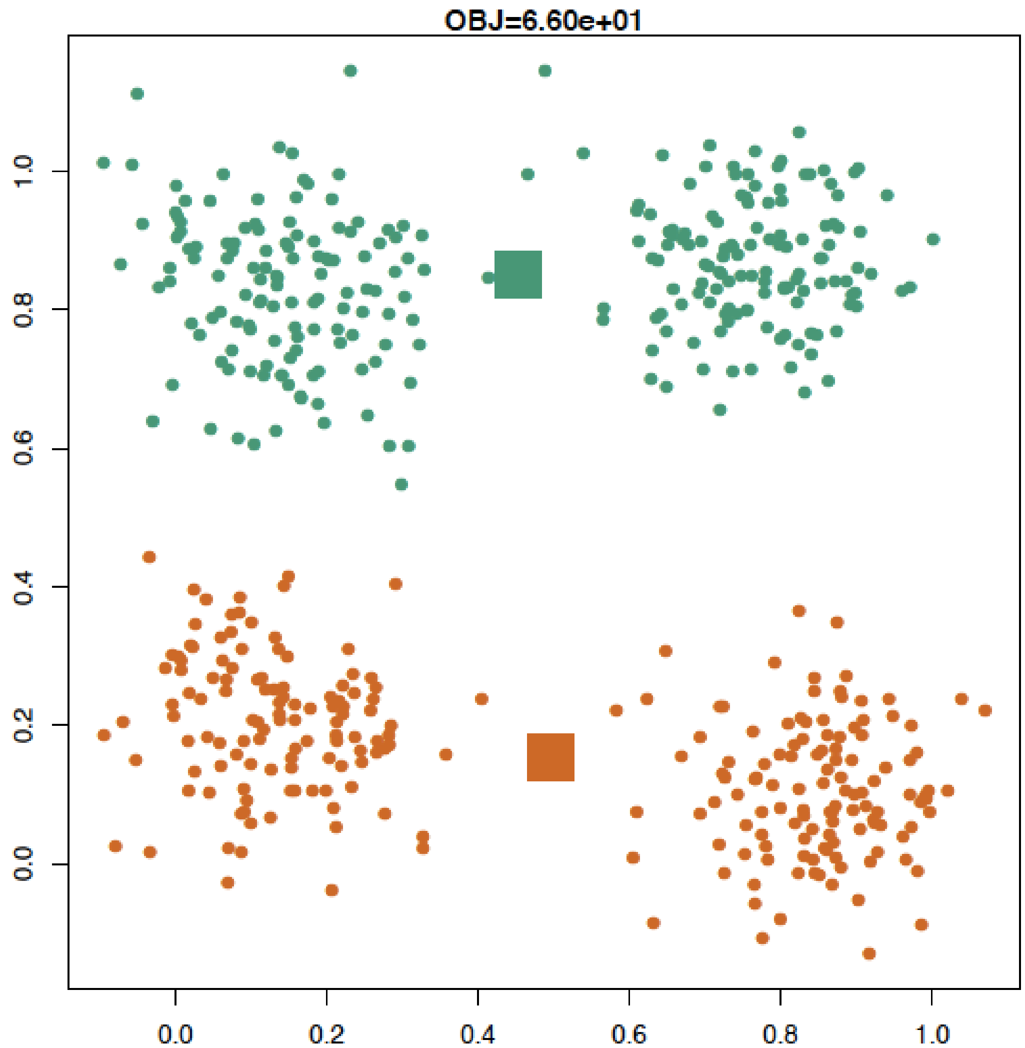
Choosing number of clusters

- Choosing k is a nagging problem in cluster analysis
- Sometimes, the problem determines k
  - A certain required compression in VQ
  - Clustering customers for k salespeople in a business
- Usually, we seek the "natural" clustering, but what does this mean?
- It is not well-defined.

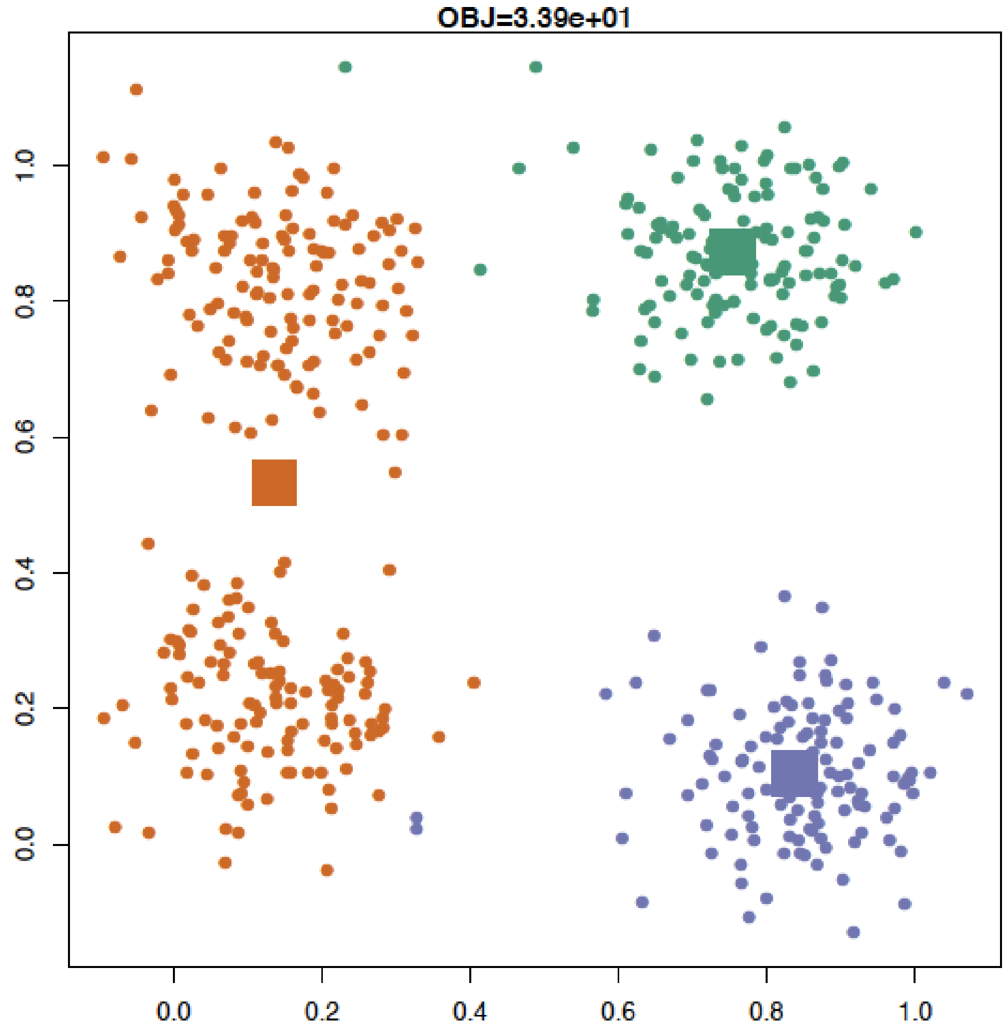
What happens as k increases?



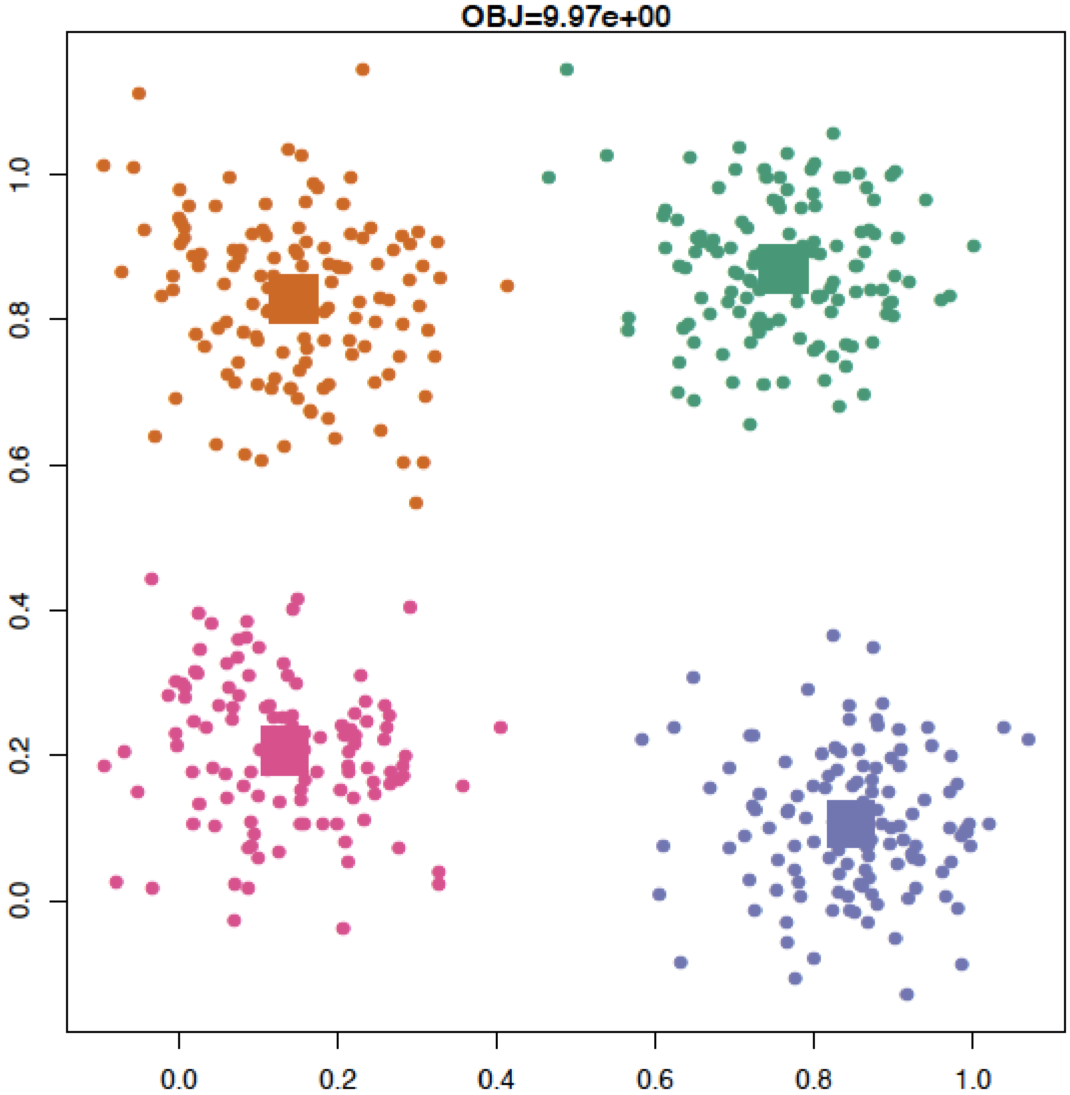
What happens as k increases?



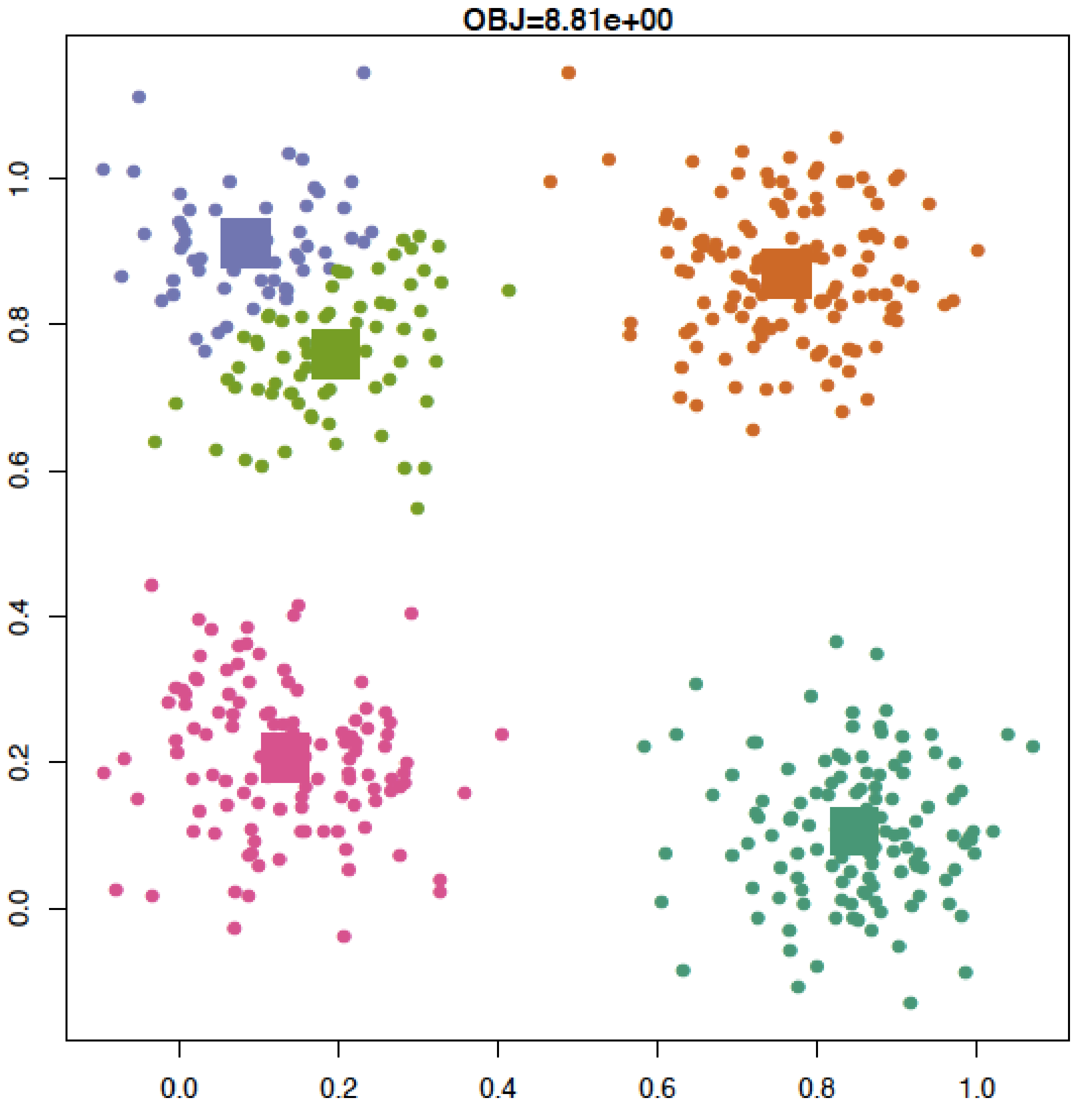
What happens as k increases?



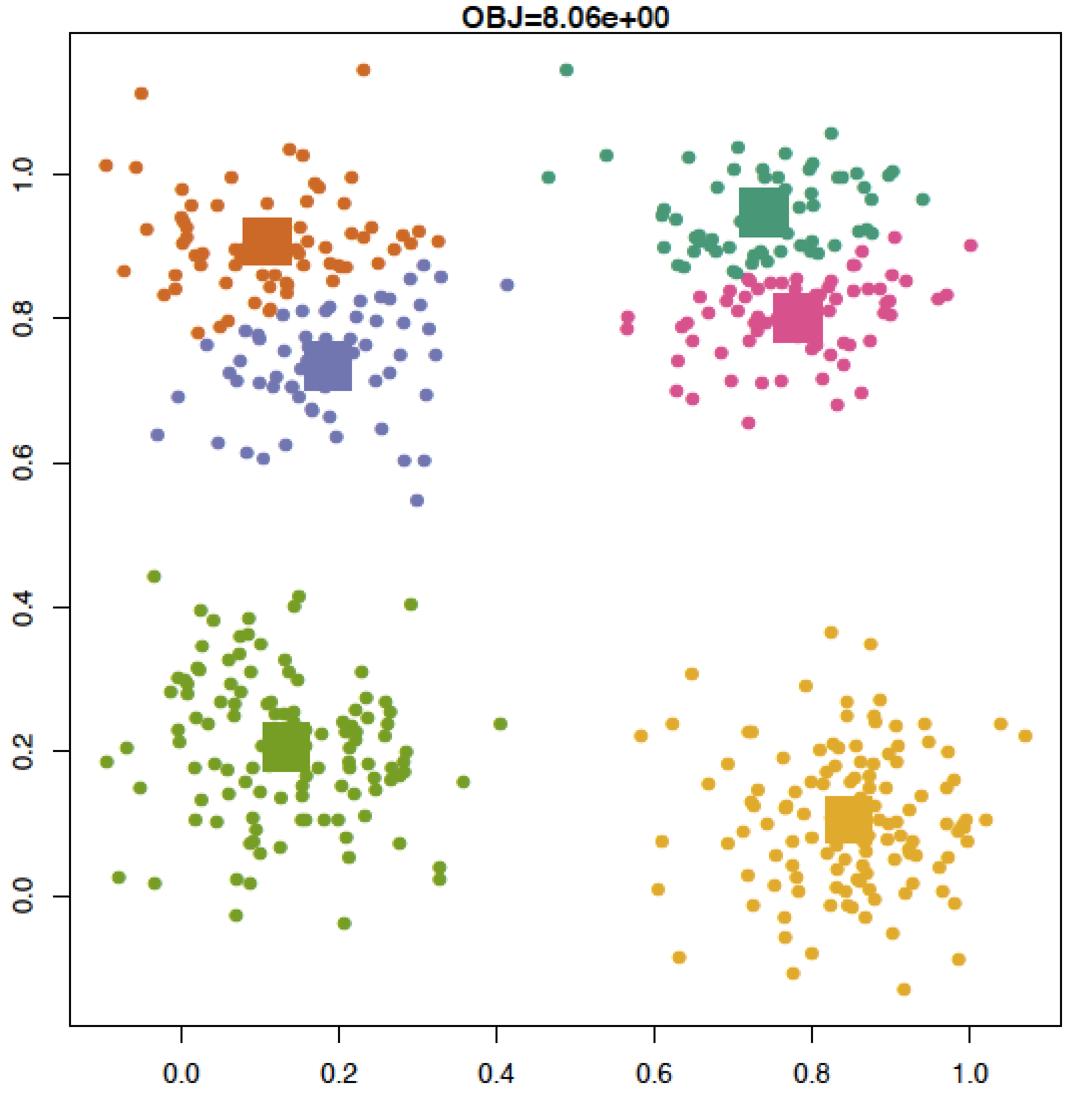
What happens as k increases?



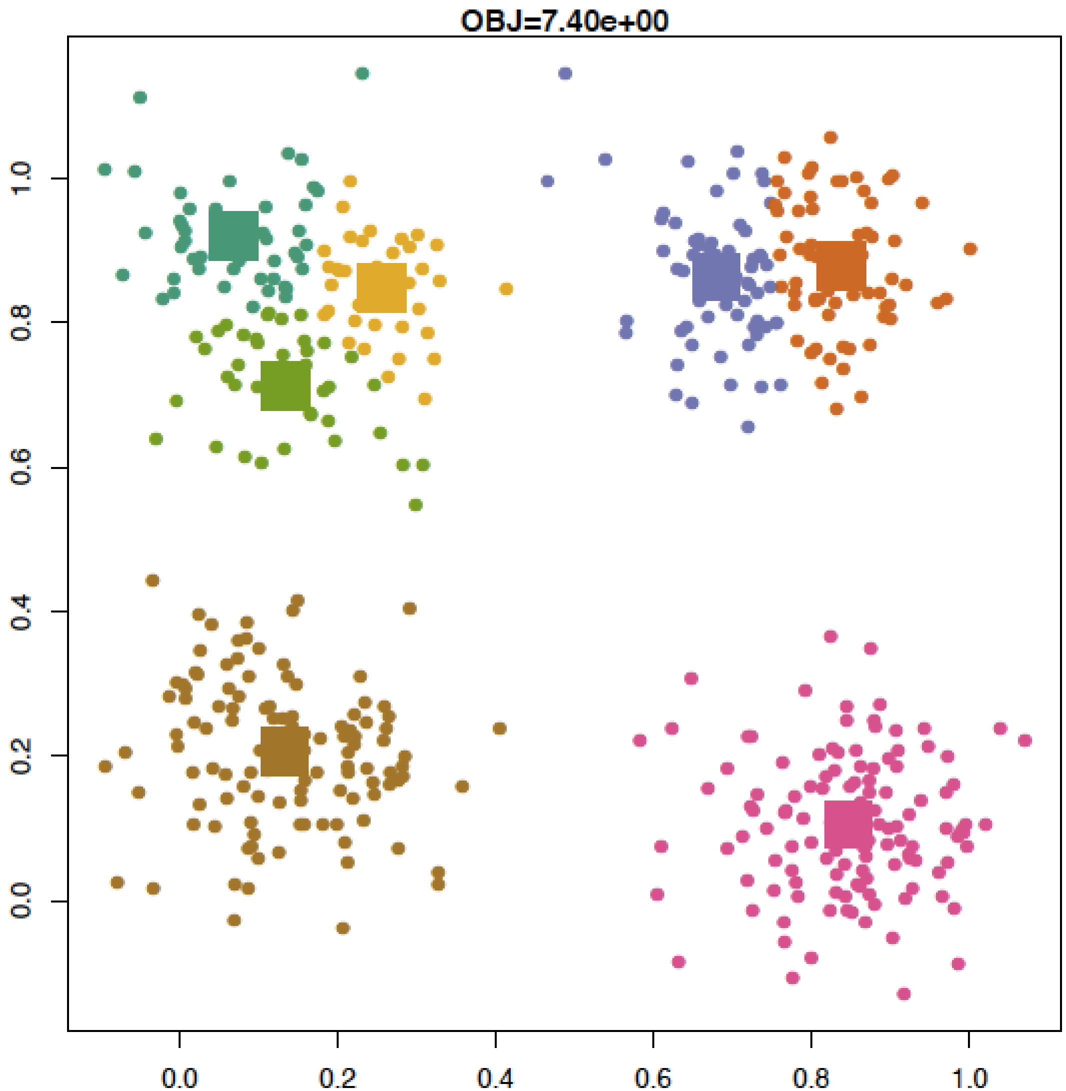
What happens as k increases?



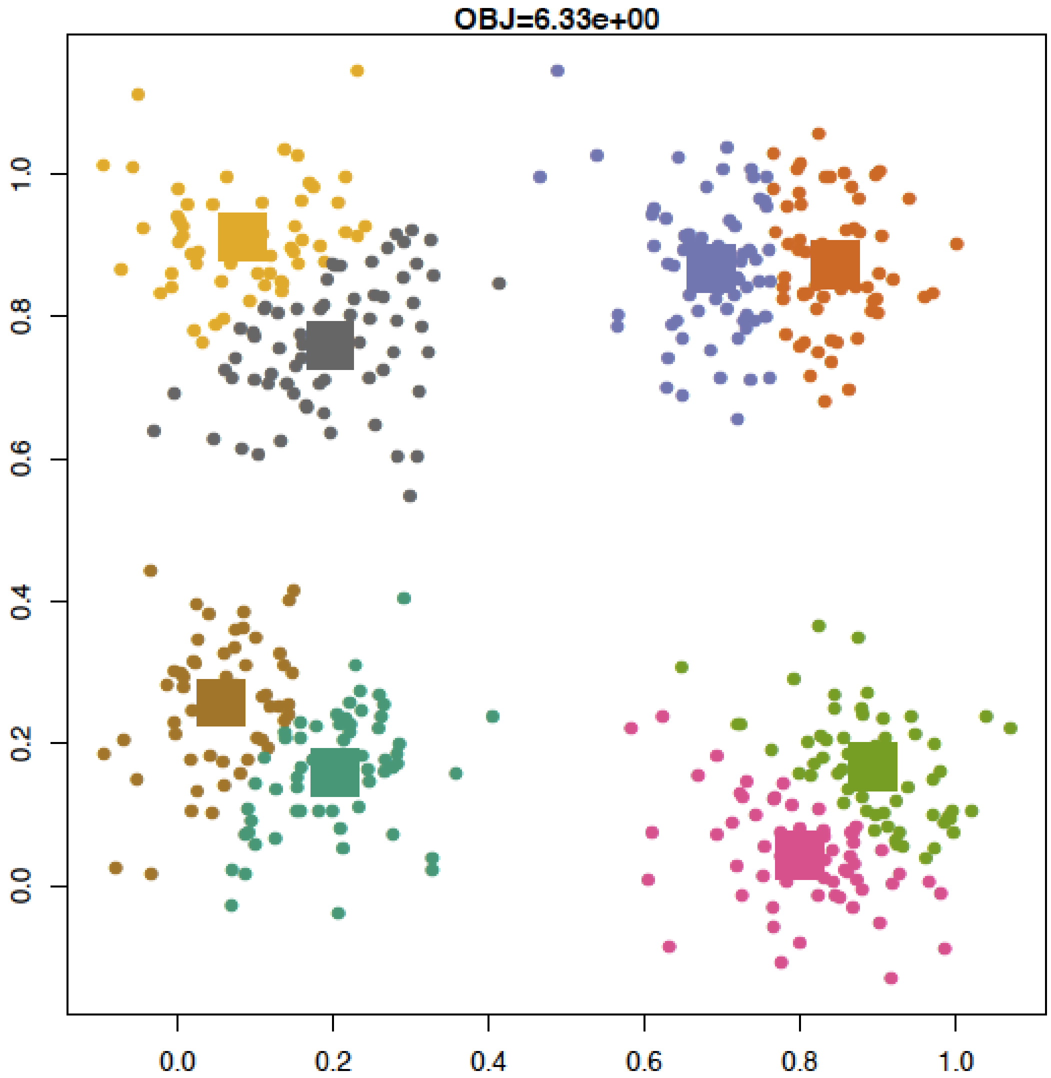
What happens as k increases?



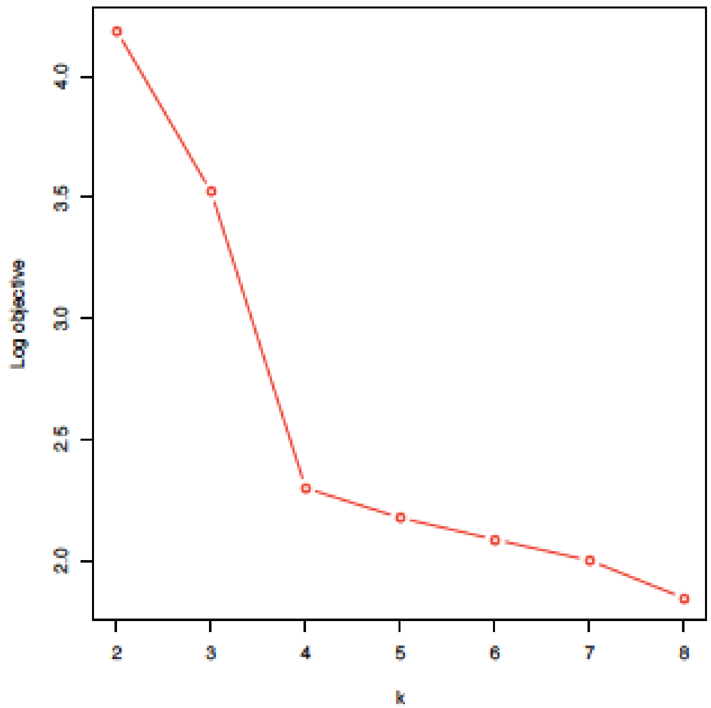
What happens as k increases?



What happens as k increases?



Heuristic: A kink in the objective



Notice the “kink” in the objective between 3 and 5.

This suggests that 4 is the right number of clusters.

Tibshirani (2001) presents a method for finding this kink.

Hierarchical clustering

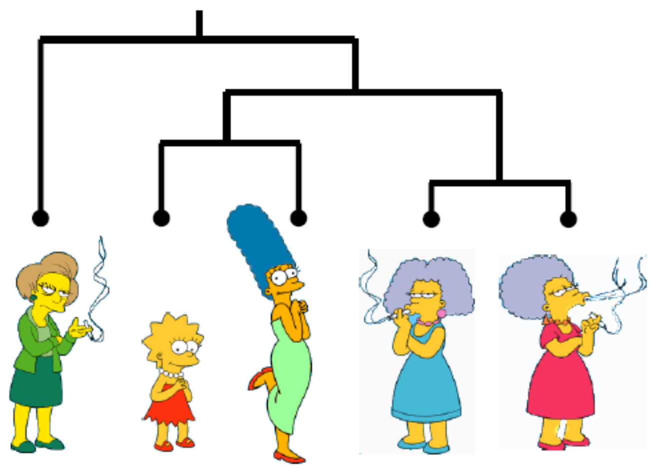
- Hierarchical clustering is a widely used data analysis tool.
- The idea is to build a binary tree of the data that successively merges similar groups of points.
- Visualizing this tree provides a useful summary of the data.
- Advantages
  - Hierarchical clustering only requires a measure of similarity between groups of data points.
  - No specification of number of clusters (k)

Two Types of Clustering

• **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)

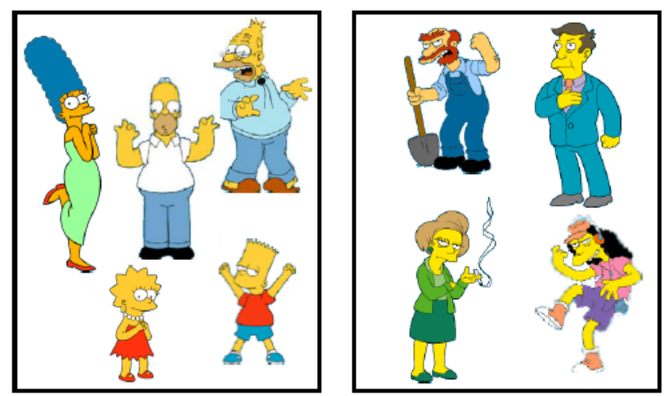
Bottom up or top down

Hierarchical



Top down

Partitional



Agglomerative clustering

- Algorithm:
  - Place each data point into its own singleton group (cluster)
  - Repeat
    - Iteratively merge *the two closest groups/clusters*
  - Until: stopping condition is satisfied
- Output
  - Set of clusters
  - Dendrogram (tree of how data was merged)
- Need to define distance or similarity between groups

Hierarchical Clustering

We begin with a distance matrix which contains the distances between every pair of objects in our database.

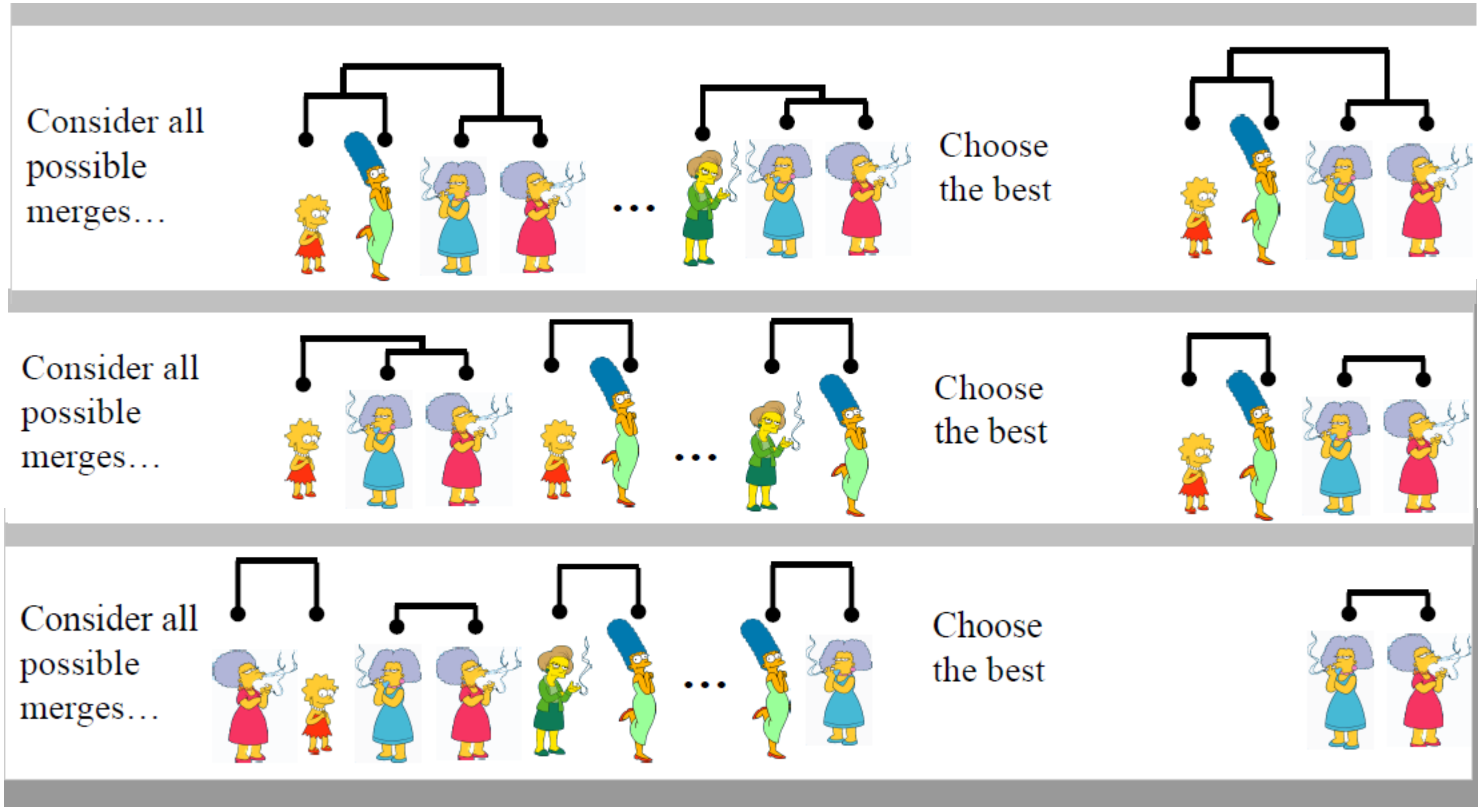
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

D() = 8

D() = 1

Hierarchical Clustering

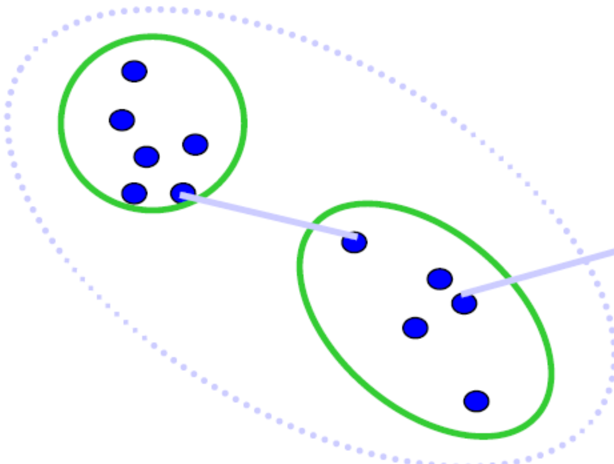
Bottom-Up (agglomerative):
Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



Computing distance between clusters

Single Linkage

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

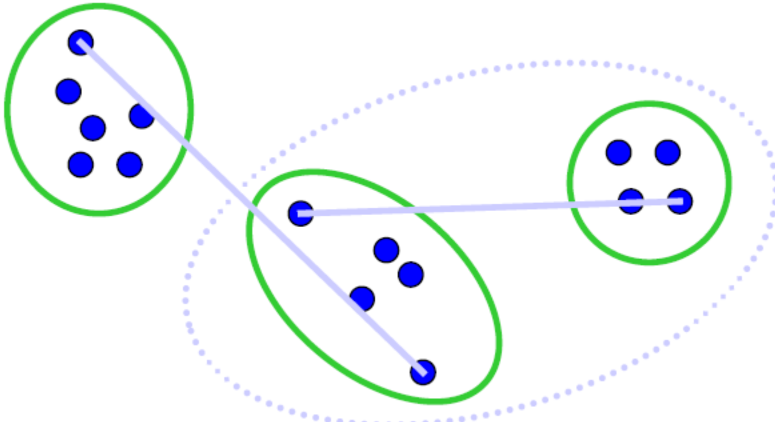


- Potentially long and skinny clusters

Computing distance between clusters

Complete Linkage

- cluster distance = distance of two farthest members

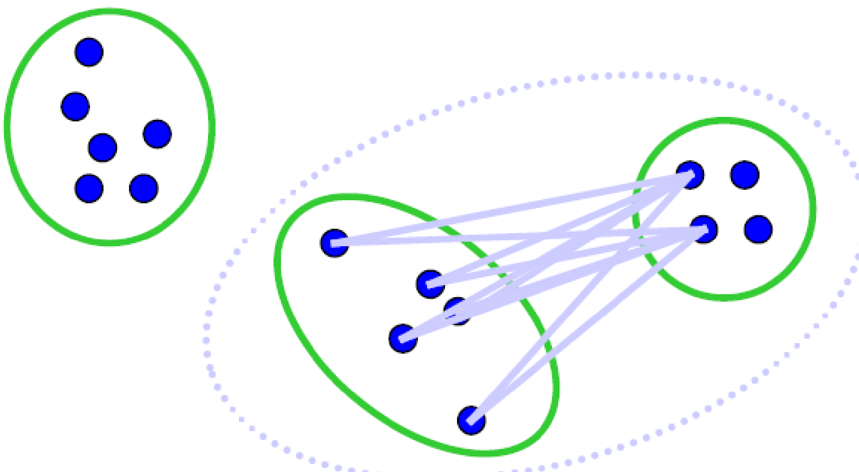


\+ tight clusters

Computing distance between clusters

Average Linkage

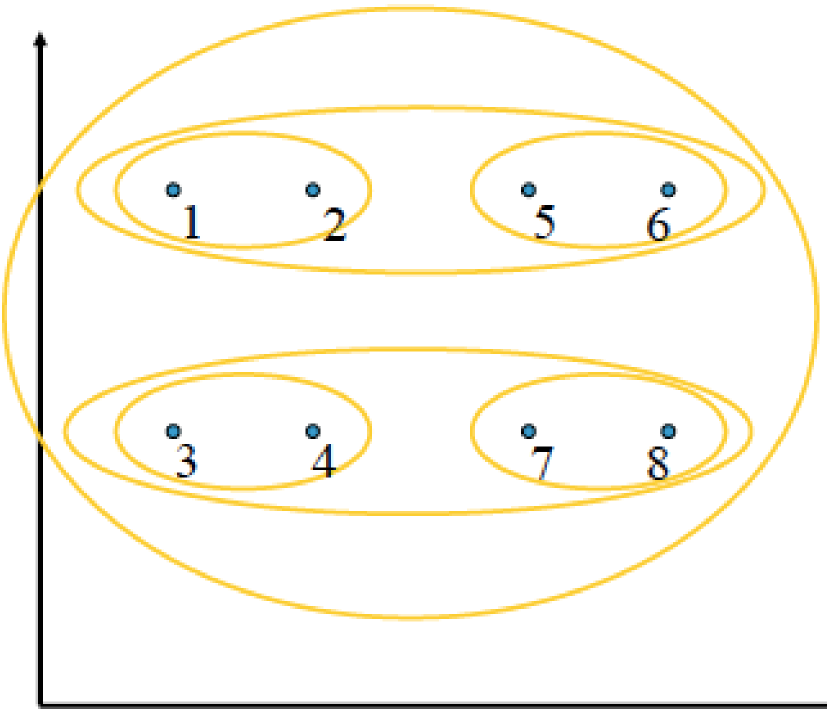
- cluster distance = average distance of all pairs



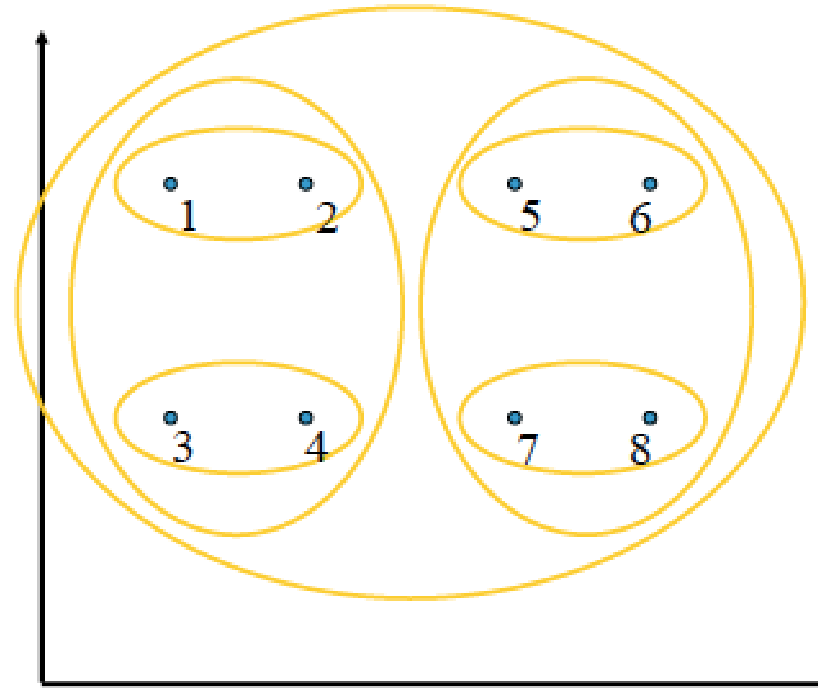
the most widely used measure

Robust against noise

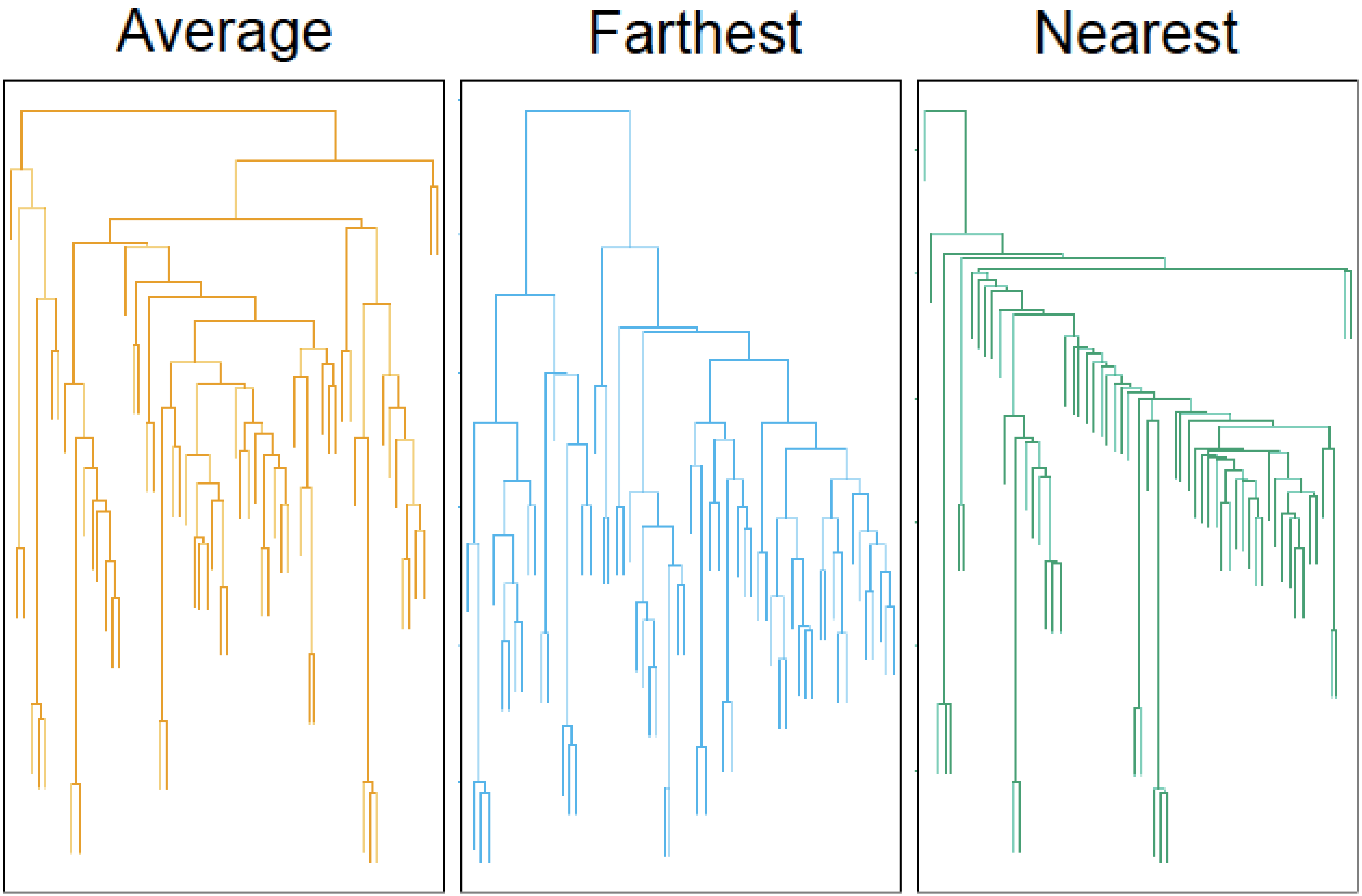
Closest pair
(single-link clustering)



Farthest pair
(complete-link clustering)



[Pictures from Thorsten Joachims]



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

Acknowledgements

- Slides made using resources from:
  - Lu Wang
  - David M. Blei
  - Eric Eaton
- Thanks!