Machine Learning – COMS3007

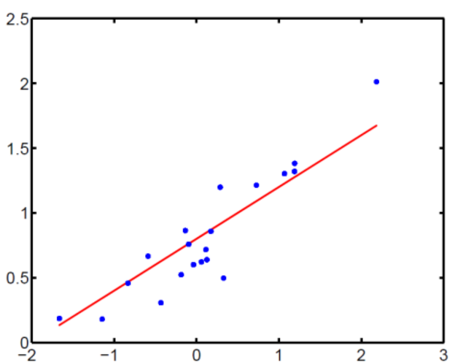
Clustering

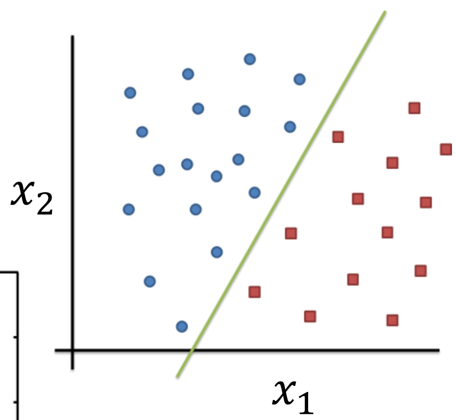
Benjamin Rosman

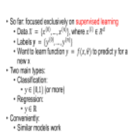
Based heavily on course notes by Chris Williams, Victor Lavrenko, Charles Sutton, David Blei, David Sontag, Shimon Ullman, Tomaso Poggio, Danny Harari, Daneil Zysman, Darren Seibart, and Clint van Alten

Previously on ML…

- So far: focused exclusively on supervised learning
  - Data $X = \{x^{(0)}, \ldots, x^{(n)}\}$, where $x^{(i)} \in R^d$
  - Labels $\mathbf{y} = \{y^{(0)}, \ldots, y^{(n)}\}$
  - Want to learn function $y = f(x, \theta)$ to predict y for a new x
- Two main types:
  - Classification:
    - $y \in \{0,1\}$ (or more)
  - Regression:
    - $y \in \mathbb{R}$
- Conveniently:
  - Similar models work







Unsupervised learning

- In supervised learning, we know what we are looking for
  - We have appropriately labelled data

- This isn't always the case!

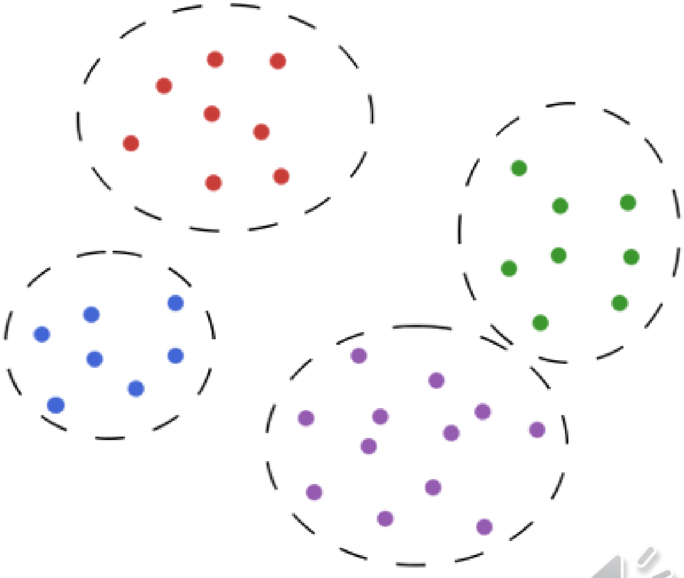
- Unsupervised learning:
  - Find patterns in the data (without labels)
  - Understanding the hidden structure of the data
  - Useful when you don't know what you're looking for

- Data:
  - Given $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where each $\boldsymbol{x} \in \mathbb{R}^d$
  - No labels!

Clustering

- Clustering: one of the most common unsupervised learning problems

- Involves automatically segmenting data into groups of similar points
- Why?
  - Automatically organising data
  - Understanding structure of the data
    - Finding sub-populations
  - Representing high dimensional data in a low dimensional space



Examples

- Make groupings from data, such as:
    - Customers based on their purchase histories
    - Genes according to expression profile
    - Search results according to topic
    - Facebook users according to interests
    - Artifacts in a museum according to visual similarity

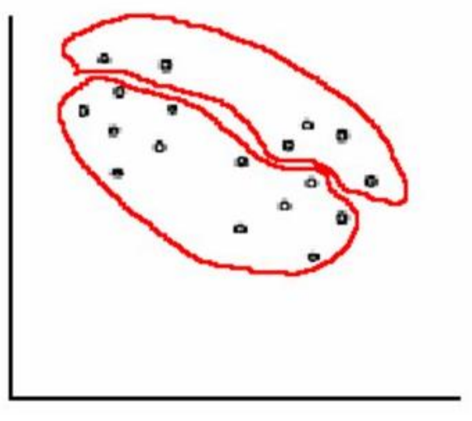
Note: this is different to classifying. We don’t even know what the classes are! This gives us a way to discover them.

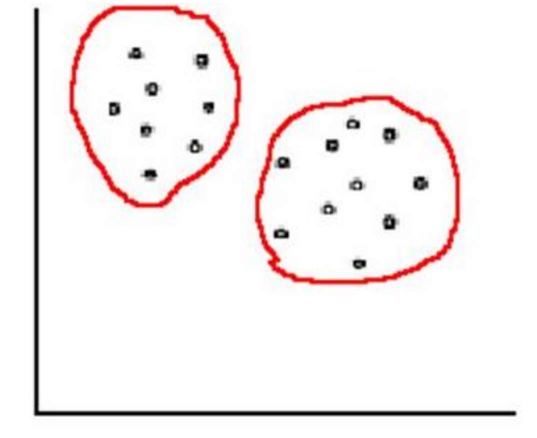
Properties

- What makes a good clustering?

- Intra-cluster cohesion (compactness)
  - Points in the same cluster are close together

- Inter-cluster separation (isolation)
  - Points in different clusters are far apart

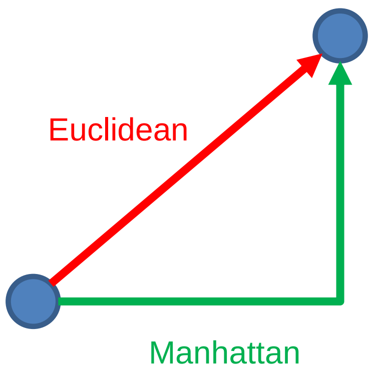




Distance metrics

- Notions of “closeness” require a distance metric

- Euclidean distance
  - $d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$



- Manhattan (city block) distance
  - $d(x_i, x_j) = \sum_{k=1}^{d} |x_i^{(k)} - x_j^{(k)}|$
  - Approximation to Euclidean distance

- Both are special cases of Minkowski distance:
  - $d(x_i, x_j) = \left(\sum_{k=1}^{d} \left|x_i^{(k)} - x_j^{(k)}\right|^p\right)^{\frac{1}{p}}$ p is a positive integer

K-means

- K-means is one of the most commonly used clustering algorithms
- Partitional clustering algorithm (maintains partitions over the space)
- Data points $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where each $\boldsymbol{x} \in \mathbb{R}^d$
- K-means partitions the data into k clusters
    - Each cluster has a **cluster centre**, called the **centroid**
    - k is user specified

K-means algorithm

- Input: $D = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$, where each $\boldsymbol{x} \in \mathbb{R}^d$
- Place centroids $c_1, c_2, \dots, c_k$ at random locations in $\mathbb{R}^d$
- Repeat until convergence (cluster assignments don't change):
  - For each point x_i:
    - Find the closest centroid $c_j = argmin_{c_j}\, d(x_i, c_j)$
    - Assign x_i to cluster j
  - For each cluster j:
    - Move the cluster centre c_j to the average of the assigned points: $c_j = \frac{1}{n_j} \sum_{i:x_i \to j} x_i$

Choose distance metric $d(\cdot,\cdot)$ appropriately

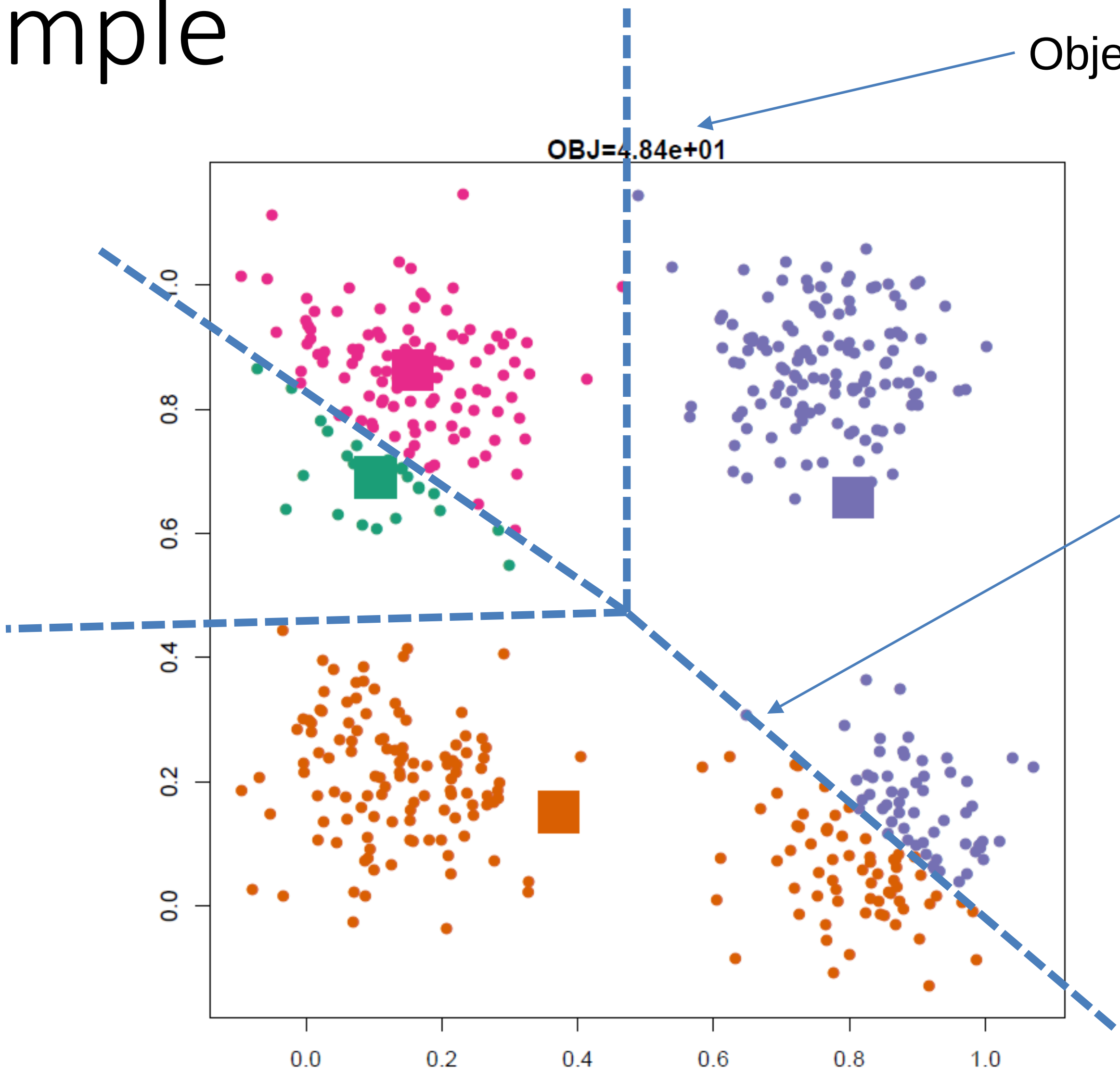
Can compute median, etc., instead of mean

Performance

- Need an objective function to measure performance of the algorithm
- K-means objective function is the sum of squared distances of each point to its assigned mean.
- Let x_i be assigned to cluster z_i
- Then
  - $J(x_{1:N}, c_{1:K}) = \frac{1}{2}\sum_{i=1}^{N} \left\|x_i - c_{z_i}\right\|^2$

Distance between point and its cluster

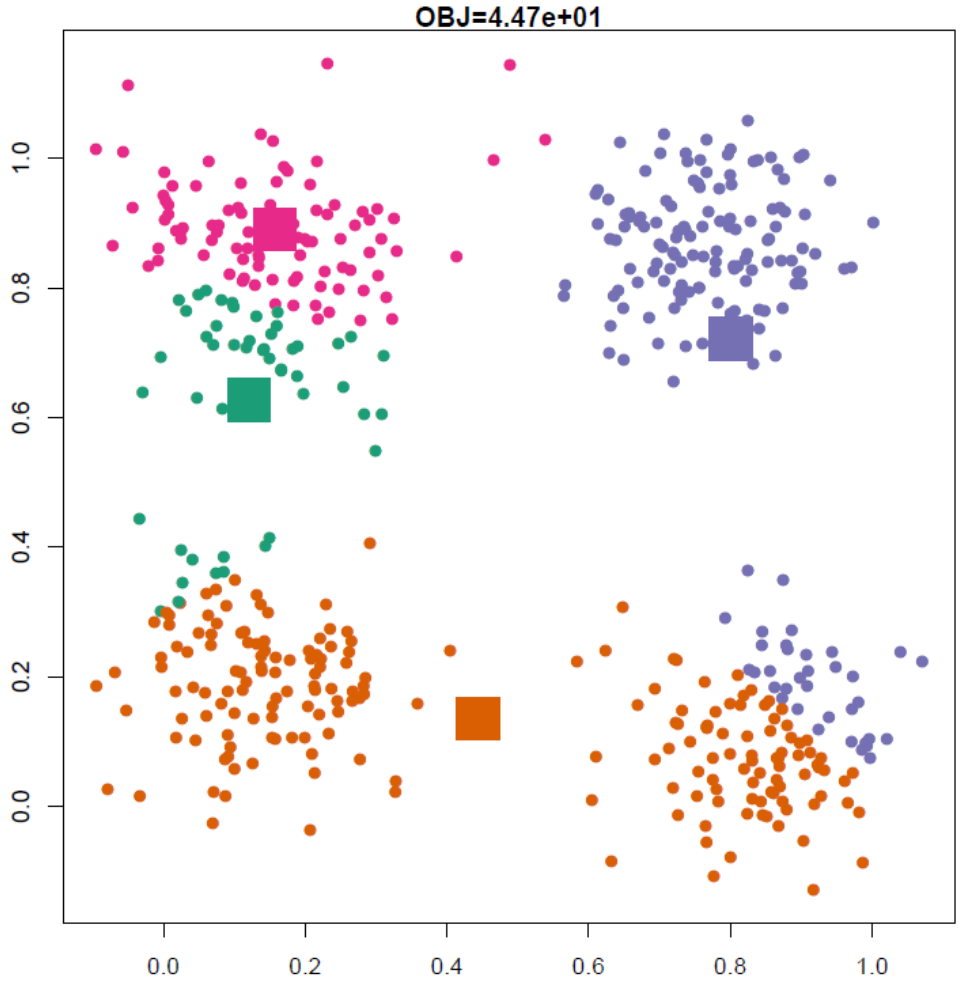
Example



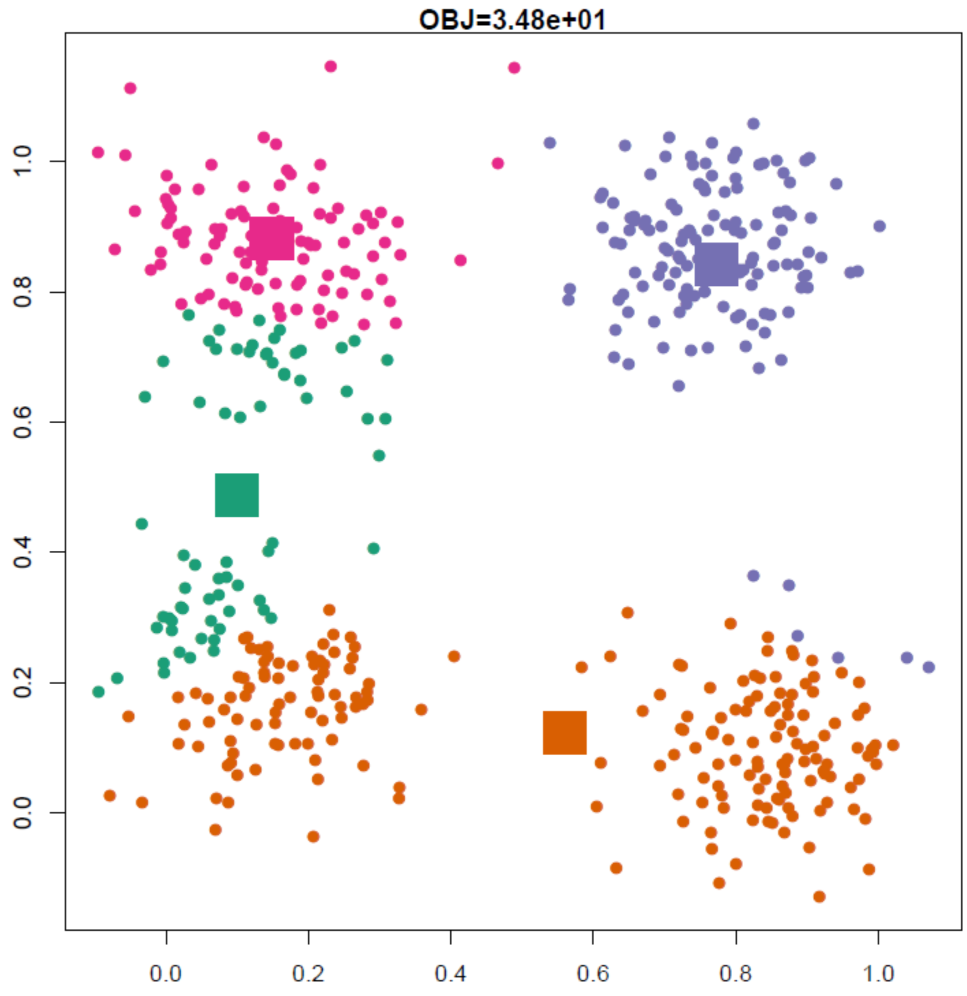
Objective value

Cluster boundaries: Points equidistant from 2 cluster centres. Defines a partitioning.

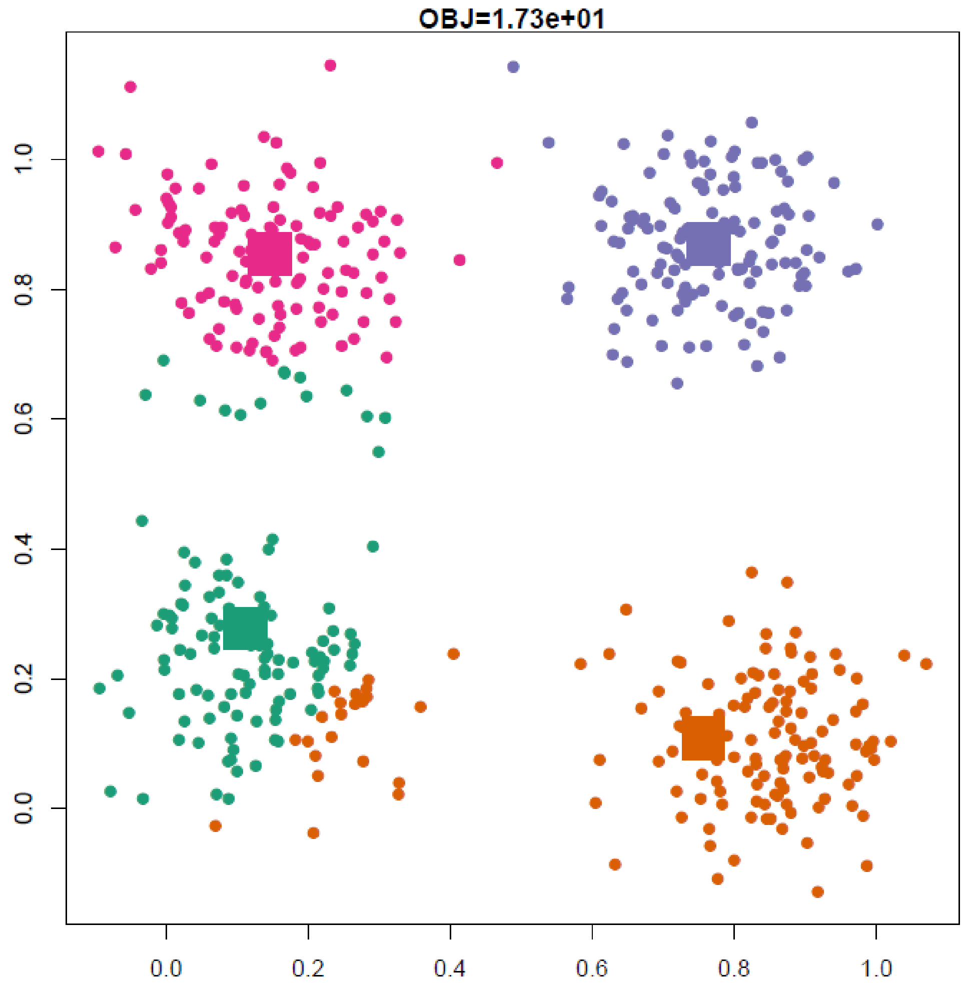
Example



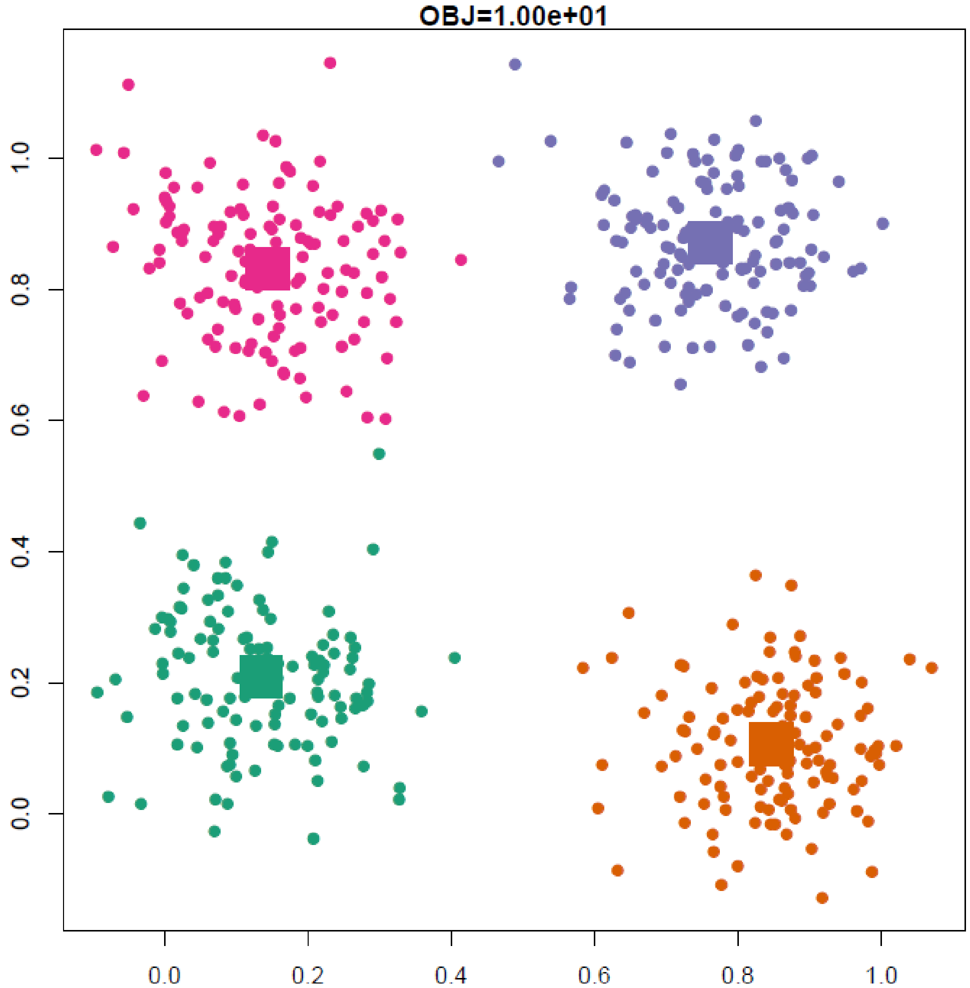
Example



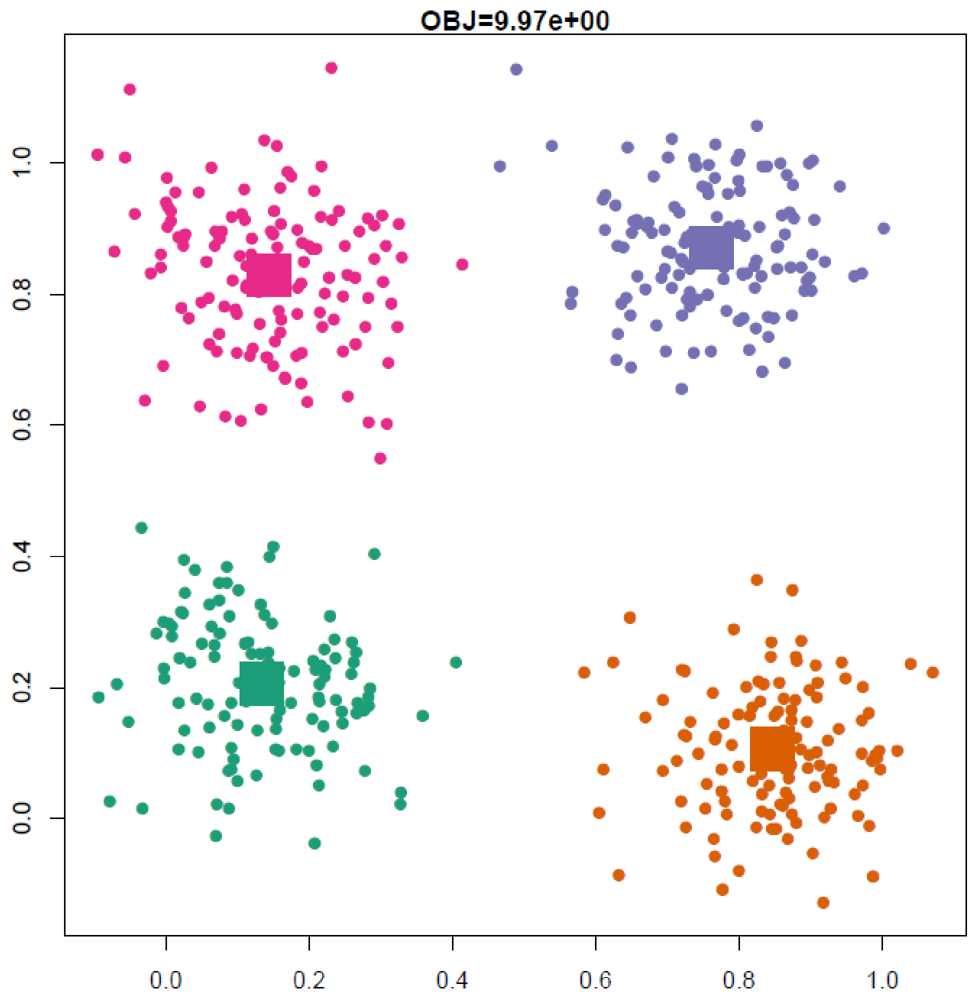
Example



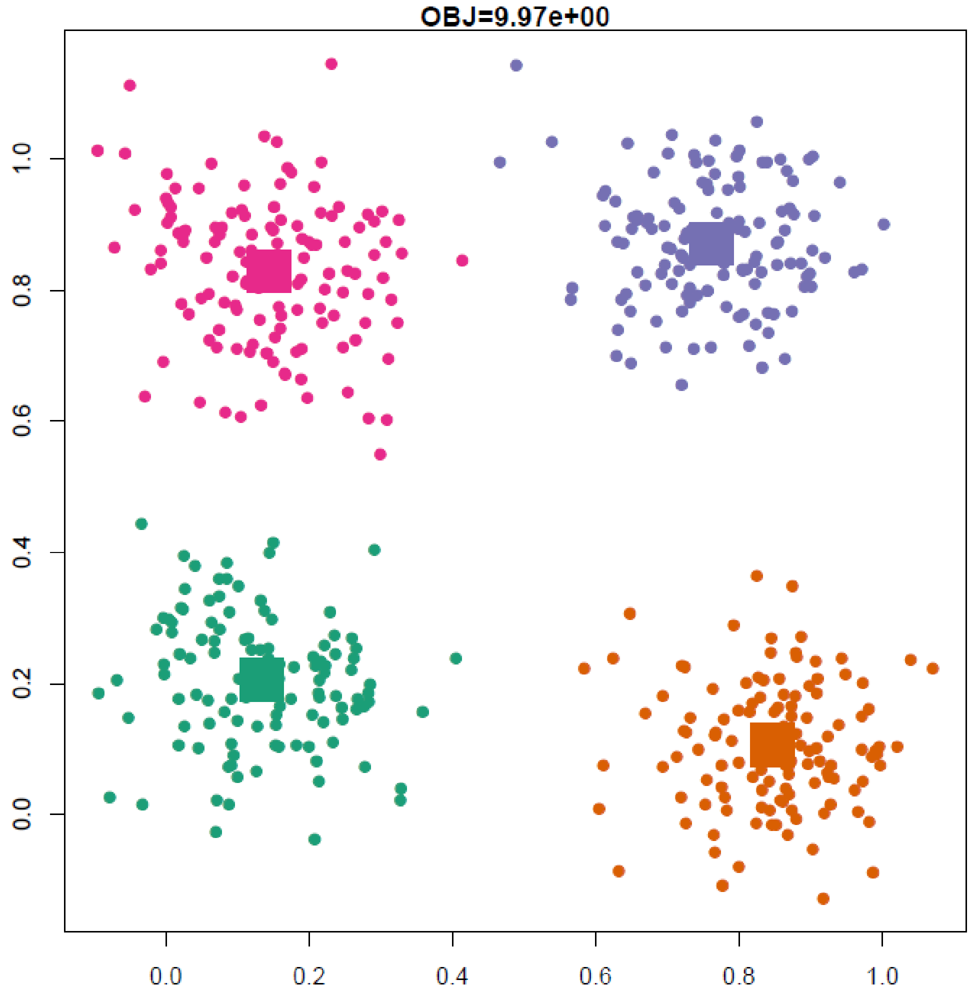
Example



Example



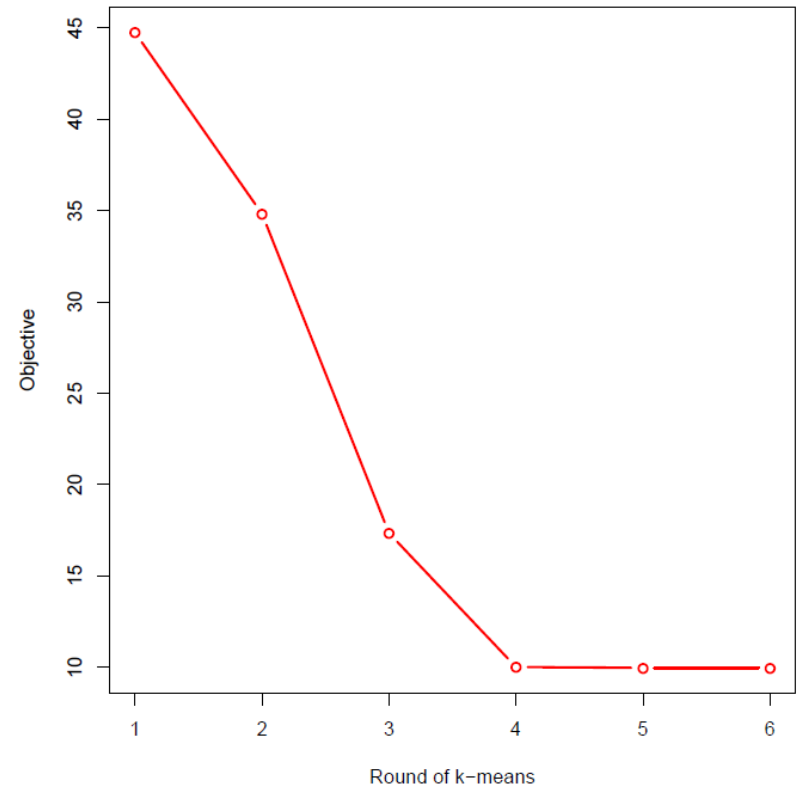
Example



Converged!

Convergence

- Note the decreasing objective from the example
- K-means takes an alternating optimization approach:
  - Optimising cluster assignments
  - Optimising cluster positions
- Each step guaranteed to decrease the objective
- So guaranteed to converge!
  - But: local optimum



Properties of k-means

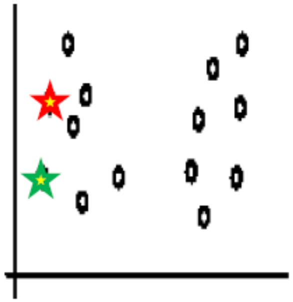
- Strengths:
  - Simple to understand and implement
  - Efficient: complexity O(NKT)
    - N = number of data points
    - K = number of clusters
    - T = number of iterations
- Weaknesses:
  - Converges to a local optimum
  - Only applicable if mean can be defined
  - K must be specified
  - Sensitive to outliers

Can you convince yourself this is right?

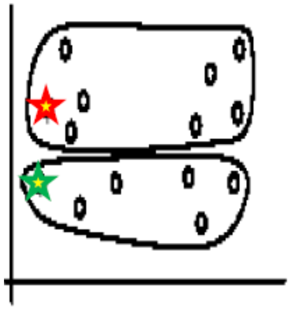
May need to use something like a median instead

Limitations

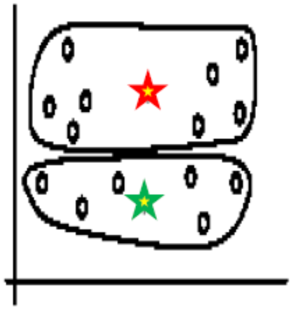
- K-means finds a local optimum
- Thus very reliant on good initialisation
- May need to restart several times



Random selection of seeds (centroids)



Iteration 1



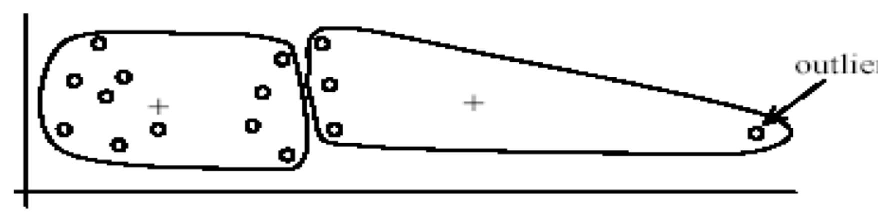
Iteration 2

Limitations

- Very sensitive to outliers
  - Points very far away from other points
- Strategies:
  - Remove outliers manually (monitor them over a few iterations first)
  - Random sampling: choose a subset of the data, less likely to contain outliers
  - Median?

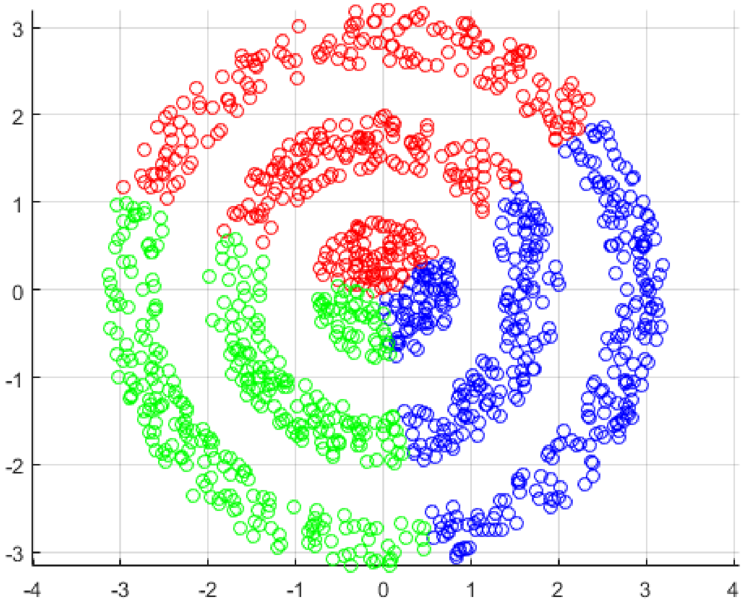
We want this:

Instead, we may get this:



Limitations

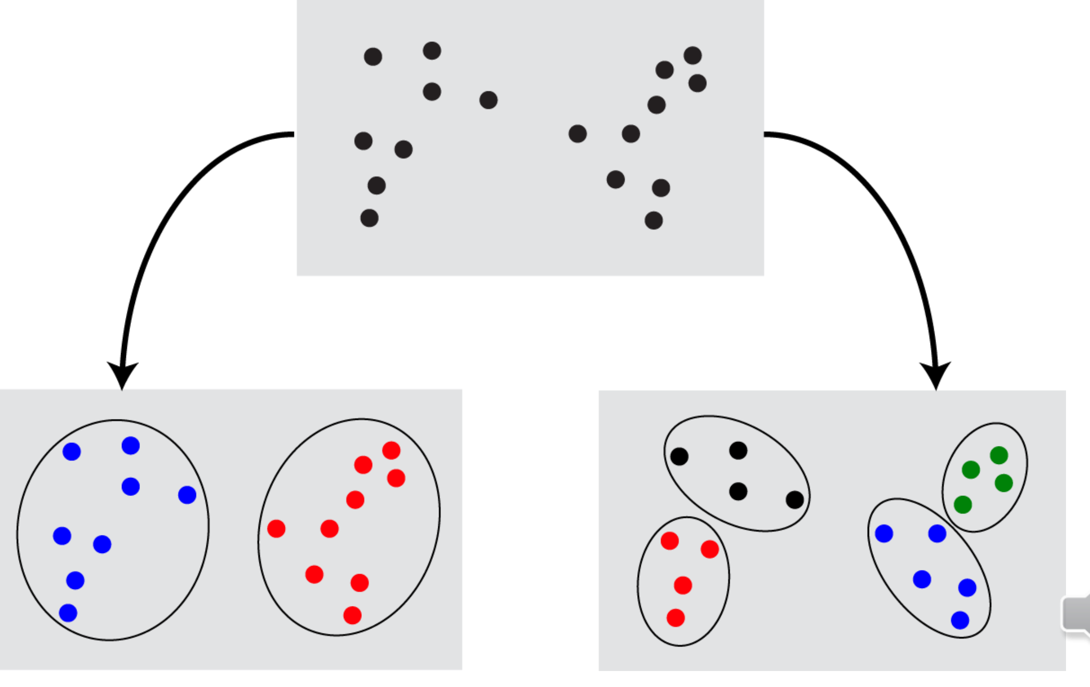
- Not suitable for clusters that are not hyper-ellipsoids



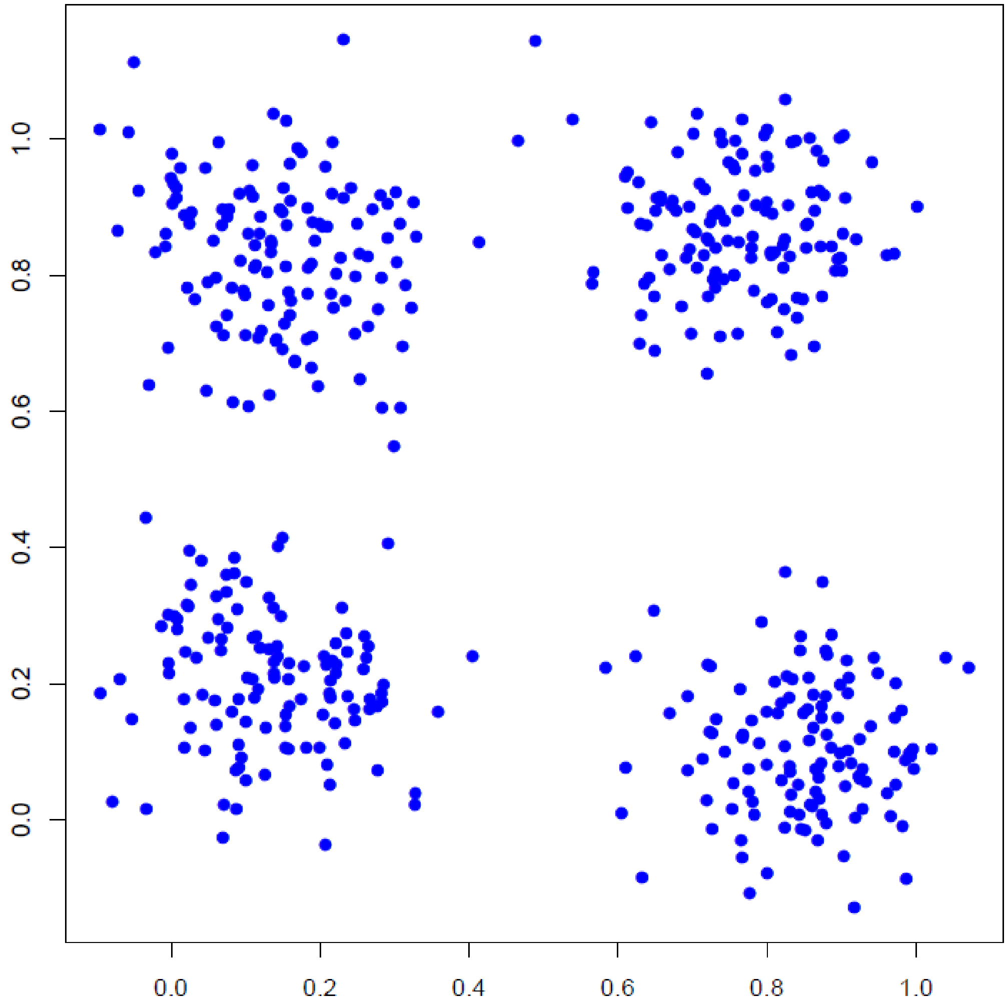
- Nonlinear features may be useful here

Choosing k

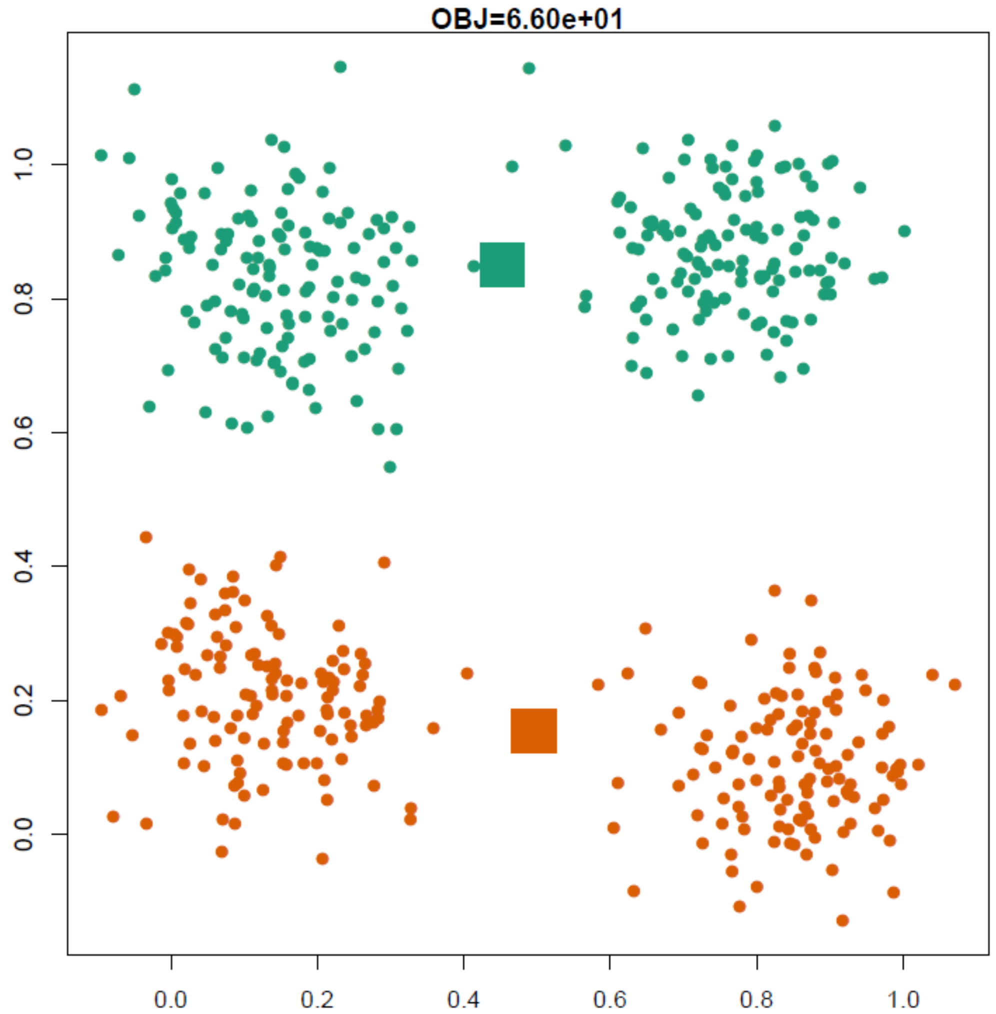
- Not always clear what the correct number of clusters is
- Often heuristics are used
  - Although there are algorithms that do this more automatically



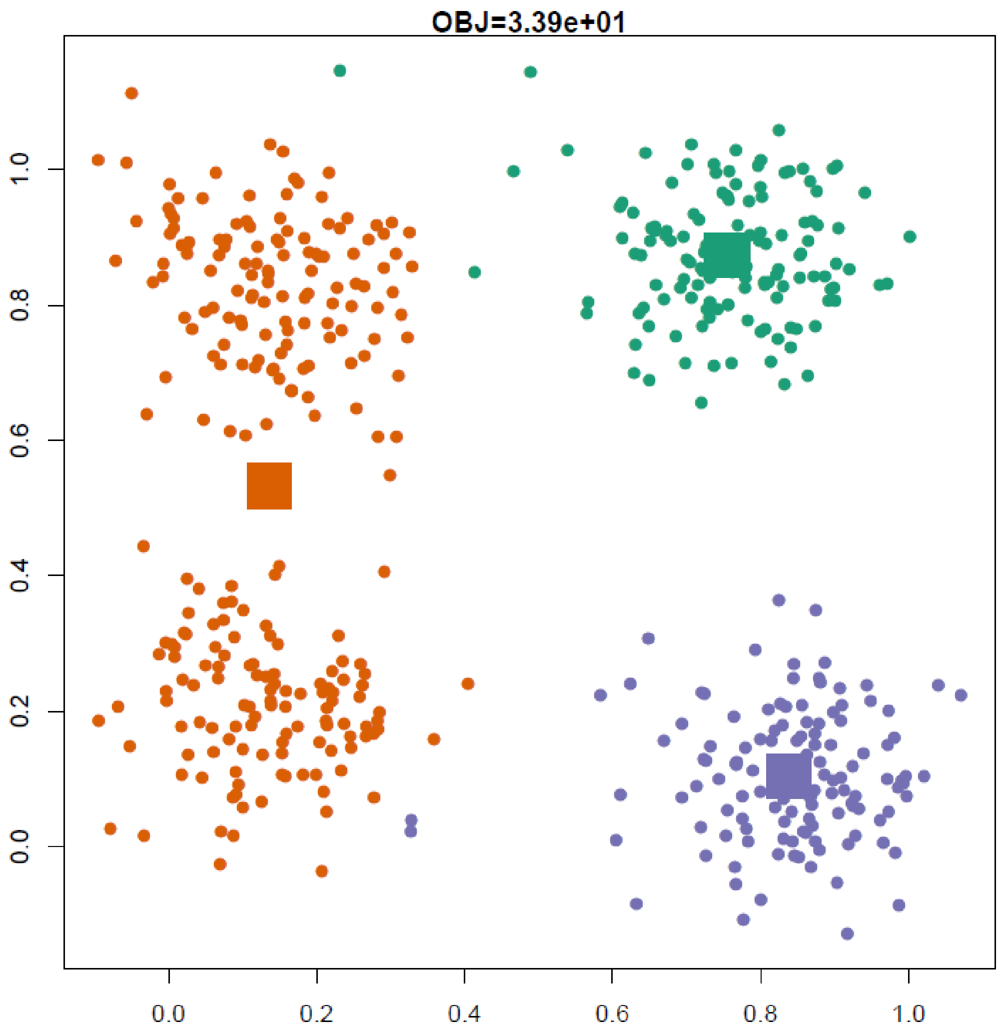
Changing values of k



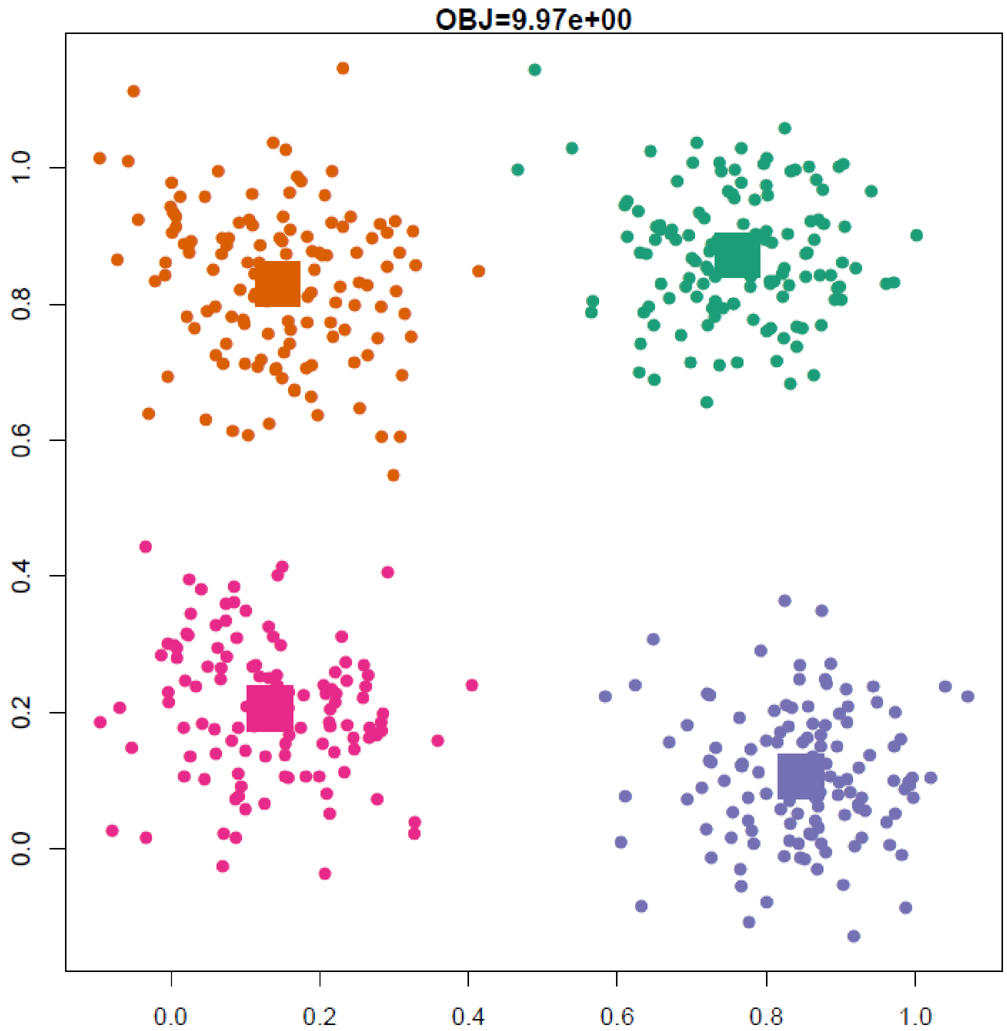
Changing values of k



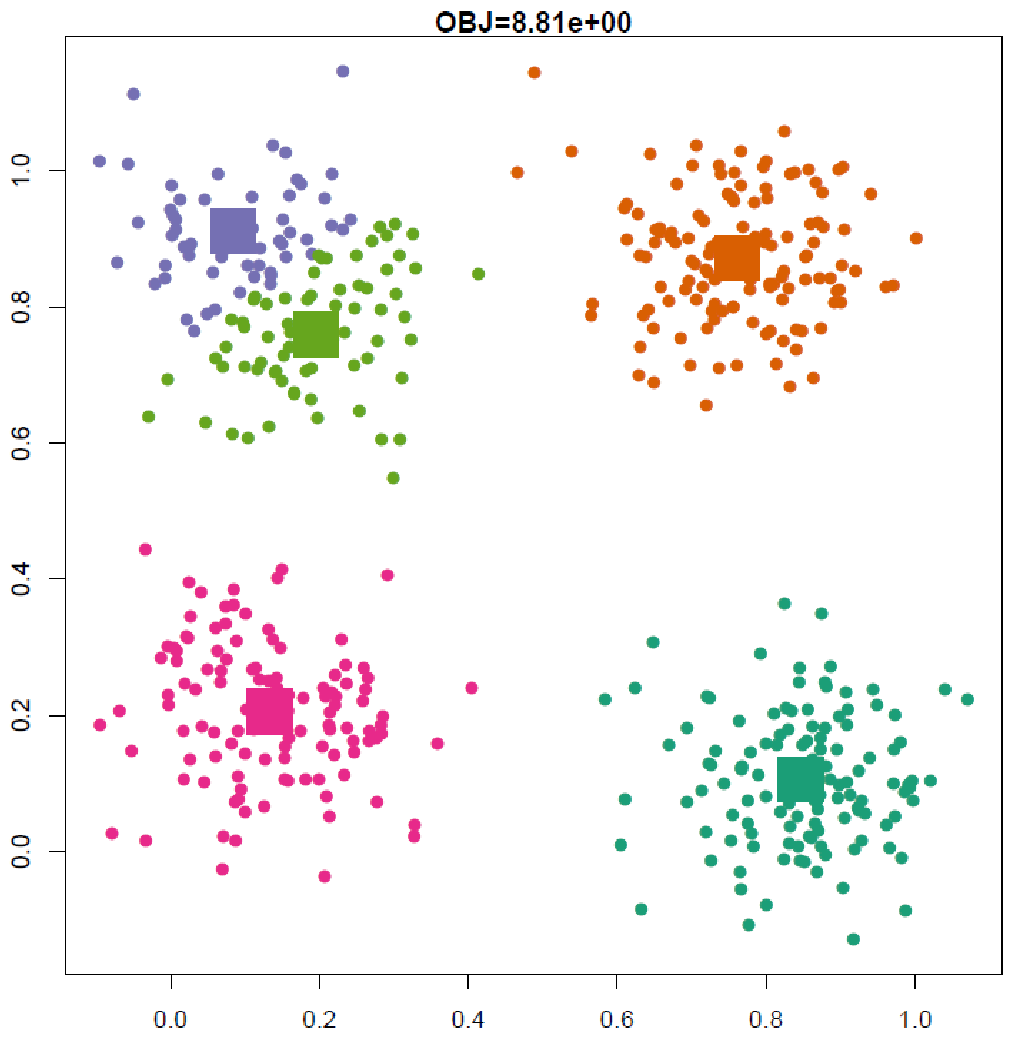
Changing values of k



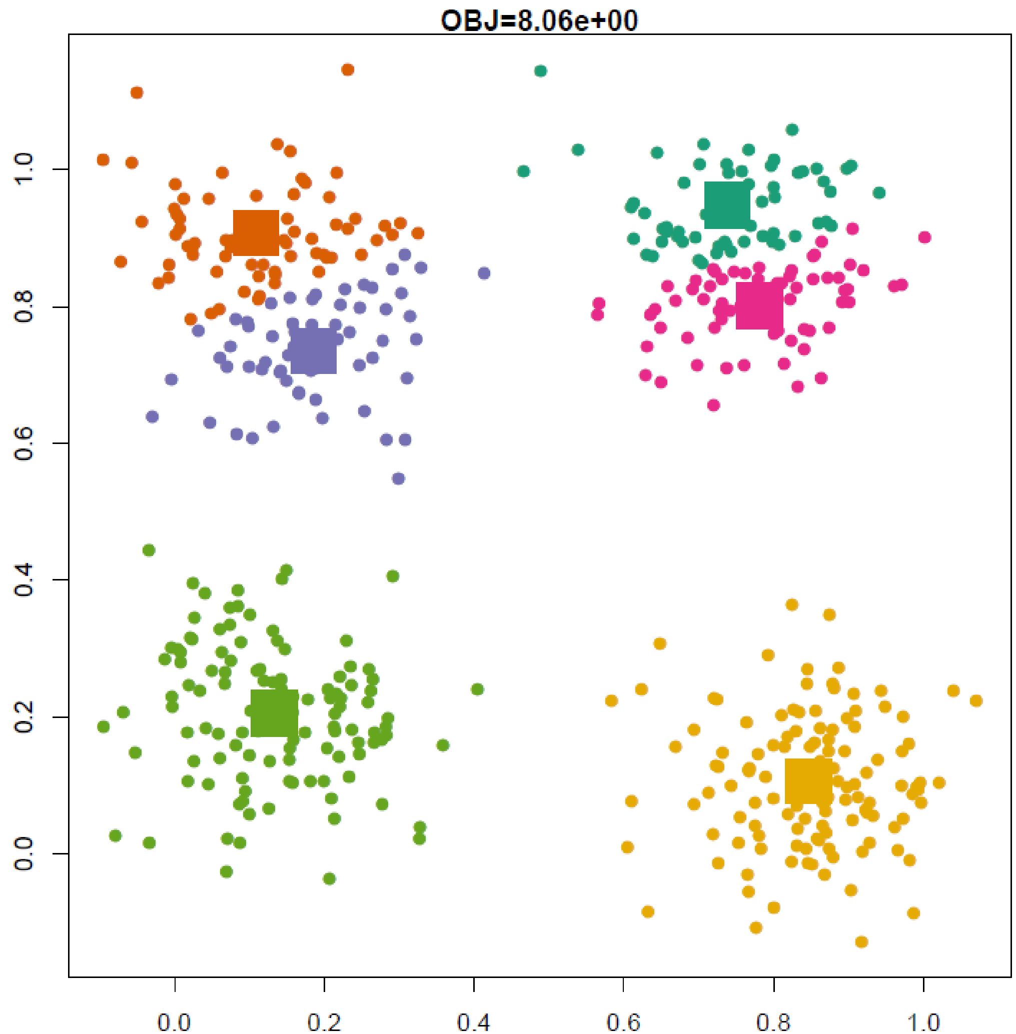
Changing values of k



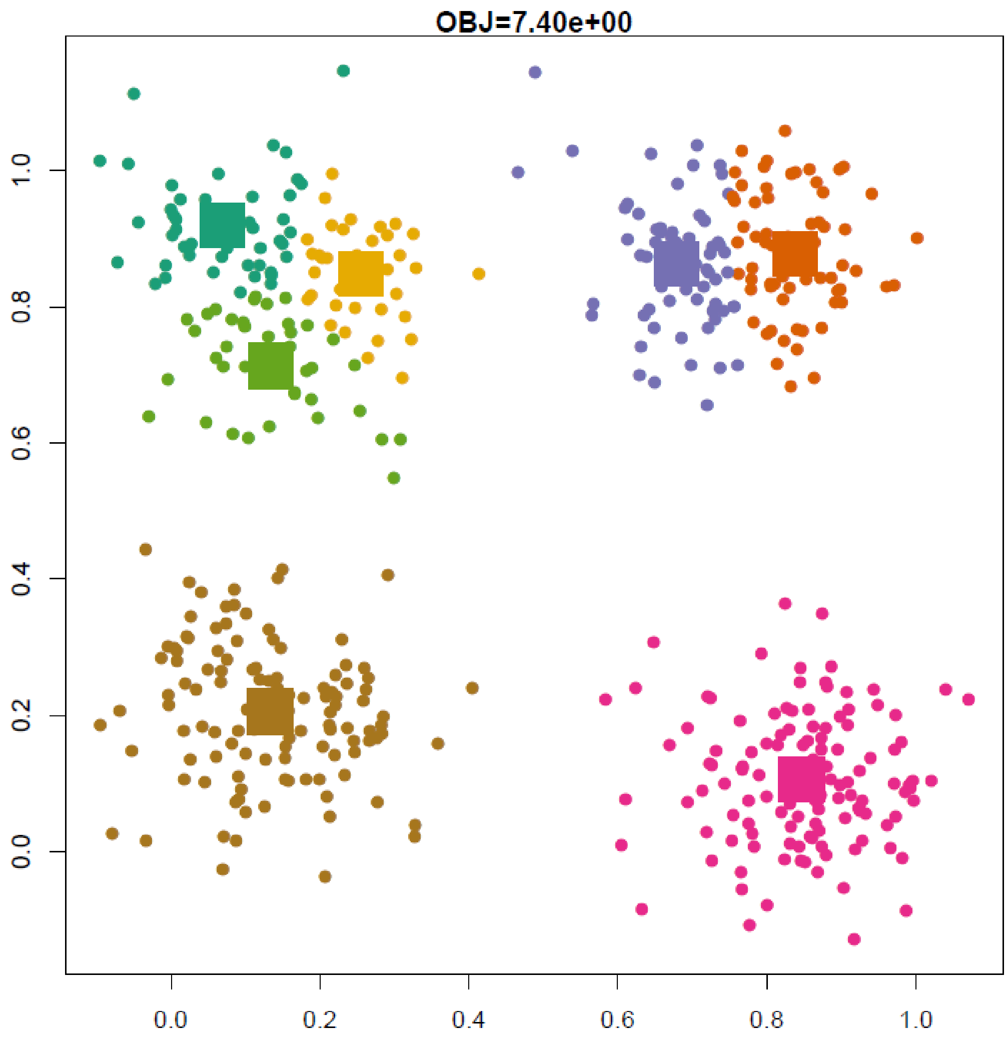
Changing values of k



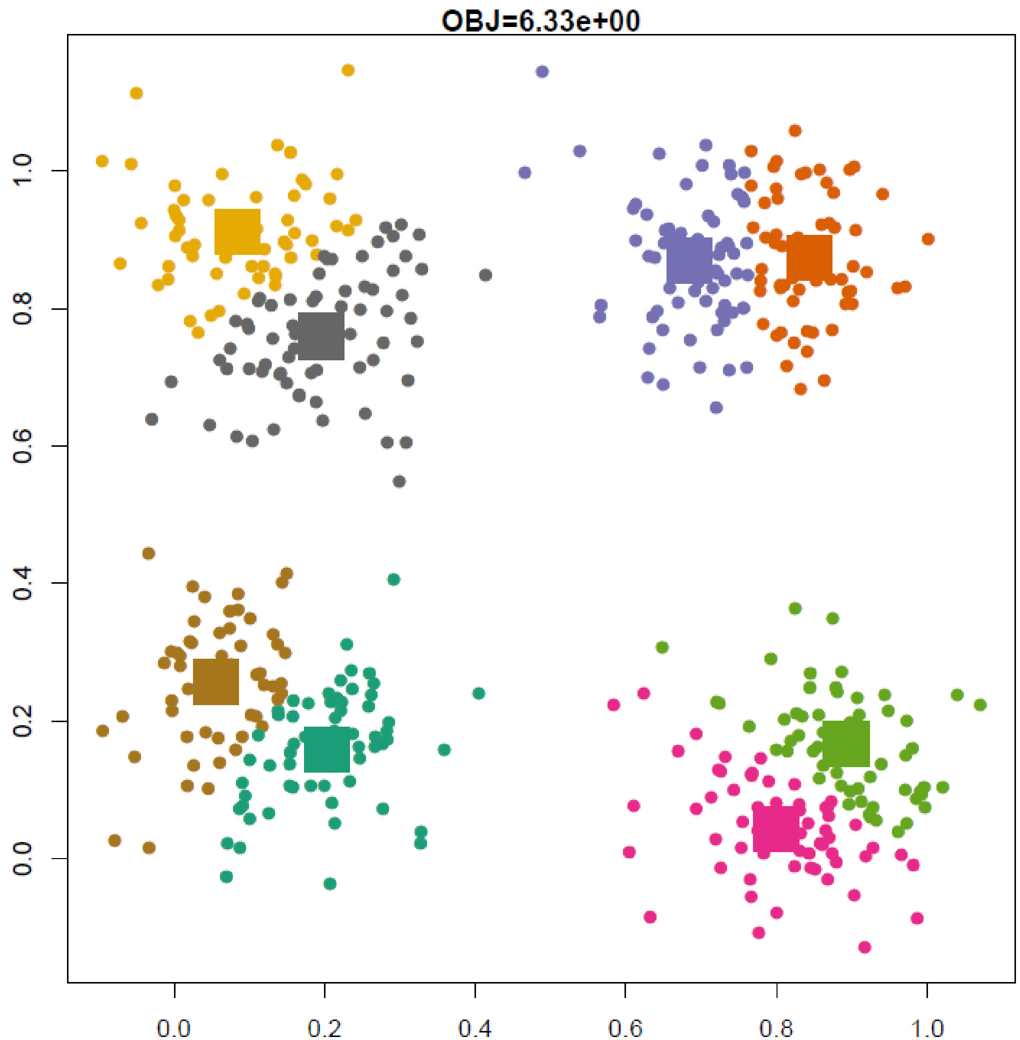
Changing values of k



Changing values of k



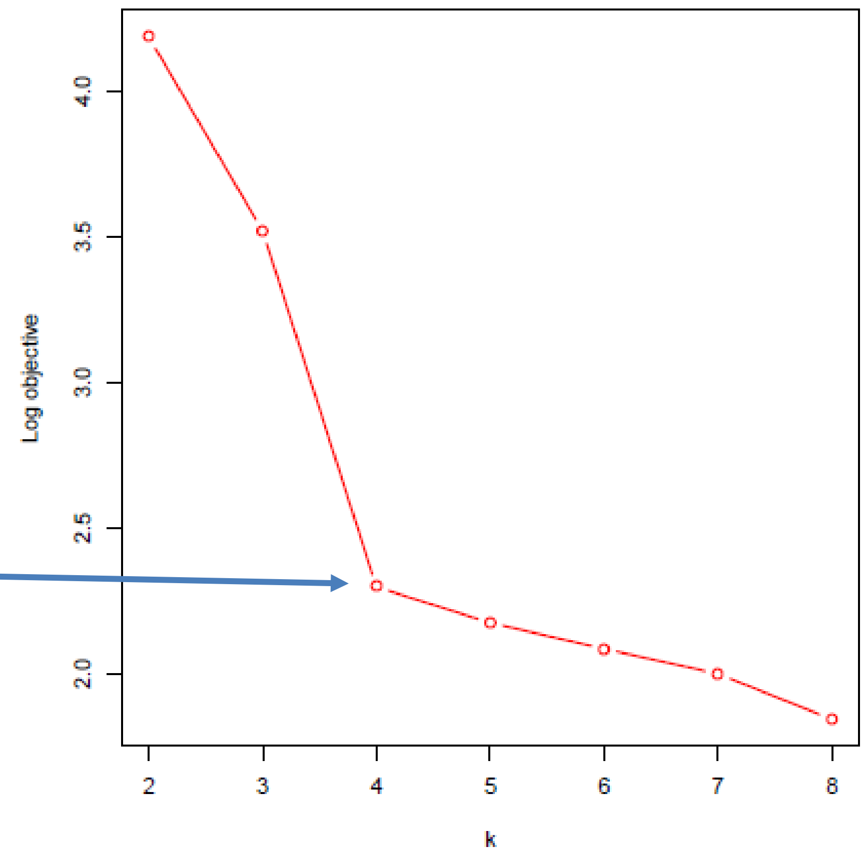
Changing values of k



Changing values of k

- A common heuristic is to look at the changing value of the objective with changing k

- The "kink" or "elbow" is often taken to indicate the best k



Running k-means online

- Online/sequential k-means
- Why? Clustering online articles as they're written

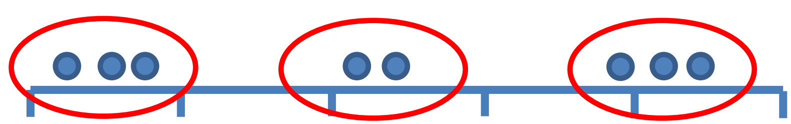
- Algorithm:

- Place centroids $c_1, c_2, \ldots, c_k$ at random locations in $\mathbb{R}^d$
- Set initial counts $n_1, n_2, \ldots, n_k = 0$
- Repeat until bored:
  - Acquire new point x_i
    - Find the closest centroid $c_j = argmin_{c_j}\, d(x_i, c_j)$
    - Assign x_i to cluster j
    - $n_j \leftarrow n_j + 1$
    - $c_j \leftarrow c_j + \frac{1}{n_j}(x_i - c_j)$

Update appropriate cluster centre by moving it closer to x_i. $\frac{1}{n_j}$ acts as an adaptive learning rate.

Applications: supervised learning

- Use clustering to discretise continuous values for supervised learning
- Instead of using a set discretisation interval
  - Cluster training data and use cluster ID
  - This can also lower the dimension of high dimensional data

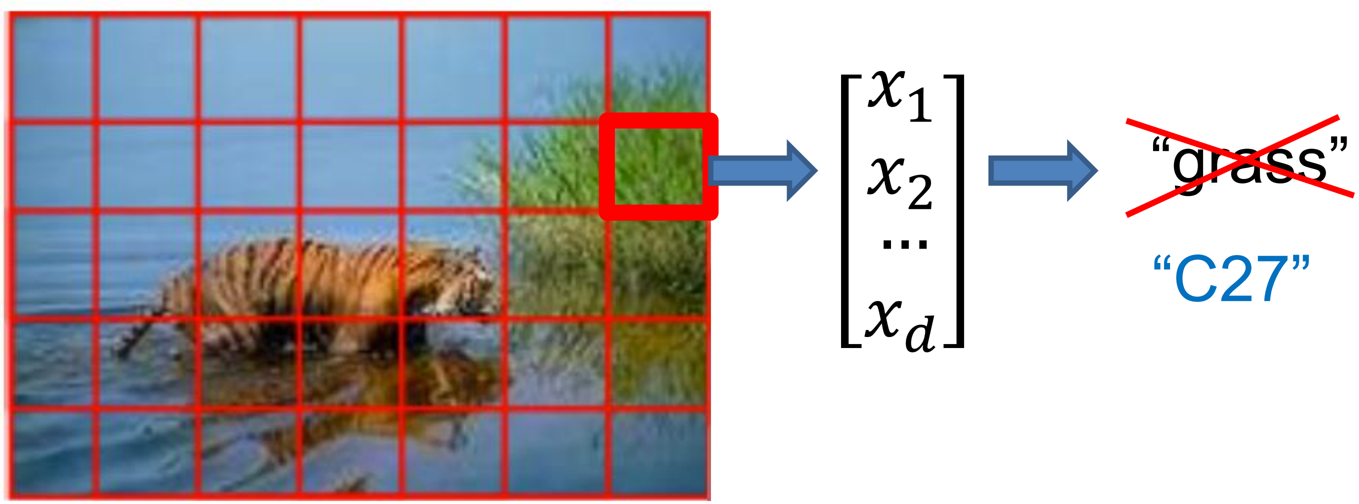


Applications: visual words

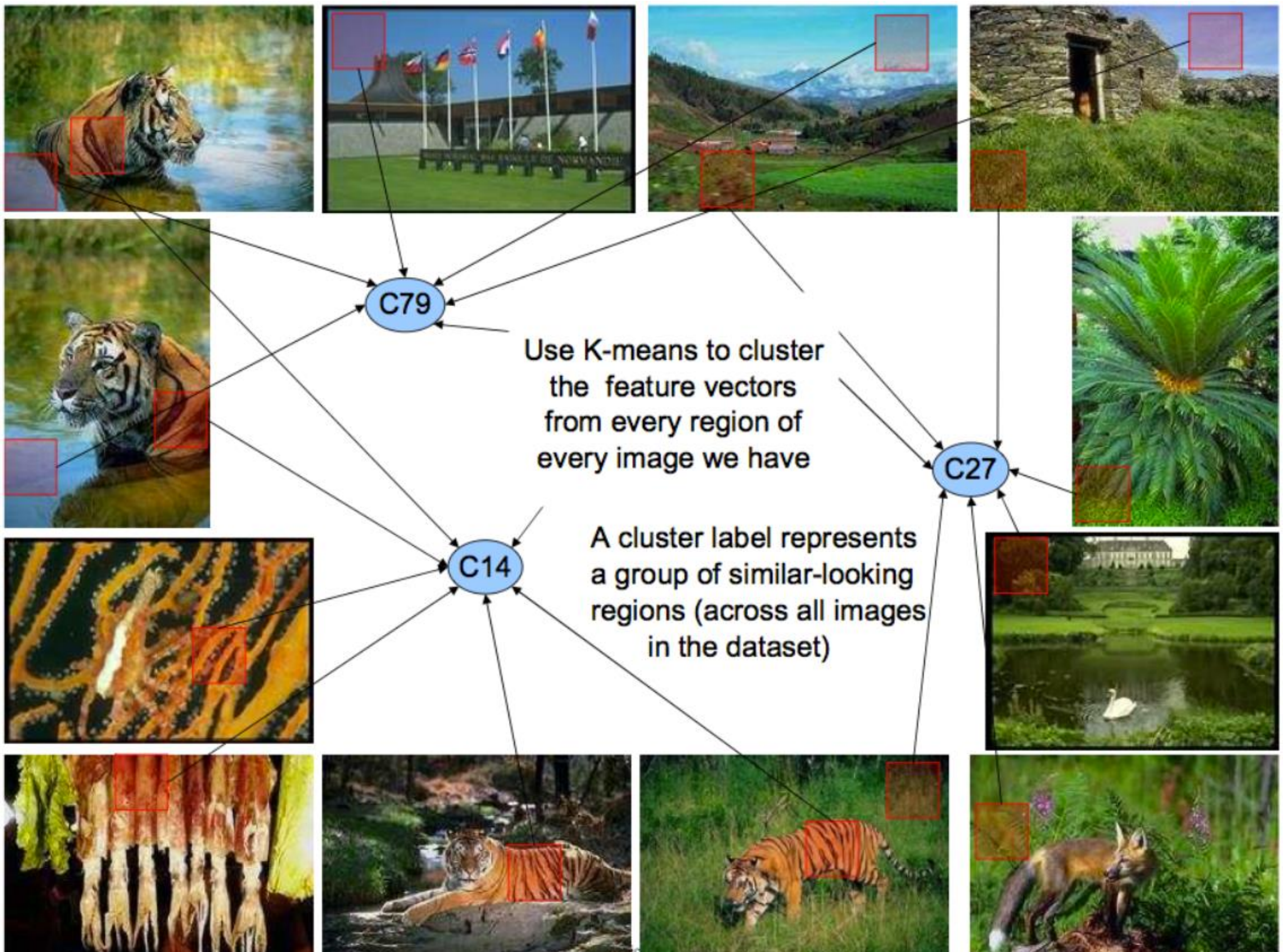
- Use a similar idea for images
- **What if you wanted to use a naïve Bayes or decision tree model to classify images?**
  - Pixels as attributes?
    - Huge space, and not useful for learning
  - Bag-of-words would be nice: {“water”, “grass”, “tiger”}
    - Needs human annotation

Applications: visual words

- Idea:
  - Break image into set of **patches**
  - Compute appearance features of each patch
    - Relative position, distribution of colours, texture, edge orientations
  - Convert to a “word” (code) to reflect patch appearance
    - Similar feature vectors → same “word”

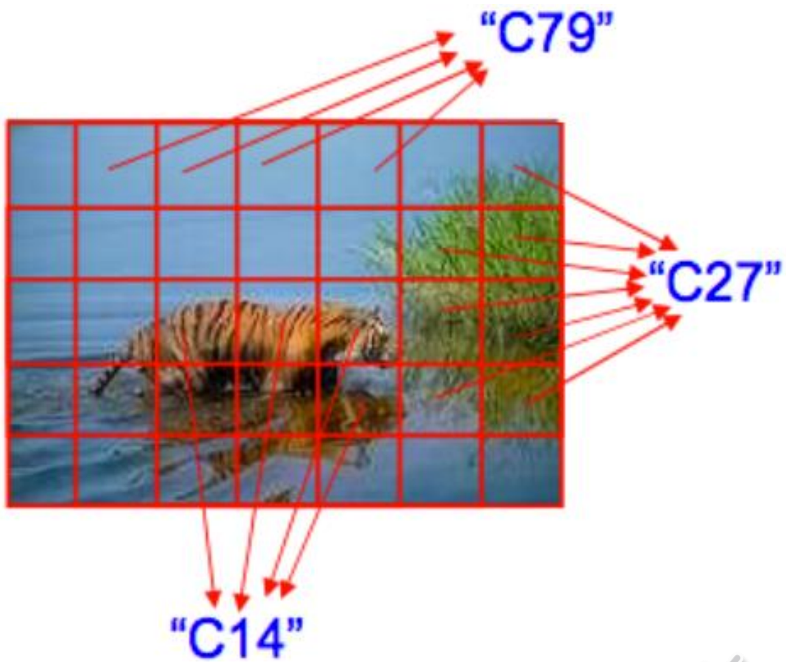


Applications: visual words



Applications: visual words

- Use k-means to:
  - Group all feature vectors from all images into K clusters
  - Provide a cluster ID for every patch in every image
    - Similar-looking patches have the same ID
- Represent patch with cluster ID
  - Image = bag of cluster IDs
  - K-dimensional representation:
  - {4 x "C14", 7 x "C27", 24 x "C79", 0 x else}

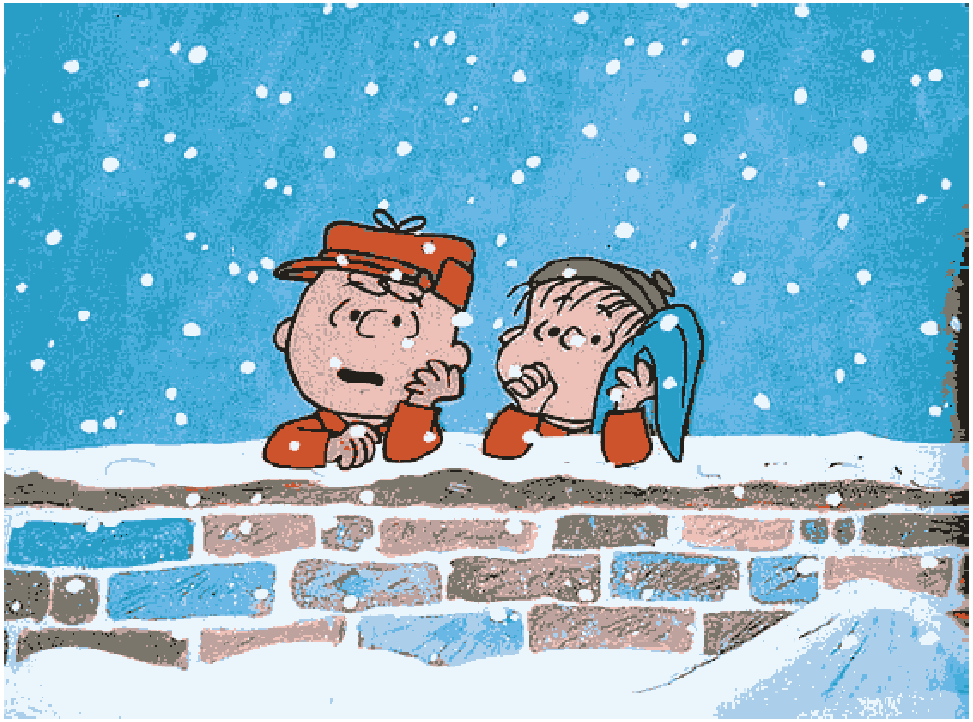


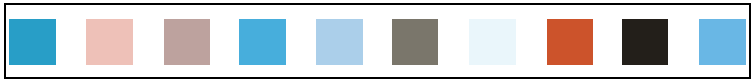
- Similar to bag-of-words
- Cluster IDs called vis-terms or "visual words"
- Plug these into a classifier

Applications: image compression

- Every pixel in an image has a red, green, blue value
- How many bits per pixel?
- We can use k-means to compress the image!

Applications: image compression





- Clustering in the colour space
- Replace each pixel x_i by its cluster centre c_{x_i}
- The k means are called the **codebook**

Applications: image compression

2 means

Applications: image compression

4 means

Applications: image compression

8 means

Applications: image compression

16 means

Applications: image compression

32 means

Applications: image compression

64 means

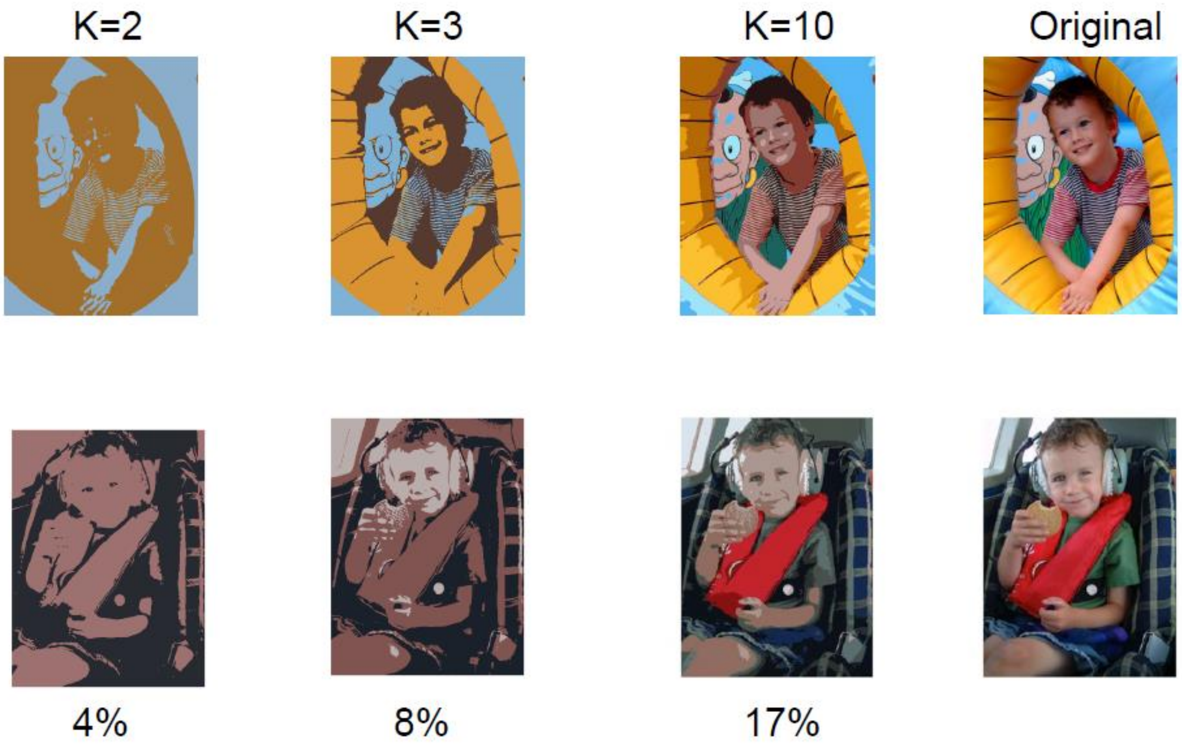
Applications: image compression

128 means

Applications: image compression

256 means

Applications: image compression



- Sometimes known as vector quantisation
- Also an easy way to do image segmentation

Recap

- Clustering and applications
- Distance metrics
- The k-means algorithm
- Limitations of k-means
- How to choose K
- Online k-means
- Representations for supervised learning
  - Visual words
- Image compression and segmentation