LAB 7

Clustering with K-means

Alex Becheru
irlab.becheru.net
irlab@becheru.net

Clustering vs Classification

Clustering = unsupervised learning

we have no knowledge of the classes

Classification = supervised learning

experts were used to classify the training set

Clustering scope

Minimise the distance between objects with similar features and maximise the distance between objects with different features.

K- Nearest Neighbour or K-means

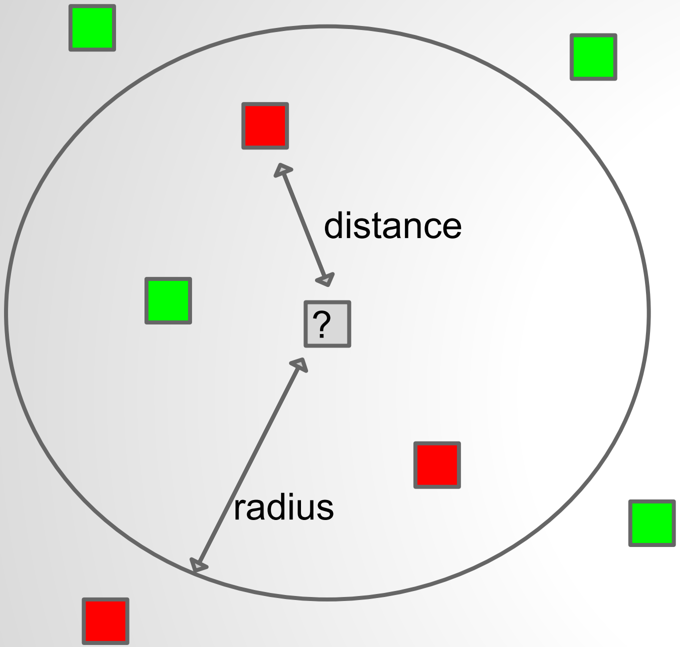
If it looks like a duck, swims like a duck, and quacks like a duck then it is probably a duck.

K-means classifies objects based on it's closest neighbours.

The white animal is among ducks, so he is probably a duck.

K-means basics

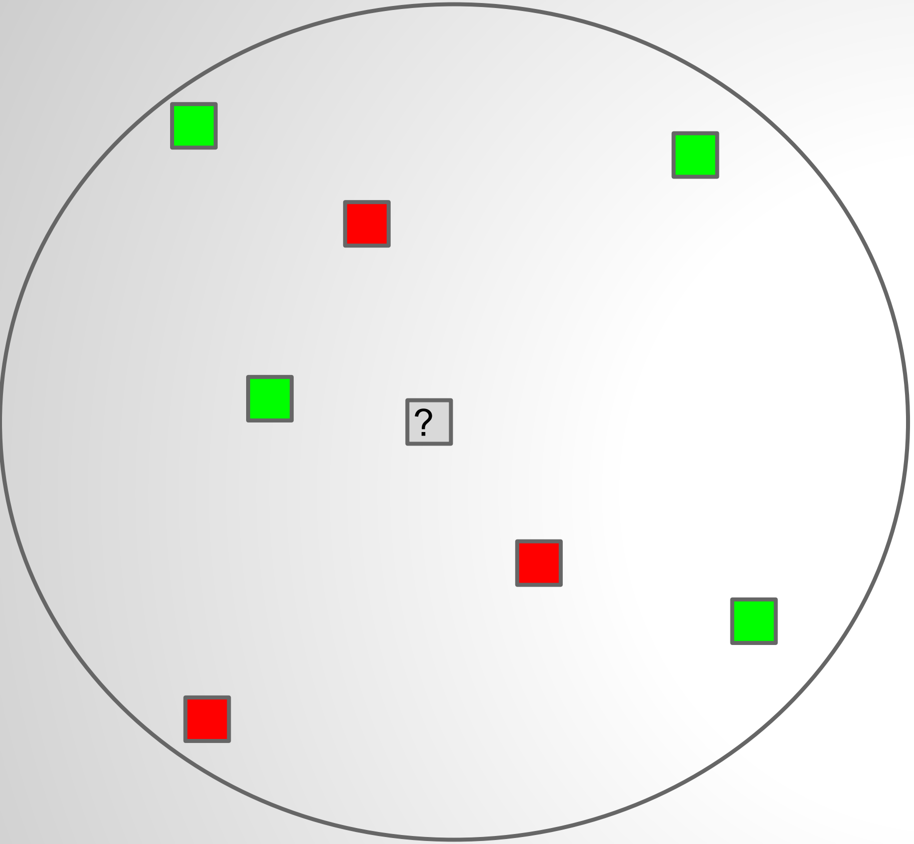
- a set of instances (training dataset)
- a distance (metric) to compute the similarity between objects
- the number of clusters: k



Naive method:

1. calculate the distance between all the training records and the new object
2. pick only the elements that have a smaller metric than the radius
3. Assign the class which is the most frequent

Can you tell what is the problem here?



The major problem comes from establishing the radius. In this case with a larger radius the object is put in a different class than before.

Solution:

1. weight the vote of a each neighbour: 1/distance^2
2. sum the votes for each class
3. the class with the biggest sum is used for the new object

Noise problems

According to k-means algorithm the “dog” is actually a little “duck”.

K-means does not identify noise. A noise object is defined as an object that has similar characteristics with the surrounding objects. But he represents a different type of objects than the ones that surround him.

Metrics

For numerical attributes: Euclidian distance
(the geometric distance)

For categorical attributes:

$$d(X_1, X_2) = \sum_i \left| \frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2} \right|,$$

X_l , l=1,2 represent the attributes
n_{li} represent the corresponding frequencies

Categorical attribute distance calculation

class	Single	Married	Divorced
Yes	2	0	1
No	2	4	1

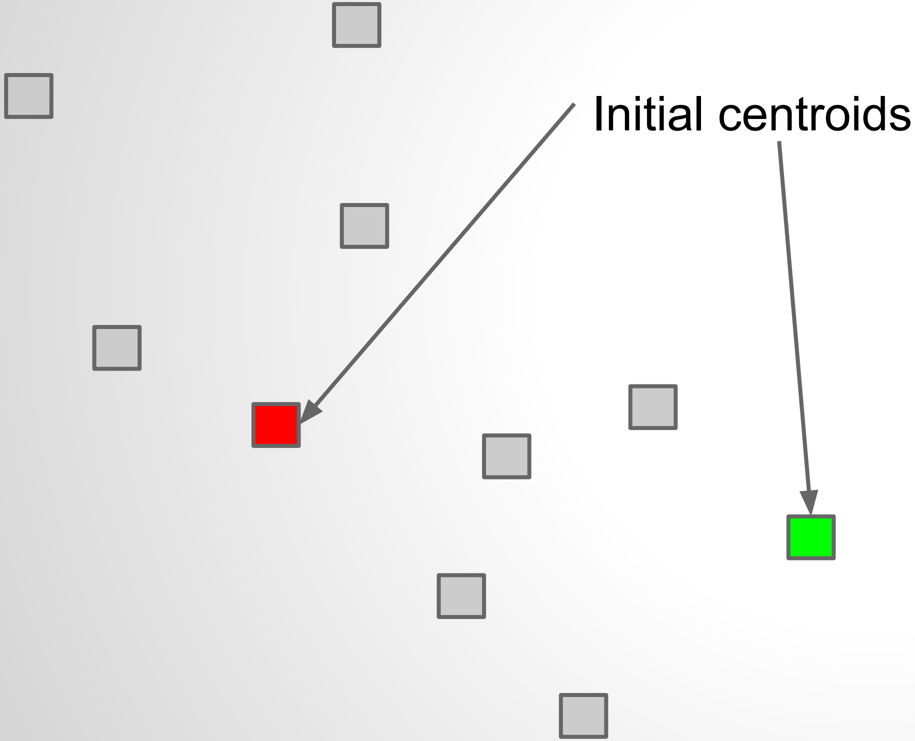
distance(Single, Married)=(single&yes / single - married&yes/married)
+ (single&no/single - married&no/married)
= (2/4 - 0/4) + (2/4 - 4/4) = 1

distance(Married, Divorced)= ??????

Algorithm description

1. use k-points as initial centroids
2. calculate all the distances between all the points and all the k-centroids
3. assign each point to it’s nearest centroid (class)
4. calculate the new centroids of each class
5. repeat steps from 2 to 5 until no object changes class

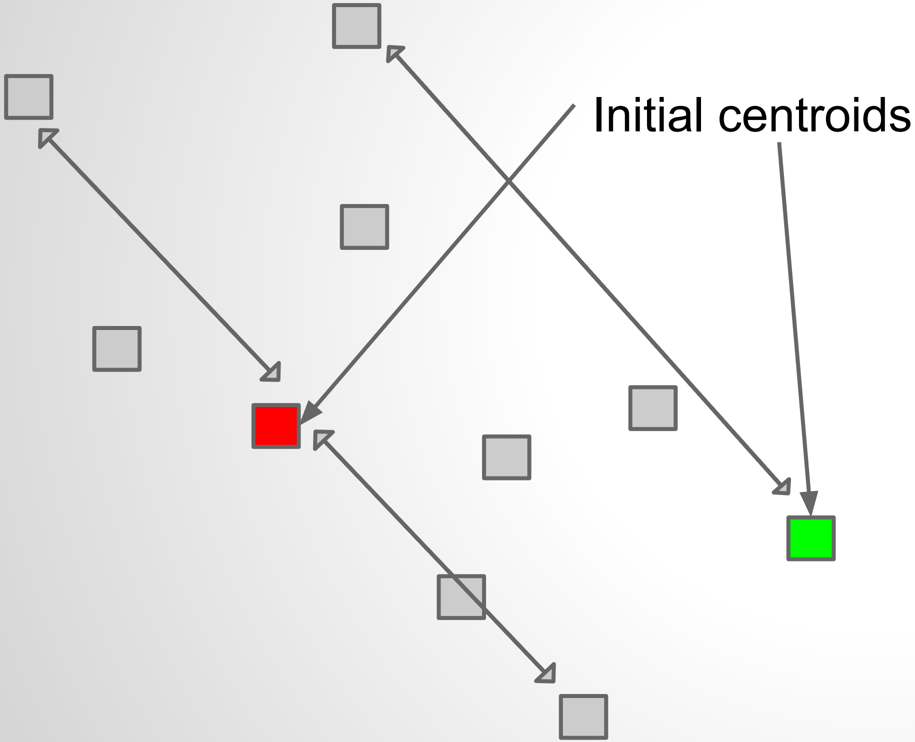
Step 1 round 1



From the objects you select k of them to be initial centroid.

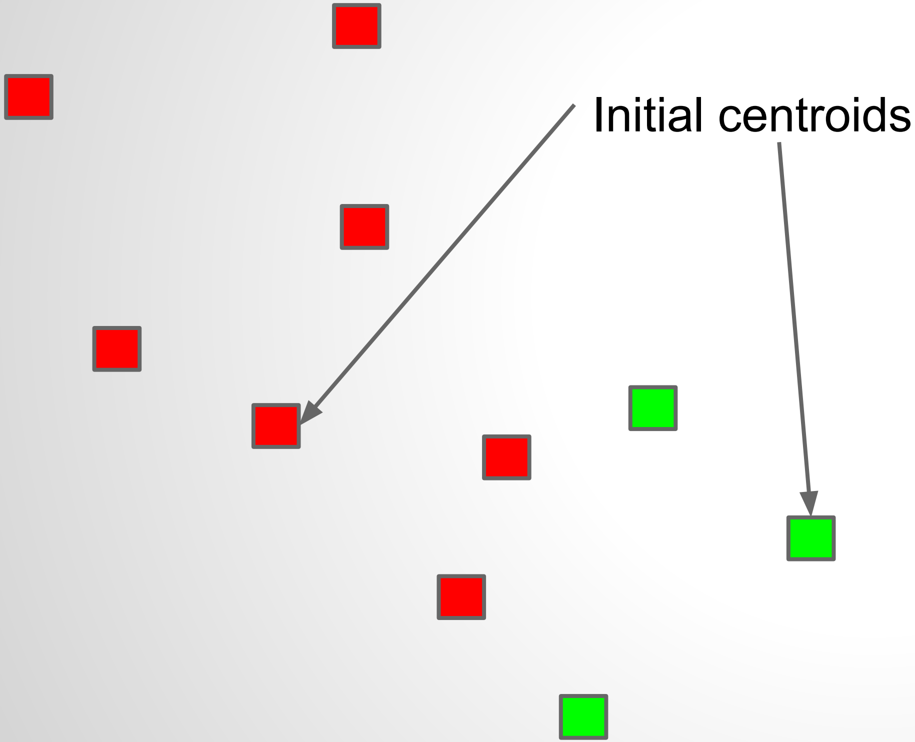
You chose as many initial centroid as the number of clusters.

Step 2 round 1



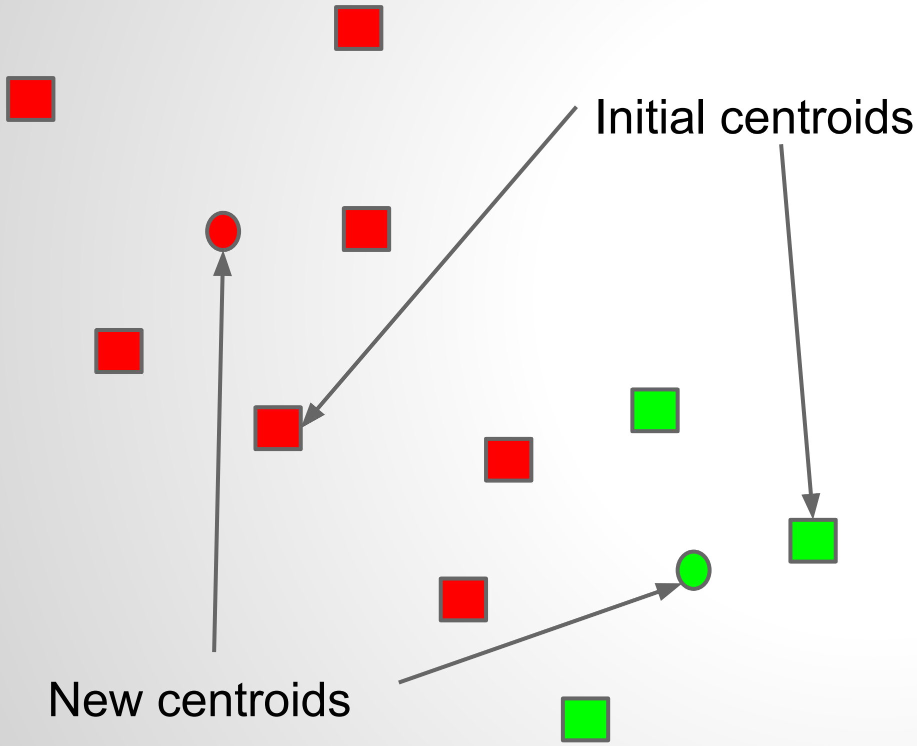
For each centroid you calculate every distances with all the other points.

Step 3 round 1



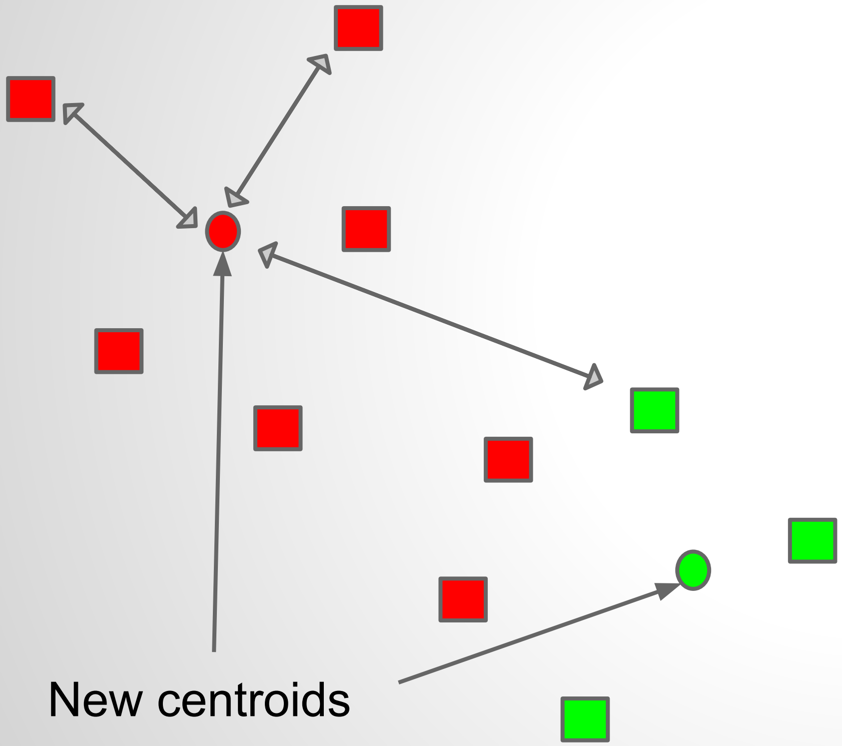
You assign to each point the class of the centroid that is closer.

Step 4 round 1



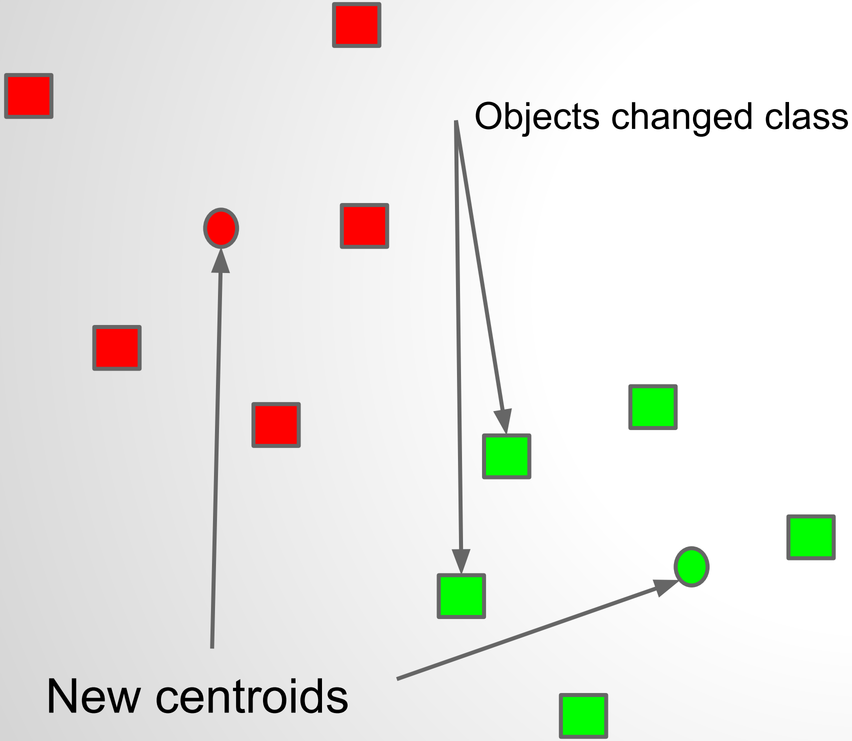
You calculate the new centroids. A centroid can be thought as the centre of gravity of each class

Step 2 round 2



For each new centroid you calculate every distances with all the other points.

Step 3 round 2



You assign to each point the class of the centroid that is closer.

In the next step centroids will be changed but in the next round no objects will change class. The algorithm will stop.

Stop conditions

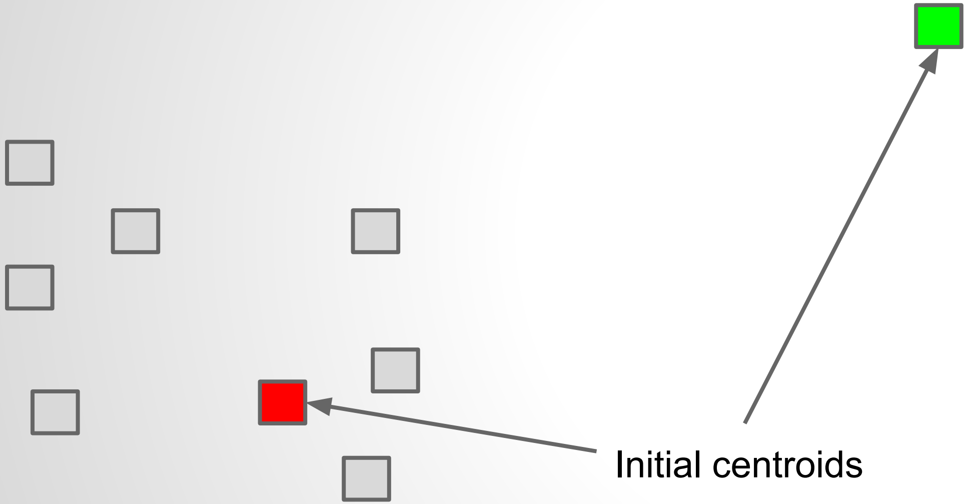
The “no change” condition is not good. On large datasets even at the 10000 round there is an objects that changes class.

Solution 1: if less than a certain percent of objects change class the algorithm will stop (1%). Hard to determine the percent as it varies from dataset to dataset.

Solution 2: if a certain number of rounds has been made. Hard to determine the number of repetitions needed.

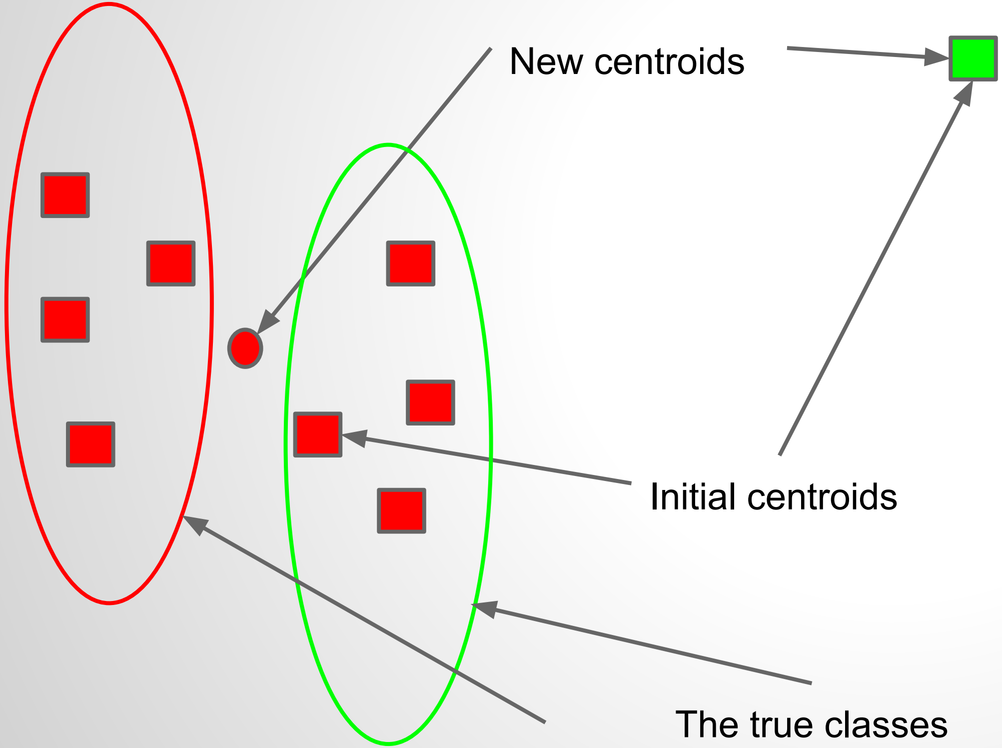
Solution 3: Combine solution 1&2.

Choosing initial centroids



Can you spot the problem here?

Although there are clearly 2 clusters here, because one initial centroid was an outlier it affected the clusterization process.



Solution:
Repeat K-means multiple times with random initial centroids. Choose the best result clusterisation.

How to determine the best clusterization?

Sum of Squared Errors

1. For each centroid you calculate the sum of the squared distances to all the nodes in that cluster.
2. Repeat step 1 for each centroid, then sum all the sums (SSE).

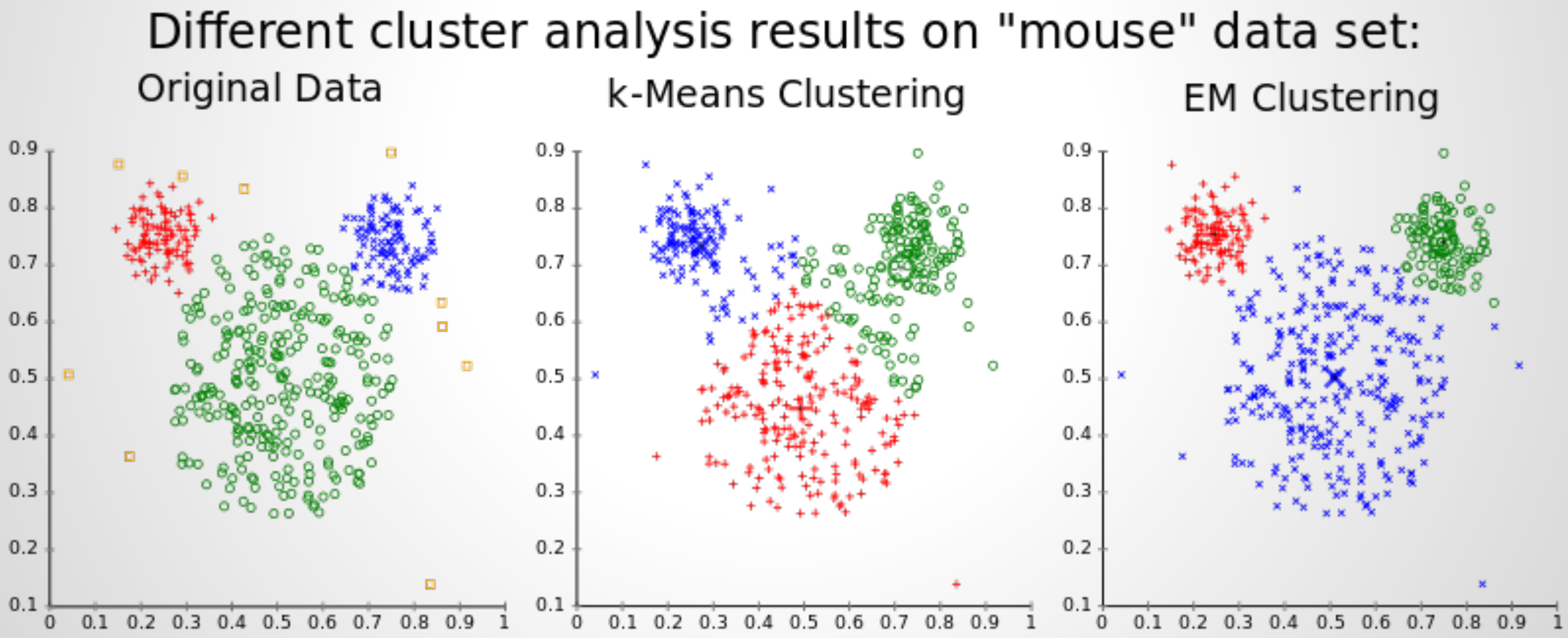
The clusterization with the smallest SSE is the best.

How to determine the number of clusters?

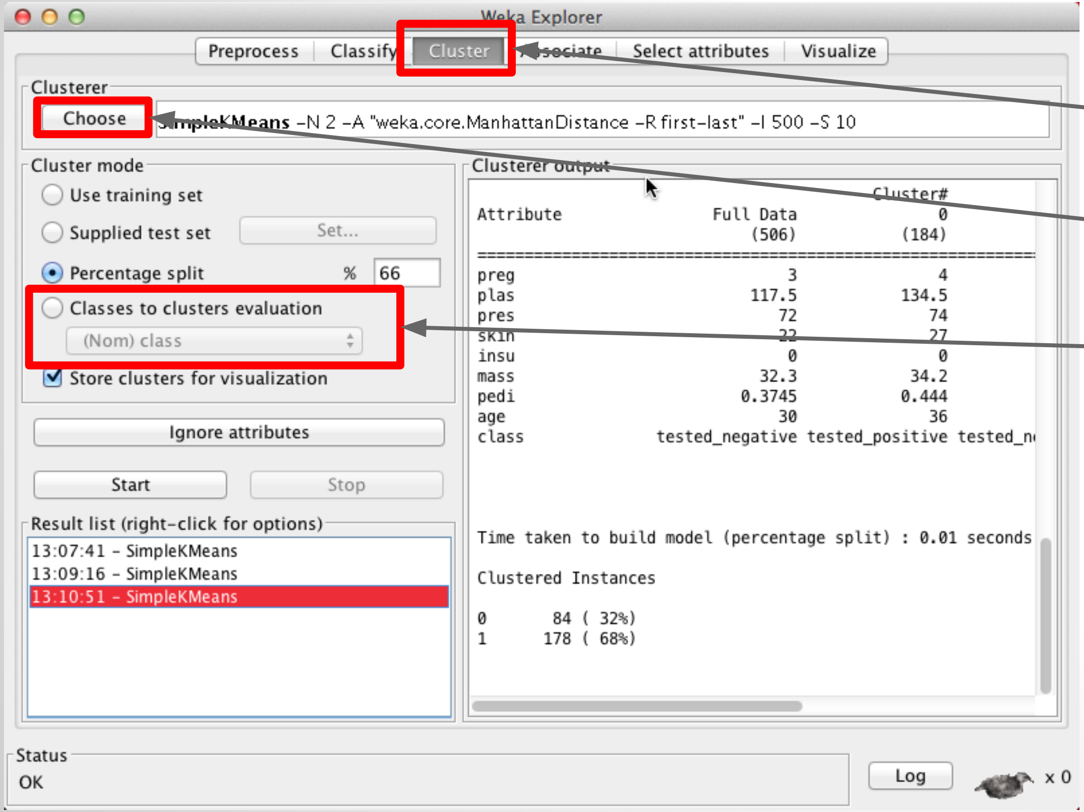
Solutions for determining K

- through visualisation
- randomly check different K and choose the one with the best result
- through hierarchical clustering

K-means enforces creation of clusters that have the same size. Which in the real world is a problem.



Clustering in Weka

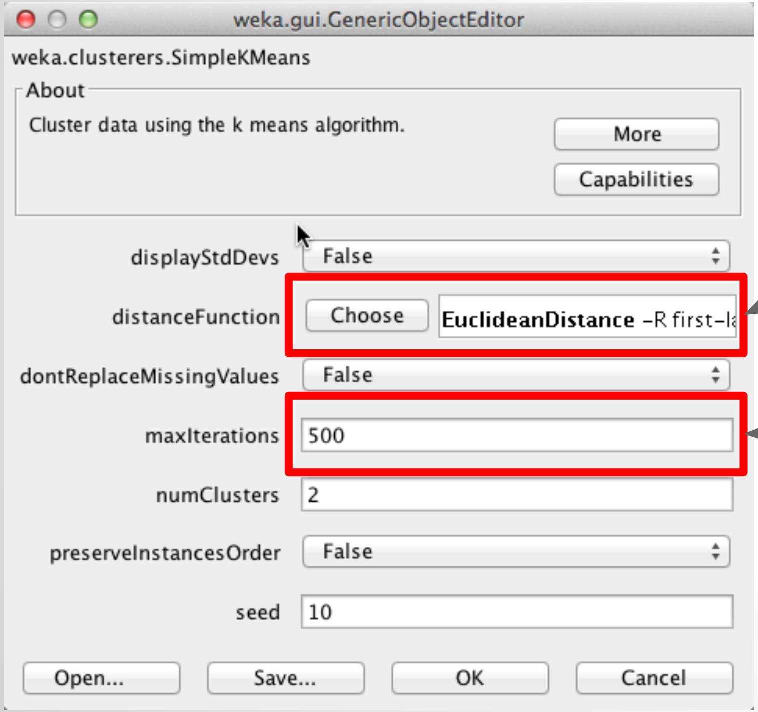


Cluster tab in Preview

Choose an algorithm

Evaluate the clustering

SimpleKmeans algorithm



Choose distance

Maximum number of iterations

Read the output of the algorithm

```
Number of iterations: 7
Within cluster sum of squared errors: 12.143688281579722
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                 (150)      (100)       (50)
============================================
sepallength     5.8433      6.262      5.006
sepalwidth       3.054      2.872      3.418
petallength     3.7587      4.906      1.464
petalwidth      1.1987      1.676      0.244
```

Number of iterations &
SSE

Centroid value for each
cluster over each attribute

```
=== Model and evaluation on training set ===

Clustered Instances

0      100 ( 67%)
1       50 ( 33%)


Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
  0 50 | Iris-setosa
 50  0 | Iris-versicolor
 50  0 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-setosa

Incorrectly clustered instances :      50.0     33.3333 %
```

Clusters size

Confusion matrix

Performance of the algorthm

Questions?