MACHINE LEARNING FOR DATA MINING
LECTURE 5: CLUSTERING

K-MEANS CLUSTERING

Siamak Sarmady (UUT, UU)
Sebastian Thrun, Katie Malone (Google)

At the Heart of Facebook's Artificial Intelligence, Human Emotions

Facebook Inc. doesn't yet have an intelligent assistant, like the iPhone's Siri. But the social-networking company says it's aiming higher, in what has become one of the biggest battles raging between Silicon Valley's behemoths: How to commercialize artificial intelligence.

The once-niche field is aimed at figuring out how computers can make decisions on a level approaching that of human intelligence. Apple Inc.'s Siri, Microsoft's Cortana and Google Inc's Google Now are all early manifestations. They are voice-recognition services that act as personal assistants on devices, helping users search for information–like finding directions or rating nearby restaurants. Both "learn" from their users, adapting to accents, for instance, and learning from previous searches about users' preferences. Facebook thinks it can do better.

"Siri and Cortana are very scripted," says Yann LeCun, director of artificial-intelligence research at Facebook, in an interview. "There's only certain things they can talk about and dialogue about. Their knowledge base is fairly limited, and their ability to dialogue is limited," he said. "We're laying the groundwork for how you give common sense to machines."

Facebook's LeCun also sees promise in natural-language processing–machines understanding what is being said in speech in a more sophisticated way than Siri or Cortana. And he said image and video recognition is the "next frontier" at Facebook. "It's clear that there's going to be a lot of progress in the way that machines can understand images and activities in video; personal interactions in video between people expressing emotions, and things like that," he said. A raised eyebrow might mean many different things in different contexts. After a computer shifts through reams of images of people raising an eyebrow, and what happens before or after, it can start to correlate that action. The basic theory is that the more images the computer analyzes and correlates, the more precise it becomes, statistically. The goal is to approach the same level of correlation that the human brain makes as it processes images sent from a person's eyes. "It's not just about looking at your face to determine your emotions, it's about understanding interactions between different people and figuring out if those people are friends, or angry at each other," LeCun said.

French-born LeCun, 55 years old, is one of the world's leading figures in artificial-intelligence research, specifically of a subset of the science called "machine learning," or mathematical algorithms that adjust, and improve, as they receive and analyze new data. While working at AT&T in the late 80s and 90s, Mr. LeCun developed handwriting recognition processing that was eventually used by banks to scan and verify checks.

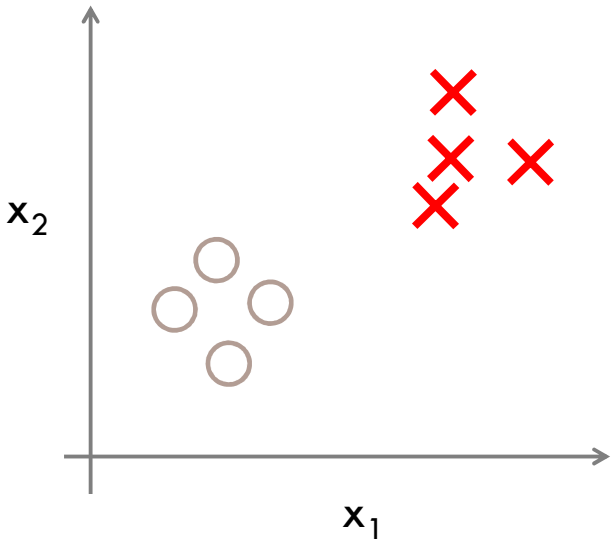
http://blogs.wsj.com/digits/2015/05/01/at-the-heart-of-facebooks-artificial-intelligence-human-emotions/

Unsupervised Learning - Clustering

- Most of the time the data we gather from the world does not include labels (or desired classes). Still we want to find some structure in the data and understand it better.
- We may use unsupervised learning algorithms to find whether the data points can be grouped into two or more clusters or not. If such groups are found, then we can guess that the data points in each group are somehow related.
- We only adjust the algorithm parameters in a way that the inputs are clustered into separate groups based on specific similarities.
  - Attributes we take into account
  - Number of clusters
  - Distance measures and its calculation details (scaling, distance calculation method)

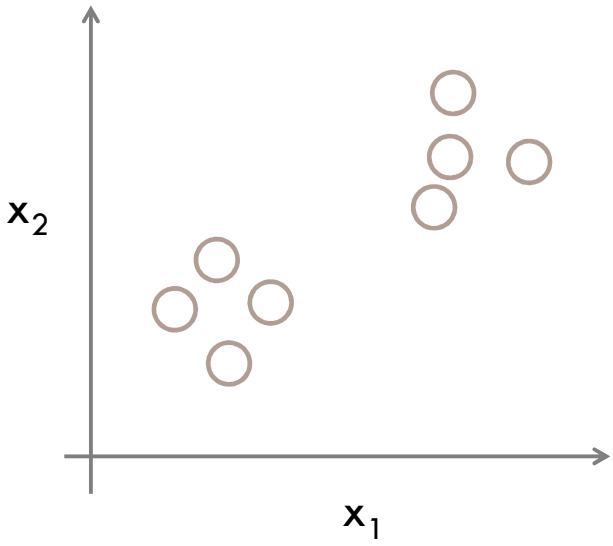
Supervised Learning (reminder)

- In supervised learning, the training data set provided the correct "labels" (i.e. classes) for individual data (e.g. O and X here).
- What if we don't have labels for the data points?



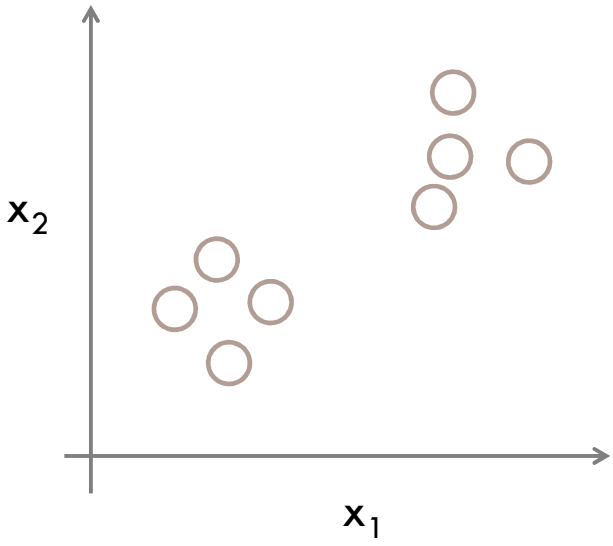
Unsupervised Learning (reminder)

- In unsupervised learning we do not have labels that determine the class of each data item.
- However, if we can visualize the data points, we might see that they are organized in identifiable groups.



Unsupervised Learning (reminder)

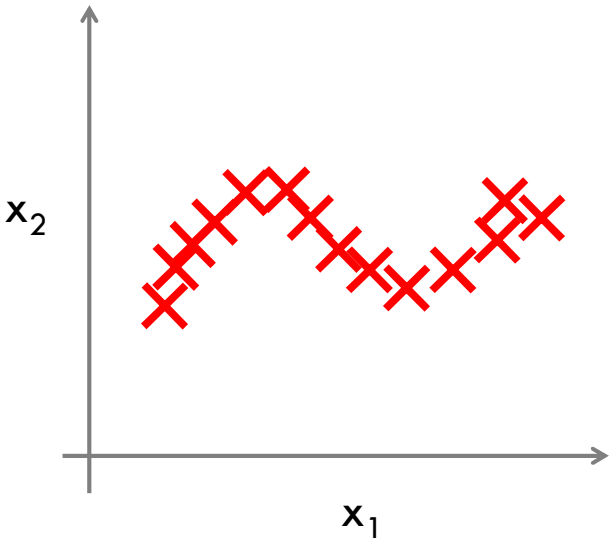
- Now if we have a learning algorithm that can identify these groups and cluster them, we may be able to understand the data better.
- The clustering algorithm might decide that in this data there are two clusters of data…



Unsupervised Learning (reminder)

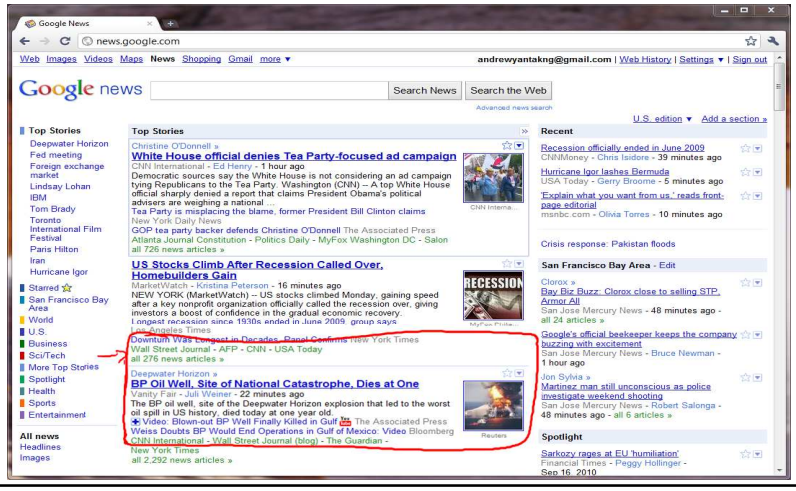
- Unsupervised learning is not just for identifying clumps (clusters) of data.

  We may find different patterns with unsupervised learning too.

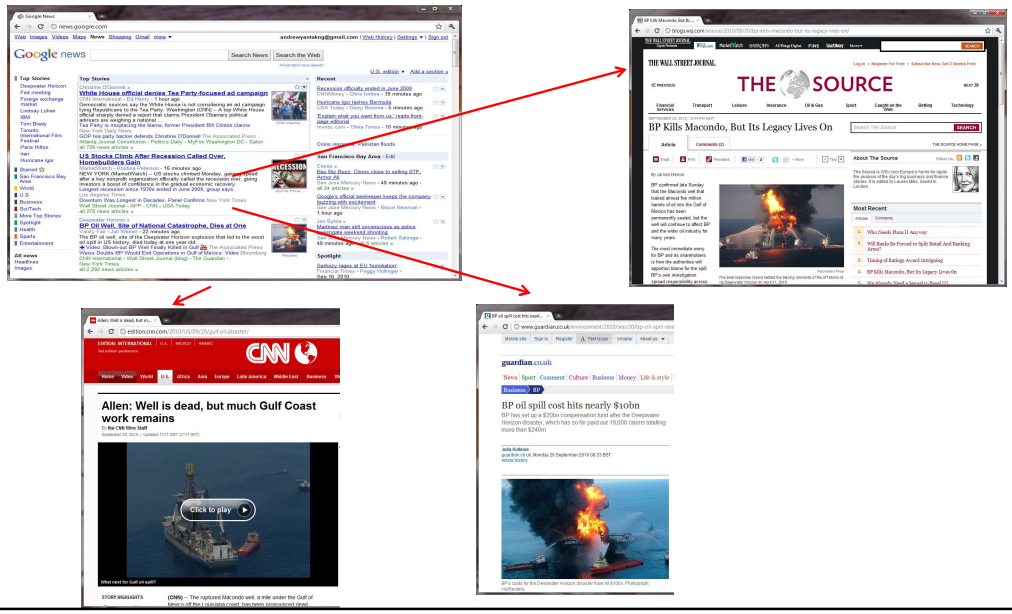


Clustering Applications (News Aggregation)

- One example of clustering is used in Google news. News items with the same subject are clustered into separate subjects (headlines). No label or supervision is provided … It just recognizes clusters of news…

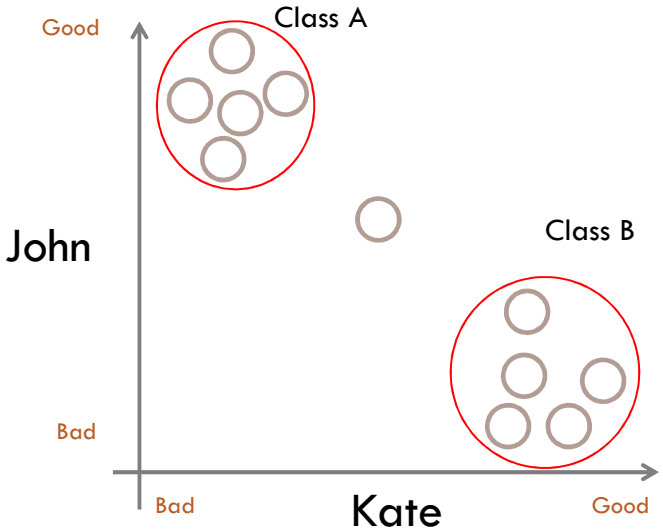


Clustering Applications (News Aggregation)



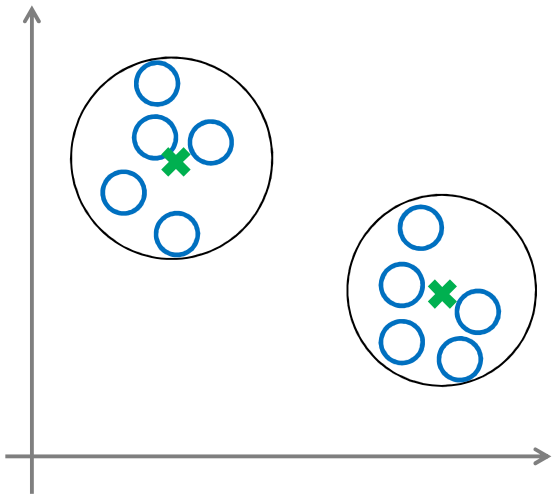
Clustering Applications (Recommender)

- Assume John and Kate look at a series of films and rate them from good to bad…
- If we do cluster analysis, we find out that John and Kate like films that fall into specific clusters or classes (A and B here) …
- Now if we want to suggest films to Kate or John, we can select films that fall within those clusters…
- What features do you suggest we use for clustering?



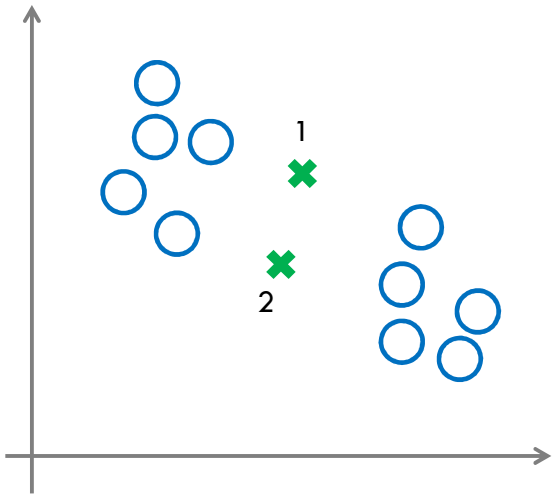
Clustering

- In the diagram, how many clusters do you see?
- In fact we cannot be sure, it might be one, it might be nine or any number in between…
- Based on common sense we would identify 2 clusters
- The green X determines the centers of the two clusters… These are the places we want to find to characterize the data



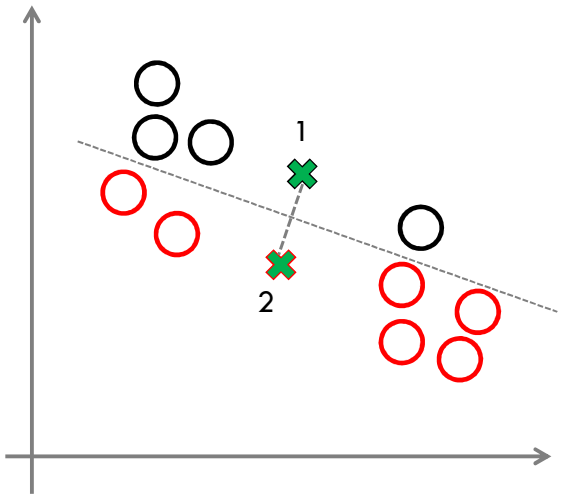
K-Means

- K-Means is the most important and common clustering algorithm.
- Assume we want to cluster the data into two classes…
- Step 0: In K-Means we first select two initial random cluster centers…
- After that K-Means operates in two steps:
- Step 1- Assign
- Step 2- Optimize



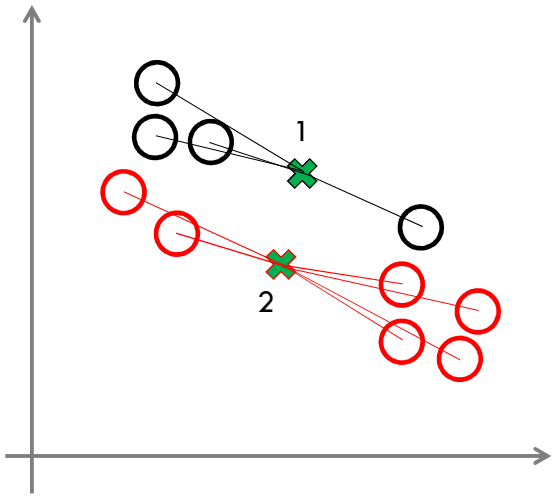
K-Means - Assign

- For each of the data points, let's guess, which of the points are nearer to each of the specified centers… (using distance measures)
- If we want to show this we can draw a line between the two center points and also a perpendicular line which divides the space into two half spaces…
- Now we assign the points to each of the two cluster centers (based on the cluster center which is nearer to the data point)



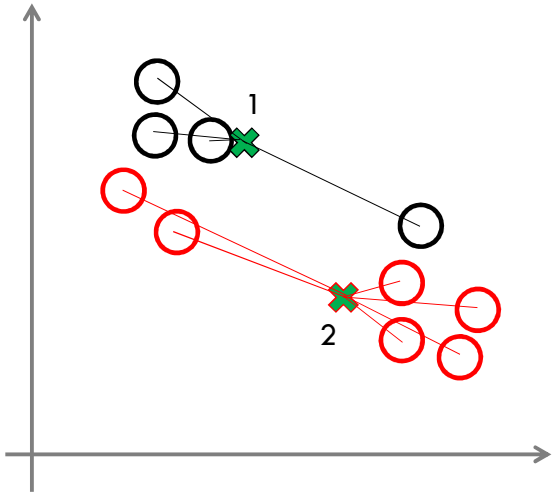
K-Means - Assign

- What we did until now was to assign each point to each of the clusters… which obviously is not enough
- Now we should perform the optimization step…
- What we do in optimization step is to minimize the total quadratic distance of our cluster centers to data points.



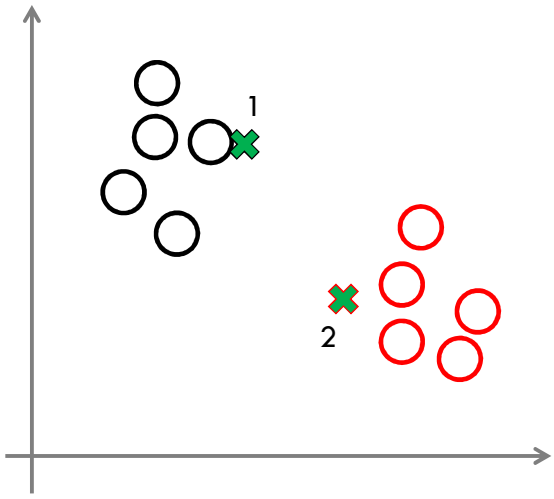
K-Means - Optimize

- We optimize the place of each of the centers in a way that the distance between the centers and their data points becomes minimum…
- To get the concept, assume the lines are rubber bands. The center will be drawn towards a point that has higher concentration of data points.
- Computationally, you can use random search (optimization) methods to find a point that has the minimum of total quadratic distance between center and points in each cluster (e.g. PSO, Hill climbing).



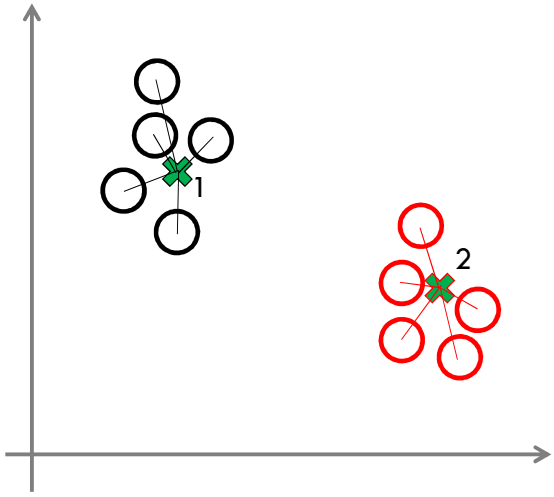
K-Means – Assign Again

- Now we repeat the Assign-Optimize Steps…
- This time the data points will be assigned to the centers differently…



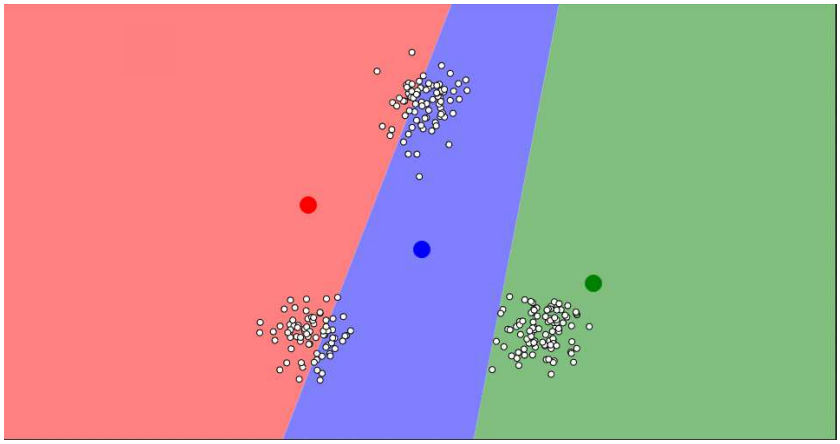
K-Means – Optimize Again

- We optimize the location of the center points again…
- As you see, within just small number of iterations, the centers of clusters move into their optimum locations and therefore we find the best clusters among the data points.



K-Means – Select Random Centers

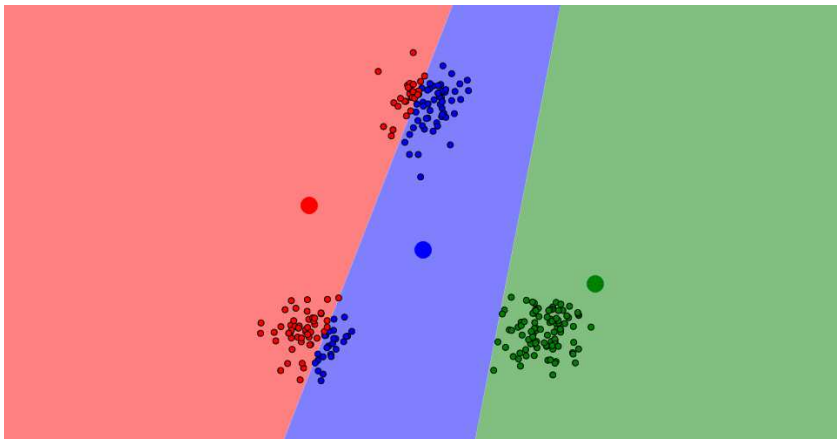
- We use an online visualization tool to show the clustering iterations and problems.
- In the first step we select random points and …



Using: http://www.naftaliharris.com/blog/visualizing-k-means-clustering/

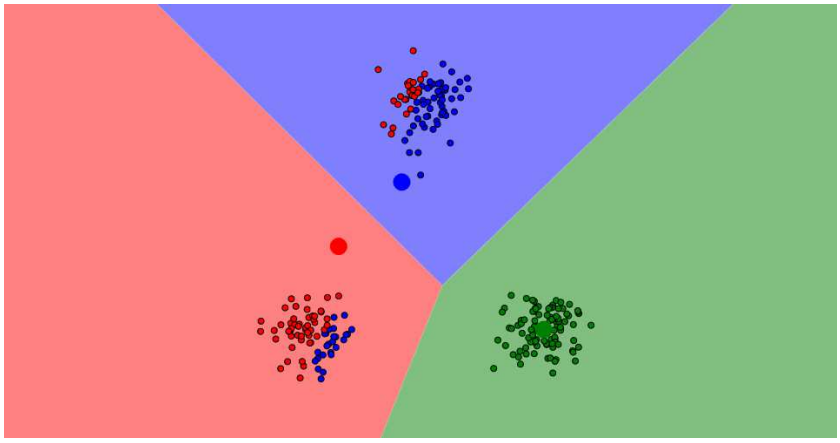
K-Means – Assign (1)

- Now we assign the data points to clusters (the color of each point shows its assignment)
- The assignment is based on the distance of each point to the 3 centers. The point is assigned to the nearest center.



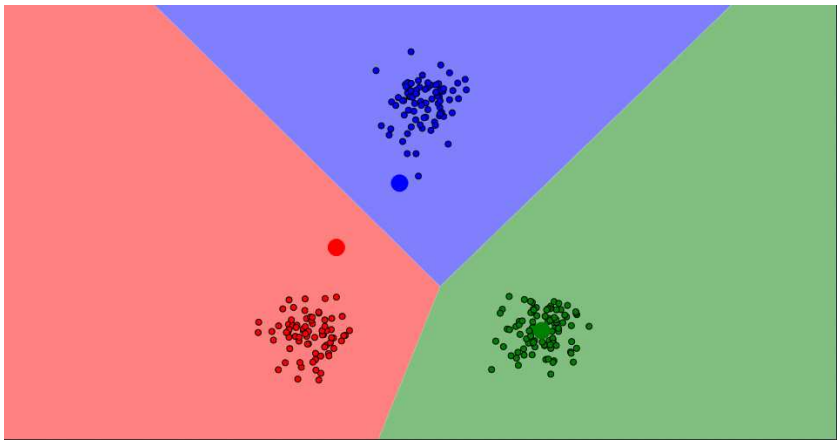
K-Means – Optimize Centroids (1)

- Now we optimize the location of each centroid, in a way that its distance is minimized to the data points in its class.



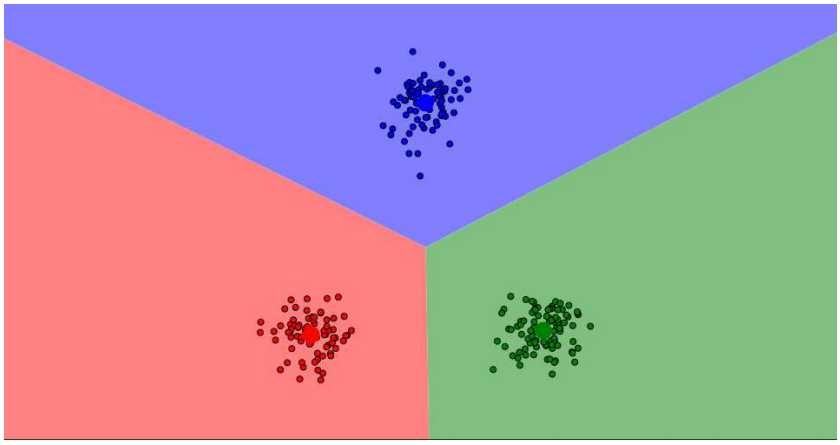
K-Means – Assign (2)

- Now we repeat the assignment step from the beginning (assigning points to the nearest centroid).



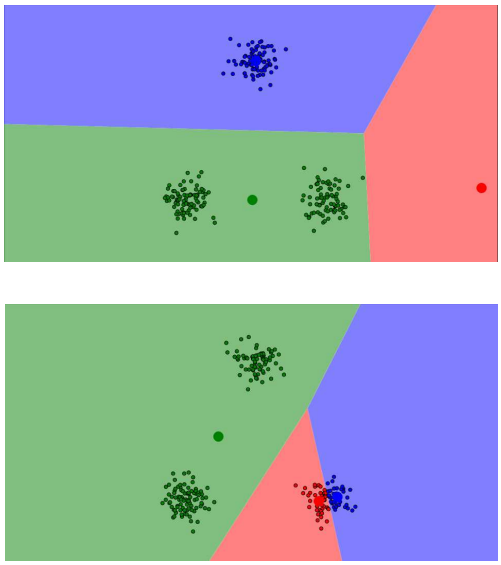
K-Means – Optimize Centroids (2)

- Now we optimize the location of the centroids again…
- In just a few iterations, as you see the centroid move to their final positions…



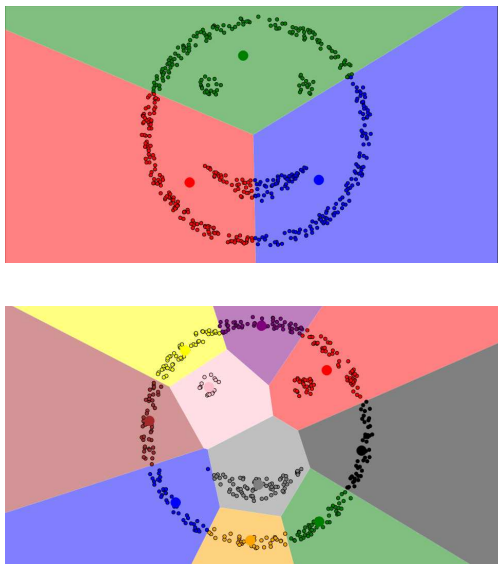
K-Means – Initial Points

- The build up of clusters depends on the initial location of centroids. Different centroids may result in quite different clusters.
- For example this is the final result of bad initial points. The algorithm determines badly selected cluster…
- K-Means normally uses hill climbing algorithm to find the best location for centroids. That is why in the optimization step, it may not find the global optimal points for the centroids…



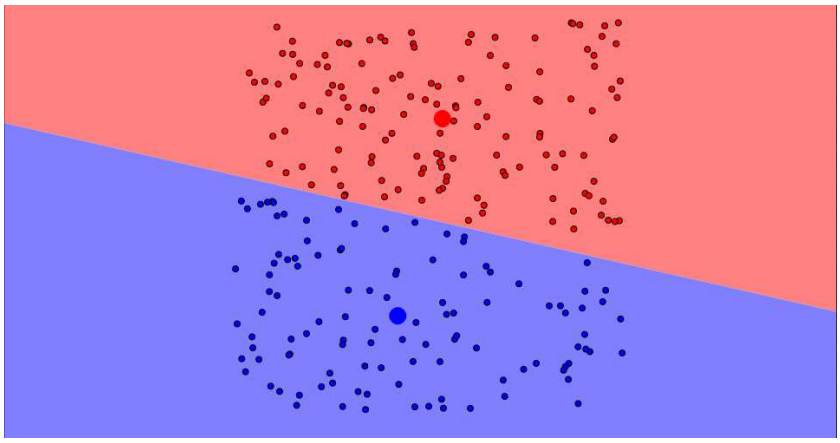
K-Means – Number of Points

- The build up of clusters also depends on the number of centroids. Different number of centroids may result in quite different clusters.



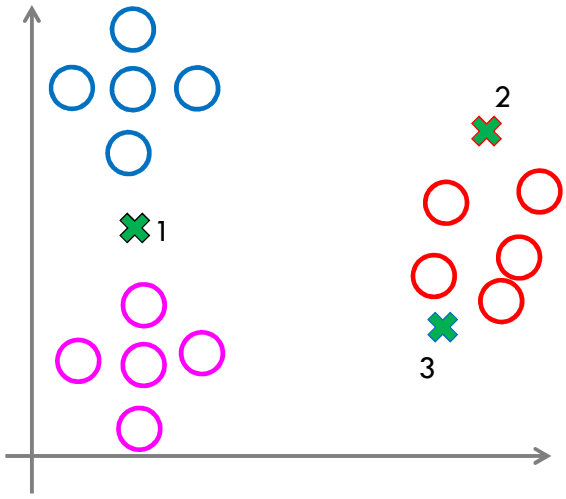
K-Means – Not Very Apparent Clusters

- If the data points are uniformly distributed and we have 2 clusters, the space will possibly be divided into to uniform half spaces…
- Clustering cannot do a better job…



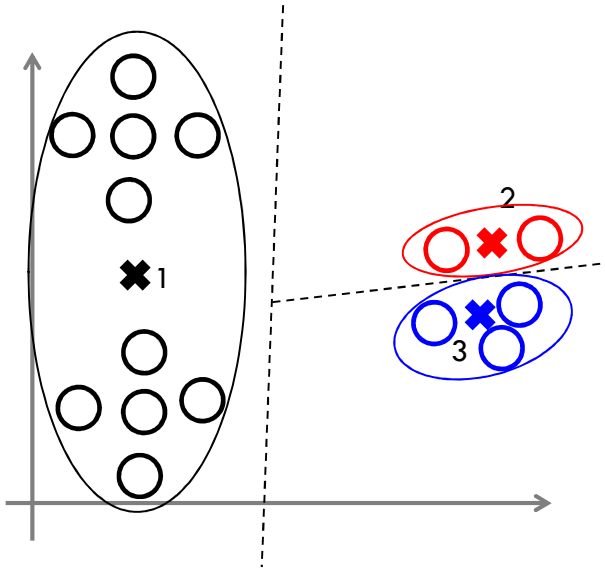
K-Means – Initial Points

- This example shows an occasion in which the centroid 1 is in between two clusters. Since the position is in the middle of the two clusters (in left), it is very likely that the optimization process cannot move it away from that point (remember the rubber bands. The point is already in a stable position …)
- The centroid 2 and 3 will compete to own the 5 points in the right hand side…



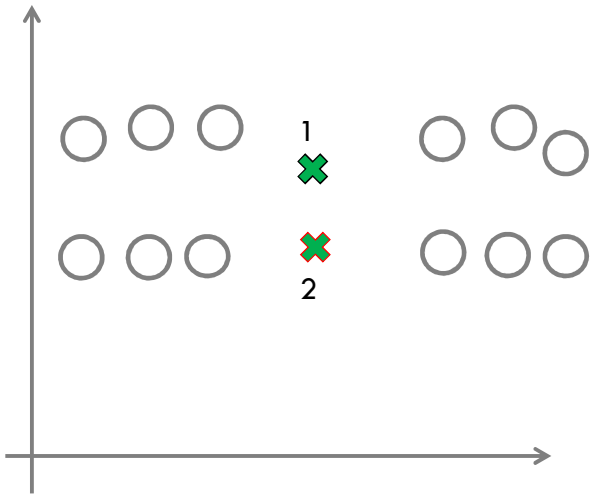
K-Means – Initial Points

- And the result is that we may have just two clusters …
- If the initial points were somewhat different, the results could be different.
- If we had 3 centroids with initial positions that are not unsuitable, we could get 3 clusters.



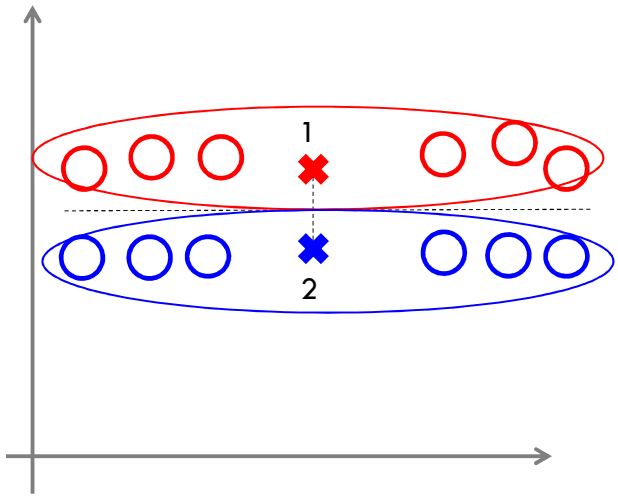
K-Means – Initial Points

- Another example of how the K-Means clustering might go wrong…
- Again we could have two initial points that cause bad local minimum….
- Assume the initial centroids are somewhere exactly between two clusters …



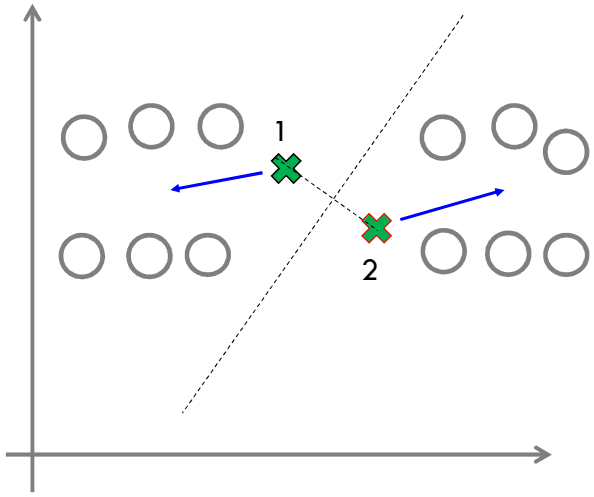
K-Means – Initial Points

- If the two points are exactly on top of each other and in the middle (a very rare case if we use good random numbers), it is possible that we get the shown clustering…



K-Means – Initial Points

- However, if the locations were slightly different, we would get a different clustering even in the first round.
- In the next iterations, the centers would eventually move to the centers of the clusters



K-Means – Number of Points

- Therefore, we should find a way to:

1) Guess proper number of clusters
2) Guess a better initial points

- S1: We can either select manually or start from 2 and go up and check the results

- S2: We can repeat the clustering say 10 times, and then either select manually from the results or build an ensemble of the 10 results (mix the found centroids, e.g. find average points of the 3 centroids, or after finding average centroids, run the optimization again) .

K-Means – Scikit Learn Example

Assuming that we have the following items (2 dimensions each), cluster the points. Start with n_clusters = 2 and increase the number of clusters until it makes sense.

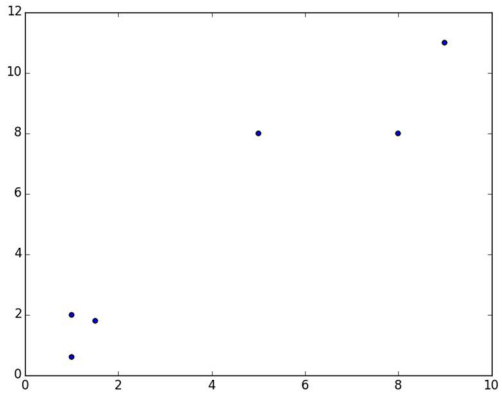
[1, 2], [5, 8], [1.5, 1.8], [8, 8], [1, 0.6], [9, 11]

K-Means – Scikit Learn Example

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

X = np.array([[1, 2],
              [5, 8],
              [1.5, 1.8],
              [8, 8],
              [1, 0.6],
              [9, 11]])

plt.scatter(X[:,0],X[:,1])
plt.show()
```

K-Means – Scikit Learn Example

- The following shows the scatter plot of the points

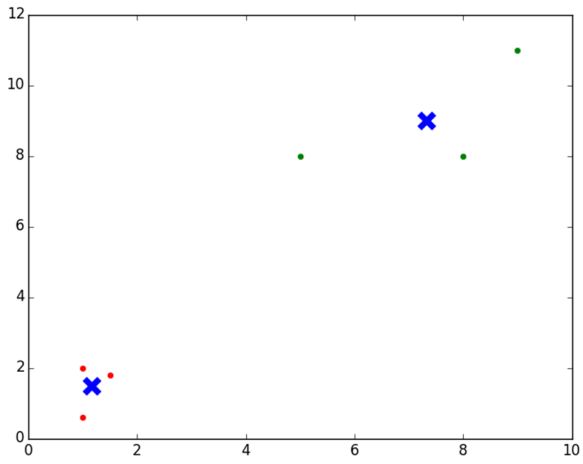


K-Means – Scikit Learn Example

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

centroids = kmeans.cluster_centers_
labels = kmeans.labels_

print(centroids)
print(labels)

colors = ["g.","r.","c.","y."]

for i in range(len(X)):
    print("coordinate:",X[i], "label:", labels[i])
    plt.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)

plt.scatter(centroids[:, 0],centroids[:, 1], marker = "x", s=150, linewidths = 5, zorder = 10)
plt.show()
```

K-Means – Scikit Learn Example

- The following shows the scatter plot of the clusters



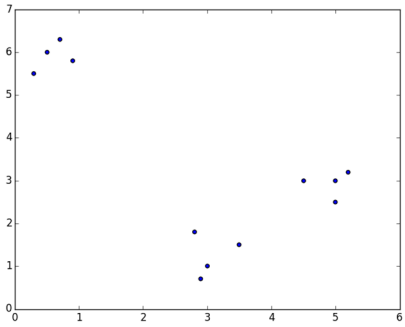
K-Means – Scikit Learn Example

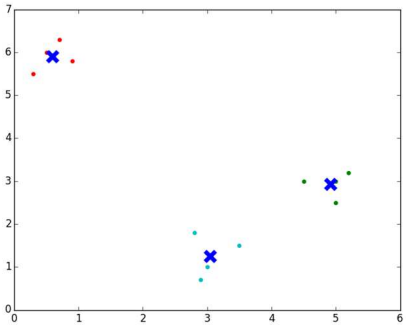
- The following shows the output console results

```
[[ 7.33333333  9.        ]
 [ 1.16666667  1.46666667]]
[1 0 1 0 1 0]
('coordinate:', array([ 1.,  2.]), 'label:', 1)
('coordinate:', array([ 5.,  8.]), 'label:', 0)
('coordinate:', array([ 1.5,  1.8]), 'label:', 1)
('coordinate:', array([ 8.,  8.]), 'label:', 0)
('coordinate:', array([ 1. ,  0.6]), 'label:', 1)
('coordinate:', array([  9.,  11.]), 'label:', 0)
```

K-Means – Exercise

- Cluster the following data with SkLearn (by changing the above program and selecting n_clusters=3). You should get graphs similar to those provided.
- [5,3], [5,2.5], [4.5,3], [5.2,3.2], [3,1], [3.5,1.5], [2.8,1.8], [2.9,0.7], [0.5,6], [0.3,5.5], [0.7,6.3], [0.9,5.8]





K-Means – More than 2 attributes

- We use a data set that comes with ScikitLearn. Iris data set is classification data set. Based on the 4 attributes a classification algorithm must learn to classify flowers into one of the flower families.
  - The attributes are: sepal length (cm), sepal width (cm), petal length (cm), petal width (cm)
  - The classes (targets) are: setosa, versicolor, virginica
  - There are 150 rows of data
- In the classification example we ignore the target classes and try to cluster the data items into 3 clusters (i.e. no use of the labels). We then observe how those clusters relate to the target classes. The classes might somehow match or don't match the clusters.

K-Means – More than 2 attributes

```
from sklearn import datasets
iris = datasets.load_iris()

from sklearn import decomposition
pca = decomposition.PCA(n_components=2)
pca.fit(iris.data)
X = pca.transform(iris.data)

# plot and select colors based on provided target classes (labels)
import pylab as pl
pl.scatter(X[:, 0], X[:, 1], c=iris.target)
pl.show()
```

K-Means – More than 2 attributes

```
# now cluster into 3 clusters
from sklearn.cluster import KMeans
km = KMeans(3)
km.fit(X)
centers = km.cluster_centers_
labels  = km.labels_

colors = ["g.","r.","c.","y."]
for i in range(len(X)):
    pl.plot(X[i][0], X[i][1], colors[labels[i]], markersize = 10)

pl.scatter(centers[:, 0], centers[:, 1], marker = "x", s=150, linewidths = 5, zorder = 10)
pl.show()
```

K-Means – More than 2 attributes

- The plot based on the target classes (left) and the plot based on cluster numbers (right) are provided. Notice that in order to draw we have used PCA method to reduce the number of attributes into 2.