



Unsupervised Learning: clustering

AAA-Python Edition



school of a

Plan

- 1- Clustering
- 2- K-mean example
- 3- Silhouette score
- 4- Mean Shift
- 5- Gaussian Mixture Model
- 6- Affinity Propagation Model

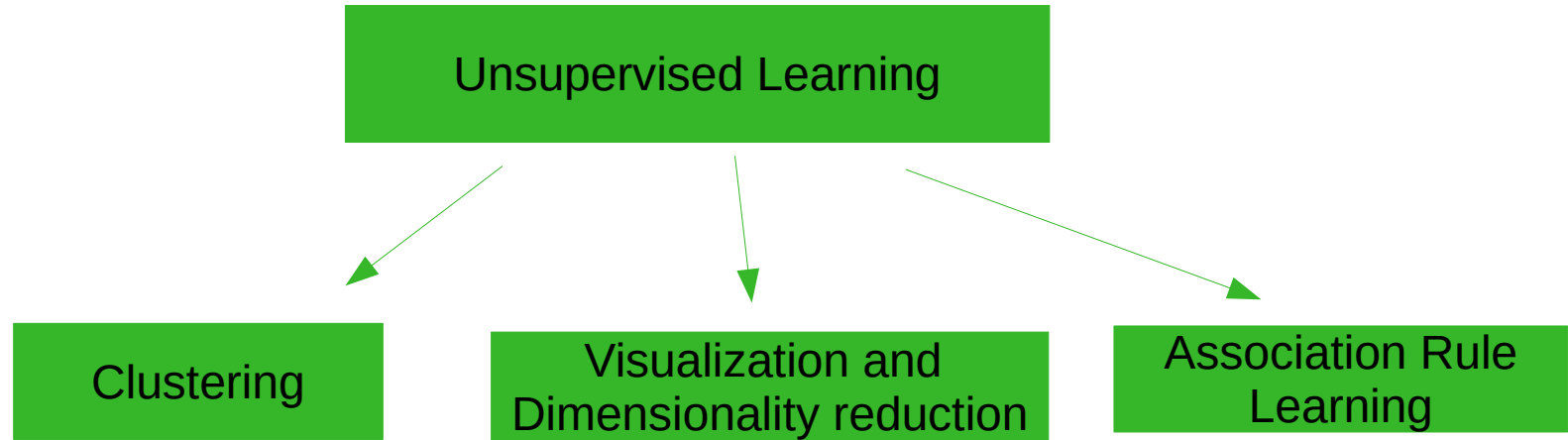


school of AI

1- Clustering

Unsupervised Learning

- Unsupervised learning is about Machine Learning using Unlabeled Data.
- It concerns several kind of tasks, that can be grouped into **3 categories**:



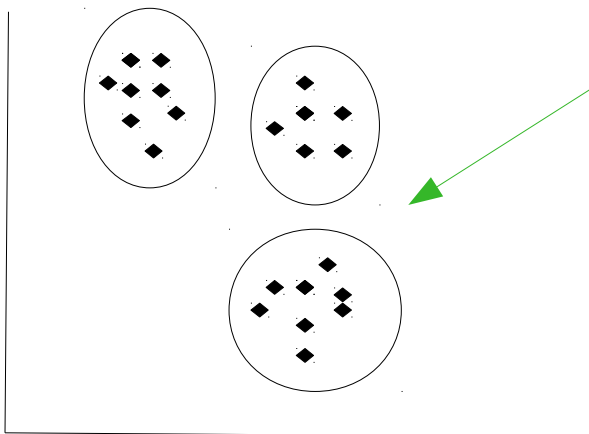


school of a

1- Clustering

Clustering

- **Clustering** means finding the **clusters** within a given **data**.
- A **cluster** can be defined as a **group** of **similar points** (samples) of data.
- One way to identify similar points, is to use **similarity metrics**. **Euclidean Distance** in one of these similarity metrics.



Data distributed
into 3 distinct
clusters



school of a

1- Clustering

K-mean algorithm

- It is one of the methods used in clustering. It uses the euclidean distance as a similarity metric.
- The steps are as follow:
 - 1- Set the number of clusters to be find: K
 - 2- Select randomly : k samples from the data ,or use an algorithm to select generally distant k points. The k samples are the initial centroids of the clusters.
 - 3- Divide the samples into these K clusters: for each sample:
 - ➔ Compute the distances from each centroid
 - ➔ Select the least one.
 - 4- Compute the new centroids of clusters u_j the mean of the samples belonging to each cluster
 - 5- Compute the inertia, for n samples, using this formula: $\sum_{i=0}^n (\|x_i - u_j\|)^2$
 - 6- Repeat from step 3, until the inertia is minimized, or the clusters assignment do not change or a maximum number of iterations is reached.



school of a

2- K-mean example

The data

- The data: 2 features →

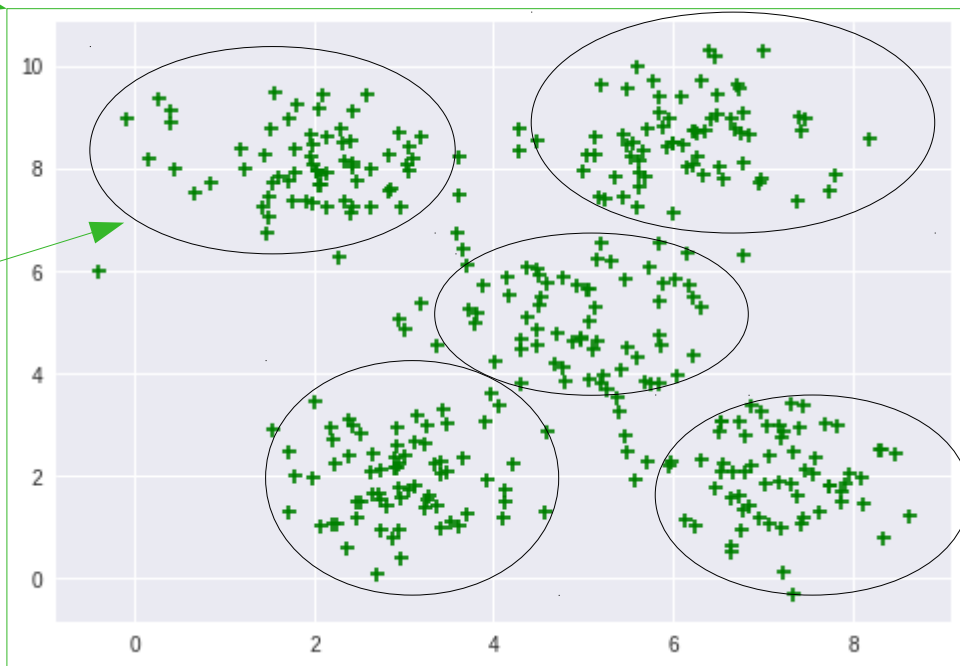
```
1 X= dataF.values
2 print (X.shape)
```

(350, 2)

```
1 dataF = pd.read_csv('data_clustering.txt',header=-1)
2 dataF.head(5)
```

	0	1
0	2.08	1.05
1	2.05	7.70
2	4.53	5.49
3	6.23	1.02
4	5.35	7.86

We can clearly
see the 5
clusters





school of a

2- K-mean example

Training

```
from sklearn.cluster import KMeans
```

```
1 # initialize the number of the clusters
2 num_clusters = 5
3 # Create KMeans object using random centroid initialization
4 kmeans = KMeans(init='random', n_clusters=num_clusters, verbose=1, n_init=1)
```

```
1 # Train the KMeans clustering model
2 kmeans.fit(X)
```

```
Initialization complete
start iteration
done sorting
end inner loop
Iteration 0, inertia 873.3646029796687
start iteration
done sorting
end inner loop
Iteration 1, inertia 762.7499327404964
```

```
Iteration 7, inertia 433.80300829447276
center shift 0.000000e+00 within tolerance 6.415290e-04
```

```
# the final centroids are
final_Centroids=kmeans.cluster_centers_
```

```
array([[2.97253521, 1.97267606],
       [6.10842857, 8.58428571],
       [7.09588235, 2.01735294],
       [1.98385714, 8.04942857],
       [4.92605634, 5.0184507 ]])
```

The algorithm will run **n_init** times with different random centroid initialization at each time. Then select the best result (the best final inertia value)

Those are the means of the samples belonging to each cluster.



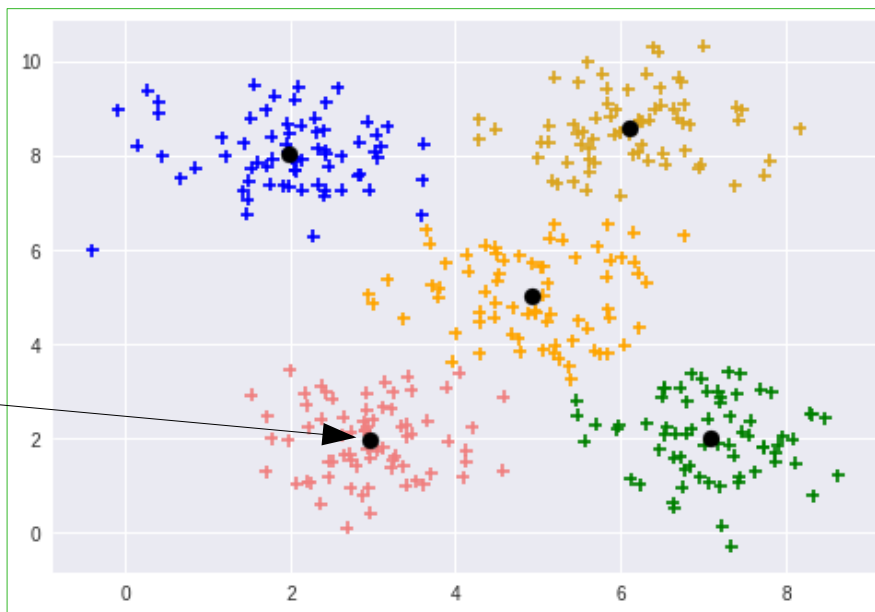
school of a

2- K-mean example

Visualization

```
1 kmeans.labels_  
  
array([3, 4, 1, 2, 0, 3, 4, 1, 1, 0, 3, 4,  
       1, 3, 1, 3, 4, 1, 2, 0, 3, 4, 1, 2])  
  
clus_colors=[ "goldenrod" if y ==1 else "green" if y == 2 else "blue" if y==3 else "orange" if y== 4  
              else "lightcoral" for y in kmeans.labels_]  
  
plt.figure()  
plt.scatter(X[:,0], X[:,1], marker='+',c=clus_colors)  
plt.scatter(final_Centroids[:,0],final_Centroids[:,1],marker="o",facecolor="black",  
            s =80)
```

The
centroids



[By Amina Delali]



school of ants

3- Silhouette score

Definition

- **Silhouette score** is a measure that is used to indicate the quality of the clustering: it shows at which degree the data samples are similar to the others samples belonging to the same cluster.

- For one sample, it is computed as follow:

$$s = \frac{b - a}{\max(a, b)}$$

➤ Where:

- **a**: the mean distance between a sample and all other samples in the same cluster.
- **b**: the mean distance between a sample and all other samples in the next nearest cluster.
- The **distance** is computed by considering the samples as points and the features as coordinates. The distance between two samples, is computed as the euclidean distance between 2 points.

For example: the distance between $c(x_1, y_1)$ and $d(x_2, y_2)$ is computed as follow: $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$



school of ants

3- Silhouette score

Use

- For one sample, the silhouette value score ranges from 1 to -1. Many values close to 1 indicates **good** clustering. And in the other hand, many values close to -1, indicates **bad** clustering (too many or too few clusters).
- We can compute the overall silhouette score by computing the **mean** of all the samples silhouette scores. A value close to zero indicates cluster overlapping.
- The overall silhouette score can be used to find the best **k** number of clusters.
- Sklearn implements the silhouette score calculation: the overall (mean) and the singular coefficient (for one sample).
 - `sklearn.metrics.silhouette_score`: the mean
 - `sklearn.metrics.silhouette_samples`: for each sample

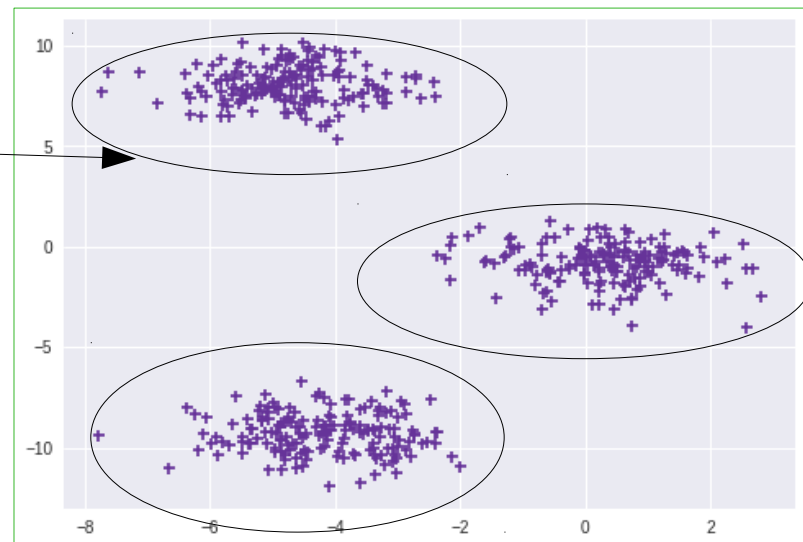
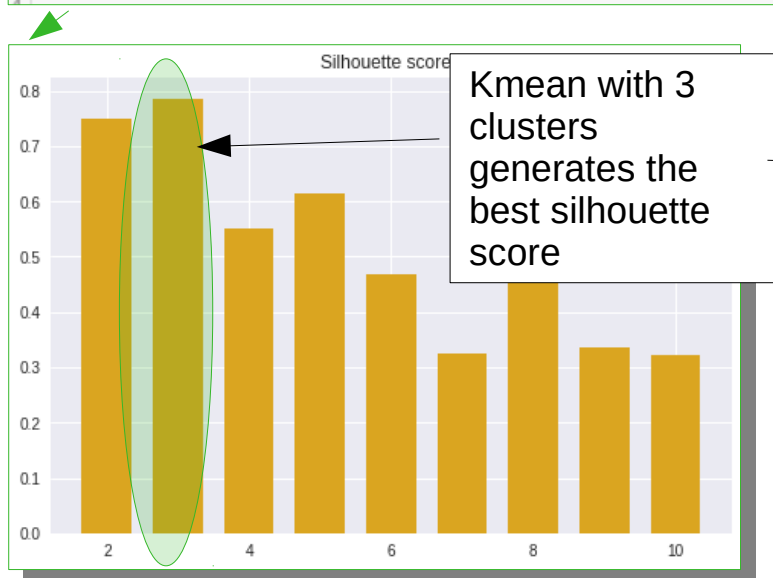


school of a

3- Silhouette score

Example

```
1 # testing cluster numbers: from 2 to 10
2 k_values = range(2,11)
3 scores = []
4 for n_clusters in k_values:
5     # initialise a kmean clusterer with the actual tested
6     # cluster number
7     myKmean = KMeans (init='random', n_clusters=n_clusters, n_init=1)
8     # train it
9     myKmean.fit(X)
10    # compute the overall silhouette score
11    overall_score= silhouette_score(X, myKmean.labels_, metric='euclidean', sample_size=len(X))
12    # add it to list of previously computed scores
13    scores.append(overall_score)
```





school of ants

4- Mean Shift

Concept & Steps

- **Mean shift** clustering is an other method used to identify clusters in a dataset. It supposes that the data distribution may tend to be dense around certain points: the centers of the clusters (centroids).
- To identify these points, it defines the following steps:
 - Make a copy of your data
 - For each sample in the copy, repeat these steps:
 - Compute the mean of samples contained in a window (fixed size) around the sample. The mean is computed as follow:

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

Where: K is a function that will affect weights to the neighbors of x, regarding their distance from x (bigger distance, small weight). N(x) represents the points near x(neighbors of x).

- Update the sample's location to that new mean.
 - Stop the process when the samples doesn't move, or the shift is not significant. Then, Filter out near-similar points to keep the final centroids.



school of a

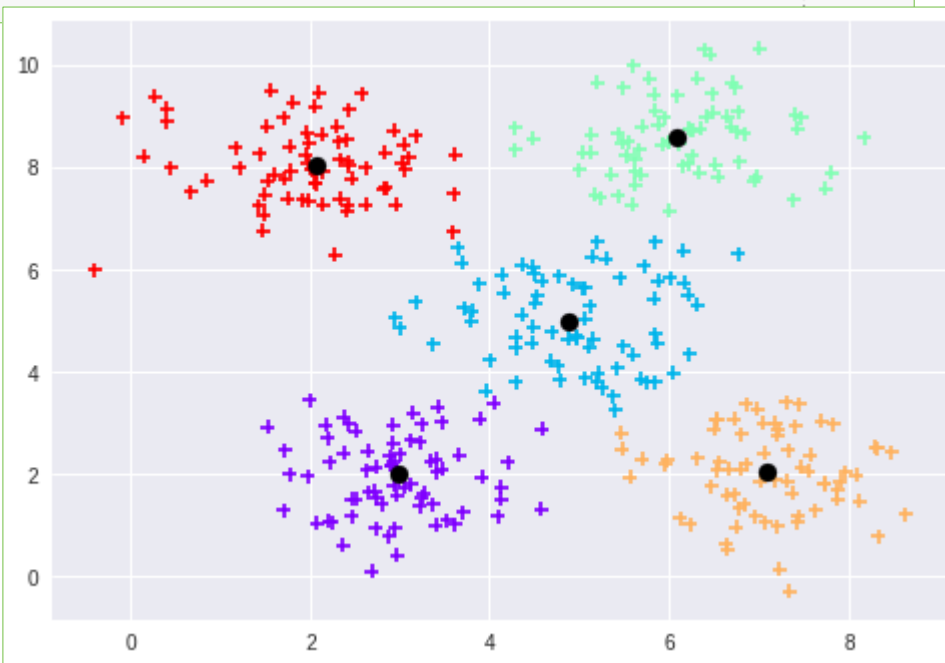
4- Mean Shift

Example

```
1 # Cluster data with MeanShift
2 myMeanShift = MeanShift(bandwidth=2)
3 myMeanShift.fit(X)
```

```
1 labels = myMeanShift.labels_
2 final_Centroids2 = myMeanShift.cluster_centers_
3 plt.figure()
4 plt.scatter(X[:,0], X[:,1], marker='+', c=labels, cmap="rainbow")
5 plt.scatter(final_Centroids2[:,0], final_Centroids2[:,1], marker="o", facecolor="black",
6             s=80)
```

We don't know the number of clusters in advance. We have to plot according to labels values.





school of a

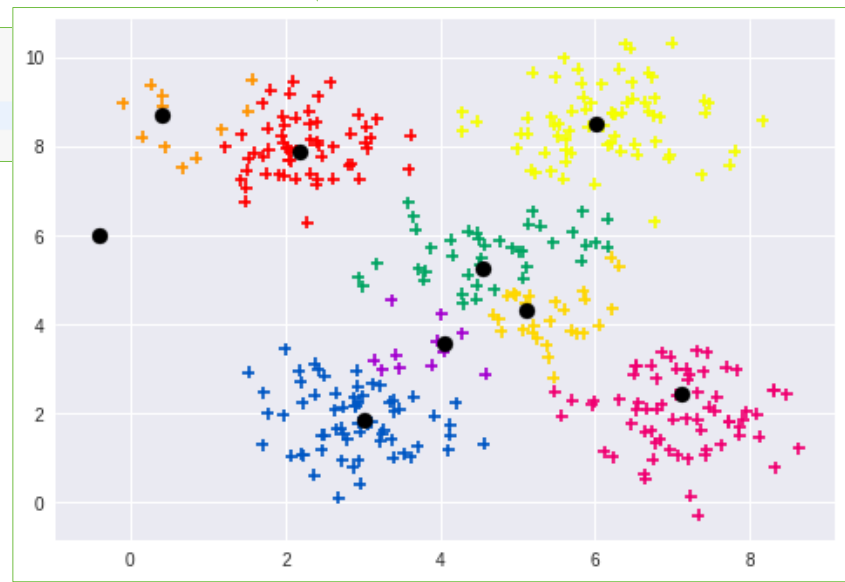
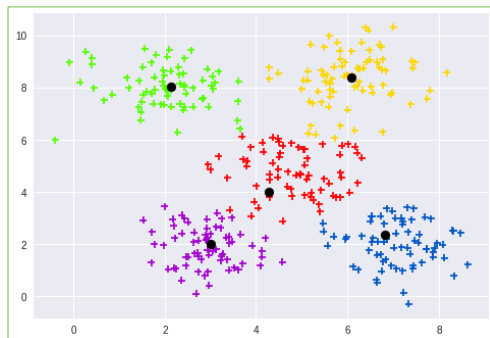
4- Mean Shift

Bandwidth

- If you change the bandwidth value, you will have different clustering:

```
1 from sklearn.cluster import estimate_bandwidth
2
3 myBW = estimate_bandwidth(X, quantile=0.2);
4 print("The estimated bandwidth is:\n", myBW)
```

The estimated bandwidth is: 2.35



Bandwidth == 1

Concept



school of AI

5- Gaussian Mixture Model

- The **Gaussian Mixture Model** is simply a **mixture** of **Gaussian models**.
- A **Gaussian Model**, is a model that makes the hypothesis that the data is drawn following the Gaussian distribution, or more exactly, the function:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

The parameters of that model are
 μ : the mean,
and σ : the standard deviation

- Gaussian Mixture clustering consist of identifying to which distribution each point belongs.
- This identification is done using algorithms like: **Expectation-Maximization (EM)** algortihm.



school of ai

5- Gaussian Mixture Model

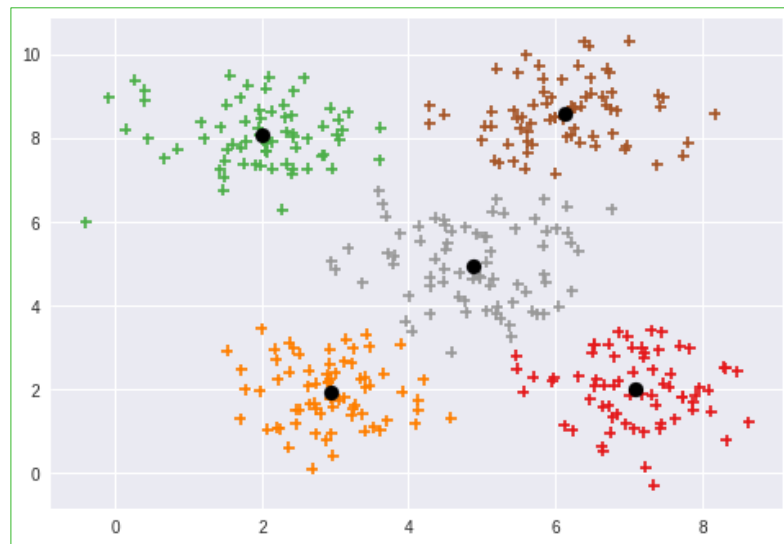
Example with scikit-learn

- Scikit-learn will use the expectation maximization algorithm to determine the different Gaussian distributions.

```
myGMM = GaussianMixture(n_components= 5,)  
myGMM.fit(X)
```

There is no cluster centers but mean of each identified distribution.

```
1 labels = myGMM.predict(X)  
2 centroids= myGMM.means_  
3 plt.figure()  
4 plt.scatter(X[:,0], X[:,1], marker='+', c=labels,cmap = "Set1")  
5 plt.scatter(centroids[:,0],centroids[:,1],marker="o",facecolor="black",  
6             s =80)
```



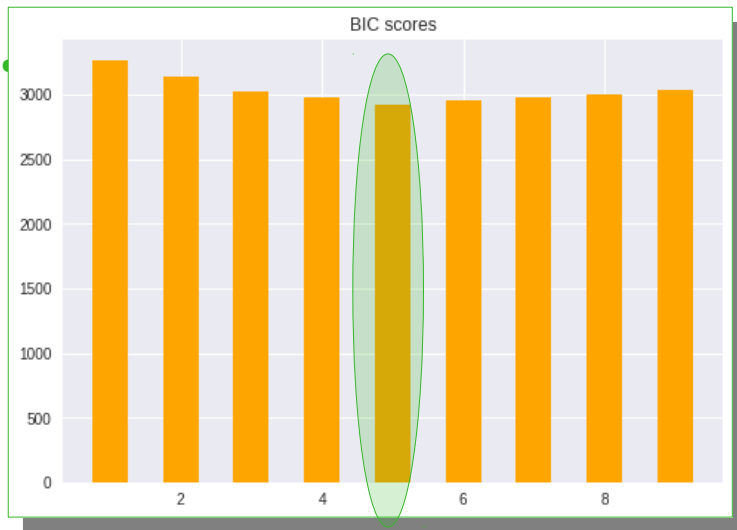


school of a

5- Gaussian Mixture Model

Select n_components

We compute the BIC score (or the AIC score) for each model corresponding to each number of components. Then, select the number of components corresponding to the lowest score



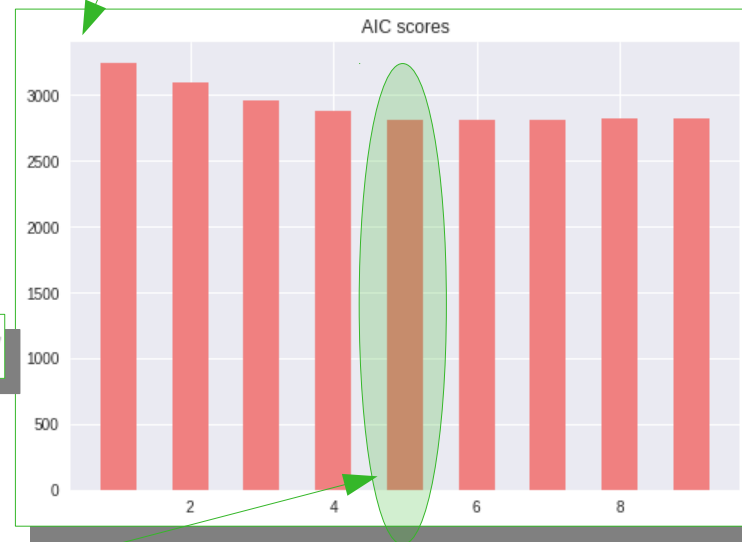
```
np.round(bics,2)
```

```
array([3263.51, 3139.39, 3028.19, 2973.55, 2920.83, 2954.91, 2983.97, 3002.4, 3034.91])
```

```
1 np.round(aics,2)
```

```
array([3244.22, 3096.95, 2962.61, 2884.81, 2808.95, 2819.88, 2825.8, 2821.08, 2830.44])
```

```
1 n_components = range(1,10)
2 bics=[]
3 aics=[]
4 for i in n_components:
5     model = GaussianMixture(n_components=i)
6     model.fit(X)
7     bics.append(model.bic(X))
8     aics.append(model.aic(X))
```



[By Amina Delali]

Concept

- An other clustering method: **Affinity propagation model**. It is defined by these terms : **Exemplars, responsibility, availability, preference, damping factor** and **message passing**.
 - **Exemplars** are the clusters centers.
 - **Preference** influence the resulting number of clusters by affecting a preference value for a sample to be an exemplar (its similarity score between it and itself).
 - **Message**: responsibility or availability values.
 - **Message passing**: responsibility and availability calculation
 - **Responsibility $r(i,k)$** : accumulated evidence that the sample **k** should be the **exemplar** of **i**.
 - **Availability $a(i,k)$** : accumulated evidence that the sample **i** should select **k** to be its **exemplar**.
 - **Damping factor λ** : influence the resulting number of clusters by affecting a value for the relationship between new and current messages values. Must be ≥ 0.5 and < 1



school of a

Process

- Select a similarity measure. For example, the **negative** of Euclidean Distance
- Select preferences and a damping factor values.
- Set **r** and **a matrices** to **0**. Fill **S** with the similarity scores between all points. Its diagonal is filled with the preferences values.
- **Repeat until convergence** (changes for exemplars values are too small or non-existent) **is reached**:
 - Compute r using this formulas:
 - ➔ $r_{t+1}(i, k) = s(i, k) - \max[a_t(i, k') + s(i, k') \forall k' \neq k]$
 - ➔ $r_{t+1}(i, k) = \lambda \cdot r_t(i, k) + (1 - \lambda) \cdot r_{t+1}(i, k)$
 - Compute a using this formulas:
 - ➔ $a_{t+1}(i, k) = \min[0, r_{t+1}(k, k) + \sum_{i', i' \notin [i, k]} r_{t+1}(i', k)]$
 - ➔ $a_{t+1}(i, k) = \lambda \cdot a_t(i, k) + (1 - \lambda) \cdot a_{t+1}(i, k)$
- **Select the exemplars** from: $r(i, i) + a(i, i) > 0$



school of a



school of a

6-Affinity Propagation Model

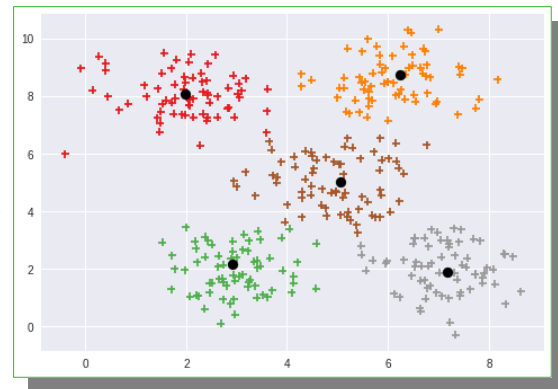
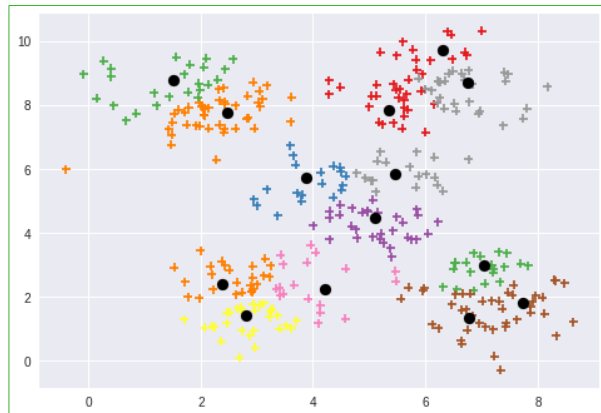
Example

```
from sklearn.cluster import AffinityPropagation  
# initialization of the model with default values  
myModel= AffinityPropagation()
```

```
myModel.fit(X)
```

Default value for **preferences** is the **median** of all similarity scores (median = the value in the **middle** of ordered similarity scores)

```
myModel= AffinityPropagation(preference=-100)  
myModel.fit(X)
```





school of a

References

- Aurélien Géron. Hands-on machine learning with Scikit-Learn and Tensor-Flow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc, 2017.
- matplotlib.org. color example code: colormaps_reference.py. On-line at https://matplotlib.org/examples/color/colormaps_reference.html. Accessed on 24-12-2018.
- matplotlib.org. color example code: named_colors.py. On-line at https://matplotlib.org/examples/color/named_colors.html. Accessed on 24-12-2018.
- Joshi Prateek. Artificial intelligence with Python. Packt Publishing, 2017.
- Vink Ritchie. Algorithm breakdown: Affinity propagation. On-line at <https://www.ritchievink.com/blog/2018/05/18/algorithm-breakdown-affinity-propagation/>, May 2018. Accessed on 24-12-2018.
- Scikit-learn.org. scikit-learn, machine learning in python. On-line at <https://scikit-learn.org/stable/>. Accessed on 03-11-2018.
- wikipedia.org. Affinity propagation. On-line at https://en.wikipedia.org/wiki/Affinity_propagation. Accessed on 24-12-2018.
- wikipedia.org. Mean shift. On-line at https://en.wikipedia.org/wiki/Mean_shift. Accessed on 24-12-2018.



Thank you!

FOR ALL YOUR TIME