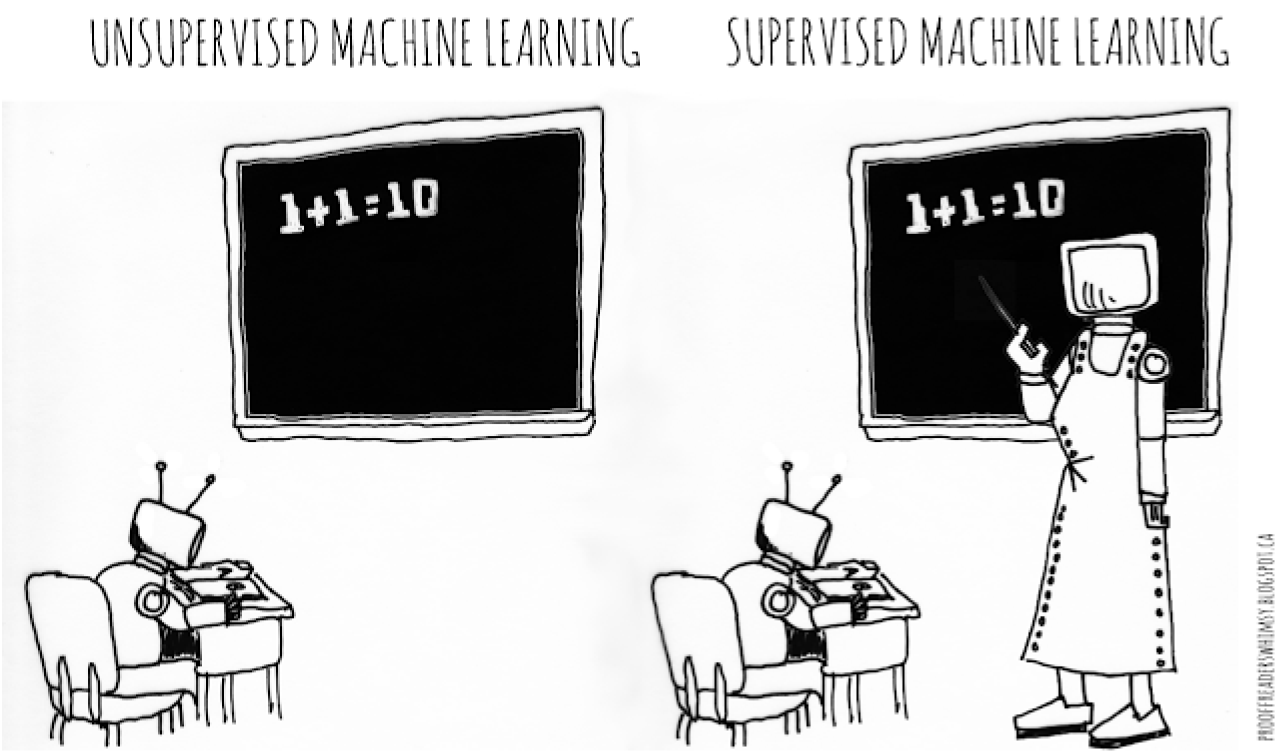
Unsupervised Learning

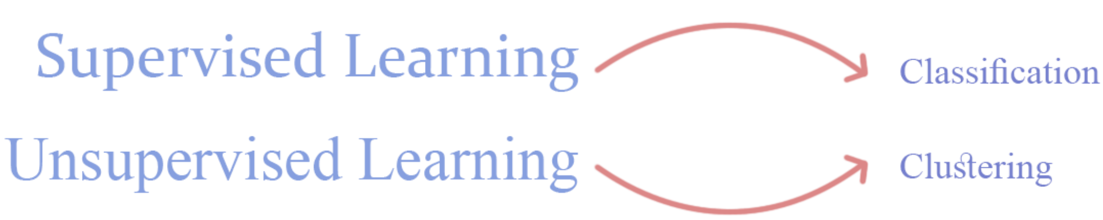
> Unsupervised learning finds the pattern in data

Supervised learning : (x, y)
Unsupervised learning : (x)



by David Taylor

[http://prooffreaderswhimsy.blogspot.com/2014/11/machine-learning.html (http://prooffreaderswhimsy.blogspot.com/2014/11/machine-learning.html)](http://prooffreaderswhimsy.blogspot.com/2014/11/machine-learning.html)



K-Means

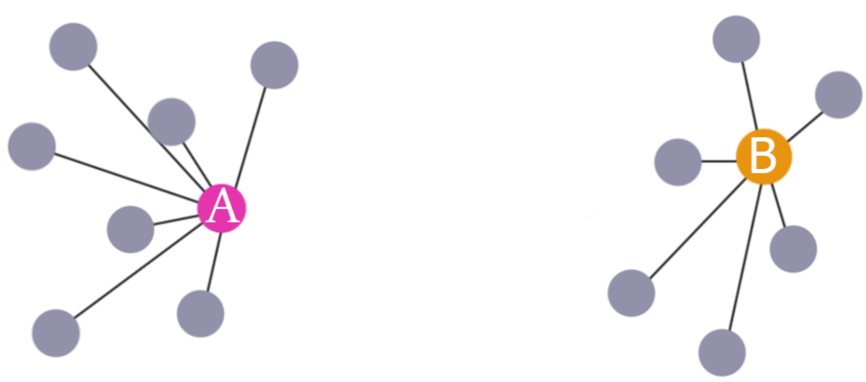
> K-Means finds culster of samples

K-Means

Step one : initialization

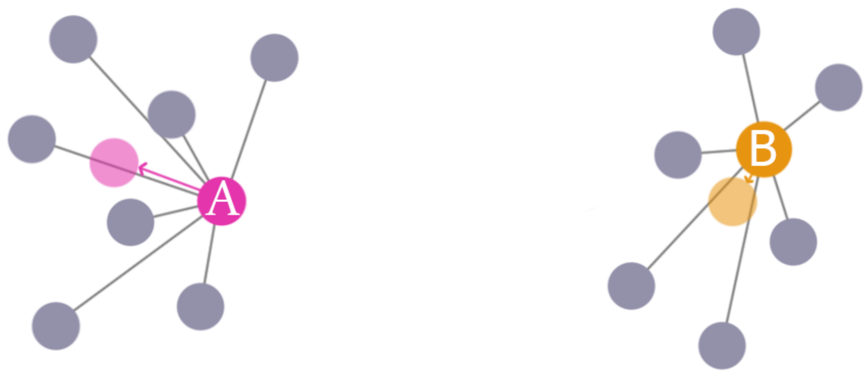
K-Means

Step two : Assignment



K-Means

Step three : Update



Visualizing K-Means algorithm :

[http://tech.nitoyon.com/en/blog/2013/11/07/k-means/ (http://tech.nitoyon.com/en/blog/2013/11/07/k-means/)](http://tech.nitoyon.com/en/blog/2013/11/07/k-means/)

In [1]:

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

In [2]:

```
iris = load_iris()
```

In [3]:
```
iris.data
```

```
Out[3]: array([[ 5.1,  3.5,  1.4,  0.2],
               [ 4.9,  3. ,  1.4,  0.2],
               [ 4.7,  3.2,  1.3,  0.2],
               [ 4.6,  3.1,  1.5,  0.2],
               [ 5. ,  3.6,  1.4,  0.2],
               [ 5.4,  3.9,  1.7,  0.4],
               [ 4.6,  3.4,  1.4,  0.3],
               [ 5. ,  3.4,  1.5,  0.2],
               [ 4.4,  2.9,  1.4,  0.2],
               [ 4.9,  3.1,  1.5,  0.1],
               [ 5.4,  3.7,  1.5,  0.2],
               [ 4.8,  3.4,  1.6,  0.2],
               [ 4.8,  3. ,  1.4,  0.1],
               [ 4.3,  3. ,  1.1,  0.1],
               [ 5.8,  4. ,  1.2,  0.2],
               [ 5.7,  4.4,  1.5,  0.4],
               [ 5.4,  3.9,  1.3,  0.4],
               [ 5.1,  3.5,  1.4,  0.3],
               [ 5.7,  3.8,  1.7,  0.3],
               [ 5.1,  3.8,  1.5,  0.3],
               [ 5.4,  3.4,  1.7,  0.2],
               [ 5.1,  3.7,  1.5,  0.4],
               [ 4.6,  3.6,  1. ,  0.2],
               [ 5.1,  3.3,  1.7,  0.5],
               [ 4.8,  3.4,  1.9,  0.2],
               [ 5. ,  3. ,  1.6,  0.2],
               [ 5. ,  3.4,  1.6,  0.4],
               [ 5.2,  3.5,  1.5,  0.2],
               [ 5.2,  3.4,  1.4,  0.2],
               [ 4.7,  3.2,  1.6,  0.2],
               [ 4.8,  3.1,  1.6,  0.2],
               [ 5.4,  3.4,  1.5,  0.4],
               [ 5.2,  4.1,  1.5,  0.1],
               [ 5.5,  4.2,  1.4,  0.2],
               [ 4.9,  3.1,  1.5,  0.1],
               [ 5. ,  3.2,  1.2,  0.2],
               [ 5.5,  3.5,  1.3,  0.2],
               [ 4.9,  3.1,  1.5,  0.1],
               [ 4.4,  3. ,  1.3,  0.2],
               [ 5.1,  3.4,  1.5,  0.2],
               [ 5. ,  3.5,  1.3,  0.3],
               [ 4.5,  2.3,  1.3,  0.3],
               [ 4.4,  3.2,  1.3,  0.2],
               [ 5. ,  3.5,  1.6,  0.6],
               [ 5.1,  3.8,  1.9,  0.4],
               [ 4.8,  3. ,  1.4,  0.3],
               [ 5.1,  3.8,  1.6,  0.2],
               [ 4.6,  3.2,  1.4,  0.2],
               [ 5.3,  3.7,  1.5,  0.2],
               [ 5. ,  3.3,  1.4,  0.2],
               [ 7. ,  3.2,  4.7,  1.4],
               [ 6.4,  3.2,  4.5,  1.5],
               [ 6.9,  3.1,  4.9,  1.5],
               [ 5.5,  2.3,  4. ,  1.3],
               [ 6.5,  2.8,  4.6,  1.5],
               [ 5.7,  2.8,  4.5,  1.3],
               [ 6.3,  3.3,  4.7,  1.6],
               [ 4.9,  2.4,  3.3,  1. ],
               [ 6.6,  2.9,  4.6,  1.3],
               [ 5.2,  2.7,  3.9,  1.4],
               [ 5. ,  2. ,  3.5,  1. ],
               [ 5.9,  3. ,  4.2,  1.5],
               [ 6. ,  2.2,  4. ,  1. ],
               [ 6.1,  2.9,  4.7,  1.4],
               [ 5.6,  2.9,  3.6,  1.3],
               [ 6.7,  3.1,  4.4,  1.4],
               [ 5.6,  3. ,  4.5,  1.5],
               [ 5.8,  2.7,  4.1,  1. ],
               [ 6.2,  2.2,  4.5,  1.5],
               [ 5.6,  2.5,  3.9,  1.1],
               [ 5.9,  3.2,  4.8,  1.8],
               [ 6.1,  2.8,  4. ,  1.3],
               [ 6.3,  2.5,  4.9,  1.5],
               [ 6.1,  2.8,  4.7,  1.2],
```

```
       [ 6.4,  2.9,  4.3,  1.3],
       [ 6.6,  3. ,  4.4,  1.4],
       [ 6.8,  2.8,  4.8,  1.4],
       [ 6.7,  3. ,  5. ,  1.7],
       [ 6. ,  2.9,  4.5,  1.5],
       [ 5.7,  2.6,  3.5,  1. ],
       [ 5.5,  2.4,  3.8,  1.1],
       [ 5.5,  2.4,  3.7,  1. ],
       [ 5.8,  2.7,  3.9,  1.2],
       [ 6. ,  2.7,  5.1,  1.6],
       [ 5.4,  3. ,  4.5,  1.5],
       [ 6. ,  3.4,  4.5,  1.6],
       [ 6.7,  3.1,  4.7,  1.5],
       [ 6.3,  2.3,  4.4,  1.3],
       [ 5.6,  3. ,  4.1,  1.3],
       [ 5.5,  2.5,  4. ,  1.3],
       [ 5.5,  2.6,  4.4,  1.2],
       [ 6.1,  3. ,  4.6,  1.4],
       [ 5.8,  2.6,  4. ,  1.2],
       [ 5. ,  2.3,  3.3,  1. ],
       [ 5.6,  2.7,  4.2,  1.3],
       [ 5.7,  3. ,  4.2,  1.2],
       [ 5.7,  2.9,  4.2,  1.3],
       [ 6.2,  2.9,  4.3,  1.3],
       [ 5.1,  2.5,  3. ,  1.1],
       [ 5.7,  2.8,  4.1,  1.3],
       [ 6.3,  3.3,  6. ,  2.5],
       [ 5.8,  2.7,  5.1,  1.9],
       [ 7.1,  3. ,  5.9,  2.1],
       [ 6.3,  2.9,  5.6,  1.8],
       [ 6.5,  3. ,  5.8,  2.2],
       [ 7.6,  3. ,  6.6,  2.1],
       [ 4.9,  2.5,  4.5,  1.7],
       [ 7.3,  2.9,  6.3,  1.8],
       [ 6.7,  2.5,  5.8,  1.8],
       [ 7.2,  3.6,  6.1,  2.5],
       [ 6.5,  3.2,  5.1,  2. ],
       [ 6.4,  2.7,  5.3,  1.9],
       [ 6.8,  3. ,  5.5,  2.1],
       [ 5.7,  2.5,  5. ,  2. ],
       [ 5.8,  2.8,  5.1,  2.4],
       [ 6.4,  3.2,  5.3,  2.3],
       [ 6.5,  3. ,  5.5,  1.8],
       [ 7.7,  3.8,  6.7,  2.2],
       [ 7.7,  2.6,  6.9,  2.3],
       [ 6. ,  2.2,  5. ,  1.5],
       [ 6.9,  3.2,  5.7,  2.3],
       [ 5.6,  2.8,  4.9,  2. ],
       [ 7.7,  2.8,  6.7,  2. ],
       [ 6.3,  2.7,  4.9,  1.8],
       [ 6.7,  3.3,  5.7,  2.1],
       [ 7.2,  3.2,  6. ,  1.8],
       [ 6.2,  2.8,  4.8,  1.8],
       [ 6.1,  3. ,  4.9,  1.8],
       [ 6.4,  2.8,  5.6,  2.1],
       [ 7.2,  3. ,  5.8,  1.6],
       [ 7.4,  2.8,  6.1,  1.9],
       [ 7.9,  3.8,  6.4,  2. ],
       [ 6.4,  2.8,  5.6,  2.2],
       [ 6.3,  2.8,  5.1,  1.5],
       [ 6.1,  2.6,  5.6,  1.4],
       [ 7.7,  3. ,  6.1,  2.3],
       [ 6.3,  3.4,  5.6,  2.4],
       [ 6.4,  3.1,  5.5,  1.8],
       [ 6. ,  3. ,  4.8,  1.8],
       [ 6.9,  3.1,  5.4,  2.1],
       [ 6.7,  3.1,  5.6,  2.4],
       [ 6.9,  3.1,  5.1,  2.3],
       [ 5.8,  2.7,  5.1,  1.9],
       [ 6.8,  3.2,  5.9,  2.3],
       [ 6.7,  3.3,  5.7,  2.5],
       [ 6.7,  3. ,  5.2,  2.3],
       [ 6.3,  2.5,  5. ,  1.9],
       [ 6.5,  3. ,  5.2,  2. ],
```

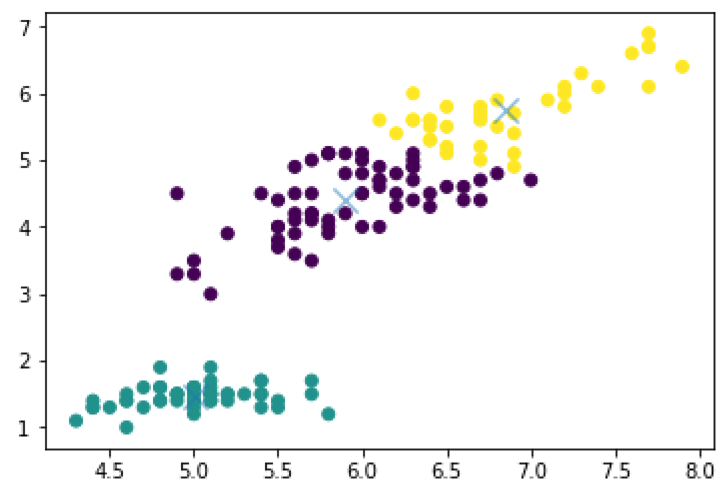
```
       [ 6.2,  3.4,  5.4,  2.3],
       [ 5.9,  3. ,  5.1,  1.8]])
```

In [4]:
```python
kmn = KMeans(n_clusters=3)
kmn.fit(iris.data)
labels = kmn.predict(iris.data)
```

In [5]:
```python
labels
```

Out[5]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 2, 2, 2, 0, 0,
       2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2, 2, 2,
       0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

In [6]:
```python
xs = iris.data[:,0]
ys = iris.data[:,2]

centroids = kmn.cluster_centers_

plt.scatter(xs, ys, c=labels)
plt.scatter(centroids[:,0],centroids[:,2],marker='x',s=150,alpha=0.5)
plt.show()
```



evaluation a clustering

> **inertia :**distance from each sample to centroids of its cluster or how spread out the clusters. (Lower is better)

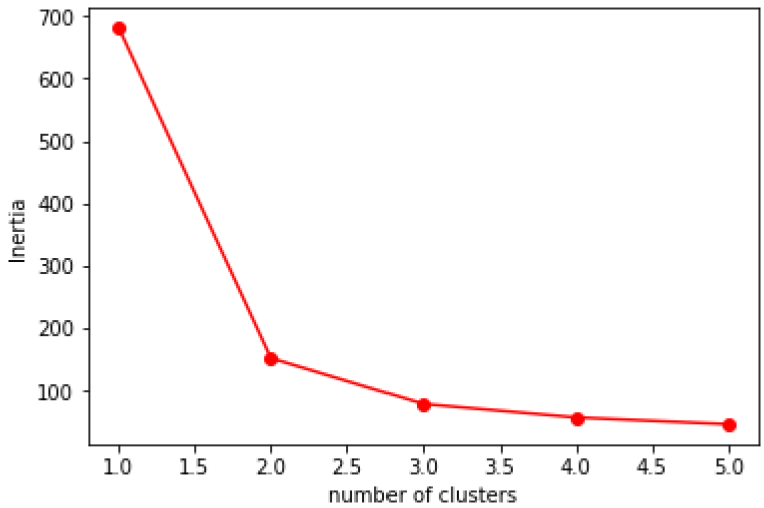
In [7]:
```python
print(kmn.inertia_)
```

```
78.9408414261
```

In [8]:
```python
inertia_list = []

for k in np.arange(1, 6):
    kmn = KMeans(n_clusters=k)
    kmn.fit(iris.data)
    inertia_list.append(kmn.inertia_)

inertia_list
```

Out[8]:
```
[680.82439999999997,
 152.36870647733906,
 78.940841426146022,
 57.345409315718165,
 46.535582051282049]
```

In [9]:
```
plt.plot(np.arange(1,6),inertia_list,'ro-')
plt.xlabel('number of clusters')
plt.ylabel('Inertia')
plt.show()
```



Hierarchical Clustering

Iran

India

France

Italy

Greece

- every country begins in a seperate cluster .
- at each step , the two closest clusters are merged .
- continue until all countries in a single cluster .

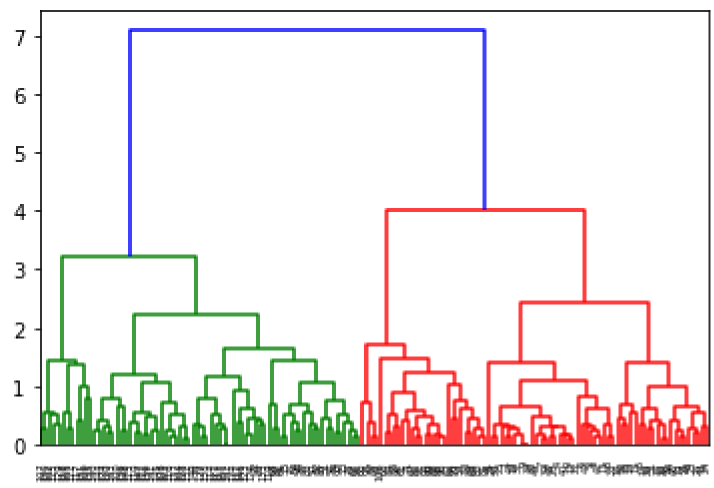
dendrogram.png

> **Wikipedia :**
> a dendrogram (from Greek dendro "tree" and gramma "drawing") is a tree diagram frequently used to illustrate the arrangement of the clusters produced by hierarchical clustering.

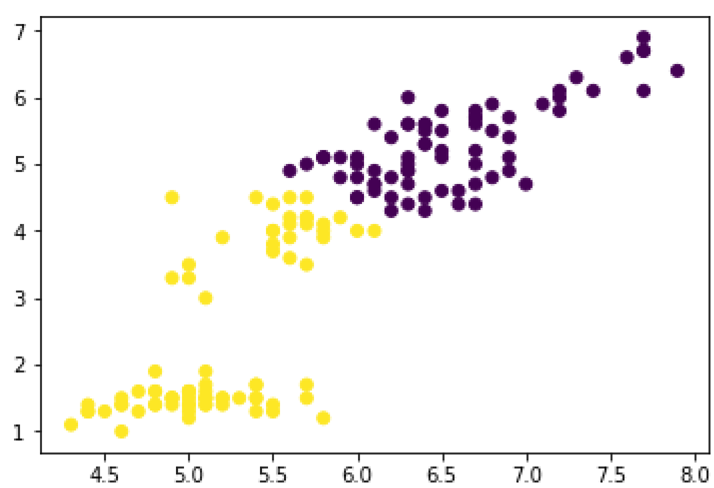
This is a "bottom up" approach called **Agglomerative**.

In [10]:
```
from scipy.cluster.hierarchy import linkage,dendrogram,fcluster
import matplotlib.pyplot as plt

hir = linkage(iris.data,method='complete')

labels = fcluster(hir, 6, criterion='distance')

print(labels)

dendrogram(hir,leaf_rotation=90)
plt.show()
```

```
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 1 2 1 2 1 2 2 2 2 1 2 1 2 2 1 2 1 2 1 1
 1 1 1 1 1 2 2 2 2 1 2 1 1 1 2 2 2 1 2 2 2 2 2 1 2 2 1 1 1 1 1 1 2 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1]
```



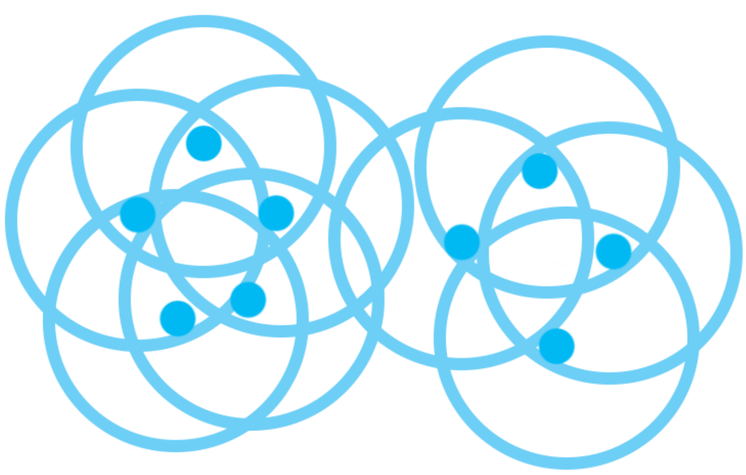
In [11]:
```
plt.scatter(iris.data[:,0], iris.data[:,2], c=labels)
plt.show()
```



Meanshift

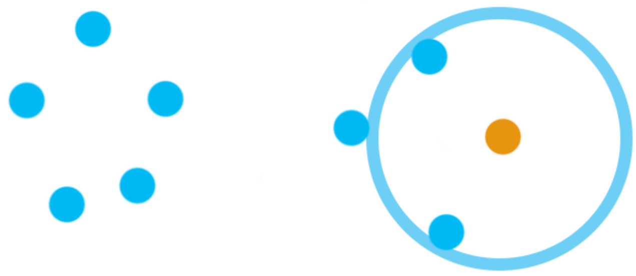
MeanShift

MeanShift



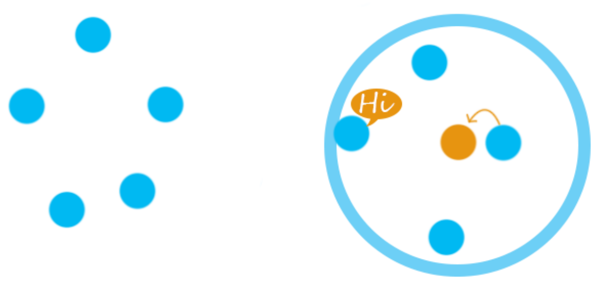
Bandwidth

MeanShift

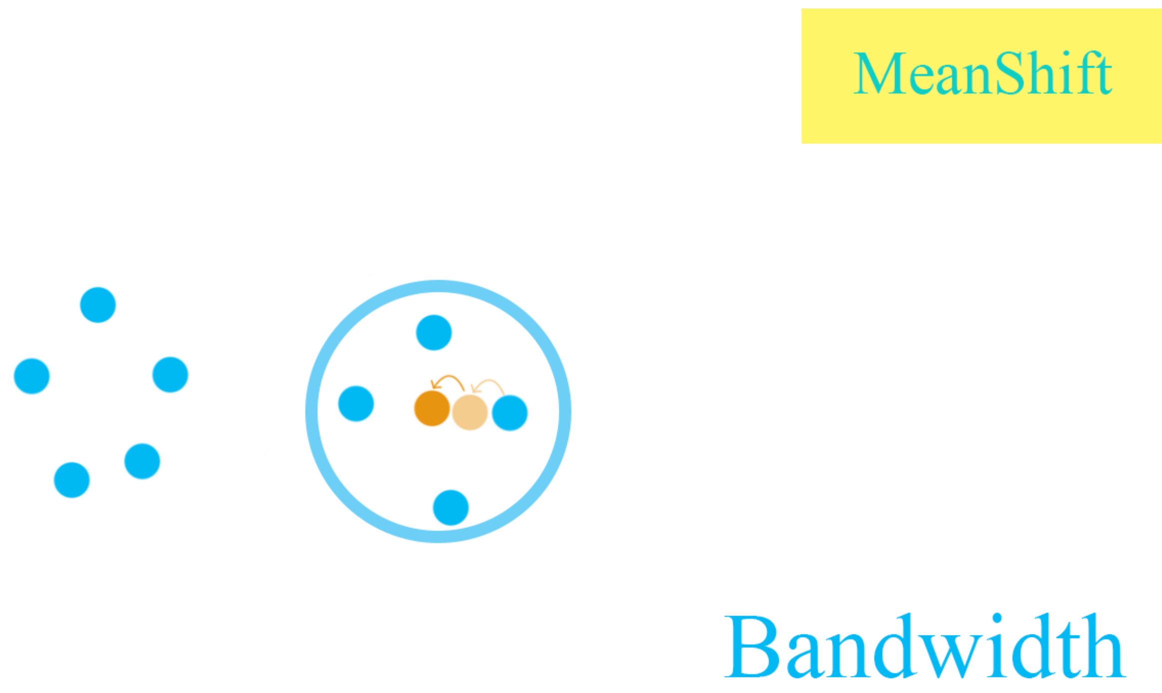


Bandwidth

MeanShift



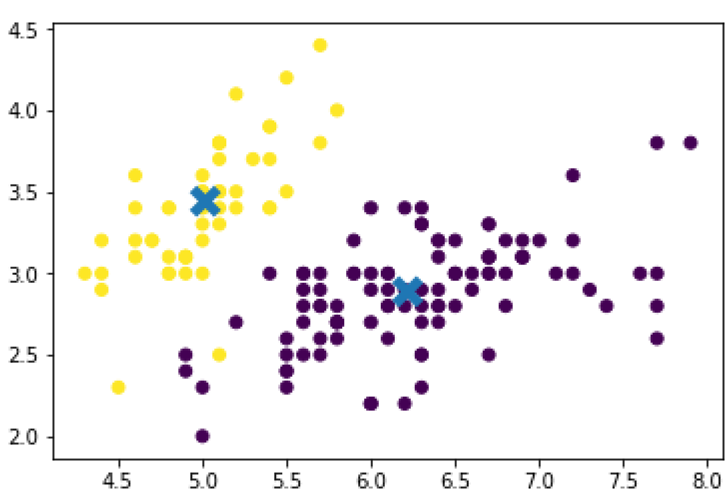
Bandwidth



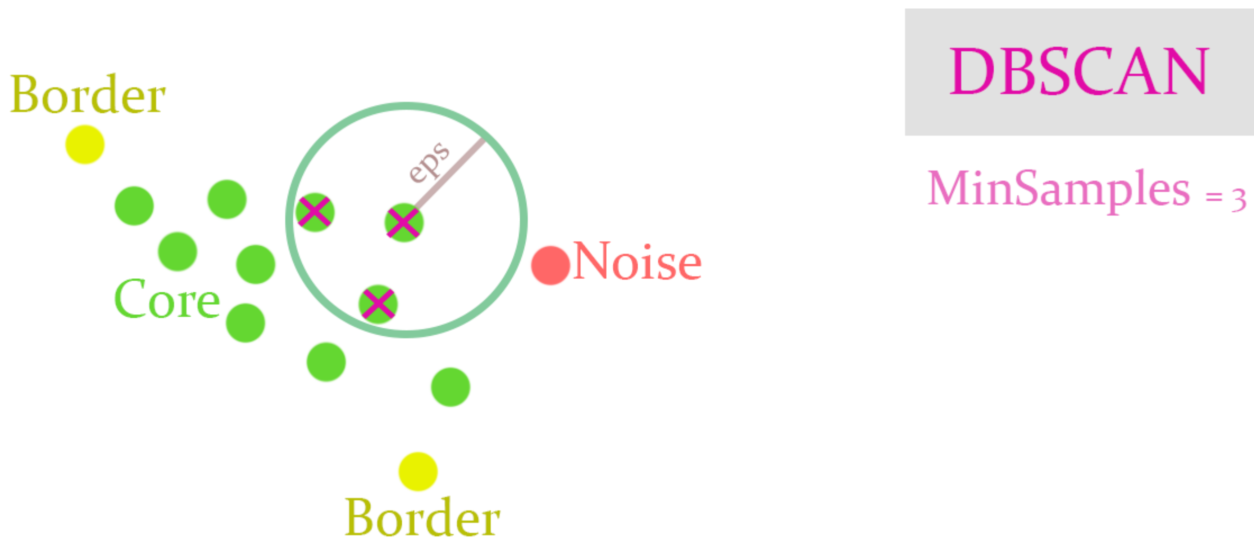
In [12]:
```
from sklearn.cluster import MeanShift
```

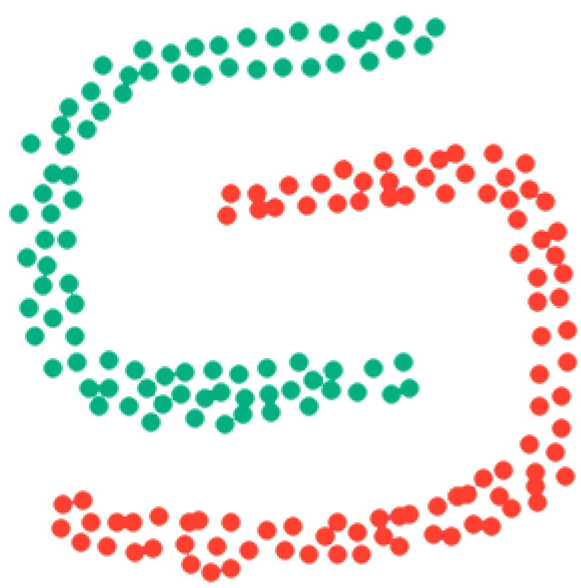
In [13]:
```
x = iris.data
ms = MeanShift()
ms.fit(x)
labels = ms.labels_

cluster_center = ms.cluster_centers_
n_cluster = len(np.unique(labels))

print('Number of estimated cluster:' ,n_cluster)


plt.scatter(x[:,0], x[:,1], c=labels)
plt.scatter(cluster_center[:,0], cluster_center[:,1], marker='x', s=150, linewidth=5, zorder=10)
plt.show()
```

```
Number of estimated cluster: 2
```



DBSCAN : Density-Based Spatial Clustering of Applications with Noise

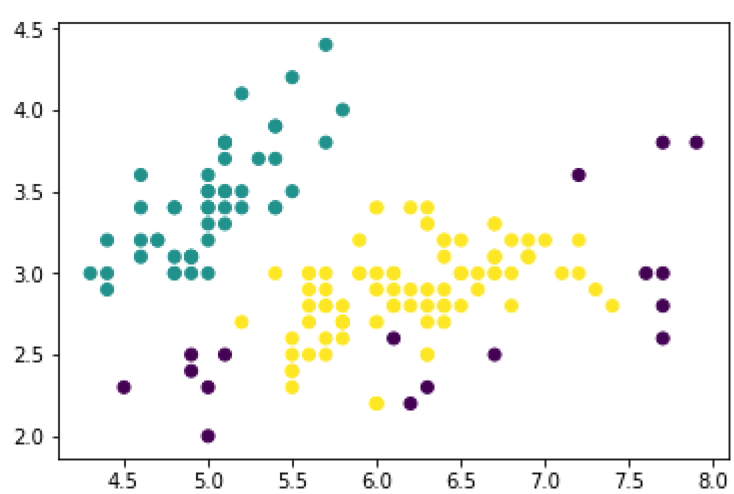




In [14]:
```
from sklearn.cluster import DBSCAN
```

In [15]:
```
dbscan = DBSCAN()
dbscan.fit(iris.data)
labels = dbscan.labels_
```

In [16]:
```
xx = iris.data[:,0]
yy = iris.data[:,1]

plt.scatter(xx,yy, c=labels)
plt.show()
```



Dimensity Reduction