K Means by Yunsu Han

K-means Algorithm

K-means algorithm is an unsupervised learning. It requires the input of the sample, but not the output of it. The steps are as follows:

1. pick random centroids
2. separate into regions
3. rescale the centroids
4. repeat the process until the centroids do not move

Import Packages

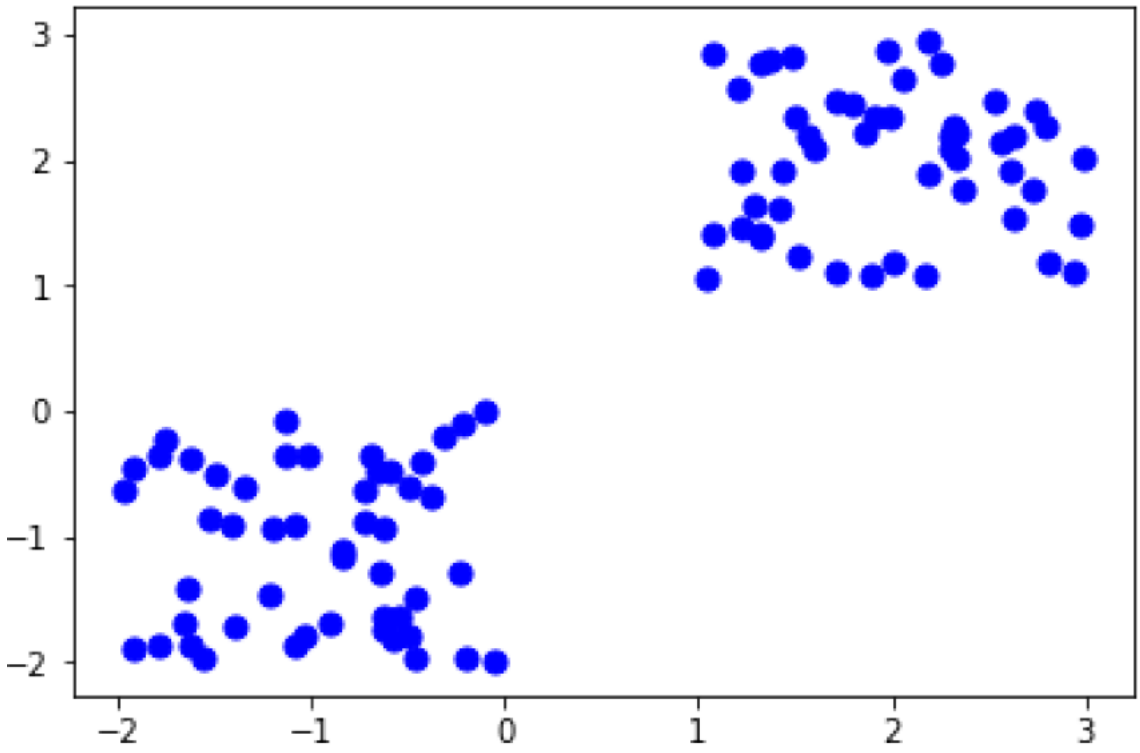
The following code imports several necessary packages for K-means algorithm. Panda and NumPy are packages that most machine learning programs use. Matplotlib allows the programmer to visualize the result in scatterplots. K means from sklearn package is the most important one in this program; it allows the programmer to use the K-means algorithm itself.

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.cluster import KMeans
        %matplotlib inline
```

Create Groups

First, create 50 dots each for two different groups. Since K-means algorithm is an unsupervised learning, do not indicate which group is which. Only the input data is used in this algorithm. Visualize the dots by using matplotlib's scatter function.

```
In [2]: X= -2 * np.random.rand(100,2)
        X1 = 1 + 2 * np.random.rand(50,2)
        X[50:100, :] = X1
        plt.scatter(X[ : , 0], X[ :, 1], s = 50, c = 'b')
        plt.show()
```



Compute the Clustering

May have noticed -- the letter "k" in "K-means algorithm" is a number of the clusters. In this program, set the value of k as 2. Then, the "fit" function computes k-mean clustering.

```
In [3]: from sklearn.cluster import KMeans
        Kmean = KMeans(n_clusters=2)
        Kmean.fit(X)

Out[3]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
            n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
            random_state=None, tol=0.0001, verbose=0)
```

Find the Coordinates

"cluster_centers_" are coordinates of cluster centers. If the algorithm stops before fully converging, these will not be consistent with labels_.

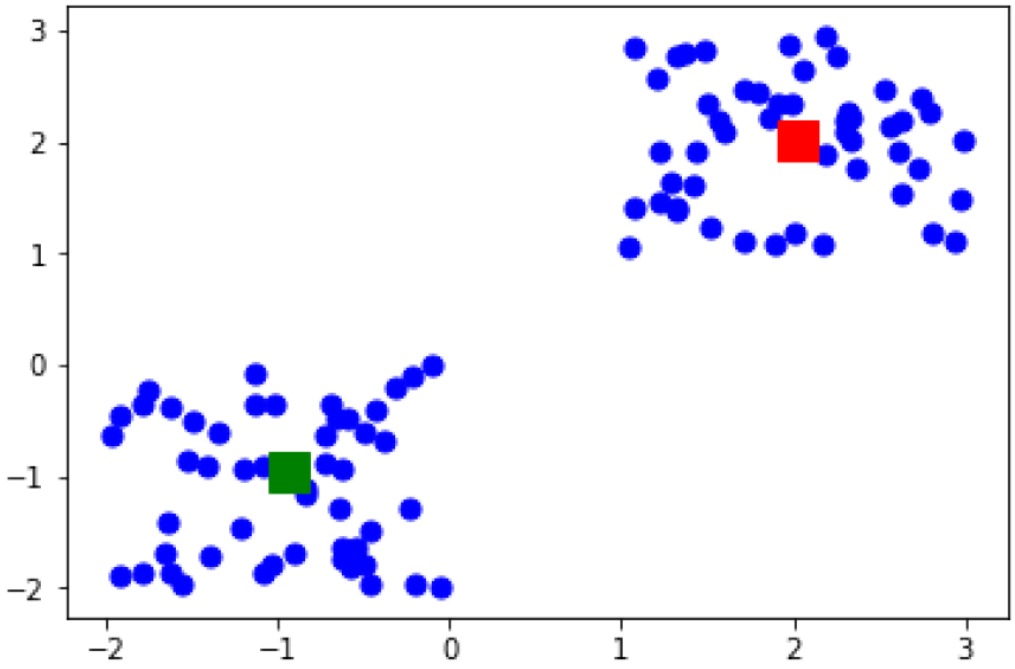
```
In [4]: Kmean.cluster_centers_

Out[4]: array([[-0.97779583, -1.06746949],
               [ 1.96953448,  1.99754973]])
```

Visualize

Lastly, visualize the dots by using two different colors. This is not necessary, but, in order to show the results to non-experts, a scatterplot is the best example to show the results right away.

```
In [5]: plt.scatter(X[ : , 0], X[ : , 1], s =50, c='b')
        plt.scatter(-0.94665068, -0.97138368, s=200, c='g', marker='s')
        plt.scatter(2.01559419, 2.02597093, s=200, c='r', marker='s')
        plt.show()
```



Label

"labels_" function labels each point either 0 or 1 since we created two groups beforehand.

```
In [6]: Kmean.labels_

Out[6]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0, 0, 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])
```

Test

This runs a second test. It tests where the coordinate (-3.0, -3.0) is included in: 0 or 1? It prints out that it is included in group 0, which is obviously correct.

```
In [7]: sample_test=np.array([-3.0,-3.0])
        second_test=sample_test.reshape(1, -1)
        Kmean.predict(second_test)

Out[7]: array([0])
```

Full Code

In [8]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
%matplotlib inline
X= -2 * np.random.rand(100,2)
X1 = 1 + 2 * np.random.rand(50,2)
X[50:100, :] = X1
plt.scatter(X[ : , 0], X[ :, 1], s = 50, c = 'b')
plt.show()
from sklearn.cluster import KMeans
Kmean = KMeans(n_clusters=2)
Kmean.fit(X)
Kmean.cluster_centers_
plt.scatter(X[ : , 0], X[ : , 1], s =50, c='b')
plt.scatter(-0.94665068, -0.97138368, s=200, c='g', marker='s')
plt.scatter(2.01559419, 2.02597093, s=200, c='r', marker='s')
plt.show()
Kmean.labels_
sample_test=np.array([-3.0,-3.0])
second_test=sample_test.reshape(1, -1)
Kmean.predict(second_test)
```