K-Means Clustering Implementation in Python

Python notebook using data from Iris Species · 48,241 views · 1y ago

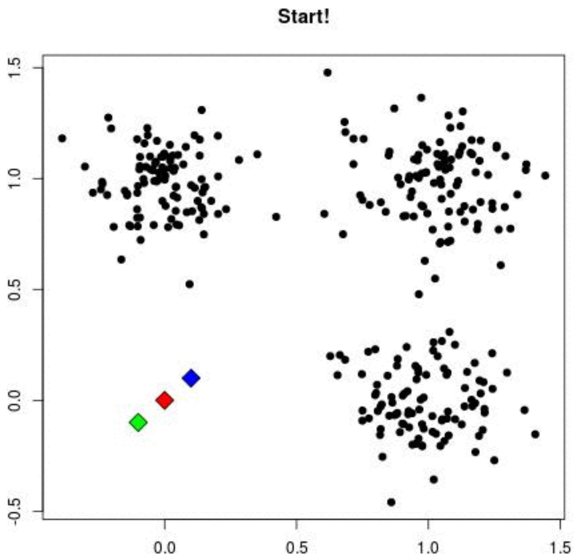
In [1]:

```
# Import necessary libraries
from copy import deepcopy
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_c
sv)
from matplotlib import pyplot as plt
```

K-Means Clustering

This work is based on Mubaris' great work (https://mubaris.com/2017/10/01/kmeans-clustering-in-python/ (https://mubaris.com/2017/10/01/kmeans-clustering-in-python/)).

A description of the algorithm can be found:
https://github.com/andrewxiechina/DataScience/blob/master/K-Means/cs229-notes7a%202.pdf
(https://github.com/andrewxiechina/DataScience/blob/master/K-Means/cs229-notes7a%202.pdf)



Generate Random Data

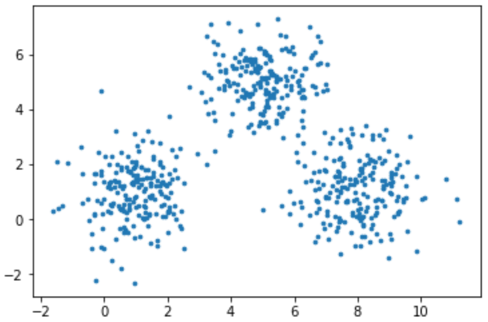
Generate random data normally distributed around 3 centers, with a noise.

In [2]:

```
# Set three centers, the model should predict similar results
center_1 = np.array([1,1])
center_2 = np.array([5,5])
center_3 = np.array([8,1])

# Generate random data and center it to the three centers
data_1 = np.random.randn(200, 2) + center_1
data_2 = np.random.randn(200,2) + center_2
data_3 = np.random.randn(200,2) + center_3

data = np.concatenate((data_1, data_2, data_3), axis = 0)

plt.scatter(data[:,0], data[:,1], s=7)
```

Out[2]:

```
<matplotlib.collections.PathCollection at 0x7fcfbc78d198>
```



Create K-Means Algorithm

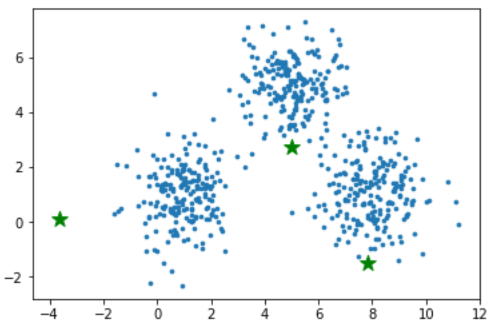
Generate random data normally distributed around 3 centers, with a noise.

In [3]:

```
# Number of clusters
k = 3
# Number of training data
n = data.shape[0]
# Number of features in the data
c = data.shape[1]

# Generate random centers, here we use sigma and mean to ensure it r
epresent the whole data
mean = np.mean(data, axis = 0)
std = np.std(data, axis = 0)
centers = np.random.randn(k,c)*std + mean

# Plot the data and the centers generated as random
plt.scatter(data[:,0], data[:,1], s=7)
plt.scatter(centers[:,0], centers[:,1], marker='*', c='g', s=150)
```

Out[3]:

```
<matplotlib.collections.PathCollection at 0x7fcfbc7253c8>
```



In [4]:

```
centers_old = np.zeros(centers.shape) # to store old centers
centers_new = deepcopy(centers) # Store new centers

data.shape
clusters = np.zeros(n)
distances = np.zeros((n,k))

error = np.linalg.norm(centers_new - centers_old)

# When, after an update, the estimate of that center stays the same,
exit loop
```

```
while error != 0:
    # Measure the distance to every center
    for i in range(k):
        distances[:,i] = np.linalg.norm(data - centers[i], axis=1)
    # Assign all training data to closest center
    clusters = np.argmin(distances, axis = 1)

    centers_old = deepcopy(centers_new)
    # Calculate mean for every cluster and update the center
    for i in range(k):
        centers_new[i] = np.mean(data[clusters == i], axis=0)
    error = np.linalg.norm(centers_new - centers_old)
centers_new
```

Out[4]:

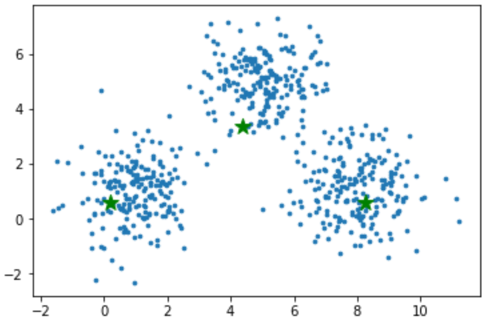
```
array([[ 0.17597931,  0.59153053],
       [ 8.25327096,  0.58760236],
       [ 4.37760992,  3.37202415]])
```

In [5]:

```
# Plot the data and the centers generated as random
plt.scatter(data[:,0], data[:,1], s=7)
plt.scatter(centers_new[:,0], centers_new[:,1], marker='*', c='g',
s=150)
```

Out[5]:

```
<matplotlib.collections.PathCollection at 0x7fcfbc69b080>
```



Test on Iris Dataset

In [6]:

```
df = pd.read_csv("../input/Iris.csv") #load the dataset
df.drop('Id',axis=1,inplace=True) # Se elimina la columna no requer
ida
```

In [7]:

```
df.head()
```

Out[7]:

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [8]:

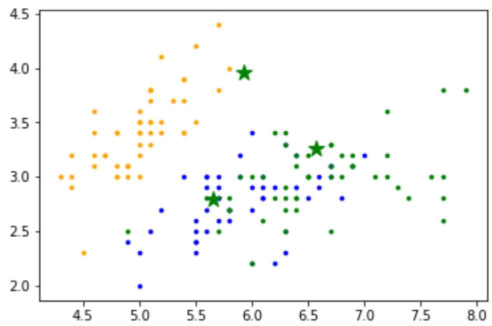
```
# Change categorical data to number 0-2
df["Species"] = pd.Categorical(df["Species"])
df["Species"] = df["Species"].cat.codes
# Change dataframe to numpy matrix
data = df.values[:, 0:4]
category = df.values[:, 4]
```

In [9]:

```
# Number of clusters
k = 3
# Number of training data
n = data.shape[0]
# Number of features in the data
c = data.shape[1]

# Generate random centers, here we use sigma and mean to ensure it r
epresent the whole data
mean = np.mean(data, axis = 0)
std = np.std(data, axis = 0)
centers = np.random.randn(k,c)*std + mean

# Plot the data and the centers generated as random
colors=['orange', 'blue', 'green']
for i in range(n):
    plt.scatter(data[i, 0], data[i,1], s=7, color = colors[int(cat
egory[i])])
plt.scatter(centers[:,0], centers[:,1], marker='*', c='g', s=150)
```

Out[9]:

```
<matplotlib.collections.PathCollection at 0x7fcfbc4e1ba8>
```



In [10]:

```
centers_old = np.zeros(centers.shape) # to store old centers
centers_new = deepcopy(centers) # Store new centers

data.shape
clusters = np.zeros(n)
distances = np.zeros((n,k))

error = np.linalg.norm(centers_new - centers_old)

# When, after an update, the estimate of that center stays the same,
exit loop
while error != 0:
    # Measure the distance to every center
    for i in range(k):
        distances[:,i] = np.linalg.norm(data - centers[i], axis=1)
    # Assign all training data to closest center
    clusters = np.argmin(distances, axis = 1)

    centers_old = deepcopy(centers_new)
    # Calculate mean for every cluster and update the center
    for i in range(k):
        centers_new[i] = np.mean(data[clusters == i], axis=0)
    error = np.linalg.norm(centers_new - centers_old)
```

```
    error = np.linalg.norm(centers_new - centers_old)
centers_new
```

Out[10]:

```
array([[ 4.4       ,  2.76666667,  1.23333333,  0.2       ],
       [ 6.49090909,  2.92597403,  5.17662338,  1.80779221],
       [ 5.19285714,  3.20714286,  2.30714286,  0.57142857]])
```

In [11]:

```
# Plot the data and the centers generated as random
```


Did you find this Kernel useful?
Show your appreciation with an upvote

24

Data

Data Sources

- Iris Species
  - Iris.csv 150 x 6
  - database.sqlite
    - Iris 150 x 6

Iris Species

Classify iris plants into three species in this classic dataset

Last Updated: 3 years ago (Version 2)

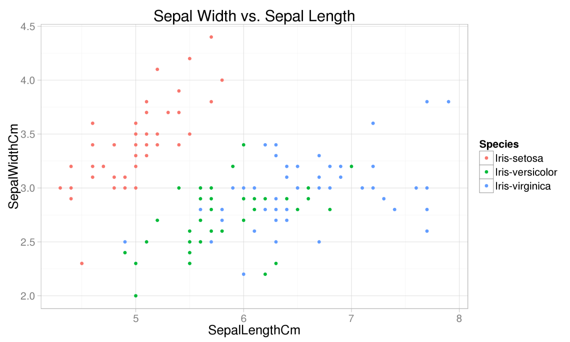
About this Dataset

The Iris dataset was used in R.A. Fisher's classic 1936 paper, The Use of Multiple Measurements in Taxonomic Problems, and can also be found on the UCI Machine Learning Repository.

It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

The columns in this dataset are:

- Id
- SepalLengthCm
- SepalWidthCm
- PetalLengthCm
- PetalWidthCm
- Species



Run Info

Succeeded	True	Run Time	10.9 seconds
Exit Code	0	Queue Time	0 seconds
Docker Image Name	kaggle/python (Dockerfile)	Output Size	0
Timeout Exceeded	False	Used All Space	False

Failure Message

Log

Download Log

```
Time   Line #  Log Message
4.9s        1  [NbConvertApp] Converting notebook script.ipynb to html
4.9s        2  [NbConvertApp] Executing notebook with kernel: python3
10.8s       3  [NbConvertApp] Support files will be in __results___files/
               [NbConvertApp] Making directory __results___files
10.9s       4  [NbConvertApp] Making directory __results___files
               [NbConvertApp] Making directory __results___files
               [NbConvertApp] Making directory __results___files
               [NbConvertApp] Making directory __results___files
               [NbConvertApp] Writing 282768 bytes to __results__.html
10.9s       5
10.9s       7  Complete. Exited with code 0.
```

Comments (7)

Sort by

All Comments | Hotness

Click here to comment...

Anurag Pand... • Posted on Latest Version • 3 months ago • Options • Reply 0

'distances[:,i] = np.linalg.norm(data - centers[i], axis=1)'
In the evaluation of new distance, shouldn't we be considering the new centers for evaluating the distances??
Or else we would have same distance everytime.

> **Leandro Li...** • Posted on Latest Version • 25 days ago • Options • Reply 0
>
> You are right. This line should be:
>
> `distances[:,i] = np.linalg.norm(data - centers_new[i], axis=1)`

Soner Paycı • Posted on Latest Version • 5 months ago • Options • Reply 0

Thanks for all!

sunstary • Posted on Latest Version • 5 months ago • Options • Reply 0

After calculating K -means value, how do we know which record is in which cluster in python?

> **Soner Paycı** • Posted on Latest Version • 5 months ago • Options • Reply 0
>
> Hi there,
> I just made additions for reaching cluster info. You can check it from Forks tab.

Dave • Posted on Latest Version • 6 months ago • Options • Reply 0

thanks

Leonardo Fer... • Posted on Latest Version • a year ago • Options • Reply

0

Nice one