Khoi Duong

Prof. Yang

CS483L

8/17/2022

FINAL

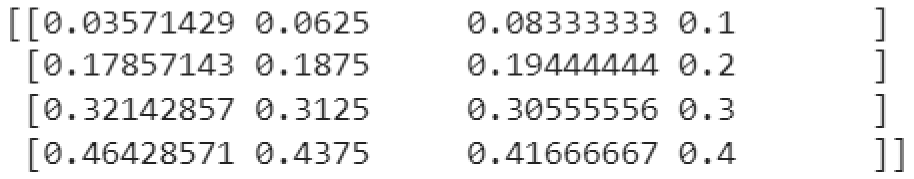
Part 1

1.

Source code:

```
# normalize matrix x by scikit learn functions
from sklearn.preprocessing import normalize
x = [[1,2,3,4], [5,6,7,8], [9,10,11,12], [13,14,15,16]]
x_norm = normalize(x, axis=0, norm='l1')
print(x_norm)
```

Run program & result:



```
[[0.03571429 0.0625     0.08333333 0.1       ]
 [0.17857143 0.1875     0.19444444 0.2       ]
 [0.32142857 0.3125     0.30555556 0.3       ]
 [0.46428571 0.4375     0.41666667 0.4       ]]
```

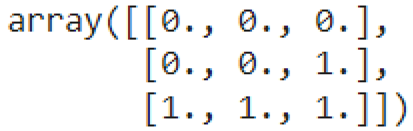
2.

Source code:

```
#Binarize matrix x by scikit learn function(s) with threshold value = 5.5
from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold=5.5)
x = [[1.1,2.2,3.3], [4.4,5.5,6.6], [7.7,8.8,9.9]]
```

```
binarizer.fit(x)
binarizer.transform(x)
```

Run program & result:



```
array([[0., 0., 0.],
       [0., 0., 1.],
       [1., 1., 1.]])
```

3.

X = [[6], [8], [10], [14], [18]]

y = [[7], [9], [13], [17.5], [18]]

Source code:

```
from sklearn.linear_model import LinearRegression
# Training data
x = [[6], [8], [10], [14], [18]]
y = [[7], [9], [13], [17.5], [18]]
# Create and fit the model
model=LinearRegression() # model created
model.fit(x,y) # model fit
#  find coefficients in 1st order linear hypothesis function
#  y = ax + b
a = model.coef_[0][0]
a
```

Run program & result:

```
0.9762931034482755
```

X_test = [[8], [9], [11], [16], [12]]

y_test = [[11], [8.5], [15], [18], [11]]

Source code:

```
from sklearn.linear_model import LinearRegression
# Training data
```

```
x_test = [[8], [9], [11], [16], [12]]
y_test = [[11], [8.5], [15], [18], [11]]
# Create and fit the model
model=LinearRegression() # model created
model.fit(x_test,y_test) # model fit
#  find coefficients in 1st order linear hypothesis function
#  y = ax + b
a = model.coef_[0][0]
a
```

Run program & result:

```
0.9871134020618553
```

4.

We know that N = 12, $k = \sqrt{12} \approx 3$

Source code:

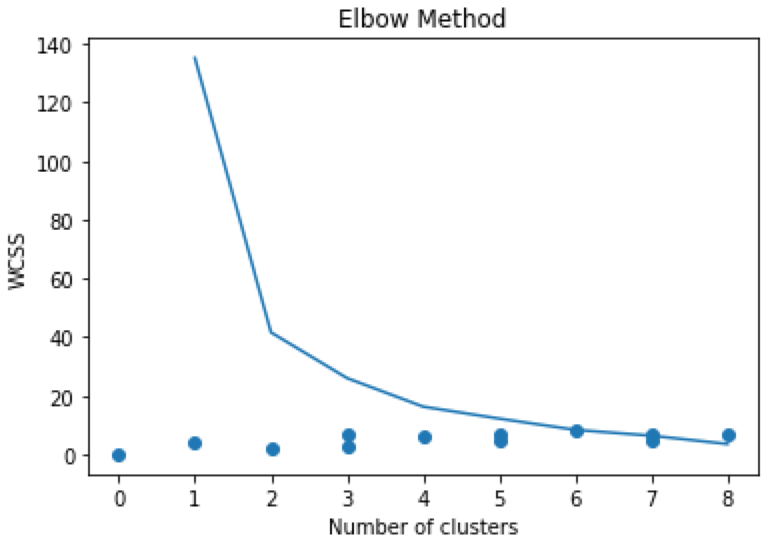
```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import KMeans

X = np.array([[7,5],
              [5,7],
              [7,7],
              [3,3],
              [4,6],
              [1,4],
              [0,0],
              [2,2],
              [8,7],
              [6,8],
              [5,5],
              [3,7]])
y = np.array([0, 2, 1, 0, 1, 1, 2, 0, 1, 2, 0, 1])

plt.scatter(X[:,0], X[:,1])
print(y.shape, X.shape)
```

```
wcss = []
for i in range(1, 9):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 9), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
print(wcss)
```

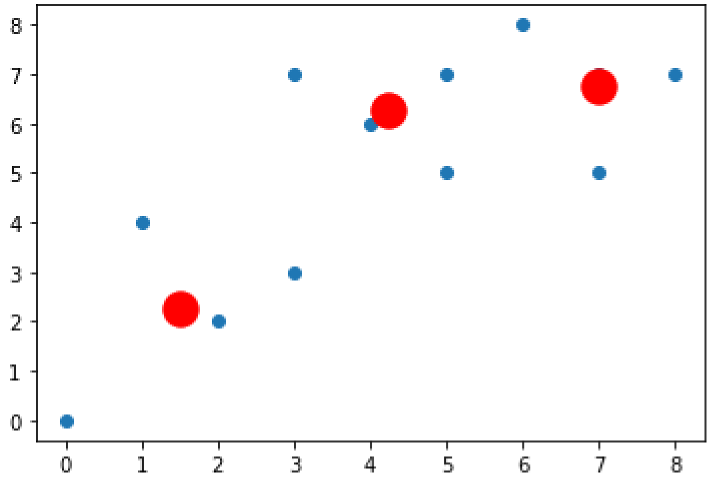
```
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, n_init=10, random_state=0)
pred_y = kmeans.fit_predict(X)
plt.scatter(X[:,0], X[:,1])
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='red')
plt.show()
```

Run program & result:

```
(12,) (12, 2)
```



```
[135.16666666666666, 41.625, 26.0, 16.25, 12.166666666666666,
8.333333333333332, 6.333333333333334, 3.5]
```



Part 2

5.

Source code:

```
import numpy as np

x = [1,2,3]
y = np.matrix([[4],[5],[6]])
x*y
```

Run program & result: `matrix([[32]])`

6.

Source code:

```
import numpy as np
x = np.matrix([[1,2,3], [4,5,6]])
# transpose the matrix
print(x.T)
```

Run program & result:

```
[[1 4]
 [2 5]
 [3 6]]
```