

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

سهیل عبداللہی ۹۸۲۳۰۶۳

مہشید جعفری ۹۹۲۳۰۱۵

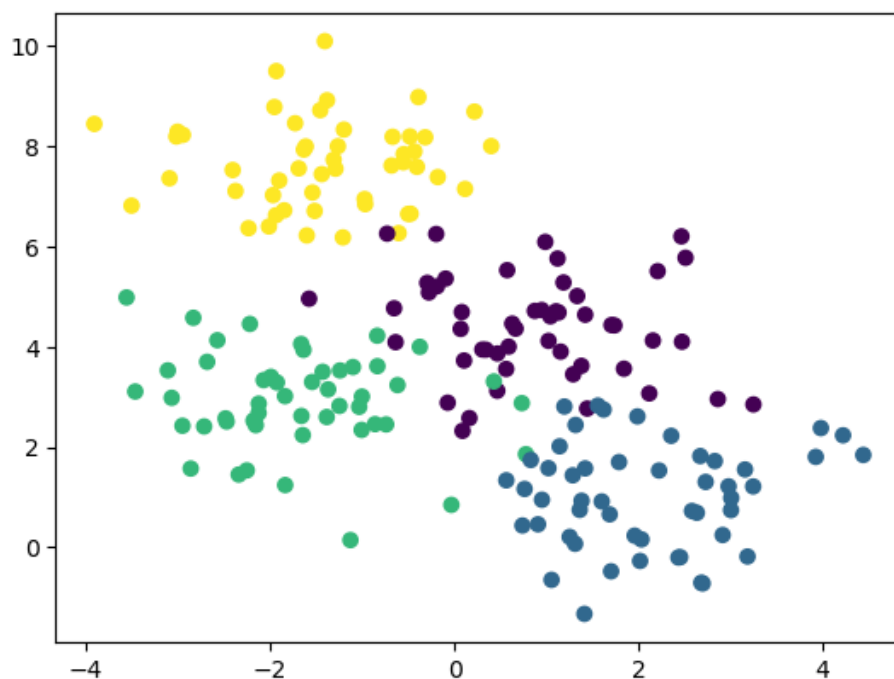
تمرین چہارم از ہوش محاسباتی

استاد: جناب حیدریان

سوال (۱)

کتابخانه های لازم را ایمپورت کرده و بکمک دستور `make_blobs` تعداد ۲۰۰ نقطه در ۴ دسته تولید میکنیم و نتیجه را میبینیم:

```
import numpy as np
from sklearn.datasets import make_blobs
X, y = make_blobs(n_samples=200, centers=4, n_features=2, random_state=0)
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y)
```



حال کلاس `kmeans` را ایجاد و مدل را طراحی میکنیم. کلاس `kmeans` را ایجاد کرده و در متد `init`، متغیر های تعداد کلاس و تعداد دفعات ترین و خطا و لیست ارورها را تعریف میکنیم.

متد بعدی، **fit** است که توضیح آن به شرح زیر است:

از بین داده های ورودی، ۴ داده بصورت رندوم، بعنوان مرکز خوشه انتخاب میشوند.

همچنین داده ها را در متغیر دیگری به اسم **picture** میریزیم. حال در حلقه تکرار

آموزش، متغیر **cluster** را که لیستی شامل ۴ لیست خالی است را میسازیم.

سپس در حلقه دیگری، درایه هارا در لیستی میریزیم که نزدیک ترین فاصله به مرکز

خوشه آن را دارد. نهایتا میانگین اعضای هر لیست را بعنوان مرکز دسته جدید در نظر

میگیریم و این کار تکرار میشود.

در هر مرحله، فاصله درایه ها از مرکز دسته آن مرحله را در لیست **error** میریزیم.

سپس میانگین ارور های هر دسته را محاسبه کرده و در به لیست **all_error** اضافه

میکنیم.

```
class kmeans:
    def __init__(self,n_classes):
        self.number_of_classes=n_classes
        self.max_iter=100
        self.tolorance=1e-8
        self.all_error=[[[]],[[]],[[]],[[]]]
    def fit(self,x):
        self.random_center_index=np.random.choice(range(len(x)),4)
        self.random_center=x[self.random_center_index]
        for j in range(self.max_iter):
            self.clustered=[[[]],[[]],[[]],[[]]]
            self.error=[[[]],[[]],[[]],[[]]]
            for i in range(200):
                distance=np.linalg.norm(self.random_center-x[i],axis=-1)
                minimum_index=np.argmin(distance)
                self.clustered[minimum_index].append(x[i])
                self.error[minimum_index].append(distance)
            for i in range(4):
                self.random_center[i]=np.mean(self.clustered[i], axis=0)
                self.error[i]=np.mean(self.error[i], axis=0)
                self.all_error[i].append(self.error[i][0])
```

حال از مدل استفاده میکنیم و ورودی ها را در آن فیت میکنیم. متغیر `all_error` میانگین خطا ها را در هر مرحله آموزش نشان میدهد. نهایتا نمودار میانگین خطا بر حسب تعداد دفعات آموزش را رسم میکنیم. مشاهده میشود که این میانگین به مقدار ثابتی میل میکند:



سوال ۲)

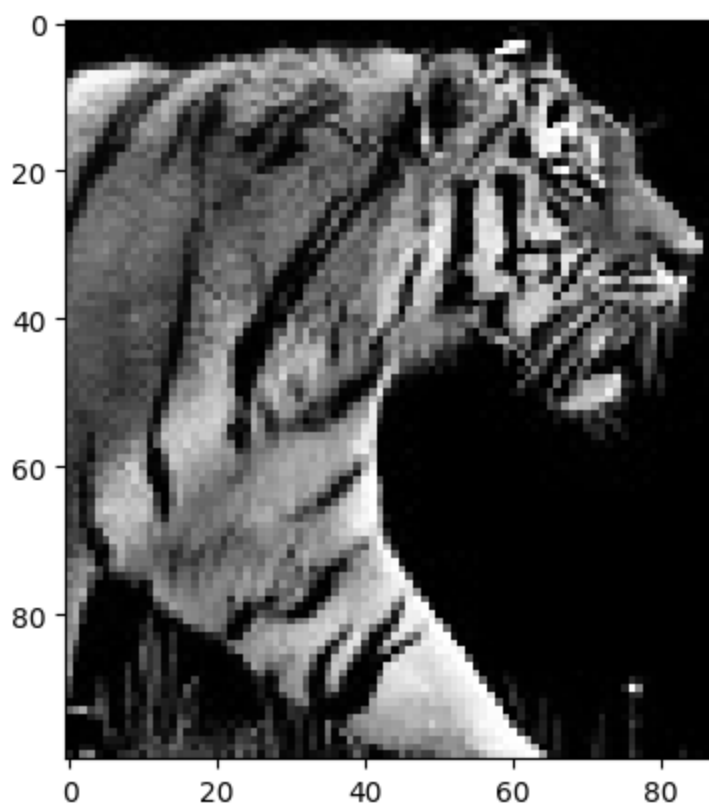
ابتدا کتابخانه های لازم را ایمپورت میکنیم:

```
import numpy as np
from sklearn.datasets import make_blobs
from matplotlib import pyplot as plt
import cv2
```

با استفاده از کتابخانه **opencv** عکس را خوانده و سیاه سفید میکنیم و نتیجه را

میبینیم:

```
picture1 = cv2.imread("./img4.jpg")
picture1 = cv2.cvtColor(picture1, cv2.COLOR_BGR2GRAY)
plt.imshow(picture1, cmap="gray")
```



```

[43] picture1.shape
... (100, 88)

[44] input_array=picture1.reshape(-1)

[45] len(input_array)
... 8800

```

ایعدا عکس را بررسی کرده و آن را به آرایه
ی تک بعدی تبدیل میکنیم. نهایتاً تعداد
درایه های آرایه را بررسی میکنیم:

تا اینجا عملیات preprocess انجام شد. حال باید مدل را طراحی کنیم.

کلاس **kmeans** را ایجاد کرده و در متد **init**، متغیرهای تعداد کلاس و تعداد دفعات
ترین و خطا را تعریف میکنیم:

```

class kmeans:
    def __init__(self, n_classes):
        self.number_of_classes = n_classes
        self.max_iter = 1000
        self.tolerance = 1e-8

```

متد بعدی، **fit** است که توضیح آن به شرح زیر است:

از بین داده های ورودی، ۱۶ داده بصورت رندوم، بعنوان مرکز خوشه انتخاب میشوند.

همچنین داده ها را در متغیر دیگری به اسم **picture** میریزیم. حال در حلقه تکرار
آموزش، متغیر **cluster** را که لیستی شامل ۱۶ لیست خالی است را میسازیم.

سپس در حلقه دیگری، درایه هارا در لیستی میریزیم که نزدیک ترین فاصله به مرکز خوشه آن را دارد. نهایتا میانگین اعضای هر لیست را بعنوان مرکز دسته جدید در نظر میگیریم و این کار تکرار میشود.

در مرحله آخر، درایه های متغیر **picture** را با نزدیک ترین مرکز خوشه جایگزین میکنیم.

```
def fit(self,x):
    self.random_center_index=np.random.choice(range(len(x)),16)
    self.random_center=x[self.random_center_index]
    self.picture=x
    for j in range(self.max_iter):
        self.clustered=[]
        for i in range(8800):
            distance=np.abs(self.random_center-x[i])
            minimum_index=np.argmin(distance)
            self.clustered[minimum_index].append(x[i])
        for i in range(16):
            self.random_center[i]=np.mean(self.clustered[i], axis=0)
    for i in range(8800):
        distance=np.abs(self.random_center-x[i]*np.ones(16))
        minimum_index=np.argmin(distance)
        self.picture[i]=self.random_center[minimum_index]
```

حال از مدل استفاده میکنیم. متغیر **picture** از مدل را درون متغیر **picture2** میریزیم و ابعاد آن را به ابعاد عکس بر میگردانیم و نتیجه را مشاهده میکنیم:

```
[48] model=kmeans(16)

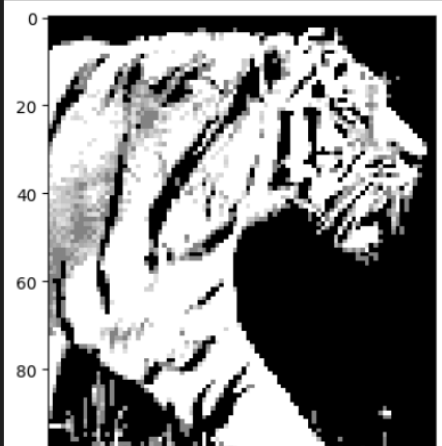
[49] model.fit(input_array)

... /usr/local/lib/python3.10/dist-packages/numpy/core/fromnumeric.py:3432: RuntimeWarning: Mean of empty slice.
    return _methods._mean(a, axis=axis, dtype=dtype,
/usr/local/lib/python3.10/dist-packages/numpy/core/_methods.py:190: RuntimeWarning: invalid value encountered in double_scalars
    ret = ret.dtype.type(ret / rcount)

[51] picture2=model.picture.reshape(100,88)

> ▾
[53] plt.imshow(picture2, cmap="gray")

... <matplotlib.image.AxesImage at 0x7c1941f038e0>

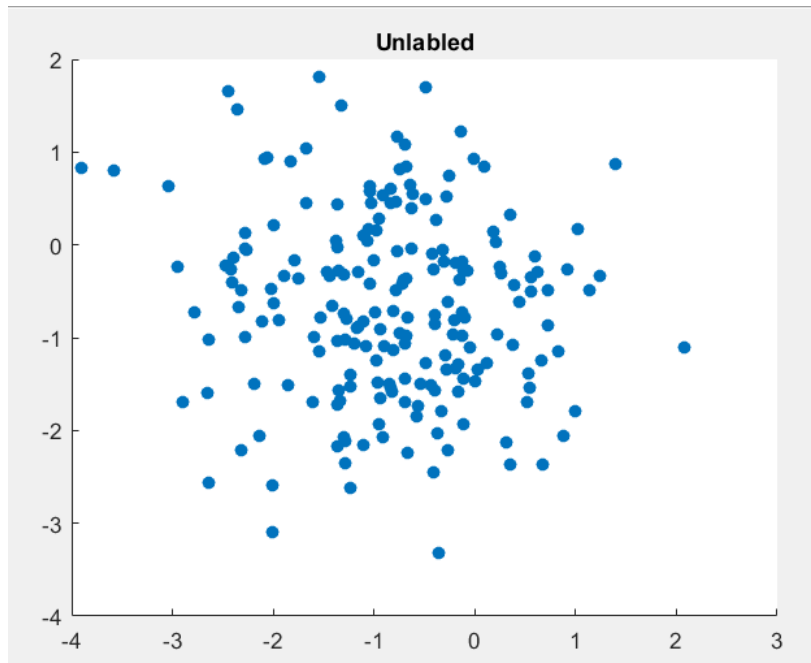
... 
```


سوال ۳)

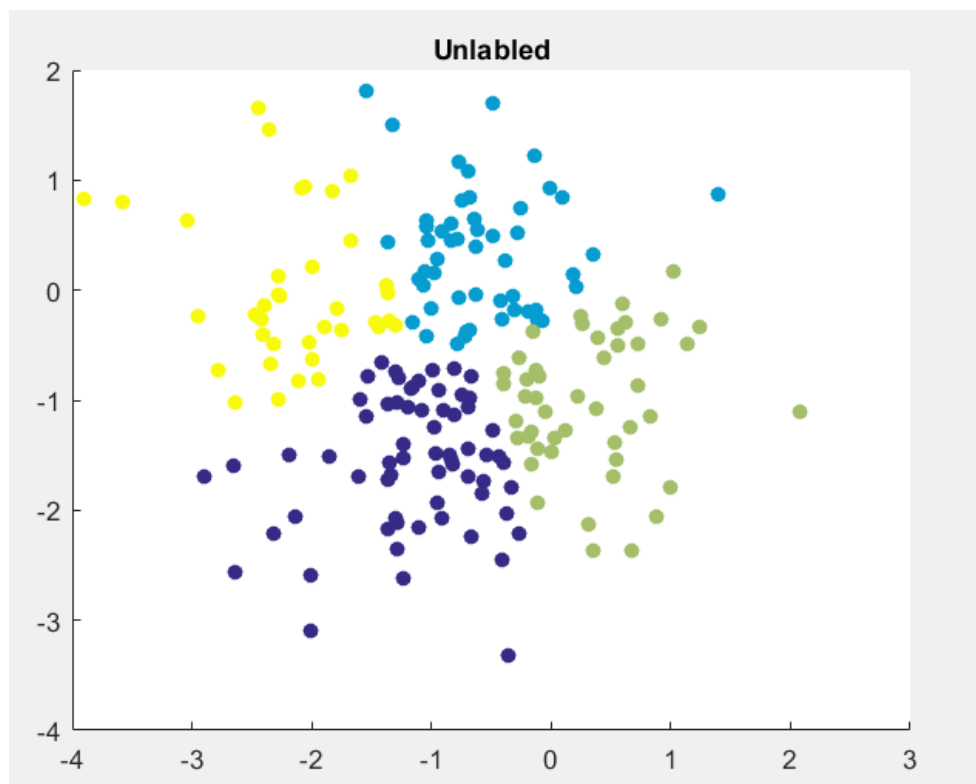
کد پیاده شده در سوال ۱ در متلب در تصویر زیر آورده شده است:

```
n_classes = 4;  
X = 200;  
data_location = randn(X, 2) + randn(1, 2);  
figure;  
scatter(data_location(:, 1), data_location(:, 2), 36, 'filled');  
title('Unlabeled');  
[idx, centers] = kmeans(data_location, n_classes);  
figure;  
scatter(data_location(:, 1), data_location(:, 2), 36, idx, 'filled');  
title('Unlabeled');
```

تعداد دسته ها همانند سوال ۱، ۴ دسته انتخاب شده است. همچنین ۲۰۰ نقطه را بررسی خواهیم کرد. موقعیت نقاط را در متغیر **data_location** بصورت رندوم انتخاب میکنیم. حال ابتدا نمودار نقاط بدون لیبل را میبینیم:



سپس بکمک تابع `kmeans` در متلب، داده ها را دسته بندی کرده و نتیجه را مجدداً میبینیم:



سوال (۴)

ابتدا باید اطلاعاتی درمورد شناسه مشتریان، سن، جنسیت و درآمد سالانه آنها از دیتاست به دست آورد. سپس باید آنها را بر اساس هر کدام از فاکتورها دسته بندی کنیم تا یک لیست دقیق برای ما به دست بیاید و بتوانیم راحت تر نمره خرید را دسته بندی کنیم.

پس از دسته بندی تمام فاکتورها براساس سن، جنسیت و نمره خرید باید الگوریتم `kmeans` را پیاده سازی کنیم تا متوجه شویم برای چه مشتریانی باید تبلیغ کرد تا فروش بیشتری داشت.

میتوان از کتابخانه ها و دستور های موجود در **Kaggle** استفاده کرد و الگوریتم خرید را باید پیاده سازی کرد.

حائز اهمیت است باید از کتابخانه **sklearn** برای پیاده سازی الگوریتم استفاده کنیم.

دستور های لازم عبارت است از:

plt.scatter()

kmeans.fit()

kmeans.fit_predict()

grid,plot,xticks

... و add_subplot()

مراحل زیر برای دسته بندی باید انجام شوند:

1. Data Preparation

2. Exploring the content of variables

- 2.1 Countries
- 2.2 Customers and products
 - 2.2.1 Cancelling orders
 - 2.2.2 StockCode
 - 2.2.3 Basket price

3. Insight on product categories

- 3.1 Product description
- 3.2 Defining product categories
 - 3.2.1 Data encoding
 - 3.2.2 Clusters of products
 - 3.2.3 Characterizing the content of clusters

4. Customer categories

- 4.1 Formating data
 - 4.1.1 Grouping products
 - 4.1.2 Time splitting of the dataset
 - 4.1.3 Grouping orders
- 4.2 Creating customer categories
 - 4.2.1 Data encoding
 - 4.2.2 Creating categories

5. Classifying customers

- 5.1 Support Vector Machine Classifier (SVC)
 - 5.1.1 Confusion matrix
 - 5.1.2 Learning curves
- 5.2 Logistic regression
- 5.3 k-Nearest Neighbors
- 5.4 Decision Tree
- 5.5 Random Forest
- 5.6 AdaBoost
- 5.7 Gradient Boosting Classifier
- 5.8 Let's vote !

6. Testing the predictions

7. Conclusion

Customer Segmentation

1154 Copy & Edit 5453

Table of Contents >

Customer segmentation

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|------------------|-----------|-----------|-------------|----------|----------------|-----------|------------|---------|
| column type | object | object | object | int64 | datetime64[ns] | float64 | object | object |
| null values (nb) | 0 | 0 | 1454 | 0 | 0 | 0 | 135080 | 0 |
| null values (%) | 0 | 0 | 0.268311 | 0 | 0 | 0 | 24.9267 | 0 |

| | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|---|-----------|-----------|-------------------------------------|----------|---------------------|-----------|------------|----------------|
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 2010-12-01 08:26:00 | 2.55 | 17850 | United Kingdom |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 2010-12-01 08:26:00 | 3.39 | 17850 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 2010-12-01 08:26:00 | 2.75 | 17850 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 2010-12-01 08:26:00 | 3.39 | 17850 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 2010-12-01 08:26:00 | 3.39 | 17850 | United Kingdom |

مرجع: سایت kaggle