COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

Week 19
Clustering Models

Applied Data Science

Columbia University - Columbia Engineering

Course Agenda

- Week 10: Organizing and Analyzing Data with NumPy and Pandas
- Week 11: Cleaning and Visualizing Data with Pandas and Matplotlib
- Week 12: Statistical Distributions
- Week 13: Statistical Sampling
- Week 14: Hypothesis Testing
- Week 15: Regression Models in Python
- Week 16: Evaluating Data Models
- Week 17: Classification with K-Nearest Neighbors
- Week 18: Decision Tree Models
- **Week 19: Clustering Models**
- Week 20: Text Mining in Python -- Analyzing Sentiment
- Week 21: Text Mining in Python -- Topic Modeling

COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

Unsupervised learning

- The algorithm tries to group similar data together (clusters)
- Using the values of the feature space

K-Means Clusterng

- partitions the dataspace into clusters
- minimizes distance between the mean of a cluster and the data points
- the desired number of clusters must be known in advance

Do imports

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.datasets import load_digits
from sklearn.preprocessing import scale
```

Load data

In [3]:
```
digits = load_digits()
digits
```

Out[3]:
```
{'DESCR': "Optical Recognition of Handwritten Digits Data Set\n===================
========================\n\nNotes\n-----\nData Set Characteristics:\n    :Number of
s: 5620\n    :Number of Attributes: 64\n    :Attribute Information: 8x8 image of i
xels in the range 0..16.\n    :Missing Attribute Values: None\n    :Creator: E. Al
lpaydin '@' boun.edu.tr)\n    :Date: July; 1998\n\nThis is a copy of the test set
I ML hand-written digits datasets\nhttp://archive.ics.uci.edu/ml/datasets/Optical+
on+of+Handwritten+Digits\n\nThe data set contains images of hand-written digits: 1
where\neach class refers to a digit.\n\nPreprocessing programs made available by N
used to extract\nnormalized bitmaps of handwritten digits from a preprinted form.
otal of 43 people, 30 contributed to the training set and different 13\nto the tes
x32 bitmaps are divided into nonoverlapping blocks of\n4x4 and the number of on pi
counted in each block. This generates\nan input matrix of 8x8 where each element i
```

Scaling Data to Normal Distribution

```
In [4]: data = scale(digits.data)

In [5]: data

Out[5]: array([[ 0.        , -0.33501649, -0.04308102, ..., -1.14664746,
                -0.5056698 , -0.19600752],
               [ 0.        , -0.33501649, -1.09493684, ...,  0.54856067,
                -0.5056698 , -0.19600752],
               [ 0.        , -0.33501649, -1.09493684, ...,  1.56568555,
                 1.6951369 , -0.19600752],
               ...,
               [ 0.        , -0.33501649, -0.88456568, ..., -0.12952258,
                -0.5056698 , -0.19600752],
               [ 0.        , -0.33501649, -0.67419451, ...,  0.8876023 ,
                -0.5056698 , -0.19600752],
               [ 0.        , -0.33501649,  1.00877481, ...,  0.8876023 ,
                -0.26113572, -0.19600752]])
```

Rendering the Digit and their Associated Values

COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

```
In [6]: def print_digits(images,y,max_n=10):
            # set up the figure size in inches
            fig = plt.figure(figsize=(12, 12))
            fig.subplots_adjust(left=0, right=1, bottom=0, top=1,
                    hspace=.05, wspace=.5)
            i = 0
            while i <max_n and i <images.shape[0]:
                # plot the images in a matrix of 20x20
                p = fig.add_subplot(20, 20, i + 1, xticks=[],
                        yticks=[])
                p.imshow(images[i], cmap=plt.cm.bone)
                # label the image with the target value
                p.text(0, 14, str(y[i]))
                i = i + 1
        print_digits(digits.images, digits.target, max_n=10)
```

Training and Testing Samples


```
In [9]: from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test, images_train,images_test = train_test_split(
                data, digits.target, digits.images,  test_size=0.25,
                  random_state=42)

        n_samples, n_features = X_train.shape
        n_digits = len(np.unique(y_train))
        labels = y_train
        labels

Out[9]: array([5, 2, 0, ..., 2, 7, 1])
```

```
In [ ]: len(np.unique(y_train))
```

```
In [10]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test, images_train,images_test = train_test_split(
                 data, digits.target, digits.images,  test_size=0.25,
                   random_state=42)

         n_samples, n_features = X_train.shape
         n_digits = len(np.unique(y_train))
         labels = y_train
         X_train

Out[10]: array([[ 0.        , -0.33501649, -0.67419451, ..., -1.14664746,
                 -0.5056698 , -0.19600752],
                [ 0.        ,  5.17802955,  2.2710018 , ..., -0.12952258,
                 -0.26113572, -0.19600752],
                [ 0.        , -0.33501649, -0.25345218, ..., -0.80760583,
                 -0.5056698 , -0.19600752],
                ...,
                [ 0.        , -0.33501649,  0.79840364, ...,  1.56568555,
                 -0.01660165, -0.19600752],
```

```
In [11]: len(np.unique(y_train))

Out[11]: 10
```

COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

```
In [12]: from sklearn import cluster
         clf = cluster.KMeans(init='k-means++',n_clusters=10, random_state=42)
         clf.fit(X_train)

Out[12]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
             n_clusters=10, n_init=10, n_jobs=1, precompute_distances='auto',
             random_state=42, tol=0.0001, verbose=0)
```

k-means++ runs an initializer before using the k-means algorithm

```
In [ ]: images_train
```

Cluster Numbers


```
In [ ]: print_digits(images_train, clf.labels_, max_n=20)
```

```
In [ ]: print_digits(images_train, clf.labels_, max_n=20)
```

```
In [14]: print_digits(images_train, clf.labels_, max_n=20)
```

1 3 2 0 6 8 6 2 3 3 8 8 8 6 8 4 8 8

Testing Samples


```
In [ ]: y_pred=clf.predict(X_test)
        y_pred
```

```
In [ ]: def print_cluster(images, y_pred, cluster_number):
            images = images[y_pred==cluster_number]
            y_pred = y_pred[y_pred==cluster_number]
            print_digits(images, y_pred,max_n=15)
        for i in range(10):
             print_cluster(images_test, y_pred, i)
```

- Adjusted rand index: A measure of the similarity between two groups
- We'll use it to see how similar the y_test actuals and predicted groupings are
- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html
- 0.0 indicates that there is no similarity and any overlap is explainable as totally random
- 1.0 indicates that the two groups are identical

```
In [18]: from sklearn import metrics
         print("Adjusted rand score: {0:2}".format(metrics.adjusted_rand_score(y_test, y_
```

```
Adjusted rand score: 0.5674467844660916
```

Confusion Matrix

- Each row corresponds to a number (y_test)
- Each column to y_pred (the cluster number)
- Data is the number of times y_test was assigned to the corresponding y_pred
- For example, 0 is fully assigned to cluster 2 (Row 0, Column 2)
- 8 is assigned to cluster 0 21 times (Row 8, Column 0)
- 7, which is cluster 6 is assigned to cluster 6 34 times (Row 7, Column 6)

```
In [19]: print(metrics.confusion_matrix(y_test, y_pred))

[[ 0  0 43  0  0  0  0  0  0  0]
 [20  0  0  7  0  0  0 10  0  0]
 [ 5  0  0 31  0  0  0  1  1  0]
 [ 1  0  0  1  0  1  4  0 39  0]
 [ 1 50  0  0  0  0  1  2  0  1]
 [ 1  0  0  0  1 41  0  0 16  0]
 [ 0  0  1  0 44  0  0  0  0  0]
 [ 0  0  0  0  0  1 34  1  0  5]
 [21  0  0  0  0  3  1  2 11  0]
 [ 0  0  0  0  0  2  3  3 40  0]]
```

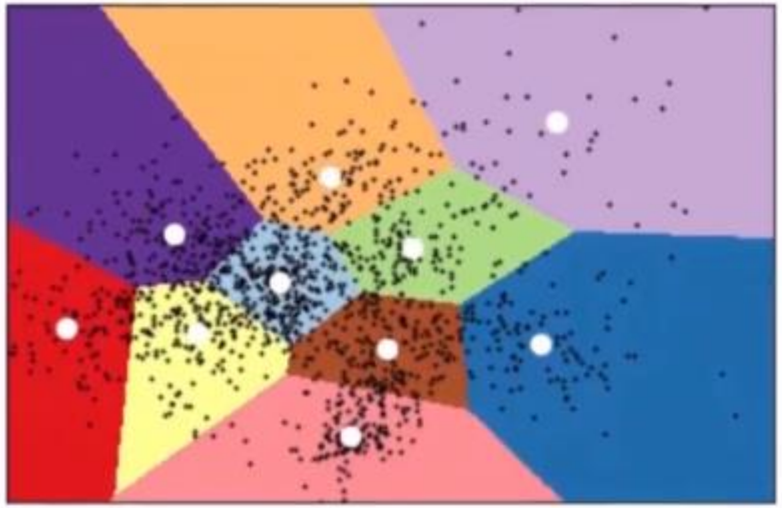
- First reduce the x dimensions to 2 using principle component analysis
- https://en.wikipedia.org/wiki/Principal_component_analysis
- Then figure out the range of values and define the grid
- Run k-means on the reduced (2 component) data set
- Draw a color map and plot the pca points on this map
- Find the cluster centroids and plot them on the color map

Graphical View of Clusters

COLUMBIA | ENGINEERING
EXECUTIVE EDUCATION

In [ ]:
```python
from sklearn import decomposition
pca = decomposition.PCA(n_components=2).fit(X_train)
reduced_X_train = pca.transform(X_train)
# Step size of the mesh.
h = .01
# point in the mesh [x_min, m_max]x[y_min, y_max].
x_min, x_max = reduced_X_train[:, 0].min() + 1, reduced_X_train[:, 0].max() - 1
y_min, y_max = reduced_X_train[:, 1].min() + 1, reduced_X_train[:, 1].max() - 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
    np.arange(y_min, y_max, h))
kmeans = cluster.KMeans(init='k-means++', n_clusters=n_digits,
    n_init=10)
kmeans.fit(reduced_X_train)
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(1)
plt.clf()
plt.imshow(Z, interpolation='nearest', extent=(xx.min(), xx.max(), yy.min(), yy.m
plt.plot(reduced_X_train[:, 0], reduced_X_train[:, 1], 'k.',
    markersize=2)
# Plot the centroids as a white X
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1],marker='.',
    s=169, linewidths=3, color='w', zorder=10)
plt.title('K-means clustering on the digits dataset (PCA reduced data)\nCentroids
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

K-means clustering on the digits dataset (PCA reduced data)
Centroids are marked with white dots



EMERITUS