深 圳 大 学 实 验 报 告

课程名称：机器学习

实验项目名称：Machine Learning Task 4

学院：电子与信息工程学院

专业：电子信息工程

指导教师：欧阳乐

报告人：余韦藩 学号：2020285102

班级：文华班

实验时间：2022.5.30 ——2022.6.6

实验报告提交时间：2022.6.6

教务处制

Aim of Experiment:

(1) Understand and learn how to the Non-Negative Matrix Factorization for and Clustering.
(2) Compare the performance of Non-Negative Matrix Factorization and the other clustering method.
(3) Learn to use the cross table to show the result.

Experiment Content:

(1) Load the seeds datasets.
(2) View the datasets, extract the 'grain_variety' column and characteristic data respectively, and convert the characteristic data into numpy array.
(3) Repetitively perform Non-Negative Matrix Factorization on the seeds datasets.(rank=3, repeat_times=20)
(4) Calculate the silhouette score of each clustering result, draw a line graph, and mark the best score.
(5) Transform the best clustering results into a DataFrame, and make a cross table for varieties and the best clustering results.
(6) (Addition) Compare the performance of Non-Negative Matrix Factorization, kmeans cluster and spectral cluster.

Experiment Process：

A. **Load the seeds datasets.**

We use the pd.read_csv() function to load the seeds datasets and print the first five line for the seeds datasets. The codes are shown in the following figure.

```
# 加载数据集
df_seeds = pd.read_csv('seeds.csv')
```

```
# 查看数据集
df_seeds.head(5)
```

In order to better visual the distribution of the seeds datasets. We plot the parallel coordinates and use the PCA to plot the distribution in the new coordinates.

For plotting parallel coordinates, we use pandas.plotting.parallel_coordinate() function to achieve. The codes are shown in the following figure.

```
# Parallel coordinates plots each feature on a separate column and then draws lines connecting the features for each data sample
plt.figure(figsize=(15,4))
pd.plotting.parallel_coordinates(df_seeds, 'grain_variety')
```

For PCA, we first use decomposition.PCA().fit_transform() function for dimension reduction setting its parameter n_components=2. At the beginning, seeds datasets have 7 attributes. After we perform PCA, we map the value of these 7 attributes into new 2D space and record them into pos DataFrame. Then we plot the distribution of three kinds of grain variety in 2D space to observe their distribution. 'Kama wheat', 'Canadian wheat' and 'Rose color' are presented by blue, green and red color respectively. The codes are shown in the following figure.

```
# Use PCA to observe its distribution
pca = decomposition.PCA(n_components=2)
X = pca.fit_transform(df_seeds.iloc[:,:-1].values)
pos=pd.DataFrame()
pos['X'] =X[:, 0]
pos['Y'] =X[:, 1]
pos['grain_variety'] = df_seeds['grain_variety']
ax = pos[pos['grain_variety']=='Kama wheat'].plot(kind='scatter', x='X', y='Y', color='blue', label='Kama wheat')
pos[pos['grain_variety']=='Canadian wheat'].plot(kind='scatter', x='X', y='Y', color='green', label='Canadian wheat', ax=ax)
pos[pos['grain_variety']=='Rosa wheat'].plot(kind='scatter', x='X', y='Y', color='red', label='Rosa wheat', ax=ax)
```

B. View the datasets, extract the 'grain_variety' column and characteristic data respectively, and convert the characteristic data into numpy array.

In order to extra the characteristic data, we only drop the 'grain variety' column for the original seeds DataFrame. Then we use .value to convert DataFrame into numpy array. In convince, we transpose the matrix and its shape is 7×210.

```
# 分别提出品种列和特征数据，并将特征数据转化为 numpy 数组
seeds = df_seeds.drop(['grain_variety'],axis=1).values
seeds = seeds.T
```

C. Repetitively perform Non-Negative Matrix Factorization on the seeds datasets.(rank=3, repeat_times=20)

We first write a Non-Negative Matrix Factorization function. It decreases the loss between V and WH in every iteration. By default, we set parameter rank=10 and iter=100. The codes are shown in the following figure.

```
def update_H(W, H, V):
    numerator = W.T.dot(V)
    denominator = W.T.dot(W).dot(H) + 1e-10
    H = H*(numerator / denominator)
    return H

def update_W(W, H, V):
    numerator = V.dot(H.T)
    denominator = W.dot(H).dot(H.T) + 1e-10
    W = W*(numerator / denominator)
    return W

def do_nnmf(V, rank=10, iter=100):
    # Initialize
    n, m = V.shape
    W = np.abs(np.random.randn(1, n, rank))[0]
    H = np.abs(np.random.randn(1, rank, m))[0]
    loss = []
    for i in range(iter):
        H = update_H(W, H, V)
        W = update_W(W, H, V)
        loss.append(sum((V - W.dot(H)).flatten()**2))
    return H, W, loss
```

In the experiment, we conduct totally 20 repeat times for Non-Negative Matrix Factorization and create a cluster_id list to record clustering result for each sample in every repeat time. Now, we focus on principle. After performing Non-Negative Matrix Factorization, the shape of H is 3×210 and the shape of W is 7×3. The matrix H indicates the clustering result for each sample. Each column of H represents the score which related to the probability of this sample belongs to corresponding variety. We use numpy.where() function to find the highest score for each sample. Then the number of row for the highest score is the clustering result of each sample. The codes are shown in the following figure.

```
# 对数据多次运行非负矩阵分解(rank=3, repeat_times=20)
repeat_times = 20
cluster_id = []
for repeat_time in range(repeat_times):
    H, W, loss = do_nnmf(seeds, rank=3, iter=500)
    temp = []
    for x in range(H.shape[1]):
        temp.append(np.where(np.max(H[:, x]) == H[:, x])[0][0])
    cluster_id.append(temp)
```

D. Calculate the silhouette score of each clustering result, draw a line graph, and mark the best score.

We first transpose the seed matrix and respectively calculate the silhouette score for each iteration by using silhouette_score() function. We find the best repeat time corresponding to the maximum silhouette score by using numpy.argmax() function. Then we plot the silhouette score for each repeat time and record the maximum silhouette score with red square point. The codes are shown in the

following figure.

```
# 计算每次聚类结果的轮廓系数，并绘制折线图，标注最佳分数
silhouette_scores = [silhouette_score(seeds.T, label) for label in cluster_id]
best_iter = np.argmax(silhouette_scores)
best_score = silhouette_scores[best_iter]

plt.figure(figsize=(8, 3))
plt.plot(range(repeat_times), silhouette_scores, "bo-")
plt.xlabel("repeat time", fontsize=14)
plt.ylabel("Silhouette score", fontsize=14)
plt.plot(best_iter, best_score, "rs")
plt.show()
```

E. Transform the best clustering results into a DataFrame, and make a cross table for varieties and the best clustering results.

We convert the best clustering result into a DataFrame by using panda.DataFrame(). And make a cross table by using pandas.crosstab() function. In the experiment, we make two kinds of cross table, the first one is statistical data and the second one is percentage data. The codes are shown in the following figure.

```
# 将最佳分数对应的聚类结果转化为 Dataframe，做品种和最佳聚类结果的交叉表
df_seeds = pd.read_csv('seeds.csv')
best_cluster = cluster_id[best_iter]
df_best_cluster = pd.DataFrame(data=best_result, columns=['cluster id'])
pd.crosstab(df_seeds['grain_variety'], df_best_cluster['cluster id'], margins = True)
```

```
# 将最佳分数对应的聚类结果转化为 Dataframe，做品种和最佳聚类结果的交叉表（百分比）
df_seeds = pd.read_csv('seeds.csv')
best_cluster = cluster_id[best_iter]
df_best_cluster = pd.DataFrame(data=best_result, columns=['cluster id'])
pd.crosstab(df_seeds['grain_variety'], df_best_cluster['cluster id'], normalize='index')
```

F. (Addition) Compare the performance of Non-Negative Matrix Factorization, k-means cluster and spectral cluster.

Additionally, in order to objectively judge the performance of Non-Negative Matrix Factorization, we also perform k-means cluster and spectral cluster on the seeds datasets.

For k-means cluster, we use cluster.KMeans() function setting its parameter n_cluster=3, max_iter=50 and random state=1. Then we also make two cross table for varieties and the clustering results. The codes are shown in the following figure.

```
# 对数据进行kmeans聚类，并输出聚类结果，并做品种和最佳聚类结果的交叉表
seeds = df_seeds.drop(['grain_variety'],axis=1)
k_means = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
k_means.fit(seeds)
labels = k_means.labels_
df_kmeans_cluster = pd.DataFrame(labels, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_kmeans_cluster['cluster id'], margins = True)
```

```
# 对数据进行kmeans聚类，并输出聚类结果，并做品种和最佳聚类结果的交叉表（百分比）
seeds = df_seeds.drop(['grain_variety'],axis=1)
k_means = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
k_means.fit(seeds)
labels = k_means.labels_
df_kmeans_cluster = pd.DataFrame(labels, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_kmeans_cluster['cluster id'], normalize='index')
```

For spectral cluster, we use cluster.SpectralClustering() function setting its parameter n_cluster=3, random state=1, affinity=rbf and gamma=1. In fact, when we use the Gaussian kernel, the discriminative ability becomes poor as the increase of gamma. Then we also make two cross table for varieties and the clustering results. The codes are shown in the following figure.

```
# 对数据进行spectral聚类，并输出聚类结果，并做品种和最佳聚类结果的交叉表
df_seeds = pd.read_csv('seeds.csv')
seeds = df_seeds.drop(['grain_variety'],axis=1)
spectral = cluster.SpectralClustering(n_clusters=3, random_state=1, affinity='rbf', gamma=1)
spectral.fit(seeds)
df_spectral_cluster = pd.DataFrame(spectral.labels_, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_spectral_cluster['cluster id'], margins = True)
```

```
# 对数据进行spectral聚类，并输出聚类结果，并做品种和最佳聚类结果的交叉表（百分比）
df_seeds = pd.read_csv('seeds.csv')
seeds = df_seeds.drop(['grain_variety'],axis=1)
spectral = cluster.SpectralClustering(n_clusters=3, random_state=1, affinity='rbf', gamma=1)
spectral.fit(seeds)
df_spectral_cluster = pd.DataFrame(spectral.labels_, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_spectral_cluster['cluster id'], normalize='index')
```

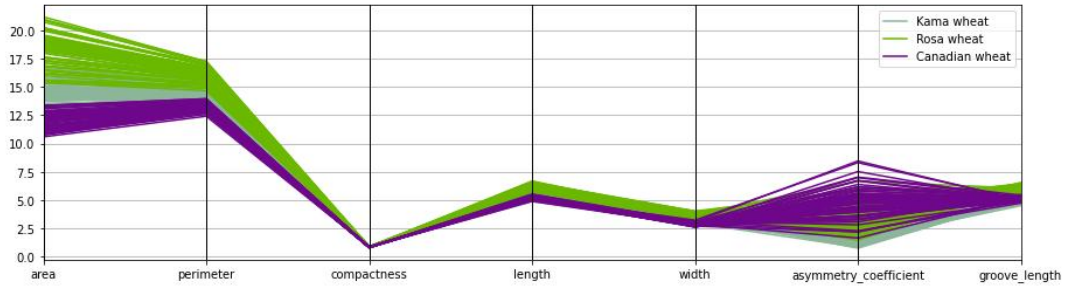
Experimental Results and Analysis:

A. Load the seeds datasets.

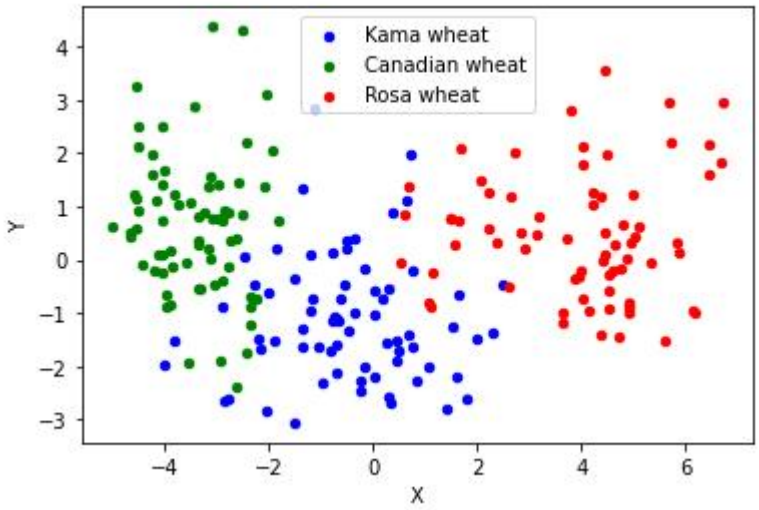
The following figure shows the first five lines for the seeds datasets. We observe that the seeds datasets has 7 attribute and 1 class label. And it has three kinds of class labels in total.

	area	perimeter	compactness	length	width	asymmetry_coefficient	groove_length	grain_variety
0	15.26	14.84	0.8710	5.763	3.312	2.221	5.220	Kama wheat
1	14.88	14.57	0.8811	5.554	3.333	1.018	4.956	Kama wheat
2	14.29	14.09	0.9050	5.291	3.337	2.699	4.825	Kama wheat
3	13.84	13.94	0.8955	5.324	3.379	2.259	4.805	Kama wheat
4	16.14	14.99	0.9034	5.658	3.562	1.355	5.175	Kama wheat

The following figure shows the parallel coordinates for the seeds datasets. The parallel coordinates plots each feature on a separate column and then draws lines connecting the features for each data sample. From the figure, we observe that for some kinds of feature such ‘area’, ‘perimeter’ and ‘length’, different grain variety distributes on specific range which indicates that we could utilize this 7 attribute to predict the grain variety. So in the rest of the experiment, we mainly use Non-Negative Matrix Factorization, k-means cluster and spectral cluster to predict the grain variety for each sample.

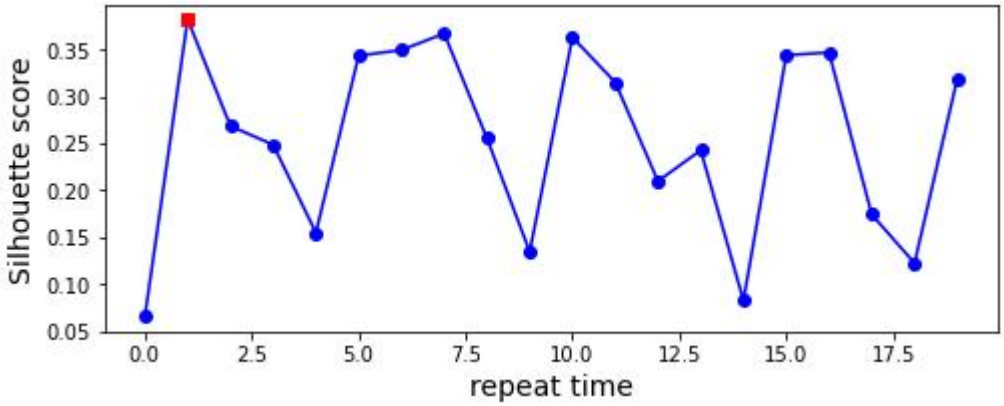


The following figure shows the seeds datasets distribution after performing PCA. From the figure, we observe that three kinds of grain variety distributes in different area which indicates that the value of 7 attributes data decides the variety of grain. Due to the special space distribution of each kind of grain variety, we can correctly distinguish or predict its variety by using clustering method. It is worth to say that some kinds of sample from Kama wheat and some kinds of sample from Canadian wheat is overlap and mixed together. That may cause the cluster method to fault to distinguish the Kama wheat and Canadian wheat.



B. Calculate the silhouette score of each clustering result, draw a line graph, and mark the best score.

Silhouette score is a kind of evaluation for the clustering method with the range of [-1,1]. The higher silhouette score prove that the better performance of the cluster. The following figure shows the silhouette score of each clustering result. We observe that the repeat time corresponding to the best silhouette score is 2 repeat time and its silhouette score reaches to 0.37.



C. Transform the best clustering results into a DataFrame, and make a cross table for varieties and the best clustering results.

From the cross table, we observe that the correct cluster rate is 78.57% for Canadian wheat and 71.43% for Kama wheat and 97.14% for Rosa wheat. Rosa wheat achieves the best clustering performance. The result proves that the possibility we had said before: 'That may cause the cluster method to fault to distinguish the Kama wheat and Canadian wheat'. Although there exists around 25% error rate for the Canadian wheat and Kama wheat, its successful clustering rate is acceptable.

cluster id	0	1	2	All
grain_variety				
Canadian wheat	15	0	55	70
Kama wheat	50	13	7	70
Rosa wheat	2	68	0	70
All	67	81	62	210

cluster id	0	1	2
grain_variety			
Canadian wheat	0.214286	0.000000	0.785714
Kama wheat	0.714286	0.185714	0.100000
Rosa wheat	0.028571	0.971429	0.000000

D. (Addition) Compare the performance of Non-Negative Matrix Factorization, kmeans cluster and spectral cluster.

The following figure shows the cross table for the k-means clustering. From the cross table, we observe that the correct cluster rate is 97.14% for Canadian wheat and 85.71% for Kama wheat and 85.71% for Rosa wheat. Canadian wheat achieves the best clustering performance. Although there exists around 15% error rate for the Canadian wheat and Kama wheat, its successful clustering rate is acceptable.

cluster id	0	1	2	All
grain_variety				
Canadian wheat	0	68	2	70
Kama wheat	1	9	60	70
Rosa wheat	60	0	10	70
All	61	77	72	210

cluster id	0	1	2
grain_variety			
Canadian wheat	0.000000	0.971429	0.028571
Kama wheat	0.014286	0.128571	0.857143
Rosa wheat	0.857143	0.000000	0.142857

The following figure shows the cross table for the spectral clustering. From the cross table, we observe that the correct cluster rate is 97.14% for Canadian wheat and 87.14% for Kama wheat and 78.57% for Rosa wheat. Canadian wheat achieves the best clustering performance. Although there exists around 20% error rate for the Canadian wheat and Kama wheat, its successful clustering rate is acceptable.

cluster id	0	1	2	All
grain_variety				
Canadian wheat	0	68	2	70
Kama wheat	0	9	61	70
Rosa wheat	55	0	15	70
All	55	77	78	210

cluster id	0	1	2
grain_variety			
Canadian wheat	0.000000	0.971429	0.028571
Kama wheat	0.000000	0.128571	0.871429
Rosa wheat	0.785714	0.000000	0.214286

Summary: Compared the three clustering method, we draw a conclusion that the k-means clustering achieves the best performance on seeds datasets and spectral clustering achieves the second best performance on seeds datasets and Non-Negative Matrix Factorization achieves the poor performance on seeds datasets especially for Kama wheat.

指导教师批阅意见： 成绩评定： 指导教师签字： 年　　月　　日
备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。

experiment_four

June 5, 2022

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
from sklearn import decomposition
from sklearn import cluster
```

[2]:
```
def update_H(W, H, V):
    numerator = W.T.dot(V)
    denominator = W.T.dot(W).dot(H) + 1e-10
    H = H*(numerator / denominator)
    return H

def update_W(W, H, V):
    numerator = V.dot(H.T)
    denominator = W.dot(H).dot(H.T) + 1e-10
    W = W*(numerator / denominator)
    return W

def do_nnmf(V, rank=10, iter=100):
    # Initialize
    n, m = V.shape
    W = np.abs(np.random.randn(1, n, rank))[0]
    H = np.abs(np.random.randn(1, rank, m))[0]
    loss = []
    for i in range(iter):
        H = update_H(W, H, V)
        W = update_W(W, H, V)
        loss.append(sum((V - W.dot(H)).flatten()**2))
    return H, W, loss
```

[3]:
```
#
df_seeds = pd.read_csv('seeds.csv')
```

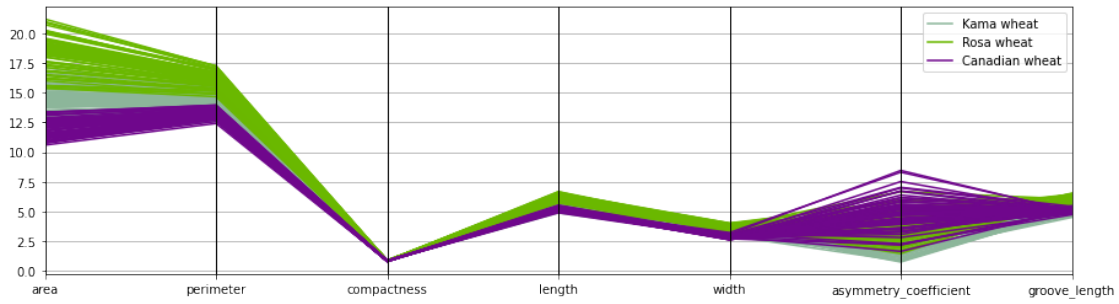
[55]:
```
#
df_seeds.head(5)
```

[55]:
```
     area  perimeter  compactness  length  width  asymmetry_coefficient  \
0   15.26      14.84       0.8710   5.763  3.312                  2.221
1   14.88      14.57       0.8811   5.554  3.333                  1.018
2   14.29      14.09       0.9050   5.291  3.337                  2.699
3   13.84      13.94       0.8955   5.324  3.379                  2.259
4   16.14      14.99       0.9034   5.658  3.562                  1.355

   groove_length grain_variety
0          5.220    Kama wheat
1          4.956    Kama wheat
2          4.825    Kama wheat
3          4.805    Kama wheat
4          5.175    Kama wheat
```

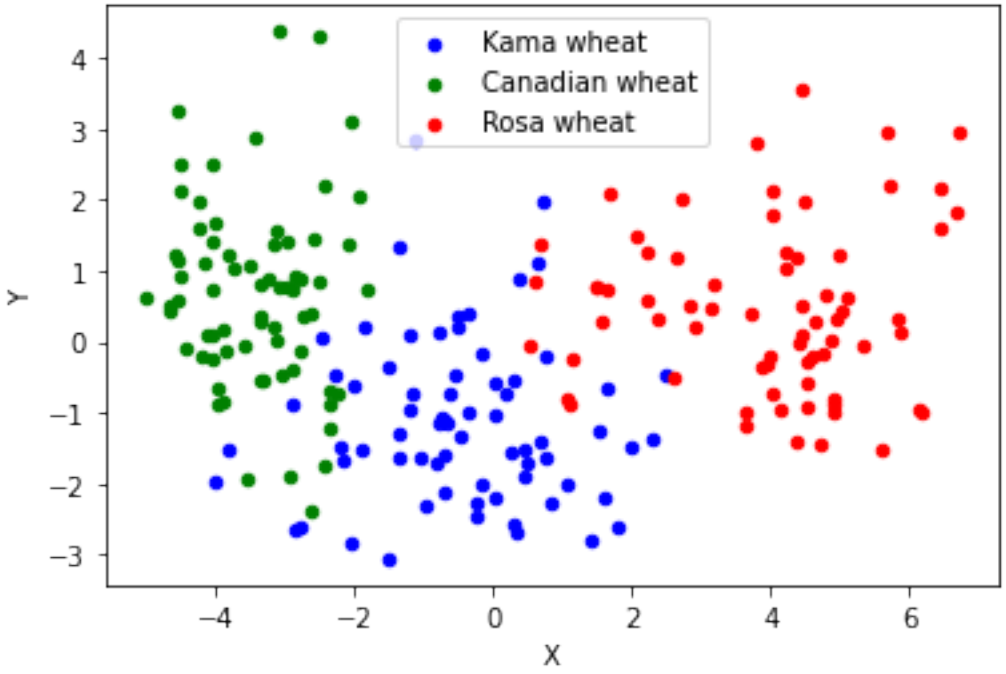
[65]:
```
# Parallel coordinates plots each feature on a separate column and then draws␣
 ↪lines connecting the features for each data sample
plt.figure(figsize=(15,4))
pd.plotting.parallel_coordinates(df_seeds, 'grain_variety')
```

[65]: <AxesSubplot:>



[71]:
```
# Use PCA to observe its distribution
pca = decomposition.PCA(n_components=2)
X = pca.fit_transform(df_seeds.iloc[:,:-1].values)
pos=pd.DataFrame()
pos['X'] =X[:, 0]
pos['Y'] =X[:, 1]
pos['grain_variety'] = df_seeds['grain_variety']
ax = pos[pos['grain_variety']=='Kama wheat'].plot(kind='scatter', x='X', y='Y',␣
 ↪color='blue', label='Kama wheat')
pos[pos['grain_variety']=='Canadian wheat'].plot(kind='scatter', x='X', y='Y',␣
 ↪color='green', label='Canadian wheat', ax=ax)
pos[pos['grain_variety']=='Rosa wheat'].plot(kind='scatter', x='X', y='Y',␣
 ↪color='red', label='Rosa wheat', ax=ax)
```

[71]: <AxesSubplot:xlabel='X', ylabel='Y'>



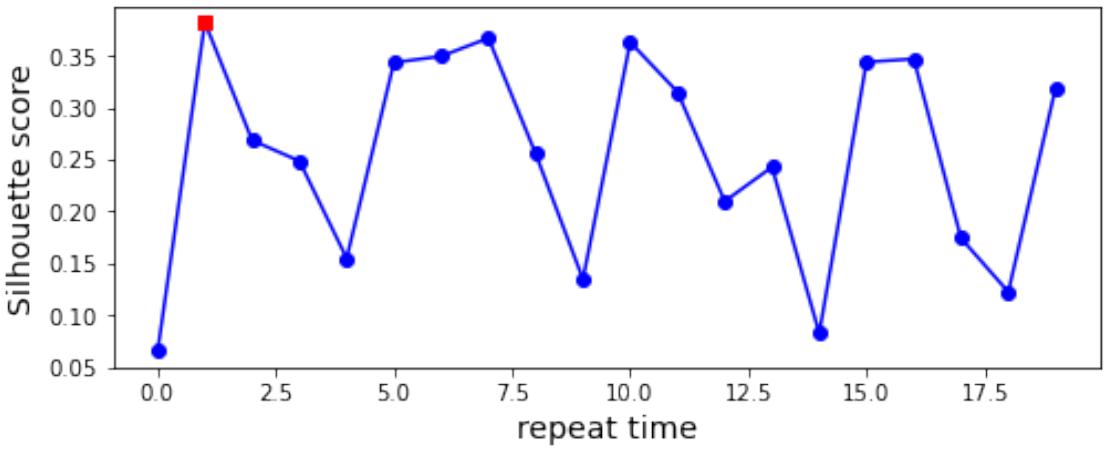
[10]:
```
#              numpy
seeds = df_seeds.drop(['grain_variety'],axis=1).values
seeds = seeds.T
```

[11]:
```
#        (rank=3, repeat_times=20)
repeat_times = 20
cluster_id = []
for repeat_time in range(repeat_times):
    H, W, loss = do_nnmf(seeds, rank=3, iter=500)
    temp = []
    for x in range(H.shape[1]):
        temp.append(np.where(np.max(H[:, x]) == H[:, x])[0][0])
    cluster_id.append(temp)
```

[58]:
```
#
silhouette_scores = [silhouette_score(seeds.T, label) for label in cluster_id]
best_iter = np.argmax(silhouette_scores)
best_score = silhouette_scores[best_iter]

plt.figure(figsize=(8, 3))
plt.plot(range(repeat_times), silhouette_scores, "bo-")
plt.xlabel("repeat time", fontsize=14)
```

```
plt.ylabel("Silhouette score", fontsize=14)
plt.plot(best_iter, best_score, "rs")
plt.show()
```



[74]:
```
#          Dataframe
df_seeds = pd.read_csv('seeds.csv')
best_cluster = cluster_id[best_iter]
df_best_cluster = pd.DataFrame(data=best_result, columns=['cluster id'])
pd.crosstab(df_seeds['grain_variety'], df_best_cluster['cluster id'], margins =␣
 ↪True)
```

[74]:

cluster id	0	1	2	All
grain_variety				
Canadian wheat	15	0	55	70
Kama wheat	50	13	7	70
Rosa wheat	2	68	0	70
All	67	81	62	210

[73]:
```
#          Dataframe
df_seeds = pd.read_csv('seeds.csv')
best_cluster = cluster_id[best_iter]
df_best_cluster = pd.DataFrame(data=best_result, columns=['cluster id'])
pd.crosstab(df_seeds['grain_variety'], df_best_cluster['cluster id'],␣
 ↪normalize='index')
```

[73]:

cluster id	0	1	2
grain_variety			
Canadian wheat	0.214286	0.000000	0.785714
Kama wheat	0.714286	0.185714	0.100000
Rosa wheat	0.028571	0.971429	0.000000

[78]:
```
#    kmeans
seeds = df_seeds.drop(['grain_variety'],axis=1)
k_means = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
k_means.fit(seeds)
labels = k_means.labels_
df_kmeans_cluster = pd.DataFrame(labels, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_kmeans_cluster['cluster id'], margins␣
 ↪= True)
```

[78]:
```
cluster id       0   1   2  All
grain_variety
Canadian wheat   0  68   2   70
Kama wheat       1   9  60   70
Rosa wheat      60   0  10   70
All             61  77  72  210
```

[79]:
```
#    kmeans
seeds = df_seeds.drop(['grain_variety'],axis=1)
k_means = cluster.KMeans(n_clusters=3, max_iter=50, random_state=1)
k_means.fit(seeds)
labels = k_means.labels_
df_kmeans_cluster = pd.DataFrame(labels, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_kmeans_cluster['cluster id'],␣
 ↪normalize='index')
```

[79]:
```
cluster id             0         1         2
grain_variety
Canadian wheat  0.000000  0.971429  0.028571
Kama wheat      0.014286  0.128571  0.857143
Rosa wheat      0.857143  0.000000  0.142857
```

[109]:
```
#    spectral
df_seeds = pd.read_csv('seeds.csv')
seeds = df_seeds.drop(['grain_variety'],axis=1)
spectral = cluster.SpectralClustering(n_clusters=3, random_state=1,␣
 ↪affinity='rbf', gamma=1)
spectral.fit(seeds)
df_spectral_cluster = pd.DataFrame(spectral.labels_, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_spectral_cluster['cluster id'],␣
 ↪margins = True)
```

[109]:
```
cluster id       0   1   2  All
grain_variety
Canadian wheat   0  68   2   70
```

```
Kama wheat       0   9  61   70
Rosa wheat      55   0  15   70
All             55  77  78  210
```

[110]:
```
#    spectral
df_seeds = pd.read_csv('seeds.csv')
seeds = df_seeds.drop(['grain_variety'],axis=1)
spectral = cluster.SpectralClustering(n_clusters=3, random_state=1,␣
 ↪affinity='rbf', gamma=1)
spectral.fit(seeds)
df_spectral_cluster = pd.DataFrame(spectral.labels_, columns=['cluster id'])
df_seeds = pd.read_csv('seeds.csv')
pd.crosstab(df_seeds['grain_variety'], df_spectral_cluster['cluster id'],␣
 ↪normalize='index')
```

[110]:
```
cluster id             0         1         2
grain_variety
Canadian wheat  0.000000  0.971429  0.028571
Kama wheat      0.000000  0.128571  0.871429
Rosa wheat      0.785714  0.000000  0.214286
```