第四次上机作业

注意事项

1. 上机作业所需上传的文件，应打包并命名为"学号-姓名-上机#序号"，如"18000xxxxx-张三-上机4"，并在截止时间前上传到教学网。截止时间后仍开放提交，但会酌情扣分。
2. 上机作业为半自动评分。上传的压缩包，会根据压缩包名称自动分类，并提取压缩包名称中的有效信息。请尽可能保持压缩包命名符合规范。压缩包名中的短横线'-'可替换为下划线'_'、加号'+'，但不能是空格' '、长横线'—'。
3. 上机作业中的代码会自动测试。要求代码中读入的外部数据文件存放在代码文件上一级目录下的data文件夹下，生成的图表应嵌入报告pdf文件中，生成的数据文件等应在代码文件所在目录下。

```
root
  ├─workdir (Compress this directory and upload it)
  |     ├─cluster_digits.py (Required)
  |     ├─cluster_zinc.py (Required)
  |     ├─data.csv (Optional)
  |     └─report.pdf (Required)
  └─data (DO NOT upload this directory)
        ├─zinc_fp.csv
        └─zinc_SMILES.csv
```

4. 题目中要求生成的图表和计算的数据均列在报告中。
5. 题目中要求计算的数据，代码运算得出结果后，应输出到控制台。

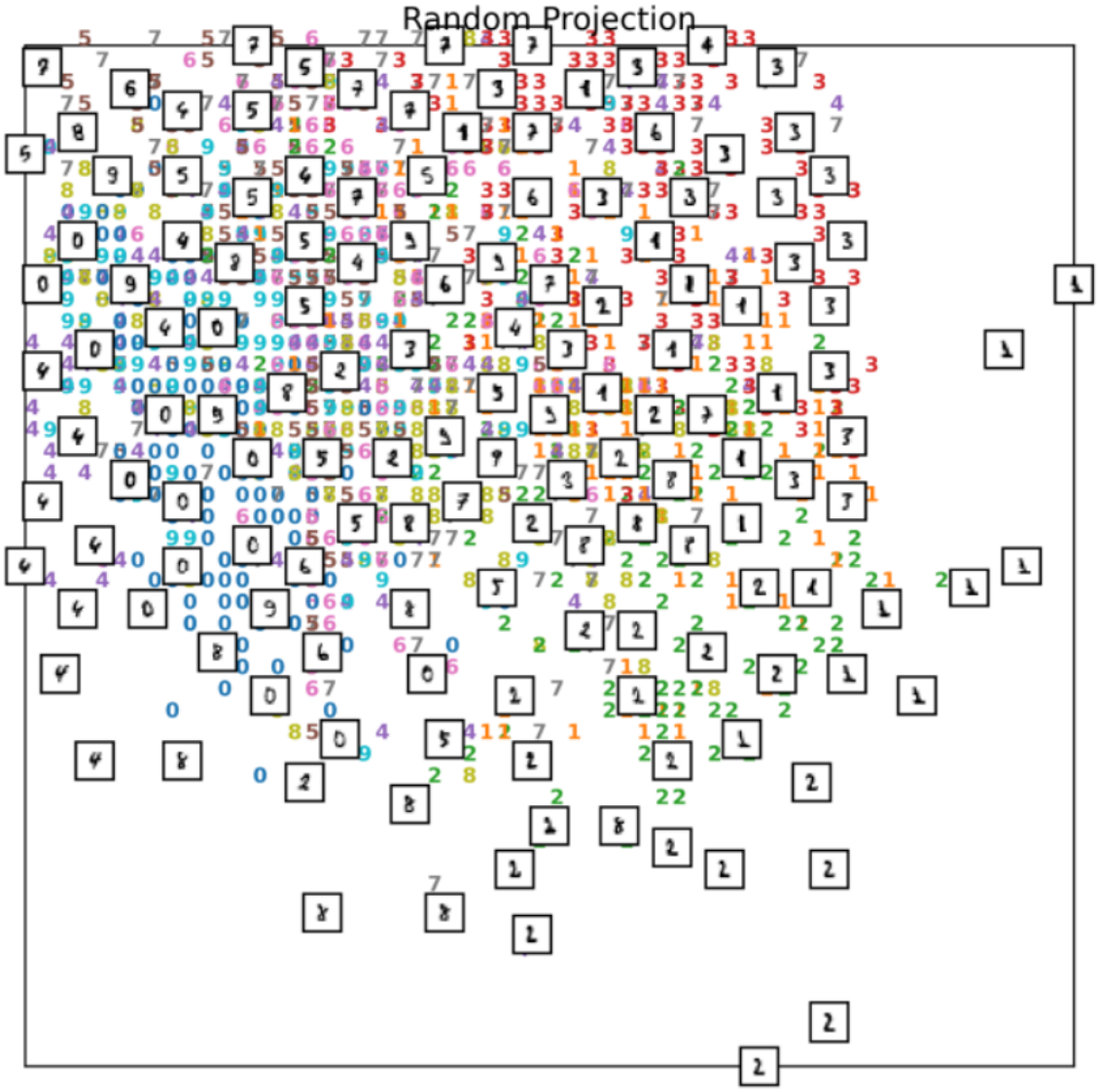
1. 调用scikit-learn提供的数字图像数据集，用无监督学习的方法将数字图像聚类，并评价聚类效果。（8分）

（1）数字图像数据集的数据特征及可视化（此部分为示例，**在代码中不做要求**）

将每个图像样本的64个像素值随机投影到正交的两个维度上，作图显示

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 12/5/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import matplotlib.pyplot as plt
import numpy as np
from matplotlib import offsetbox
from sklearn import cluster, datasets, manifold, metrics, random_projection
from sklearn.preprocessing import MinMaxScaler

NCLASS = 10
# Color for each category
category_colors = plt.get_cmap('tab10')(np.linspace(0., 1., NCLASS))
digit_styles = {'weight': 'bold', 'size': 8}


def plot2D(X, labels, images, title="", save="./2D-plot.png"):
```

```
    fig = plt.figure(figsize=(6, 6), dpi=320)
    ax = fig.add_subplot(1, 1, 1)
    X_std = MinMaxScaler().fit_transform(X)

    for xy, l in zip(X_std, labels):
        ax.text(*xy, str(l), color=category_colors[l], **digit_styles)

    image_locs = np.ones((1, 2), dtype=float)
    for xy, img in zip(X_std, images):
        dist = np.sqrt(np.sum(np.power(image_locs - xy, 2), axis=1))
        if np.min(dist) < .05:
            continue
        thumbnail = offsetbox.OffsetImage(img, zoom=.8, cmap=plt.cm.gray_r)
        imagebox = offsetbox.AnnotationBbox(thumbnail, xy)
        ax.add_artist(imagebox)
        image_locs = np.vstack([image_locs, xy])

    ax.set_xticks([])
    ax.set_yticks([])
    plt.title(title)
    plt.tight_layout()
    plt.savefig(save)


def plot3D(X, labels, title="", save="./3D-plot.png"):
    fig = plt.figure(figsize=(6, 6), dpi=320)
    ax = fig.add_subplot(1, 1, 1, projection='3d')
    X_std = MinMaxScaler().fit_transform(X)

    for xy, l in zip(X_std, labels):
        ax.text(*xy, str(l), color=category_colors[l], **digit_styles)

    plt.title(title)
    plt.tight_layout()
    plt.savefig(save)


# Load digits dataset from scikit-learn
digits = datasets.load_digits(n_class=NCLASS)
# Pixel data from dataset
X = digits.data
# Project X to 2 random components
RP = random_projection.SparseRandomProjection(
    n_components=2, random_state=20201205)
X_projected = RP.fit_transform(X)
# Labels: digits.target
# Images: digits.images
plot2D(X_projected, digits.target, digits.images, title="Random Projection")
```



（2）分别用**PCA**和**t-SNE**的方法将每个图像样本的像素数据**降维**至**2维**，并用与（1）中相似的方法**可视化。**

（3）用**kMeans**方法分别对**未降维**的数据和PCA降**至2维**后的数据聚类，聚为10类，用**Silhouette Coefficient**评估聚类效果。

$$\mathrm{SC} = \frac{1}{N}\sum_{i=1}^{N}\frac{b_i - a_i}{\max\left(a_i, b_i\right)}$$

其中a_i是样本i和所在类其他样本间的平均距离，b_i是样本i和其他类的样本间的平均距离。

（4）由于每个数字图像都有标注类别，可以用**Homogeneity**，**Completeness**和**V-measure**作为评价指标，比较聚类结果与标注的类别是否一致，以维数为自变量，绘制PCA维度与这些指标的变化曲线（例如1至64维全部计算）。

Homogeneity：衡量聚类得到的每个簇包含归属于同一类的样本的程度

Completeness：衡量归属于同一类的样本被聚类到同一簇的程度

$$\text{V-measure} = \frac{(1+\beta)\times\text{homogeneity}\times\text{completeness}}{\beta\times\text{homogeneity}+\text{completeness}}$$

其中默认$\beta = 1$。

计算出你的类别与标注类别之间的**混淆矩阵**，哪些类别之间比较相似？

更详细的介绍及scikit-learn中的相关函数见https://scikit-learn.org/stable/modules/clustering.html#homogeneity-completeness-and-v-measure

用**Homogeneity**，**Completeness**和**V-measure**评价kMeans方法对未降维的数据和降维后的数据聚类的效果。

（5）结合上述问题中的结果，**分析数据降维方法对聚类效果的影响。**

2. 现给出ZINC数据集中随机取出的10000个分子，在`zinc_SMILES.csv`中有记录每个分子的SMILES表达式，在`zinc_fp.csv`中有记录每个分子的ECFP4圆形指纹（长度为2048的bit串，每一位上或为0或为1）。用合适的方法对数据进行降维，并用kMeans方法聚类，并用Silhouette Coefficient评估聚类效果，每一类给出一个代表性分子的SMILES字符串。（2分）

对于两个分子，可以根据分子指纹的相似度定义其之间的"距离"，常用的相似度定义有

- Tanimoto Similarity

$$\text{Tanimoto Similarity} = \frac{X \cdot Y}{||X||_2^2 + ||Y||_2^2 - X \cdot Y}$$

- Cosine Similarity

$$\text{Cosine Similarity} = \frac{X \cdot Y}{||X||_2 \times ||Y||_2}$$

- Dice Similarity

$$\text{Dice Similarity} = \frac{2X \cdot Y}{||X||_1 + ||Y||_1}$$

其中

$$||X||_1 = \sum_{i=1}^{N} |X_i|, \quad ||X||_2 = \sqrt{\sum_{i=1}^{N} X_i^2}$$

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

'''
    Author: Pengbo Song
    Date created: 12/7/2020
    Python Version: Anaconda3 (Python 3.8.3)
'''

import numpy as np


def tanimoto_similarity(x, y):
    return np.dot(x, y) / (np.linalg.norm(x, ord=2) ** 2 + np.linalg.norm(y,
ord=2) ** 2 - np.dot(x, y))


def cosine_similarity(x, y):
    return np.dot(x, y) / (np.linalg.norm(x, ord=2) * np.linalg.norm(y, ord=2))


def dice_similarity(x, y):
```

```
    return 2 * np.dot(x, y) / (np.linalg.norm(x, ord=1) + np.linalg.norm(y,
ord=1))


x = np.array([1, 0, 0, 1, 0, 1, 1, 0])
y = np.array([1, 1, 0, 0, 1, 0, 1, 1])
print(f"Tanimoto Similarity = {tanimoto_similarity(x, y):.3f}")
print(f"Cosine Similarity = {cosine_similarity(x, y):.3f}")
print(f"Dice Similarity = {dice_similarity(x, y):.3f}")
"""
Tanimoto Similarity = 0.286
Cosine Similarity = 0.447
Dice Similarity = 0.444
"""
```

将文件打包上传，压缩包中应有文件

- Python源代码`cluster_digits.py`（1.）
- Python源代码`cluster_zinc.py`（2.）
- 分析报告`report.pdf`