HW1

id:110062209
name:簡晟棋

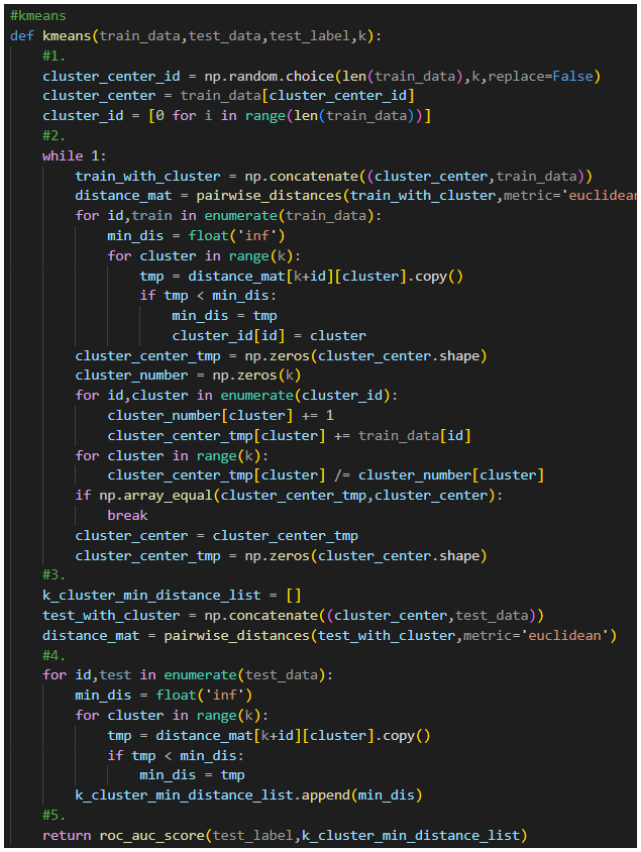
Function Implement

problem 1:

```python
#knn
def knn(train_data,test_data,test_label,k):
    #1.
    k_near_distance_list = []
    all_data = np.concatenate((train_data,test_data))
    distance_mat = pairwise_distances(all_data,metric='euclidean')
    #2.
    for id,test in enumerate(test_data):
        distance_list = distance_mat[len(train_data)+id,:len(train_data)].copy()
        distance_list.sort()
        k_near_distance_list.append(np.mean(distance_list[:k]))
    #3.
    return roc_auc_score(test_label,k_near_distance_list)
```

1.用 np.concatenate 把 train_data 跟 test_data 串在一起變成 all_data
用 pairwise_distances 算出 all_data 中各兩點間的距離矩陣 distance_mat
2.對每個 test_data 中的點取出
distance_list = distance_mat[len(train_data)+id, :len(train_data)]，即為該 test data point 與所有 train data point 的距離
將 distance_list 從小到大 sort 後
把前 k 個最近的 train_data point 的距離的平均放入 k_near_distance_list
3.用 test_label(已轉化成 0,1 的形式)與 k_near_distance_list 計算 roc_auc_score

problem 2:



```python
#kmeans
def kmeans(train_data,test_data,test_label,k):
    #1.
    cluster_center_id = np.random.choice(len(train_data),k,replace=False)
    cluster_center = train_data[cluster_center_id]
    cluster_id = [0 for i in range(len(train_data))]
    #2.
    while 1:
        train_with_cluster = np.concatenate((cluster_center,train_data))
        distance_mat = pairwise_distances(train_with_cluster,metric='euclidean
        for id,train in enumerate(train_data):
            min_dis = float('inf')
            for cluster in range(k):
                tmp = distance_mat[k+id][cluster].copy()
                if tmp < min_dis:
                    min_dis = tmp
                    cluster_id[id] = cluster
        cluster_center_tmp = np.zeros(cluster_center.shape)
        cluster_number = np.zeros(k)
        for id,cluster in enumerate(cluster_id):
            cluster_number[cluster] += 1
            cluster_center_tmp[cluster] += train_data[id]
        for cluster in range(k):
            cluster_center_tmp[cluster] /= cluster_number[cluster]
        if np.array_equal(cluster_center_tmp,cluster_center):
            break
        cluster_center = cluster_center_tmp
        cluster_center_tmp = np.zeros(cluster_center.shape)
    #3.
    k_cluster_min_distance_list = []
    test_with_cluster = np.concatenate((cluster_center,test_data))
    distance_mat = pairwise_distances(test_with_cluster,metric='euclidean')
    #4.
    for id,test in enumerate(test_data):
        min_dis = float('inf')
        for cluster in range(k):
            tmp = distance_mat[k+id][cluster].copy()
            if tmp < min_dis:
                min_dis = tmp
        k_cluster_min_distance_list.append(min_dis)
    #5.
    return roc_auc_score(test_label,k_cluster_min_distance_list)
```

1.隨機從 train_data 取 k 個點當 cluster center

2.用 np.concatenate 把 cluster_center 跟 train_data 串在一起變成 train_with_cluster

用 pairwise_distances 算出 train_with_cluster 中各兩點間的距離矩陣 distance_mat

對每個 train data point 找到離該 point 最近的 cluster center

用 cluster_id 紀錄每個 point 被分到哪個 cluster

用 cluster_number 紀錄每個 cluster 有幾個 point

用 cluster_center_tmp 紀錄目前分類下的的 cluster center(把該 cluster 每個 point 的值加起來取平均)

直到 cluster center converge 為止前持續進行 2.

3. 用 np.concatenate 把 cluster_center 跟 test_data 串在一起變成 test_with_cluster

用 pairwise_distances 算出 test_with_cluster 中各兩點間的距離矩陣 distance_mat

4.把每個 test point 與離它最近的 cluster 之間的距離放入 k_cluster_min_distance_list

5.用 test_label(已轉化成 0,1 的形式)與 k_cluster_min_distance_list 計算 roc_auc_score

problem 3:

```python
#Distance-based
def cosine(A,B):
    cos = np.sum(A*B)/((np.sum(A**2)*np.sum(B**2))**0.5)
    return 1-cos

def minkowski(A,B,p):
    if p == np.inf:
        return np.max(abs(A-B))
    return np.sum(abs(A-B)**p)**(1/p)

def mahalanobis(A,B,S_inv):
    ans = (A-B).transpose()
    ans = ans @ S_inv
    ans = ans @ (A-B)
    return ans**0.5
```

先把 cosine, minkowski 和 mahalanobis distance 的 function 準備好

```python
def k_dis(train_data,test_data,test_label,k,distance_type=None,r=None):
    k_near_distance_list = []
    distance_mat = None
    #1.
    if distance_type == 'cosine':
        distance_mat = pairwise_distances(test_data,metric=cosine)
    elif distance_type == 'minkowski':
        distance_mat = pairwise_distances(test_data,metric=minkowski,p=r)
    elif distance_type == 'mahalanobis':
        mean= np.mean(train_data)
        S = np.zeros((train_data.shape[1],train_data.shape[1]))
        for train in train_data:
            delta = train - mean
            delta = delta.reshape((train_data.shape[1],1))
            S += delta @ delta.transpose()
        S /= train_data.shape[0]
        S_inv = np.linalg.inv(S)
        distance_mat = pairwise_distances(test_data,metric=mahalanobis,S_inv=S_inv)
    for id,test in enumerate(test_data):
        distance_list = distance_mat[id,:].copy()
        distance_list.sort()
        k_near_distance_list.append(distance_list[k])
    #2.
    return roc_auc_score(test_label,k_near_distance_list)
```

1.

如果 distance_type 為’cosine’，用 pairwise_distances(test_data,metric=cosine)算出每個 test_data 中各兩點的 cosine distance

如果 distance_type 為'minkowski'，用 pairwise_distances(test_data,metric=minkowski)算出每個 test_data 中各兩點的 minkowski distance，記得使 p=r

如果 distance_type 為’mahalanobis'，用 pairwise_distances(test_data,metric=mahalanobis)加上 train_data 的 covariance matrix S 的 inverse 把 test_data 兩兩間的 mahalanobis distance 算出來

用 distance_list 取出每個 test point 與其他 test data 中的 point 的距離

將 distance_list 從小到大 sort 後取出第 k 項，也就是離自己第 k 近的距離(第 0 項是自己)

把離自己第 k 近的距離放入 k_near_distance_list

2.用 test_label(已轉化成 0,1 的形式)與 k_near_distance_list 計算 roc_auc_score

problem 4:

```python
def k_distance(distance_mat,point_id,k):
    distance_list = distance_mat[point_id,:].copy()
    distance_list.sort()
    return distance_list[k]

def reachable_distance(distance_mat,point_p_id,point_o_id,k):
    return max([k_distance(distance_mat,point_o_id,k),distance_mat[point_p_id][point_o_id]])

def lrd(distance_mat,point_id,k):
    distance_list = []
    for id in range(distance_mat.shape[0]):
        distance_list.append((distance_mat[point_id][id],id))
    distance_list = sorted(distance_list, key=lambda i: i[0])
    ans = 0
    for i in range(1,k+1):
        ans += reachable_distance(distance_mat,point_id,distance_list[i][1],k)
    return 1/(ans/k)

def lof(distance_mat,point_id,k):
    distance_list = []
    for id in range(distance_mat.shape[0]):
        distance_list.append((distance_mat[point_id][id],id))
    distance_list = sorted(distance_list, key=lambda i: i[0])
    ans = 0
    for i in range(1,k+1):
        ans += lrd(distance_mat,distance_list[i][1],k)
    ans /= lrd(distance_mat,point_id,k)
    return ans/k
```

k_distance:用 distance_mat 找出該 point_id 與其他點的距離，由小到大 sort 後取出第 k 項為離自己第 k 近的距離(第 0 項是自己)

reachable_distance:求出 max(point_o 的 k_distance , point_o 和 point_p 的距離)

lrd:把 tuple(point_p 與該 point id 的距離,point id)放入 distance_list，從小到大 sort 後第 1~k 項的 id 就是離 point_p 最近的 k 個點 (第 0 項是自己)

接著求出 point_p 的 local reachability distance(1/(離 point_p 最近的 k 個點的 reachable_distance 的平均))

lof: 把 tuple(point_p 與該 point id 的距離,point id)放入 distance_list，從小到大 sort 後第 1~k 項的 id 就是離 point_p 最近的 k 個點 (第 0 項是自己)

接著求出 point_p 的 local outlier factor(離 point_p 最近的 k 個點的 local reachability distance 的平均/point_p 的 local reachability distance)

```python
def LOF(test_data,test_label,k,drawing=False):
    #1.
    distance_list = []
    score_for_color_tmp = []
    distance_mat = pairwise_distances(test_data,metric='euclidean')
    for id,test in enumerate(test_data):
        score = lof(distance_mat,id,k)
        distance_list.append(score)
        score_for_color_tmp.append(score)
    #2.
    if drawing:
        global test_data_draw
        global test_label_draw
        global score_for_color
        test_data_draw = TSNE(n_components=2).fit_transform(test_data)
        test_label_draw = test_label.copy()
        score_for_color = score_for_color_tmp.copy()
    #3.
    return roc_auc_score(test_label,distance_list)
```

1 用 test_data 各兩點間距離矩陣 distance_mat 與上面的 lof 求出每個 test point 的 local outlier factor

把 point_id 的 local outlier factor 放入 distance_list

把 local outlier factor 放入 score_for_color_tmp

2.如果 drawing 是 true:

把 globel variable test_data_draw、test_label_draw、score_for_color 分別放入 TSNE(n_components=2).fit_transform(test_data)、test_label、score_for_color_tmp

3.用 test_label(已轉化成 0,1 的形式)與 distance_list 計算 roc_auc_score

Calculate, Record, Drawing

```
#knn
knn_score = [[],[],[]]
k1 = [1,5,10]

#kmeans
kmeans_score = [[],[],[]]
k2 = [1,5,10]

#Distance-based
Cosine_dis_score = []
Minkowski_dis_score = [[],[],[]]
Mahalanobis_dis_score = []
r = [1,2,np.inf]

#Density-based
LOF_score = []
test_data_draw = None
test_label_draw = None
score_for_color = None
```

knn_score:紀錄 K Nearest Neighbor 在不同 k(對應 k1)的結果
kmeans_score:紀錄 Cluster-based 在不同 k(對應 k2)的結果
Cosine_dis_score: 紀錄 Cosine Distance-based 的結果
Minkowski_dis_score = 紀錄 Minkowski Distance-based 在不同 r(對應 r)的結果
Mahalanobis_dis_score : 紀錄 Mahalanobis Distance-based 的結果
LOF_score :紀錄 Density-based 的結果
test_data_draw: 在 Density-based 中畫圖用的 data
test_label_draw:test_data_draw 對應的 label
score_for_color: test_data_draw 對應的 score

```
for i in tqdm.tqdm(range(10)):
    train_data = orig_train_data[orig_train_label==i]
    test_data,test_label = resample(orig_test_data,orig_test_label,target_label=i,outlier_ratio=0.1)
    # [TODO] prepare training/testing data with label==i labeled as 0, and others labeled as 1
    test_label_01 = np.zeros(test_data.shape[0])
    for j in range(len(test_label_01)):
        if test_label[j] != i:
            test_label_01[j] = 1
    test_label = test_label_01
    # [TODO] implement methods
    # [TODO] record ROC-AUC for each method
    for j in range(len(k1)):
        knn_score[j].append(knn(train_data,test_data,test_label,k=k1[j]))
    for j in range(len(k2)):
        kmeans_score[j].append(kmeans(train_data,test_data,test_label,k=k2[j]))
    Cosine_dis_score.append(k_dis(train_data,test_data,test_label,k=5,distance_type='cosine',r=None))
    for j in range(len(r)):
        Minkowski_dis_score[j].append(k_dis(train_data,test_data,test_label,k=5,distance_type='minkowski',r=r[j]))
    Mahalanobis_dis_score.append(k_dis(train_data,test_data,test_label,k=5,distance_type='mahalanobis',r=None))
    LOF_score.append(LOF(test_data,test_label,k=5,drawing=(i==0)))
```

先把 test_label 轉換成 0,1 的形式
0:normal data(指定 digit)
1:anomaly data(其他 digit)
將每個 digit 的運算結果記錄在上面的 list 中

```python
# [TODO] print the average ROC-AUC for each method
print("------Problem 1------")
for i in range(len(k1)):
    print(f"knn with k = {k1[i]}, score = {np.mean(knn_score[i])}")

print("------Problem 2------")
for i in range(len(k2)):
    print(f"kmeans with k = {k2[i]}, score = {np.mean(kmeans_score[i])}")

print("------Problem 3------")
print(f"Cosine_distance, score = {np.mean(Cosine_dis_score)}")
for i in range(len(r)):
    print(f"Minkowski_distance with r = {r[i]}, score = {np.mean(Minkowski_dis_score[i])}")
print(f"Mahalanobis_distance, score = {np.mean(Mahalanobis_dis_score)}")

print("------Problem 4------")
print(f"Local Outlier Factor, score = {np.mean(LOF_score)}")
```

印出每個紀錄 roc_auc_score 的 list 的平均值

```python
fig, axs = plt.subplots(1,2)

axs[0].set_title("predicted LOF score for normal digit=0")
x_val = [test_data_draw[i][0] for i in range(len(test_label_draw))]
y_val = [test_data_draw[i][1] for i in range(len(test_label_draw))]
sc = axs[0].scatter(x_val,y_val,c=score_for_color)
axs[1].set_title("ground truth label for normal digit=0")
indices0 = [i for i in range(len(test_label_draw)) if test_label_draw[i] == 0]
indices1 = [i for i in range(len(test_label_draw)) if test_label_draw[i] == 1]
normal_x = [test_data_draw[i][0] for i in indices0]
normal_y = [test_data_draw[i][1] for i in indices0]
anomaly_x = [test_data_draw[i][0] for i in indices1]
anomaly_y = [test_data_draw[i][1] for i in indices1]
axs[1].scatter(normal_x,normal_y,color="blue",label='normal')
axs[1].scatter(anomaly_x,anomaly_y,color="orange",label='anomaly')
axs[1].legend()

plt.colorbar(sc)
plt.tight_layout()
plt.show()
```

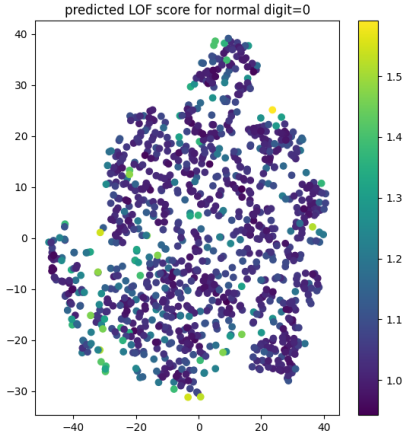
分成兩張 subplot

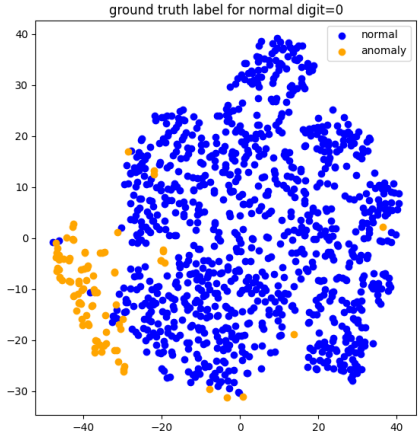
左邊用 test_data_draw 作圖，把對應的 score_for_color 當顏色塗上去，並附上 colorbar sc

右邊把 test_data_draw 用 test_data_draw 區分成 normal(blue)跟 anomaly(orange) 分別作圖

Result

```
------Problem 1------
knn with k = 1, score = 0.9724490864873104
knn with k = 5, score = 0.9723927175119265
knn with k = 10, score = 0.9707676378918354
------Problem 2------
kmeans with k = 1, score = 0.9071134126404894
kmeans with k = 5, score = 0.9584521902378764
kmeans with k = 10, score = 0.963550592620041
------Problem 3------
Cosine_distance, score = 0.9751108388280567
Minkowski_distance with r = 1, score = 0.9531355655969254
Minkowski_distance with r = 2, score = 0.9564024143505202
Minkowski_distance with r = inf, score = 0.9544573346606295
Mahalanobis_distance, score = 0.9754574261730333
------Problem 4------
Local Outlier Factor, score = 0.7814758450676039
```





Observation

problem 5:

1.在 knn 中，雖然不明顯，roc_auc_score 會隨著 k 的上升而下降，因為太大的 k 會使 normal point 的 anomaly score 相對於 anomaly point 上升過多

2.在 kmeans 中，roc_auc_score 會隨著 k 的上升而有明顯上升，因為多個 cluster center 能讓每個 normal point 都被分配到合適的 cluster 使 anomaly score 下降

3.k-distance 除了 Minkowski distance 以外的 roc_auc_score 是 4 個方法中最好的，因為能使 anomaly point 與 normal point 的 anomaly score 有所區別

Mahalanobis distance 有把不同維度的 scale 的影響考慮進去，比只考慮夾角大小，沒有考慮距離的 Cosine distance 表現好一點，而沒有把不同維度的 scale 的影響考慮進去 Minkowski distance 表現最差

在 Minkowski distance 中，r=2 時表現最好，接著是 r=inf，最後是 r=1，因為歐式距離(r=2)比起切比雪夫距離(r=inf)與曼哈頓距離(r=1)較能確切表現兩點間的距離

4.local outlier factor 的 roc_auc_score 是 4 個方法中最低的，因為從圖中可看到 normal point 與 anomaly point 的密度差距不夠明顯