```python
import numpy as np
import matplotlib.pyplot as plt
from   scipy import sparse, stats
from   scipy.linalg import fractional_matrix_power
from   scipy.sparse import csgraph, issparse
from   scipy.spatial.distance import pdist, squareform
import sklearn
from   sklearn import datasets
from   sklearn.datasets import make_circles
from   sklearn.metrics import confusion_matrix, classification_report
from   sklearn.neighbors import NearestNeighbors
from   sklearn.semi_supervised import _label_propagation
from   sklearn.utils.extmath import safe_sparse_dot
import pandas as pd
import pygcn
from   pygcn.data import cora
import os
import warnings
```

```
/opt/homebrew/Caskroom/miniforge/base/envs/CS584/lib/python3.8/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

Assignment 5

Jane Downer
A20452471
CS584-02

Problem 1: Label Propagation

1.

```python
P_0 = np.matrix([0,0,0,-1,-1,1]).T
print(P_0)
```

```
[[ 0]
 [ 0]
 [ 0]
 [-1]
 [-1]
 [ 1]]
```

2.

```python
mult_3 = lambda a,b,c: np.matmul(a,np.matmul(b,c))

def D(S):
    sums = []
    for i in range(len(S)):
        sum_ = sum(S[i][j] for j in range(len(S[i])) if i != j)
        sums.append(sum_)
    return np.round(np.diag(sums),3)

def sim_norm(S):
    D_         = D(S)
    D_neg_half = fractional_matrix_power(D_,-0.5)
    S_norm     = np.round(mult_3(D_neg_half,S,D_neg_half),3)
    return S_norm


S = np.array([[0,1,0,0,1,1],
              [1,0,1,1,0,0],
              [0,1,0,1,0,0],
              [0,1,1,0,0,0],
              [1,0,0,0,0,0],
              [1,0,0,0,0,0]])

S_norm = sim_norm(S)

print('S =\n\n{}\n'      .format(S))
print('S_norm = \n\n{}\n'.format(S_norm))

alpha = 0.8
P_1        = np.multiply(1-alpha,P_0) + np.multiply(alpha,np.matmul(S_norm,P_0))
[l0,l1,l2] = np.concatenate(P_1[:3])

print()
print()
print('P_1 = \n\n{}'.format(np.round(P_1,3)))
print('\nl0 = {}, l1 = {}, l2 = {}'.format(l0, l1, l2))
```

```
S =

[[0 1 0 0 1 1]
 [1 0 1 1 0 0]
 [0 1 0 1 0 0]
 [0 1 1 0 0 0]
 [1 0 0 0 0 0]
 [1 0 0 0 0 0]]

S_norm =

[[0.    0.333 0.    0.    0.577 0.577]
 [0.333 0.    0.408 0.408 0.    0.   ]
 [0.    0.408 0.    0.5   0.    0.   ]
 [0.    0.408 0.5   0.    0.    0.   ]
 [0.577 0.    0.    0.    0.    0.   ]
 [0.577 0.    0.    0.    0.    0.   ]]



P_1 =

[[ 0.   ]
 [-0.326]
 [-0.4  ]
 [-0.2  ]
 [-0.2  ]
 [ 0.2  ]]

l0 = [[0.]], l1 = [[-0.3264]], l2 = [[-0.4]]
```

3.

```python
P_2 = np.multiply(1-0.8,P_1) + np.multiply(0.8,np.matmul(S_norm,P_1))
[l0,l1,l2] = np.concatenate(P_2[:3])

print('P_2 = \n\n{}'.format(np.round(P_2,3)))
print('\nl0 = {}, l1 = {}, l2 = {}'.format(l0, l1, l2))
```

```
P_2 =

[[-0.087]
 [-0.261]
 [-0.267]
 [-0.307]
 [-0.04 ]
 [ 0.04 ]]

l0 = [[-0.08695296]], l1 = [[-0.26112]], l2 = [[-0.26653696]]
```

Positive values are associated with the positive class (Class 1, label 1) and negative values are associated with the negative class (Class 2, label -1).

4.

```python
pwr = lambda x,p: fractional_matrix_power(x,p)

I = np.identity(S_norm.shape[0])

I_aS_neg1 = pwr(I-alpha*S_norm,-1.0)
P_inf     = np.round((1-alpha)*np.matmul(I_aS_neg1,P_0),3)
[l0,l1,l2] = np.concatenate(P_inf[:3])
print('P_inf = \n\n{}\n'.format(P_inf))
print('l0 = {}, l1 = {}, l2 = {}'.format(l0, l1, l2))
```

```
P_inf =

[[-0.097]
 [-0.209]
 [-0.209]
 [-0.352]
 [-0.245]
 [ 0.155]]

l0 = -0.097, l1 = -0.209, l2 = -0.209
```

Positive values are associated with the positive class (Class 1, label 1) and negative values are associated with the negative class (Class 2, label -1).

5.

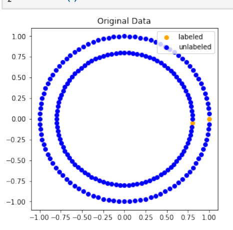
```python
D_ = D(S)
L=D_-S
L_uu = L[:3,:3]
L_ul = L[:3,3:]
Y_l  = P_0[3:]
F_u  = np.round(mult_3(-pwr(L_uu,-1.0),L_ul,Y_l),3)
[[l0],[l1],[l2]] = np.round(F_u,3).tolist()[:3]
print('Using normalized similarity matrix:\n\n')
print('F_u = \n\n{}'.format(F_u))
print('\nl0 = {}, l1 = {}, l2 = {}'.format(l0, l1, l2))
```

```
Using normalized similarity matrix:


F_u =

[[-0.231]
 [-0.692]
 [-0.846]]

l0 = -0.231, l1 = -0.692, l2 = -0.846
```

Positive values are associated with the positive class (Class 1, label 1) and negative values are associated with the negative class (Class 2, label -1).

2.

```python
n_samples    = 200
X,y          = make_circles(n_samples=n_samples,shuffle=False)
outer,inner  = 0,1
labels       = np.full(n_samples,-1)
labels[0]    = outer
labels[-1]   = inner


plt.rcParams["figure.figsize"] = (5,5)
plt.scatter(x=[X[0][0],X[-1][0]],y=[X[0][1],X[-1][1]],color='orange',label='labeled')
plt.scatter(x=X[1:-1][:,0],y=X[1:-1][:,1],color='blue',label='unlabeled')
plt.title('Original Data')
plt.legend(loc='upper right')
plt.show()
```

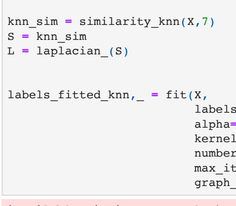



```python
'''
This block of code references the sklearn.semi-supervised._label_propagation
source code:

https://github.com/scikit-learn/scikit-learn/blob/582fa30a3/sklearn/semi_supervised/_label_propagation.py#L332

'''

def fit(X, y,
        alpha,
        kernel='knn',
        max_iter=100,
        tol=0.001,
        number_neighbors=7,
        graph_matrix=[]):

        gamma=0.25
        if graph_matrix==[]:
            graph_matrix = _build_graph1(X,kernel,gamma)
        classes = np.unique(y)
        classes = classes[classes != -1]
        classes_ = classes

        n_samples, n_classes = len(y), len(classes)

        y = np.asarray(y)
        unlabeled = y == -1

        label_distributions_ = np.zeros((n_samples, n_classes))
        for label in classes:
            label_distributions_[y == label, classes == label] = 1

        y_static = np.copy(label_distributions_)
        y_static *= 1 - alpha
        l_previous = np.zeros((X.shape[0], n_classes))

        unlabeled = unlabeled[:, np.newaxis]
        if sparse.isspmatrix(graph_matrix):
            graph_matrix = graph_matrix.tocsr()


        for n_iter_ in range(max_iter):
            if np.abs(label_distributions_ - l_previous).sum() < tol:
                break
            l_previous = label_distributions_
            label_distributions_ = safe_sparse_dot(graph_matrix, label_distributions_)
            label_distributions_ = (
                np.multiply(alpha, label_distributions_) + y_static
            )
        else:
            warnings.warn(
                "max_iter=%d was reached without convergence." % max_iter)
            n_iter_ += 1

        normalizer = np.sum(label_distributions_, axis=1)[:, np.newaxis]
        normalizer[normalizer == 0] = 1
        label_distributions_ /= normalizer

        transduction = classes_[np.argmax(label_distributions_, axis=1)]
        transduction_ = transduction.ravel()
        return transduction,label_distributions_

def _get_kernel(X, y=None,nn_fit=None,gamma=20,n_neighbors=7,n_jobs=1):
    if nn_fit is None:
        nn_fit = NearestNeighbors(
            n_neighbors=n_neighbors).fit(X)
    if y is None:
        return nn_fit.kneighbors_graph(
            nn_fit._fit_X, n_neighbors, mode="connectivity")
    else:
        return nn_fit.kneighbors(y, return_distance=False)

def _build_graph1(X,kernel,gamma):
    if kernel == "knn":
        nn_fit = None
    n_samples = X.shape[0]
    affinity_matrix = _get_kernel(X,gamma=gamma)
    L = csgraph.laplacian(affinity_matrix, normed=True)
    L = -L
    if sparse.isspmatrix(L):
        diag_mask = L.row == L.col
        L.data[diag_mask] = 0.0
    else:
        L.flat[:: n_samples + 1] = 0.0  # set diag to 0.0
    return L
```

```python
def similarity_knn(X,number_neighbors=7):
    nn = NearestNeighbors(n_neighbors=number_neighbors)#,n_jobs=None)
    nn.fit(X)
    S = nn.kneighbors_graph(X, number_neighbors,mode='connectivity')
    S = np.array(S.todense())
    normal = S.sum(axis=0)
    S /= normal[:,np.newaxis]
    return S

def laplacian_(S):
    L = -csgraph.laplacian(S,normed=True)
    L.flat[:: S.shape[0]+1]=0
    return L

knn_sim = similarity_knn(X,7)
S = knn_sim
L = laplacian_(S)


labels_fitted_knn,_ = fit(X,
                          labels,
                          alpha=0.8,
                          kernel='knn',
                          number_neighbors=13,
                          max_iter=500,
                          graph_matrix=knn_sim)
```

```
/var/folders/mq/2x_qs6yx2rdcmj160y9p9h700000gn/T/ipykernel_98670/2401231619.py:10: DeprecationWarning: elementwise comparison failed; this will raise an error in the future.
  if graph_matrix==[]:
```

```python
def split_by_class(X,labels):
    classes = list(np.unique(labels))
    class_points_dict = {c:[] for c in classes}
    for i in range(len(labels)):
        c = labels[i]
        class_points_dict[c] = class_points_dict[c] + [[X[i][0],X[i][1]]]
    points = [np.asarray(L) for L in list(class_points_dict.values())]
    return points

points_0_knn,points_1_knn = split_by_class(X,labels_fitted_knn)

plt.scatter(x=points_0_knn[:,0],y=points_0_knn[:,1],label='Cluster 1',color='blue')
plt.scatter(x=points_1_knn[:,0],y=points_1_knn[:,1],label='Cluster 2',color='orange')
plt.title('Fully Labeled Points (KNN kernel)',fontsize=14)
plt.legend(loc='upper right')
plt.show()
```



3.

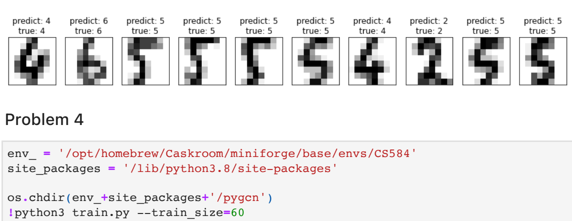
We start by learning a label propagation model with only 10 labeled points, then we select the top 5 most confident points to label. Next, we train with 15 labeled points (original 10 + 5 new ones). We repeat this process 4 times to have a model trained with 30 labeled examples. Please report accuracy and confusion matrix after learning each model. The sample code to load the digit dataset is as follows:

```python
#!pip install sklearn
from sklearn import datasets
from scipy import stats
from sklearn.metrics import confusion_matrix, classification_report


digits  = datasets.load_digits()
rng     = np.random.RandomState(0)
indices = np.arange(len(digits.data))
rng.shuffle(indices)

images  = digits.images[indices[:330]]
X3      = digits.data  [indices[:330]]
y3      = digits.target[indices[:330]]
classes = np.unique(y3)


n_total_samples   = len(y3)
n_labeled_points  = 10
all_indices       = range(n_total_samples)
unlabeled_indices = all_indices[n_labeled_points:]
labeled_indices   = all_indices[:n_labeled_points]

y_train = [-1 if i in unlabeled_indices else y3[i] for i in all_indices]
predicted_labels, P_t = fit(X3, y_train, 0.8, kernel='knn')#, gamma=0.25)
these_predicted_labels     = predicted_labels[unlabeled_indices]
```

```python
for _ in range(5):
    predicted_labels, P_t = fit(X3, y_train, 0.8, kernel='knn')#, gamma=0.25)
    these_predicted_labels     = predicted_labels[unlabeled_indices]
    true_labels           = np.array(y3)[unlabeled_indices]
    cm                    = confusion_matrix(true_labels, these_predicted_labels, labels=classes)

    print("Label Spreading model: %d labeled & %d unlabeled points (%d total)" %
         (n_labeled_points, n_total_samples - n_labeled_points, n_total_samples))
    print('\n',classification_report(true_labels, these_predicted_labels,zero_division=1),'\n')

    print('**********************************************************************')

    pred_entropies    = stats.distributions.entropy(P_t.T)
    next_5_indices = np.argsort(pred_entropies[n_labeled_points:])[:5]

    labeled_indices   = list(labeled_indices)+next_5_indices.tolist()
    unlabeled_indices = [i for i in unlabeled_indices if i not in labeled_indices]
    y_train           = [-1 if i in unlabeled_indices else y3[i] for i in all_indices]
    n_labeled_points  = len(labeled_indices)
```

```
Label Spreading model: 10 labeled & 320 unlabeled points (330 total)

              precision    recall  f1-score   support

           0       1.00      0.00      0.00        24
           1       0.00      0.00      0.00        29
           2       0.53      1.00      0.69        31
           3       1.00      0.00      0.00        28
           4       1.00      0.00      0.00        27
           5       0.89      0.69      0.77        35
           6       0.81      0.95      0.87        40
           7       0.52      1.00      0.69        36
           8       0.60      0.88      0.72        33
           9       0.71      0.78      0.74        37

    accuracy                           0.58       320
   macro avg       0.71      0.53      0.45       320
weighted avg       0.70      0.58      0.50       320


**********************************************************************
Label Spreading model: 15 labeled & 315 unlabeled points (330 total)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        23
           1       0.00      0.00      0.00        29
           2       0.48      0.93      0.64        30
           3       1.00      0.00      0.00        28
           4       0.58      0.85      0.69        26
           5       0.88      0.85      0.87        34
           6       1.00      0.93      0.96        40
           7       0.85      0.97      0.91        35
           8       0.61      0.85      0.71        33
           9       0.83      0.78      0.81        37

    accuracy                           0.73       315
   macro avg       0.72      0.72      0.66       315
weighted avg       0.73      0.73      0.68       315


**********************************************************************
Label Spreading model: 20 labeled & 310 unlabeled points (330 total)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       0.00      0.00      0.00        29
           2       0.54      0.93      0.68        30
           3       1.00      0.00      0.00        28
           4       0.77      0.92      0.84        25
           5       0.88      0.82      0.85        34
           6       1.00      0.93      0.96        40
           7       0.92      0.97      0.94        35
           8       0.57      0.97      0.72        32
           9       0.64      0.80      0.71        35

    accuracy                           0.75       310
   macro avg       0.73      0.73      0.67       310
weighted avg       0.74      0.75      0.68       310


**********************************************************************
Label Spreading model: 25 labeled & 305 unlabeled points (330 total)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       0.00      0.00      0.00        29
           2       0.57      0.93      0.71        30
           3       1.00      0.00      0.00        28
           4       0.65      0.96      0.77        23
           5       0.88      0.82      0.85        34
           6       0.76      0.97      0.85        38
           7       0.92      0.94      0.93        35
           8       0.66      0.91      0.76        32
           9       0.72      0.80      0.76        35

    accuracy                           0.74       306
   macro avg       0.71      0.73      0.66       306
weighted avg       0.71      0.74      0.67       306


**********************************************************************
Label Spreading model: 30 labeled & 300 unlabeled points (330 total)

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        22
           1       0.00      0.00      0.00        29
           2       0.72      1.00      0.84        28
           3       0.92      0.85      0.88        27
           4       0.62      0.95      0.75        22
           5       0.88      0.82      0.85        34
           6       0.95      0.97      0.96        38
           7       0.97      0.86      0.91        35
           8       0.67      0.91      0.77        32
           9       0.72      0.74      0.73        35

    accuracy                           0.81       302
   macro avg       0.74      0.81      0.77       302
weighted avg       0.75      0.81      0.77       302


**********************************************************************
/opt/homebrew/Caskroom/miniforge/base/envs/CS584/lib/python3.8/site-packages/scipy/stats/_entropy.py:77: RuntimeWarning: invalid value encountered in true_divide
  pk = 1.0*pk / np.sum(pk, axis=axis, keepdims=True)
```

```python
uncertainty_index = np.argsort(pred_entropies)[50:100]

f = plt.figure(figsize=(15, 5))
for index, image_index in enumerate(uncertainty_index):
    image = images[image_index]
    if index+1 > 20:
        break
    sub = f.add_subplot(2, 10, index+1)
    sub.imshow(image, cmap=plt.cm.gray_r)
    plt.xticks([])
    plt.yticks([])
    sub.set_title('predict: %i\ntrue: %i' % (
        predicted_labels[image_index], y3[image_index]))

f.suptitle('Learning with small amount of labeled data')
plt.show()
```



Problem 4

```python
env_ = '/opt/homebrew/Caskroom/miniforge/base/envs/CS584'
site_packages = '/lib/python3.8/site-packages'

os.chdir(env_+site_packages+'/pygcn')
!python3 train.py --train_size=60
!python3 train.py --train_size=120
!python3 train.py --train_size=180
!python3 train.py --train_size=240
```

```
Loading cora dataset...
Test set results: loss= 0.8259 accuracy= 0.7540
Loading cora dataset...
Test set results: loss= 0.7610 accuracy= 0.8120
Loading cora dataset...
Test set results: loss= 0.6889 accuracy= 0.8280
Loading cora dataset...
Test set results: loss= 0.6908 accuracy= 0.8400
```