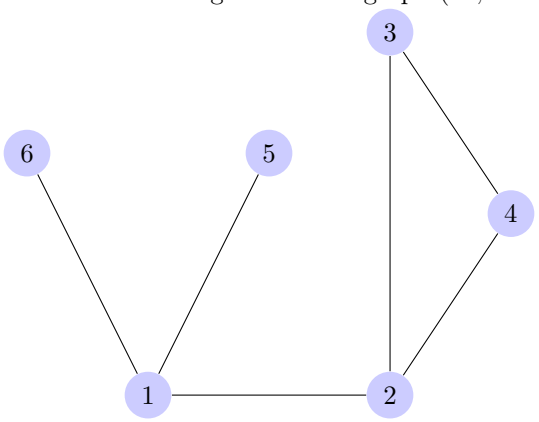
Problem 1 (25 Points): Label Propagation

Consider the following undirected graph (i.e, all edges have an edge weight 1) :



1. **3 Points.** Consider a binary classification problem. Write down the initial label vector P_0 for this graph where v_6 has observed label 1 and v_4 and v_5 have observed label 2.
2. **4 Points.** Perform 1 iteration of the label spreading algorithm with the decay parameter $\alpha = 0.8$ and determine the node labels for the unlabeled nodes v_1, v_2, and v_3, i.e., compute P_1 and provide the labels l_1, l_2, and l_3 after 1 iteration.
3. **4 Points.** Perform 2 iterations of the label spreading algorithm with the decay parameter $\alpha = 0.8$ and determine the node labels for the unlabeled nodes v_1, v_2, and v_3, i.e., compute P_2 and provide the labels l_1, l_2, and l_3 after 2 iterations.
4. **5 Points.** Perform infinite iterations of the label spreading algorithm with the decay parameter $\alpha = 0.8$ and determine the node labels for the unlabeled nodes v_1, v_2, and v_3, i.e., compute P_∞ and provide the labels l_1, l_2, and l_3 after infinite iterations.
5. **9 Points.** Determine the node labels for the unlabeled nodes v_1, v_2, and v_3 via the energy minimization algorithm.

Problem 2 (25 Points): Implementing Label Spreading

In this problem, you will implement Label Spreading to classify data points from two circles (See left figure in Figure 1). As both label groups lie inside their own distinct shape, we can see that Label Spreading can propagate labels correctly around the circle (See right figure in Figure 1).

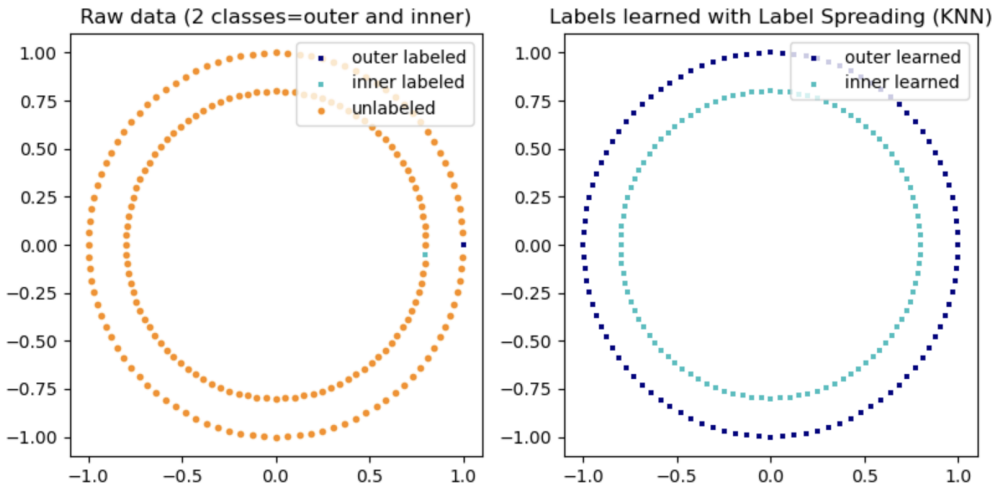


Figure 1: Dataset for Problem 2.

The following sample code is used to generate the two circles.

```
import numpy as np
from sklearn.datasets import make_circles
n_samples = 200
X, y = make_circles(n_samples=n_samples, shuffle=False)
outer, inner = 0, 1
labels = np.full(n_samples, -1.0)
labels[0] = outer
labels[-1] = inner
```

Problem 3 (25 Points): Implementing Label Propagation with Self-Training

In this problem, we will incorporate self-training into label propagation (using energy minimization) to classify handwritten digits.

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:

```
import numpy as np
from sklearn import datasets

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

X = digits.data[indices[:330]]
y = digits.target[indices[:330]]
images = digits.images[indices[:330]]

n_total_samples = len(y)
n_labeled_points = 10
```

Problem 4 (25 Points): Implementing Graph Convolutional Networks

Graph convolutional network (GCN) is a well-known graph neural network for semi-supervised classification. More details please refer to https://arxiv.org/pdf/1609.02907.pdf.

The source code is in github (https://github.com/tkipf/pygcn). Please test GCN on the Cora dataset for node classification. Specifically, show GCN's node classification accuracy on Cora with the number of labeled nodes to be 60, 120, 180, 240, 300, respectively.