Problem 1:

Part 1:

```
#Part 1
import numpy as np

P0 = np.array([0, 0, 0, 2, 2, 1])
print("Initial Label Vector P0:", P0)
```

```
Initial Label Vector P0: [0 0 0 2 2 1]
```

Part 2:

```
#Part 2
S = np.array([
    [0, 1, 0, 0, 0, 1],
    [1, 0, 1, 1, 0, 0],
    [0, 1, 0, 0, 1, 0],
    [0, 1, 0, 0, 0, 0],
    [0, 0, 1, 0, 0, 0],
    [1, 0, 0, 0, 0, 0]
])

D = np.diag(S.sum(axis=1))
S_normalized = np.linalg.inv(D) @ S
alpha = 0.8
P1 = (1 - alpha) * P0 + alpha * S_normalized @ P0
l1, l2, l3 = P1[0], P1[1], P1[2]

print("Labels after 1 iteration (P1):", P1)
print("l1, l2, l3 after 1 iteration:", l1, l2, l3)
```

```
Labels after 1 iteration (P1): [0.4        0.53333333 0.8        0.4        0.4        0.2       ]
l1, l2, l3 after 1 iteration: 0.4 0.5333333333333333 0.8
```

Part 3:

```
# Part 3
P2 = (1 - alpha) * P0 + alpha * S_normalized @ P1

l1, l2, l3 = P2[0], P2[1], P2[2]
print("Labels after 2 iterations (P2):", P2)
print("l1, l2, l3 after 2 iterations:", l1, l2, l3)
```

```
Labels after 2 iterations (P2): [0.29333333 0.42666667 0.37333333 0.82666667 1.04       0.52      ]
l1, l2, l3 after 2 iterations: 0.29333333333333333 0.4266666666666667 0.3733333333333333
```

Part 4:

```
# Part 4
epsilon = 1e-6

Pn = P2
while True:
    P_next = (1 - alpha) * P0 + alpha * S_normalized @ Pn
    if np.linalg.norm(P_next - Pn) < epsilon:
        break
    Pn = P_next

l1, l2, l3 = Pn[0], Pn[1], Pn[2]
print("Labels after infinite iterations (P∞):", Pn)
print("l1, l2, l3 after infinite iterations:", l1, l2, l3)
```

```
Labels after infinite iterations (P∞): [0.36737858 0.42454416 0.48502564 0.73963494 0.78802089 0.49390325]
l1, l2, l3 after infinite iterations: 0.36737857946634245 0.4245441597786095 0.4850256382898714
```

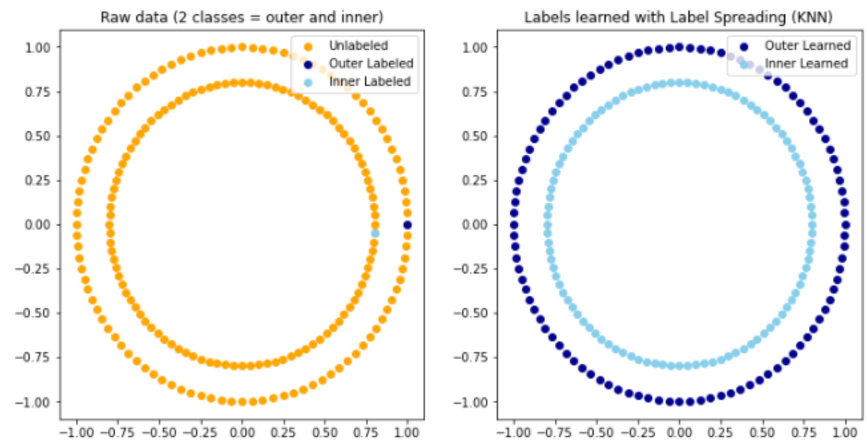
Part 5:

```python
#Part 5
from scipy.linalg import inv

L = D - S
Luu = L[:3, :3]
Lul = L[:3, 3:]
Fl = P0[3:]
Fu = -inv(Luu) @ Lul @ Fl
print("Labels for the unlabeled nodes v1, v2, and v3 after energy minimization:", Fu)
```

```
Labels for the unlabeled nodes v1, v2, and v3 after energy minimization: [1.375 1.75  1.875]
```

Problem 2:

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_circles
from sklearn.semi_supervised import LabelSpreading

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

label_spread = LabelSpreading(kernel='knn', alpha=0.8)
label_spread.fit(X, labels)

predicted_labels = label_spread.transduction_

plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.scatter(X[:, 0], X[:, 1], c='orange', label='Unlabeled')
plt.scatter(X[0, 0], X[0, 1], color='darkblue', label='Outer Labeled')
plt.scatter(X[-1, 0], X[-1, 1], color='skyblue', label='Inner Labeled')
plt.legend(loc='upper right')
plt.title("Raw data (2 classes = outer and inner)")

plt.subplot(1, 2, 2)
outer_learned_indices = np.where(predicted_labels == 0)[0]
inner_learned_indices = np.where(predicted_labels == 1)[0]

plt.scatter(X[outer_learned_indices, 0], X[outer_learned_indices, 1], color='darkblue', label='Outer Learned')
plt.scatter(X[inner_learned_indices, 0], X[inner_learned_indices, 1], color='skyblue', label='Inner Learned')
plt.legend(loc='upper right')
plt.title("Labels learned with Label Spreading (KNN)")
plt.show()
```



Problem 3:

```python
import numpy as np
from sklearn import datasets
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.semi_supervised import LabelPropagation

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
labels = np.full(n_total_samples, -1)
labels[:n_labeled_points] = y[:n_labeled_points]

for iteration in range(4):
    label_prop_model = LabelPropagation(kernel='knn')
    label_prop_model.fit(X, labels)

    probabilities = label_prop_model.predict_proba(X)
    confidence = np.max(probabilities, axis=1)
    unlabeled_indices = np.where(labels == -1)[0]
    top_5_indices = unlabeled_indices[np.argsort(confidence[unlabeled_indices])[-5:]]

    labels[top_5_indices] = y[top_5_indices]

    predicted_labels = label_prop_model.transduction_
    acc = accuracy_score(y, predicted_labels)
    conf_matrix = confusion_matrix(y, predicted_labels)

    num_labeled_points = np.sum(labels != -1)

    print(f"Iteration {iteration + 1}:")
    print(f"Labeled Points: {num_labeled_points}")
    print(f"Accuracy: {acc:.4f}")
    print("Confusion Matrix:")
    print(conf_matrix)
    print()
```

```
Iteration 1:
Labeled Points: 15
Accuracy: 0.1182
Confusion Matrix:
[[ 0 24  0  0  0  0  0  0  0  0]
 [ 0 30  0  0  0  0  0  0  0  0]
 [ 0 31  2  0  0  0  0  0  0  0]
 [ 0 28  0  0  0  0  0  0  0  0]
 [ 0 27  0  0  0  0  0  0  0  0]
 [ 0 35  0  0  0  1  0  0  0  0]
 [ 0 40  0  0  0  0  2  0  0  0]
 [ 0 36  0  0  0  0  0  1  0  0]
 [ 0 33  0  0  0  0  0  0  2  0]
 [ 0 37  0  0  0  0  0  0  0  1]]

Iteration 2:
Labeled Points: 20
Accuracy: 0.1152
Confusion Matrix:
[[24  0  0  0  0  0  0  0  0  0]
 [29  1  0  0  0  0  0  0  0  0]
 [31  0  2  0  0  0  0  0  0  0]
 [28  0  0  0  0  0  0  0  0  0]
 [26  0  0  0  1  0  0  0  0  0]
 [35  0  0  0  0  1  0  0  0  0]
 [40  0  0  0  0  0  2  0  0  0]
 [36  0  0  0  0  0  0  1  0  0]
 [31  0  0  0  0  0  0  0  4  0]
 [36  0  0  0  0  0  0  0  0  2]]
```

```
Iteration 3:
Labeled Points: 25
Accuracy: 0.1303
Confusion Matrix:
[[24  0  0  0  0  0  0  0  0  0]
 [28  2  0  0  0  0  0  0  0  0]
 [30  0  3  0  0  0  0  0  0  0]
 [27  0  0  1  0  0  0  0  0  0]
 [26  0  0  0  1  0  0  0  0  0]
 [35  0  0  0  0  1  0  0  0  0]
 [40  0  0  0  0  0  2  0  0  0]
 [35  0  0  0  0  0  0  2  0  0]
 [31  0  0  0  0  0  0  0  4  0]
 [35  0  0  0  0  0  0  0  0  3]]

Iteration 4:
Labeled Points: 30
Accuracy: 0.1455
Confusion Matrix:
[[24  0  0  0  0  0  0  0  0  0]
 [28  2  0  0  0  0  0  0  0  0]
 [30  0  3  0  0  0  0  0  0  0]
 [27  0  0  1  0  0  0  0  0  0]
 [25  0  0  0  2  0  0  0  0  0]
 [34  0  0  0  0  2  0  0  0  0]
 [40  0  0  0  0  0  2  0  0  0]
 [34  0  0  0  0  0  0  3  0  0]
 [29  0  0  0  0  0  0  0  6  0]
 [35  0  0  0  0  0  0  0  0  3]]
```

Problem 4:

train.py Code:

train.py M ×

pygcn > train.py > ...

You, 3 minutes ago | 5 authors (Thomas Kipf and others)

```python
from __future__ import division        Thomas Kipf, 7 years ago • Initial commit
from __future__ import print_function

import time
import argparse
import numpy as np
import torch
import torch.nn.functional as F
import torch.optim as optim

from pygcn.utils import load_data, accuracy
from pygcn.models import GCN

# Training settings
parser = argparse.ArgumentParser()
parser.add_argument('--no-cuda', action='store_true', default=False,
                    help='Disables CUDA training.')
parser.add_argument('--fastmode', action='store_true', default=False,
                    help='Validate during training pass.')
parser.add_argument('--seed', type=int, default=42, help='Random seed.')
parser.add_argument('--epochs', type=int, default=200,
                    help='Number of epochs to train.')
parser.add_argument('--lr', type=float, default=0.01,
                    help='Initial learning rate.')
parser.add_argument('--weight_decay', type=float, default=5e-4,
                    help='Weight decay (L2 loss on parameters).')
parser.add_argument('--hidden', type=int, default=16,
                    help='Number of hidden units.')
parser.add_argument('--dropout', type=float, default=0.5,
                    help='Dropout rate (1 - keep probability).')

args = parser.parse_args()
args.cuda = not args.no_cuda and torch.cuda.is_available()

np.random.seed(args.seed)
torch.manual_seed(args.seed)
if args.cuda:
    torch.cuda.manual_seed(args.seed)

# Load data
adj, features, labels, idx_train, idx_val, idx_test = load_data()
```

```python
# Model and optimizer
model = GCN(nfeat=features.shape[1],
            nhid=args.hidden,
            nclass=labels.max().item() + 1,
            dropout=args.dropout)
optimizer = optim.Adam(model.parameters(),
                       lr=args.lr, weight_decay=args.weight_decay)

if args.cuda:
    model.cuda()
    features = features.cuda()
    adj = adj.cuda()
    labels = labels.cuda()
    idx_train = idx_train.cuda()
    idx_val = idx_val.cuda()
    idx_test = idx_test.cuda()

def train(epoch):
    t = time.time()
    model.train()
    optimizer.zero_grad()
    output = model(features, adj)
    loss_train = F.nll_loss(output[idx_train], labels[idx_train])
    acc_train = accuracy(output[idx_train], labels[idx_train])
    loss_train.backward()
    optimizer.step()

    if not args.fastmode:
        # Evaluate validation set performance separately,
        # deactivates dropout during validation run.
        model.eval()
        output = model(features, adj)

    loss_val = F.nll_loss(output[idx_val], labels[idx_val])
    acc_val = accuracy(output[idx_val], labels[idx_val])
    print('Epoch: {:04d}'.format(epoch+1),
          'loss_train: {:.4f}'.format(loss_train.item()),
          'acc_train: {:.4f}'.format(acc_train.item()),
          'loss_val: {:.4f}'.format(loss_val.item()),
          'acc_val: {:.4f}'.format(acc_val.item()),
          'time: {:.4f}s'.format(time.time() - t))
```

```python
def test():
    model.eval()
    output = model(features, adj)
    loss_test = F.nll_loss(output[idx_test], labels[idx_test])
    acc_test = accuracy(output[idx_test], labels[idx_test])
    print("Test set results:",
          "loss= {:.4f}".format(loss_test.item()),
          "accuracy= {:.4f}".format(acc_test.item()))
    return acc_test.item()

def adjust_labeled_nodes(num_labeled):
    global idx_train
    total_train_nodes = len(idx_train)
    if num_labeled > total_train_nodes:
        num_labeled = total_train_nodes
    idx_train_new = np.random.choice(idx_train.cpu().numpy(), num_labeled, replace=False)

    return torch.LongTensor(idx_train_new).to(idx_train.device)

labeled_nodes_list = [60, 120, 180, 240, 300]
for num_labeled in labeled_nodes_list:

    idx_train_new = adjust_labeled_nodes(num_labeled)
    t_total = time.time()

    for epoch in range(args.epochs):
        train(epoch)
    print("Optimization Finished!")
    test_acc = test()
    print(f"Accuracy with {num_labeled} labeled nodes: {test_acc:.4f}")
```

Running train.py:

```
C:\Users\Abhiram\Downloads\pygcn\pygcn>python train.py
Loading cora dataset...
C:\Users\Abhiram\AppData\Local\Programs\Python\Python39\lib\site-packages\pygcn-0.1-py3.9.egg\pygcn\utils.py:80: UserWarning: torch.sparse.SparseT
ensor(indices, values, shape, *, device=) is deprecated.  Please use torch.sparse_coo_tensor(indices, values, shape, dtype=, device=). (Triggered
internally at C:\actions-runner\_work\pytorch\pytorch\builder\windows\pytorch\torch\csrc\utils\tensor_new.cpp:653.)
Epoch: 0001 loss_train: 2.0177 acc_train: 0.0571 loss_val: 2.0176 acc_val: 0.0833 time: 0.3799s
Epoch: 0002 loss_train: 2.0041 acc_train: 0.0929 loss_val: 2.0029 acc_val: 0.0833 time: 0.0180s
Epoch: 0003 loss_train: 1.9906 acc_train: 0.0929 loss_val: 1.9889 acc_val: 0.0833 time: 0.0237s
Epoch: 0004 loss_train: 1.9808 acc_train: 0.1000 loss_val: 1.9756 acc_val: 0.0833 time: 0.0255s
Epoch: 0005 loss_train: 1.9677 acc_train: 0.1143 loss_val: 1.9631 acc_val: 0.0833 time: 0.0346s
Epoch: 0006 loss_train: 1.9459 acc_train: 0.1500 loss_val: 1.9512 acc_val: 0.2033 time: 0.0297s
Epoch: 0007 loss_train: 1.9457 acc_train: 0.2000 loss_val: 1.9400 acc_val: 0.1567 time: 0.0330s
Epoch: 0008 loss_train: 1.9308 acc_train: 0.1857 loss_val: 1.9289 acc_val: 0.1567 time: 0.0286s
Epoch: 0009 loss_train: 1.9212 acc_train: 0.1929 loss_val: 1.9179 acc_val: 0.1567 time: 0.0249s
Epoch: 0010 loss_train: 1.9094 acc_train: 0.2214 loss_val: 1.9069 acc_val: 0.1567 time: 0.0297s
Epoch: 0011 loss_train: 1.9010 acc_train: 0.2071 loss_val: 1.8959 acc_val: 0.1567 time: 0.0296s
Epoch: 0012 loss_train: 1.8881 acc_train: 0.1929 loss_val: 1.8850 acc_val: 0.1567 time: 0.0293s
Epoch: 0013 loss_train: 1.8821 acc_train: 0.2000 loss_val: 1.8740 acc_val: 0.1567 time: 0.0283s
Epoch: 0014 loss_train: 1.8692 acc_train: 0.2143 loss_val: 1.8629 acc_val: 0.1567 time: 0.0290s
Epoch: 0015 loss_train: 1.8535 acc_train: 0.2071 loss_val: 1.8521 acc_val: 0.1567 time: 0.0270s
Epoch: 0016 loss_train: 1.8435 acc_train: 0.2214 loss_val: 1.8417 acc_val: 0.1567 time: 0.0262s
Epoch: 0017 loss_train: 1.8187 acc_train: 0.2857 loss_val: 1.8317 acc_val: 0.1600 time: 0.0277s
Epoch: 0018 loss_train: 1.8179 acc_train: 0.2357 loss_val: 1.8219 acc_val: 0.1667 time: 0.0292s
Epoch: 0019 loss_train: 1.8084 acc_train: 0.2571 loss_val: 1.8122 acc_val: 0.2400 time: 0.0278s
Epoch: 0020 loss_train: 1.7905 acc_train: 0.3429 loss_val: 1.8024 acc_val: 0.3667 time: 0.0225s
Epoch: 0021 loss_train: 1.7834 acc_train: 0.3214 loss_val: 1.7927 acc_val: 0.4667 time: 0.0330s
Epoch: 0022 loss_train: 1.7832 acc_train: 0.3429 loss_val: 1.7830 acc_val: 0.4433 time: 0.0313s
Epoch: 0023 loss_train: 1.7512 acc_train: 0.4214 loss_val: 1.7733 acc_val: 0.4100 time: 0.0337s
Epoch: 0024 loss_train: 1.7484 acc_train: 0.3429 loss_val: 1.7637 acc_val: 0.3700 time: 0.0374s
Epoch: 0025 loss_train: 1.7534 acc_train: 0.3857 loss_val: 1.7544 acc_val: 0.3600 time: 0.0660s
Epoch: 0026 loss_train: 1.7380 acc_train: 0.3500 loss_val: 1.7452 acc_val: 0.3533 time: 0.0538s
Epoch: 0027 loss_train: 1.7125 acc_train: 0.3643 loss_val: 1.7358 acc_val: 0.3467 time: 0.0857s
Epoch: 0028 loss_train: 1.7368 acc_train: 0.3929 loss_val: 1.7266 acc_val: 0.3467 time: 0.0453s
Epoch: 0029 loss_train: 1.6824 acc_train: 0.3500 loss_val: 1.7173 acc_val: 0.3467 time: 0.0377s
Epoch: 0030 loss_train: 1.6850 acc_train: 0.3500 loss_val: 1.7082 acc_val: 0.3467 time: 0.0230s
Epoch: 0031 loss_train: 1.6840 acc_train: 0.3357 loss_val: 1.6989 acc_val: 0.3467 time: 0.0195s
Epoch: 0032 loss_train: 1.6530 acc_train: 0.3643 loss_val: 1.6896 acc_val: 0.3467 time: 0.0210s
Epoch: 0033 loss_train: 1.6532 acc_train: 0.3571 loss_val: 1.6802 acc_val: 0.3467 time: 0.0272s
Epoch: 0034 loss_train: 1.6607 acc_train: 0.3714 loss_val: 1.6707 acc_val: 0.3467 time: 0.0225s
Epoch: 0035 loss_train: 1.6373 acc_train: 0.3571 loss_val: 1.6614 acc_val: 0.3500 time: 0.0206s
Epoch: 0036 loss_train: 1.6337 acc_train: 0.3643 loss_val: 1.6521 acc_val: 0.3533 time: 0.0231s
Epoch: 0037 loss_train: 1.6049 acc_train: 0.4000 loss_val: 1.6424 acc_val: 0.3600 time: 0.0249s
Epoch: 0038 loss_train: 1.5957 acc_train: 0.3786 loss_val: 1.6328 acc_val: 0.3633 time: 0.0201s
Epoch: 0039 loss_train: 1.5887 acc_train: 0.3857 loss_val: 1.6231 acc_val: 0.3733 time: 0.0221s
Epoch: 0040 loss_train: 1.5746 acc_train: 0.3429 loss_val: 1.6133 acc_val: 0.3967 time: 0.0219s
Epoch: 0041 loss_train: 1.5562 acc_train: 0.4000 loss_val: 1.6034 acc_val: 0.4133 time: 0.0260s
Epoch: 0042 loss_train: 1.5468 acc_train: 0.4429 loss_val: 1.5932 acc_val: 0.4233 time: 0.0240s
Epoch: 0043 loss_train: 1.5097 acc_train: 0.4143 loss_val: 1.5828 acc_val: 0.4333 time: 0.0216s
Epoch: 0044 loss_train: 1.4860 acc_train: 0.4500 loss_val: 1.5721 acc_val: 0.4433 time: 0.0229s
Epoch: 0045 loss_train: 1.4807 acc_train: 0.4857 loss_val: 1.5612 acc_val: 0.4533 time: 0.0257s
Epoch: 0046 loss_train: 1.5008 acc_train: 0.4429 loss_val: 1.5497 acc_val: 0.4633 time: 0.0240s
Epoch: 0047 loss_train: 1.4737 acc_train: 0.5286 loss_val: 1.5376 acc_val: 0.4667 time: 0.0246s
```

Accuracy with 60 Labelled Nodes:

```
Epoch: 0186 loss_train: 0.4749 acc_train: 0.9143 loss_val: 0.7158 acc_val: 0.8233 time: 0.0239s
Epoch: 0187 loss_train: 0.4836 acc_train: 0.9429 loss_val: 0.7127 acc_val: 0.8200 time: 0.0253s
Epoch: 0188 loss_train: 0.4500 acc_train: 0.9143 loss_val: 0.7105 acc_val: 0.8167 time: 0.0283s
Epoch: 0189 loss_train: 0.4627 acc_train: 0.9214 loss_val: 0.7088 acc_val: 0.8133 time: 0.0275s
Epoch: 0190 loss_train: 0.4302 acc_train: 0.9286 loss_val: 0.7078 acc_val: 0.8133 time: 0.0240s
Epoch: 0191 loss_train: 0.4649 acc_train: 0.9000 loss_val: 0.7063 acc_val: 0.8133 time: 0.0260s
Epoch: 0192 loss_train: 0.4617 acc_train: 0.8857 loss_val: 0.7056 acc_val: 0.8167 time: 0.0265s
Epoch: 0193 loss_train: 0.4580 acc_train: 0.9143 loss_val: 0.7062 acc_val: 0.8233 time: 0.0220s
Epoch: 0194 loss_train: 0.4375 acc_train: 0.9429 loss_val: 0.7066 acc_val: 0.8267 time: 0.0265s
Epoch: 0195 loss_train: 0.4264 acc_train: 0.9286 loss_val: 0.7066 acc_val: 0.8267 time: 0.0263s
Epoch: 0196 loss_train: 0.4377 acc_train: 0.9500 loss_val: 0.7069 acc_val: 0.8233 time: 0.0265s
Epoch: 0197 loss_train: 0.4219 acc_train: 0.9286 loss_val: 0.7066 acc_val: 0.8233 time: 0.0244s
Epoch: 0198 loss_train: 0.4384 acc_train: 0.9286 loss_val: 0.7073 acc_val: 0.8233 time: 0.0271s
Epoch: 0199 loss_train: 0.4256 acc_train: 0.9071 loss_val: 0.7078 acc_val: 0.8233 time: 0.0240s
Epoch: 0200 loss_train: 0.4629 acc_train: 0.9357 loss_val: 0.7078 acc_val: 0.8233 time: 0.0246s
Optimization Finished!
Test set results: loss= 0.7522 accuracy= 0.8270
Accuracy with 60 labeled nodes: 0.8270
Epoch: 0001 loss_train: 0.4040 acc_train: 0.9286 loss_val: 0.7083 acc_val: 0.8200 time: 0.0259s
Epoch: 0002 loss_train: 0.4066 acc_train: 0.9643 loss_val: 0.7092 acc_val: 0.8167 time: 0.0264s
Epoch: 0003 loss_train: 0.4332 acc_train: 0.9357 loss_val: 0.7092 acc_val: 0.8167 time: 0.0278s
Epoch: 0004 loss_train: 0.4359 acc_train: 0.9143 loss_val: 0.7075 acc_val: 0.8167 time: 0.0216s
Epoch: 0005 loss_train: 0.3760 acc_train: 0.9500 loss_val: 0.7045 acc_val: 0.8167 time: 0.0216s
Epoch: 0006 loss_train: 0.4056 acc_train: 0.9214 loss_val: 0.7016 acc_val: 0.8200 time: 0.0230s
Epoch: 0007 loss_train: 0.4542 acc_train: 0.9357 loss_val: 0.6994 acc_val: 0.8167 time: 0.0200s
Epoch: 0008 loss_train: 0.4121 acc_train: 0.9571 loss_val: 0.6968 acc_val: 0.8200 time: 0.0210s
Epoch: 0009 loss_train: 0.4141 acc_train: 0.9214 loss_val: 0.6956 acc_val: 0.8200 time: 0.0223s
Epoch: 0010 loss_train: 0.4244 acc_train: 0.9357 loss_val: 0.6929 acc_val: 0.8233 time: 0.0215s
Epoch: 0011 loss_train: 0.3870 acc_train: 0.9786 loss_val: 0.6901 acc_val: 0.8233 time: 0.0210s
Epoch: 0012 loss_train: 0.4426 acc_train: 0.9214 loss_val: 0.6877 acc_val: 0.8200 time: 0.0220s
Epoch: 0013 loss_train: 0.4403 acc_train: 0.9357 loss_val: 0.6866 acc_val: 0.8200 time: 0.0246s
Epoch: 0014 loss_train: 0.3772 acc_train: 0.9500 loss_val: 0.6861 acc_val: 0.8233 time: 0.0198s
```

Accuracy with 120 Labelled Nodes:

```
Epoch: 0188 loss_train: 0.2533 acc_train: 0.9500 loss_val: 0.6396 acc_val: 0.8033 time: 0.0220s
Epoch: 0189 loss_train: 0.2817 acc_train: 0.9571 loss_val: 0.6379 acc_val: 0.8100 time: 0.0220s
Epoch: 0190 loss_train: 0.2523 acc_train: 0.9714 loss_val: 0.6357 acc_val: 0.8067 time: 0.0226s
Epoch: 0191 loss_train: 0.2540 acc_train: 0.9714 loss_val: 0.6348 acc_val: 0.8033 time: 0.0250s
Epoch: 0192 loss_train: 0.2841 acc_train: 0.9714 loss_val: 0.6333 acc_val: 0.8033 time: 0.0210s
Epoch: 0193 loss_train: 0.2752 acc_train: 0.9643 loss_val: 0.6316 acc_val: 0.8100 time: 0.0200s
Epoch: 0194 loss_train: 0.2607 acc_train: 0.9786 loss_val: 0.6298 acc_val: 0.8100 time: 0.0226s
Epoch: 0195 loss_train: 0.2842 acc_train: 0.9714 loss_val: 0.6302 acc_val: 0.8067 time: 0.0225s
Epoch: 0196 loss_train: 0.2954 acc_train: 0.9571 loss_val: 0.6324 acc_val: 0.8133 time: 0.0220s
Epoch: 0197 loss_train: 0.2621 acc_train: 0.9714 loss_val: 0.6360 acc_val: 0.8067 time: 0.0216s
Epoch: 0198 loss_train: 0.2847 acc_train: 0.9714 loss_val: 0.6379 acc_val: 0.8100 time: 0.0225s
Epoch: 0199 loss_train: 0.2950 acc_train: 0.9571 loss_val: 0.6388 acc_val: 0.8100 time: 0.0250s
Epoch: 0200 loss_train: 0.2828 acc_train: 0.9714 loss_val: 0.6382 acc_val: 0.8067 time: 0.0217s
Optimization Finished!
Test set results: loss= 0.6355 accuracy= 0.8330
Accuracy with 120 labeled nodes: 0.8330
Epoch: 0001 loss_train: 0.2943 acc_train: 0.9714 loss_val: 0.6364 acc_val: 0.8033 time: 0.0230s
Epoch: 0002 loss_train: 0.2347 acc_train: 0.9786 loss_val: 0.6349 acc_val: 0.8033 time: 0.0253s
Epoch: 0003 loss_train: 0.2676 acc_train: 0.9643 loss_val: 0.6325 acc_val: 0.8133 time: 0.0240s
Epoch: 0004 loss_train: 0.2608 acc_train: 0.9929 loss_val: 0.6314 acc_val: 0.8100 time: 0.0210s
Epoch: 0005 loss_train: 0.2742 acc_train: 0.9714 loss_val: 0.6324 acc_val: 0.8067 time: 0.0210s
Epoch: 0006 loss_train: 0.2823 acc_train: 0.9857 loss_val: 0.6337 acc_val: 0.8067 time: 0.0230s
Epoch: 0007 loss_train: 0.2321 acc_train: 0.9643 loss_val: 0.6350 acc_val: 0.8067 time: 0.0220s
Epoch: 0008 loss_train: 0.2533 acc_train: 0.9786 loss_val: 0.6365 acc_val: 0.8067 time: 0.0210s
Epoch: 0009 loss_train: 0.2831 acc_train: 0.9643 loss_val: 0.6368 acc_val: 0.8067 time: 0.0230s
Epoch: 0010 loss_train: 0.2524 acc_train: 0.9643 loss_val: 0.6368 acc_val: 0.8133 time: 0.0215s
Epoch: 0011 loss_train: 0.2783 acc_train: 0.9786 loss_val: 0.6351 acc_val: 0.8133 time: 0.0243s
Epoch: 0012 loss_train: 0.3050 acc_train: 0.9429 loss_val: 0.6334 acc_val: 0.8133 time: 0.0250s
Epoch: 0013 loss_train: 0.2621 acc_train: 0.9786 loss_val: 0.6312 acc_val: 0.8167 time: 0.0253s
Epoch: 0014 loss_train: 0.2623 acc_train: 0.9714 loss_val: 0.6275 acc_val: 0.8167 time: 0.0226s
```

Accuracy with 180 Labelled Nodes:

```
Epoch: 0189 loss_train: 0.2620 acc_train: 0.9714 loss_val: 0.6173 acc_val: 0.8133 time: 0.0235s
Epoch: 0190 loss_train: 0.1963 acc_train: 0.9857 loss_val: 0.6173 acc_val: 0.8100 time: 0.0265s
Epoch: 0191 loss_train: 0.2098 acc_train: 0.9786 loss_val: 0.6182 acc_val: 0.8167 time: 0.0214s
Epoch: 0192 loss_train: 0.2564 acc_train: 0.9571 loss_val: 0.6198 acc_val: 0.8033 time: 0.0231s
Epoch: 0193 loss_train: 0.2637 acc_train: 0.9786 loss_val: 0.6204 acc_val: 0.8033 time: 0.0230s
Epoch: 0194 loss_train: 0.2308 acc_train: 0.9643 loss_val: 0.6227 acc_val: 0.8033 time: 0.0275s
Epoch: 0195 loss_train: 0.2262 acc_train: 0.9786 loss_val: 0.6230 acc_val: 0.8033 time: 0.0298s
Epoch: 0196 loss_train: 0.2230 acc_train: 0.9786 loss_val: 0.6202 acc_val: 0.8000 time: 0.0254s
Epoch: 0197 loss_train: 0.2273 acc_train: 0.9857 loss_val: 0.6169 acc_val: 0.8033 time: 0.0266s
Epoch: 0198 loss_train: 0.2118 acc_train: 0.9929 loss_val: 0.6136 acc_val: 0.8200 time: 0.0239s
Epoch: 0199 loss_train: 0.2304 acc_train: 0.9786 loss_val: 0.6115 acc_val: 0.8200 time: 0.0200s
Epoch: 0200 loss_train: 0.2303 acc_train: 0.9714 loss_val: 0.6105 acc_val: 0.8300 time: 0.0235s
Optimization Finished!
Test set results: loss= 0.6024 accuracy= 0.8370
Accuracy with 180 labeled nodes: 0.8370
Epoch: 0001 loss_train: 0.2453 acc_train: 0.9643 loss_val: 0.6110 acc_val: 0.8233 time: 0.0220s
Epoch: 0002 loss_train: 0.2371 acc_train: 0.9714 loss_val: 0.6135 acc_val: 0.8233 time: 0.0237s
Epoch: 0003 loss_train: 0.2289 acc_train: 0.9857 loss_val: 0.6158 acc_val: 0.8200 time: 0.0281s
Epoch: 0004 loss_train: 0.2389 acc_train: 0.9429 loss_val: 0.6186 acc_val: 0.8167 time: 0.0235s
Epoch: 0005 loss_train: 0.2624 acc_train: 0.9643 loss_val: 0.6208 acc_val: 0.8167 time: 0.0261s
Epoch: 0006 loss_train: 0.2272 acc_train: 0.9857 loss_val: 0.6230 acc_val: 0.8133 time: 0.0216s
Epoch: 0007 loss_train: 0.2150 acc_train: 0.9714 loss_val: 0.6269 acc_val: 0.8100 time: 0.0200s
Epoch: 0008 loss_train: 0.2527 acc_train: 0.9643 loss_val: 0.6264 acc_val: 0.8133 time: 0.0243s
Epoch: 0009 loss_train: 0.2249 acc_train: 0.9643 loss_val: 0.6268 acc_val: 0.8067 time: 0.0284s
Epoch: 0010 loss_train: 0.2144 acc_train: 0.9643 loss_val: 0.6274 acc_val: 0.8067 time: 0.0222s
Epoch: 0011 loss_train: 0.2838 acc_train: 0.9429 loss_val: 0.6271 acc_val: 0.8067 time: 0.0232s
Epoch: 0012 loss_train: 0.2459 acc_train: 0.9429 loss_val: 0.6238 acc_val: 0.8033 time: 0.0278s
```

Accuracy with 240 Labelled Nodes:

```
Epoch: 0193 loss_train: 0.2097 acc_train: 0.9714 loss_val: 0.6096 acc_val: 0.8167 time: 0.0271s
Epoch: 0194 loss_train: 0.2564 acc_train: 0.9643 loss_val: 0.6090 acc_val: 0.8200 time: 0.0255s
Epoch: 0195 loss_train: 0.2152 acc_train: 0.9786 loss_val: 0.6083 acc_val: 0.8200 time: 0.0221s
Epoch: 0196 loss_train: 0.2145 acc_train: 0.9786 loss_val: 0.6087 acc_val: 0.8233 time: 0.0190s
Epoch: 0197 loss_train: 0.2051 acc_train: 0.9786 loss_val: 0.6086 acc_val: 0.8233 time: 0.0191s
Epoch: 0198 loss_train: 0.2322 acc_train: 0.9786 loss_val: 0.6088 acc_val: 0.8100 time: 0.0250s
Epoch: 0199 loss_train: 0.2052 acc_train: 0.9786 loss_val: 0.6089 acc_val: 0.8100 time: 0.0230s
Epoch: 0200 loss_train: 0.1883 acc_train: 0.9857 loss_val: 0.6106 acc_val: 0.8133 time: 0.0234s
Optimization Finished!
Test set results: loss= 0.5978 accuracy= 0.8330
Accuracy with 240 labeled nodes: 0.8330
Epoch: 0001 loss_train: 0.2359 acc_train: 0.9643 loss_val: 0.6134 acc_val: 0.8200 time: 0.0270s
Epoch: 0002 loss_train: 0.2128 acc_train: 0.9857 loss_val: 0.6166 acc_val: 0.8200 time: 0.0220s
Epoch: 0003 loss_train: 0.2290 acc_train: 0.9571 loss_val: 0.6183 acc_val: 0.8233 time: 0.0190s
Epoch: 0004 loss_train: 0.1954 acc_train: 0.9786 loss_val: 0.6176 acc_val: 0.8233 time: 0.0210s
Epoch: 0005 loss_train: 0.2331 acc_train: 0.9643 loss_val: 0.6157 acc_val: 0.8200 time: 0.0202s
Epoch: 0006 loss_train: 0.2334 acc_train: 0.9857 loss_val: 0.6117 acc_val: 0.8133 time: 0.0240s
Epoch: 0007 loss_train: 0.2408 acc_train: 0.9857 loss_val: 0.6083 acc_val: 0.8167 time: 0.0220s
Epoch: 0008 loss_train: 0.2347 acc_train: 0.9714 loss_val: 0.6085 acc_val: 0.8133 time: 0.0760s
Epoch: 0009 loss_train: 0.2420 acc_train: 0.9571 loss_val: 0.6099 acc_val: 0.8200 time: 0.0320s
```

Accuracy with 300 Labelled Nodes:

```
Epoch: 0188 loss_train: 0.2341 acc_train: 0.9714 loss_val: 0.6074 acc_val: 0.8133 time: 0.0200s
Epoch: 0189 loss_train: 0.2179 acc_train: 0.9786 loss_val: 0.6078 acc_val: 0.8167 time: 0.0220s
Epoch: 0190 loss_train: 0.2257 acc_train: 0.9857 loss_val: 0.6093 acc_val: 0.8133 time: 0.0221s
Epoch: 0191 loss_train: 0.2213 acc_train: 0.9714 loss_val: 0.6110 acc_val: 0.8067 time: 0.0241s
Epoch: 0192 loss_train: 0.1672 acc_train: 0.9786 loss_val: 0.6132 acc_val: 0.8067 time: 0.0200s
Epoch: 0193 loss_train: 0.2242 acc_train: 0.9500 loss_val: 0.6166 acc_val: 0.8100 time: 0.0210s
Epoch: 0194 loss_train: 0.2250 acc_train: 0.9714 loss_val: 0.6213 acc_val: 0.8067 time: 0.0210s
Epoch: 0195 loss_train: 0.2315 acc_train: 0.9857 loss_val: 0.6246 acc_val: 0.8100 time: 0.0225s
Epoch: 0196 loss_train: 0.2194 acc_train: 0.9786 loss_val: 0.6253 acc_val: 0.8100 time: 0.0220s
Epoch: 0197 loss_train: 0.2107 acc_train: 0.9786 loss_val: 0.6218 acc_val: 0.8167 time: 0.0242s
Epoch: 0198 loss_train: 0.2088 acc_train: 0.9857 loss_val: 0.6200 acc_val: 0.8167 time: 0.0223s
Epoch: 0199 loss_train: 0.2110 acc_train: 0.9857 loss_val: 0.6160 acc_val: 0.8133 time: 0.0227s
Epoch: 0200 loss_train: 0.2348 acc_train: 0.9714 loss_val: 0.6119 acc_val: 0.8100 time: 0.0210s
Optimization Finished!
Test set results: loss= 0.5942 accuracy= 0.8330
Accuracy with 300 labeled nodes: 0.8330

C:\Users\Abhiram\Downloads\pygcn\pygcn>
```