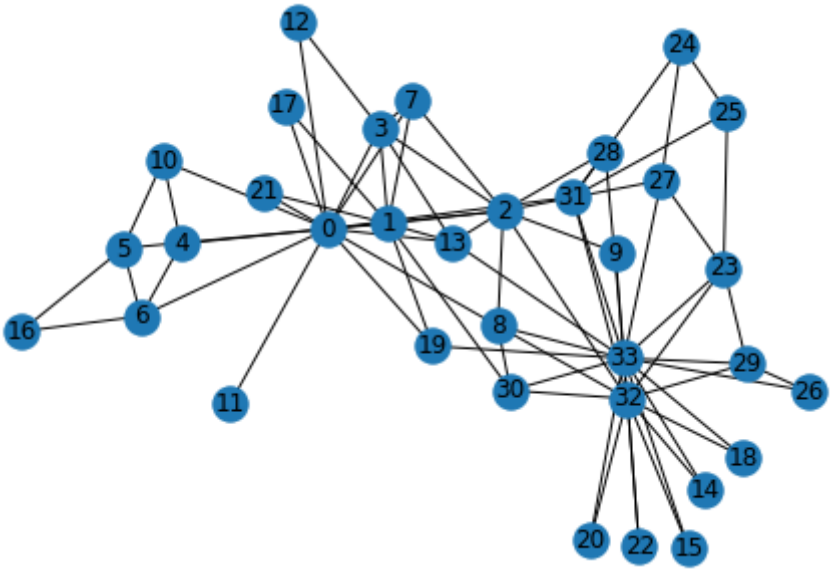
1 Graph Basics

In [1]:
```
import networkx as nx
```

In [2]:
```
G = nx.karate_club_graph()
type(G)
```

Out[2]:
```
networkx.classes.graph.Graph
```

In [3]:
```
# Visualize the graph
nx.draw(G, with_labels = True)
```



Question 1: What is the average degree of the karate club network?

In [24]:
```
def average_degree(num_edges, num_nodes):
    temp = (num_edges*2)/num_nodes
    avg_degree = round(temp,0)
    return avg_degree
num_edges = G.number_of_edges()
num_nodes = G.number_of_nodes()
avg_degree = average_degree(num_edges, num_nodes)
print("Average degree of karate club network is {}".format(avg_degree))
```

```
Average degree of karate club network is 5.0
```

Question 2: What is the average clustering coficient of the karate club network?

In [25]:
```
def average_clustering_coefficient(G):
    temp = nx.clustering(G)
    k = 0
    for key in temp.values():
        k = k + key
```

```
    avg_cluster_coef = round(k/33,2)
    return avg_cluster_coef
avg_cluster_coef = average_clustering_coefficient(G)
print("Average clustering coefficient of karate club network is {}".format(avg_cluster
```

```
Average clustering coefficient of karate club network is 0.59
```

Question 3: What is the PageRank value for node 0 (node with id 0) after one PageRank iteration?

In [38]:

```
def one_iter_pagerank(G, beta, r0, node_id):
    r1 = 0
    for ni in nx.neighbors(G,0):
        di = G.degree[ni]  #获取node_ni的度数
        r1 += (beta*r0)/di
    r1 +=(1-beta)*(r0)
    r1=round(r1,2)
    return r1

beta = 0.8
r0 = 1 / G.number_of_nodes()
node = 0
r1 = one_iter_pagerank(G, beta, r0, node)
print("The PageRank value for node 0 after one iteration is {}".format(r1))
```

```
The PageRank value for node 0 after one iteration is 0.13
```

Question 4: What is the (raw) closeness centrality for the karate club network node 5?

In [43]:

```
def closeness_centrality(G, node=5):
    closeness = nx.closeness_centrality(G,node)
    closeness = round(closeness/(G.number_of_nodes()-1),2)
    return closeness
node = 5
closeness = closeness_centrality(G, node=node)
print("The node 5 has closeness centrality {}".format(closeness))
```

```
The node 5 has closeness centrality 0.01
```

2 Graph to Tensor

In [45]:

```
import torch
print(torch.__version__)
```

```
1.10.0+cpu
```

PyTorch tensor basics

In [47]:

```
# Generate 3 x 4 tensor with all ones
ones = torch.ones(3, 4)
print(ones)

# Generate 3 x 4 tensor with all zeros
zeros = torch.zeros(3, 4)
print(zeros)
```

```
# Generate 3 x 4 tensor with random values on the interval [0, 1)
random_tensor = torch.rand(3, 4)
print(random_tensor)

# Get the shape of the tensor
print(ones.shape)
```

```
tensor([[1., 1., 1., 1.],
        [1., 1., 1., 1.],
        [1., 1., 1., 1.]])
tensor([[0., 0., 0., 0.],
        [0., 0., 0., 0.],
        [0., 0., 0., 0.]])
tensor([[0.7645, 0.2426, 0.2365, 0.4322],
        [0.2221, 0.6863, 0.0767, 0.6389],
        [0.4827, 0.7259, 0.5541, 0.5467]])
torch.Size([3, 4])
```

PyTorch tensor contains elements for a single data type, the dtype.

In [48]:

```
# Create a 3 x 4 tensor with all 32-bit floating point zeros
zeros = torch.zeros(3, 4, dtype=torch.float32)
print(zeros.dtype)

# Change the tensor dtype to 64-bit integer
zeros = zeros.type(torch.long)
print(zeros.dtype)
```

```
torch.float32
torch.int64
```

Question 5: Get the edge list of the karate club network and transform it into *torch.LongTensor* . What is the *torch.sum* value of *pos_edge_index* tensor?

In [57]:

```
# TODO: Implement the function that transforms the edge_list to
# tensor. The input edge_list is a list of tuples and the resulting
# tensor should have the shape [2 x len(edge_list)].

def graph_to_edge_list(G):
    edge_list = []
    for edge in G.edges():
        edge_list.append(edge)
    return edge_list

def edge_list_to_tensor(edge_list):
    edge_index=torch.LongTensor(edge_list).t()
    return edge_index

pos_edge_list = graph_to_edge_list(G)
pos_edge_index = edge_list_to_tensor(pos_edge_list)
print("The pos_edge_index tensor has shape {}".format(pos_edge_index.shape))
print("The pos_edge_index tensor has sum value {}".format(torch.sum(pos_edge_index)))
```

```
The pos_edge_index tensor has shape torch.Size([2, 78])
The pos_edge_index tensor has sum value 2535
```

Question 6: Please implement following function that

samples negative edges. Then answer which edges (edge_1 to edge_5) can be potential negative edges in the karate club network?

In [64]:
```
import random

# TODO: Implement the function that returns a list of negative edges.
# The number of sampled negative edges is num_neg_samples. You do not
# need to consider the corner case when the number of possible negative edges
# is less than num_neg_samples. It should be ok as long as your implementation
# works on the karate club network. In this implementation, self loops should
# not be considered as either a positive or negative edge. Also, notice that
# the karate club network is an undirected graph, if (0, 1) is a positive
# edge, do you think (1, 0) can be a negative one?

def sample_negative_edges(G, num_neg_samples):
    neg_edge_list = []
    non_edges_one_side = list(enumerate(nx.non_edges(G)))
    neg_edge_list_indices=random.sample(range(0,len(non_edges_one_side)),num_neg_samp
    for i in neg_edge_list_indices:
        neg_edge_list.append(non_edges_one_side[i][1])
    return neg_edge_list

# Sample 78 negative edges
neg_edge_list = sample_negative_edges(G, len(pos_edge_list))

# Transform the negative edge list to tensor
neg_edge_index = edge_list_to_tensor(neg_edge_list)
print("The neg_edge_index tensor has shape {}".format(neg_edge_index.shape))

# Which of following edges can be negative ones?
edge_1 = (7, 1)
edge_2 = (1, 33)
edge_3 = (33, 22)
edge_4 = (0, 4)
edge_5 = (4, 2)

print('edge_1'+(" can't" if G.has_edge(edge_1[0],edge_1[1]) else ' can')+' be negati
print('edge_2'+(" can't" if G.has_edge(edge_2[0],edge_2[1]) else ' can')+' be negati
print('edge_3'+(" can't" if G.has_edge(edge_3[0],edge_3[1]) else ' can')+' be negati
print('edge_4'+(" can't" if G.has_edge(edge_4[0],edge_4[1]) else ' can')+' be negati
print('edge_5'+(" can't" if G.has_edge(edge_5[0],edge_5[1]) else ' can')+' be negati
```

```
The neg_edge_index tensor has shape torch.Size([2, 78])
edge_1 can't be negative edge
edge_2 can be negative edge
edge_3 can't be negative edge
edge_4 can't be negative edge
edge_5 can be negative edge
```

3 Node Emebedding Learning

In [65]:
```
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
```

how to use *nn.Embedding*.

In [66]:
```
# Initialize an embedding layer
# Suppose we want to have embedding for 4 items (e.g., nodes)
# Each item is represented with 8 dimensional vector

emb_sample = nn.Embedding(num_embeddings=4, embedding_dim=8)
print('Sample embedding layer: {}'.format(emb_sample))
```

```
Sample embedding layer: Embedding(4, 8)
```

We can select items from the embedding matrix, by using Tensor indices

In [67]:
```
# Select an embedding in emb_sample
id = torch.LongTensor([1])
print(emb_sample(id))

# Select multiple embeddings
ids = torch.LongTensor([1, 3])
print(emb_sample(ids))

# Get the shape of the embedding weight matrix
shape = emb_sample.weight.data.shape
print(shape)

# Overwrite the weight to tensor with all ones
emb_sample.weight.data = torch.ones(shape)

# Let's check if the emb is indeed initilized
ids = torch.LongTensor([0, 3])
print(emb_sample(ids))
```

```
tensor([[ 0.7509,  1.5107, -0.6926, -0.6811, -2.3892, -0.7916, -1.1280, -0.7188]],
       grad_fn=<EmbeddingBackward0>)
tensor([[ 0.7509,  1.5107, -0.6926, -0.6811, -2.3892, -0.7916, -1.1280, -0.7188],
        [ 0.0562, -0.7202,  1.1536,  0.2445,  0.8610, -0.0050,  0.0500, -0.2087]],
       grad_fn=<EmbeddingBackward0>)
torch.Size([4, 8])
tensor([[1., 1., 1., 1., 1., 1., 1., 1.],
        [1., 1., 1., 1., 1., 1., 1., 1.]], grad_fn=<EmbeddingBackward0>)
```

Now, it's your time to create node embedding matrix for the graph we have! We want to have 16 dimensional vector for each node in the karate club network. We want to initalize the matrix under uniform distribution, in the range of . We suggest you using *torch.rand*.

In [69]:
```
# Please do not change / reset the random seed
torch.manual_seed(1)

# TODO: Implement this function that will create the node embedding matrix.
# A torch.nn.Embedding layer will be returned. You do not need to change
# the values of num_node and embedding_dim. The weight matrix of returned
# layer should be initialized under uniform distribution.
```

```
def create_node_emb(num_node=34, embedding_dim=16):
    emb = nn.Embedding(num_node, embedding_dim)
    shape = emb.weight.data.shape
    emb.weight.data = torch.rand(shape)
    return emb

emb = create_node_emb()
ids = torch.LongTensor([0, 3])

# Print the embedding layer
print("Embedding: {}".format(emb))

# An example that gets the embeddings for node 0 and 3
print(emb(ids))
```

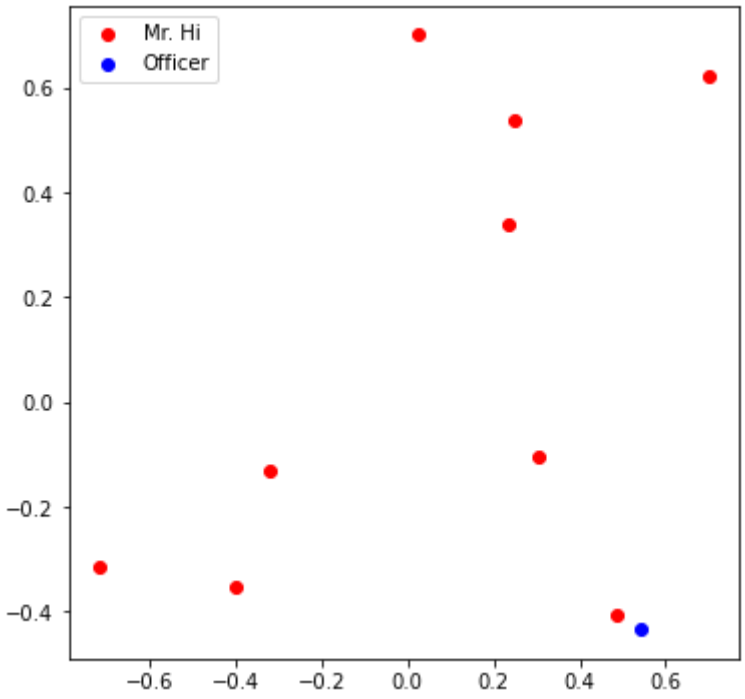
```
Embedding: Embedding(34, 16)
tensor([[0.2114, 0.7335, 0.1433, 0.9647, 0.2933, 0.7951, 0.5170, 0.2801, 0.8339,
         0.1185, 0.2355, 0.5599, 0.8966, 0.2858, 0.1955, 0.1808],
        [0.7486, 0.6546, 0.3843, 0.9820, 0.6012, 0.3710, 0.4929, 0.9915, 0.8358,
         0.4629, 0.9902, 0.7196, 0.2338, 0.0450, 0.7906, 0.9689]],
       grad_fn=<EmbeddingBackward0>)
```

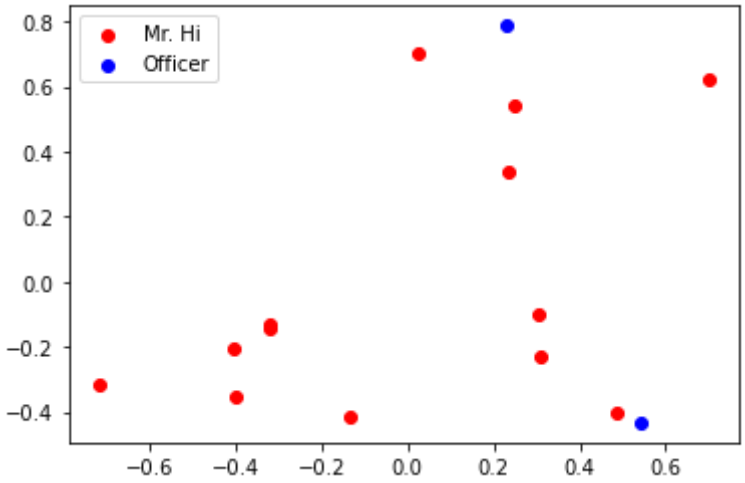
Visualize the initial node embeddings

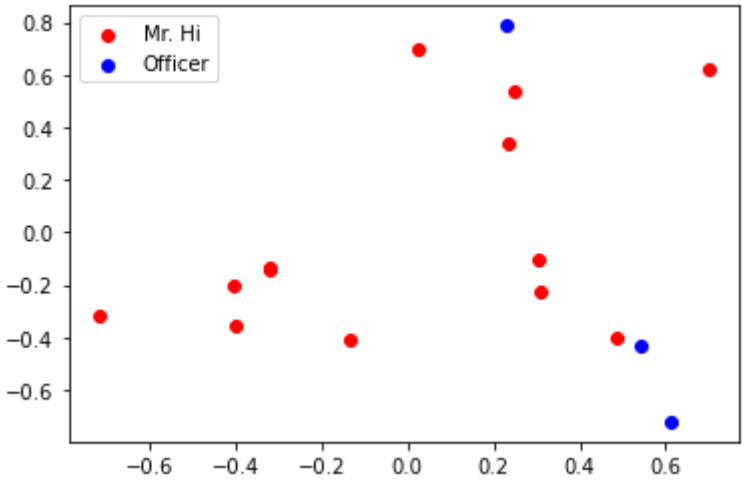
One good way to understand an embedding matrix, is to visualize it in a 2D space. Here, we have implemented an embedding visualization function for you. We ¦rst do PCA to reduce the dimensionality of embeddings to a 2D space. Then we visualize each point, colored by the community it belongs to.

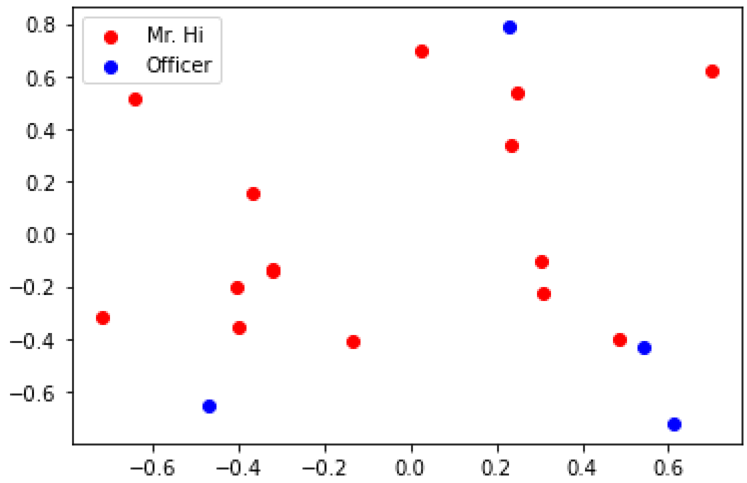
In [71]:

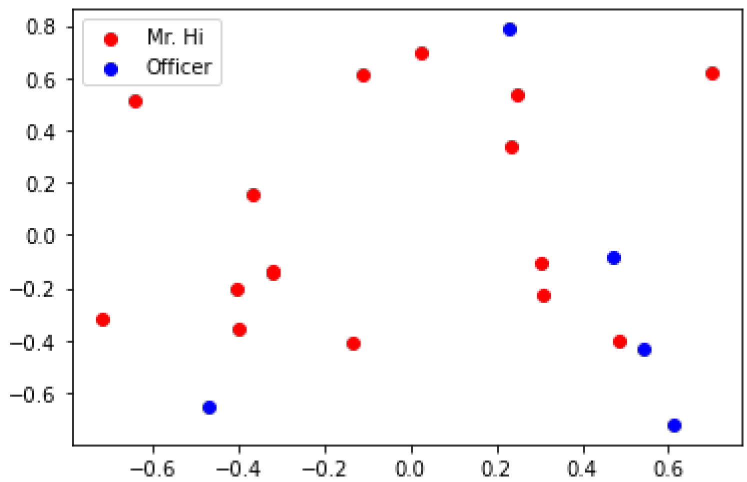
```
def visualize_emb(emb):
    X = emb.weight.data.numpy()
    pca = PCA(n_components=2)
    components = pca.fit_transform(X)
    plt.figure(figsize=(6, 6))
    club1_x = []
    club1_y = []
    club2_x = []
    club2_y = []
    for node in G.nodes(data=True):
        if node[1]['club'] == 'Mr. Hi':
            club1_x.append(components[node[0]][0])
            club1_y.append(components[node[0]][1])
        else:
            club2_x.append(components[node[0]][0])
            club2_y.append(components[node[0]][1])
            plt.scatter(club1_x, club1_y, color="red", label="Mr. Hi")
            plt.scatter(club2_x, club2_y, color="blue", label="Officer")
            plt.legend()
            plt.show()

# Visualize the initial random embeddding
visualize_emb(emb)
```

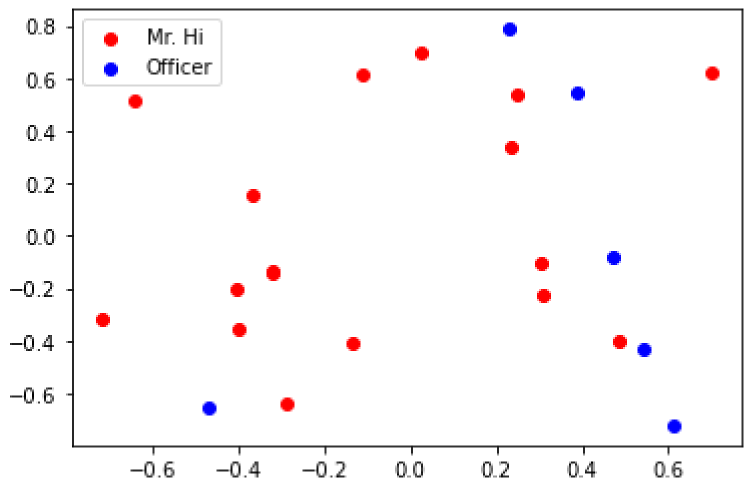

Mr. Hi
Officer
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

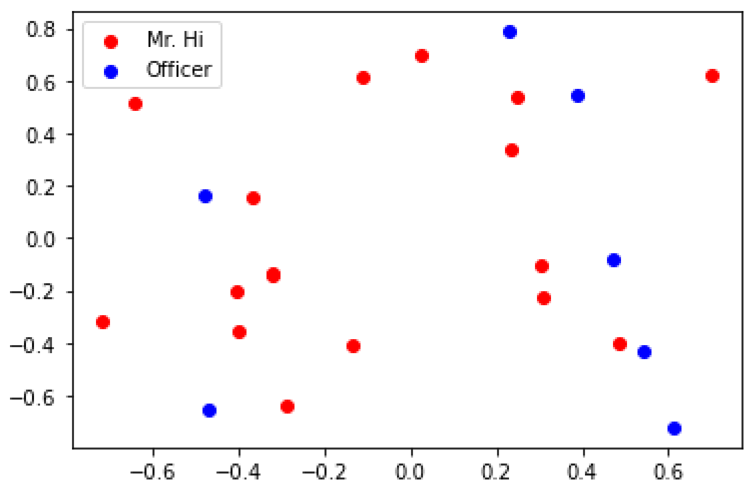

Mr. Hi
Officer
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

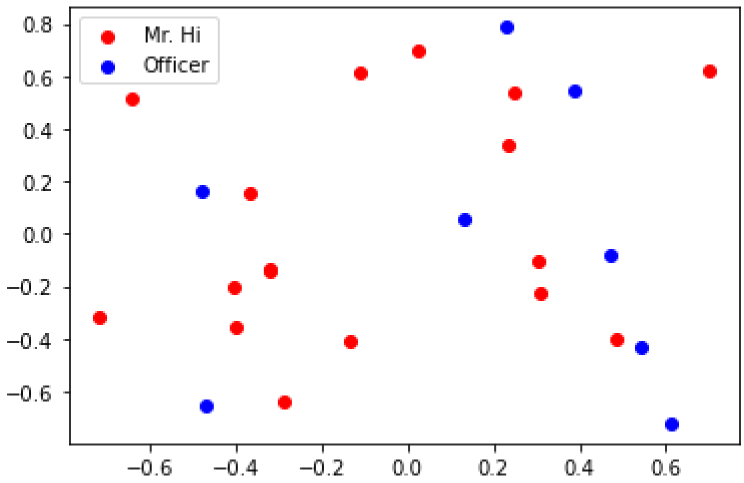

Mr. Hi
Officer
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

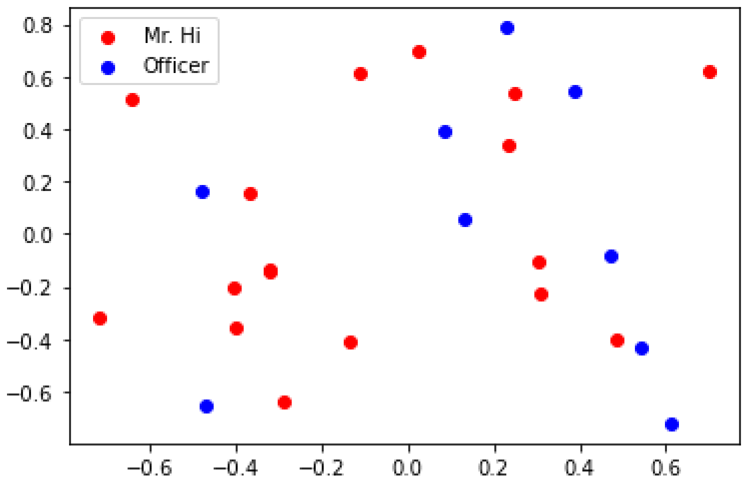

Mr. Hi
Officer

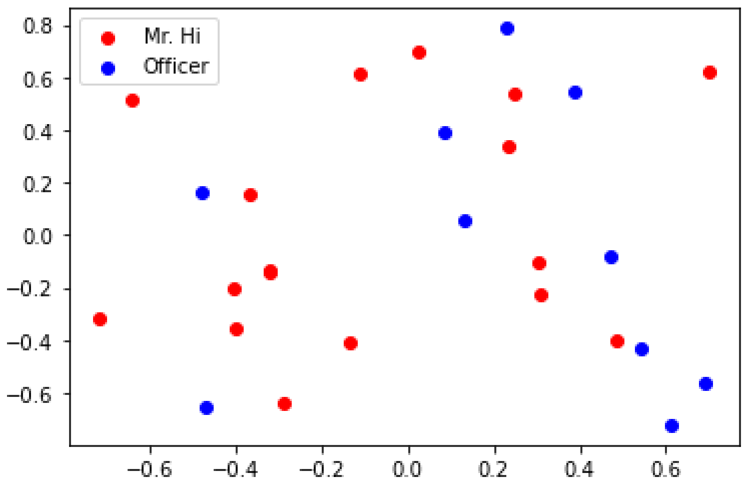

Mr. Hi
Officer

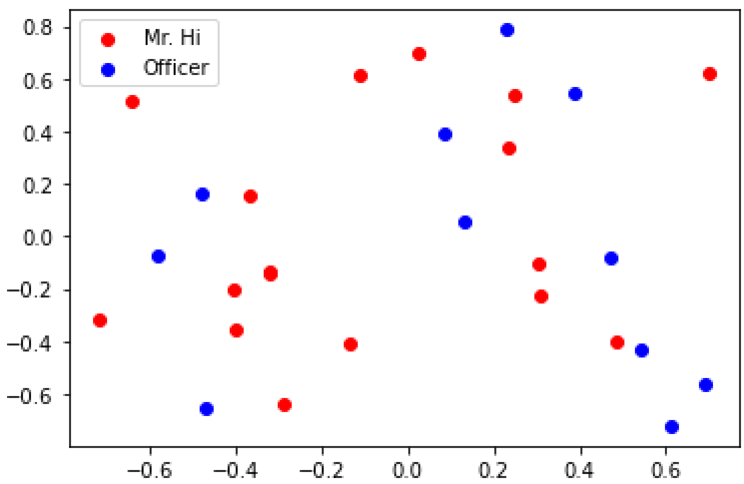

Mr. Hi
Officer

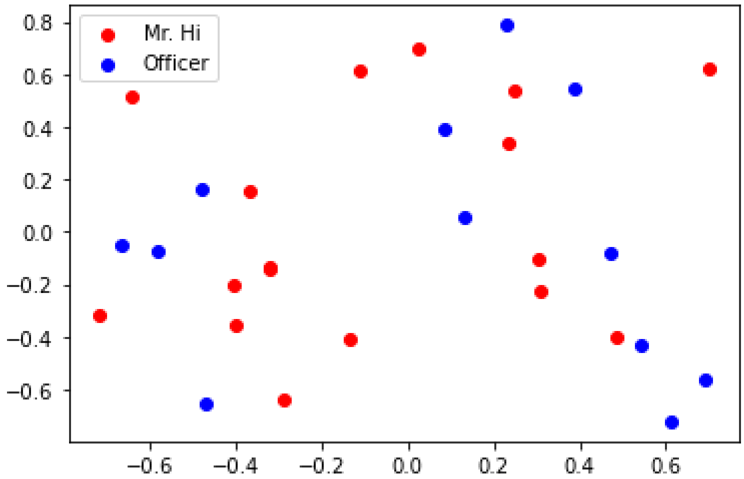

Mr. Hi
Officer

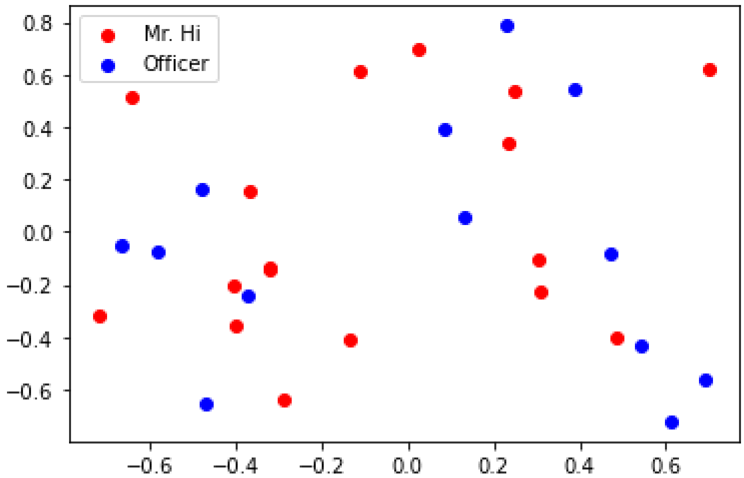

Mr. Hi
Officer

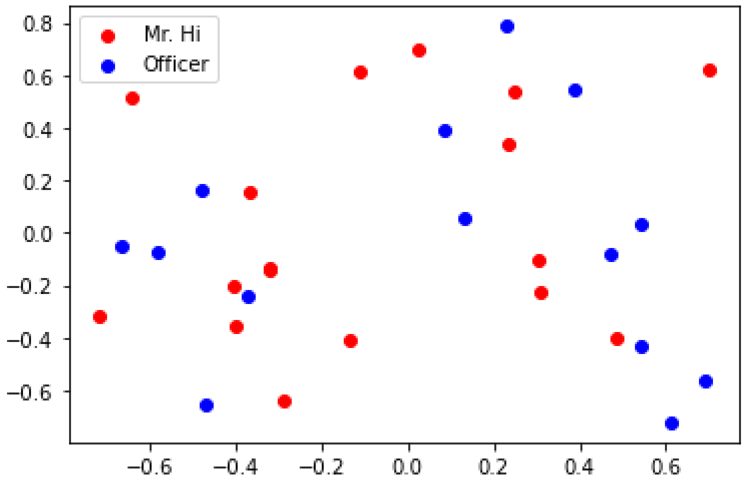

Mr. Hi
Officer

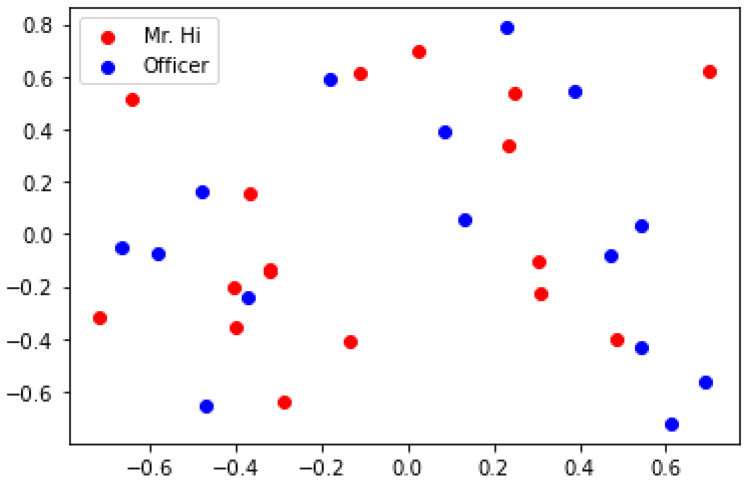

Mr. Hi
Officer

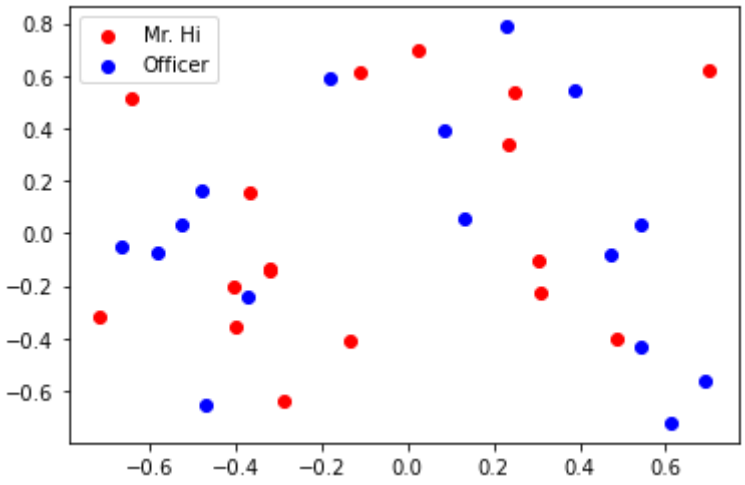

Mr. Hi
Officer

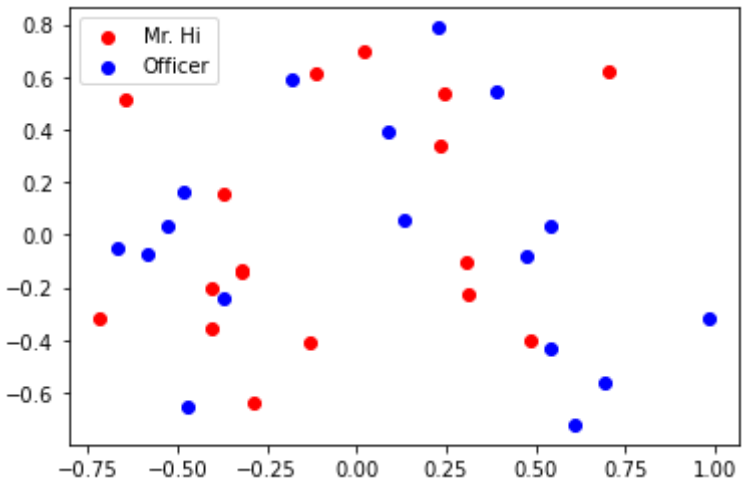

Mr. Hi
Officer
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


Mr. Hi
Officer
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


Mr. Hi
Officer
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


Mr. Hi
Officer
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6





Question 7: Training the embedding! What is the best performance you can get? Please report both the best loss and accuracy on Gradescope.

In [72]:

```
from torch.optim import SGD
import torch.nn as nn

# TODO: Implement the accuracy function. This function takes the
# pred tensor (the resulting tensor after sigmoid) and the label
# tensor (torch.LongTensor). Predicted value greater than 0.5 will
# be classified as label 1. Else it will be classified as label 0.
# The returned accuracy should be rounded to 4 decimal places.
# For example, accuracy 0.82956 will be rounded to 0.8296.

def accuracy(pred, label):
    accu=round(((pred>0.5)==label).sum().item()/(pred.shape[0]),4)
    return accu

# TODO: Train the embedding layer here. You can also change epochs and
# learning rate. In general, you need to implement:
# (1) Get the embeddings of the nodes in train_edge
# (2) Dot product the embeddings between each node pair
# (3) Feed the dot product result into sigmoid
# (4) Feed the sigmoid output into the loss_fn
# (5) Print both loss and accuracy of each epoch
# (6) Update the embeddings using the loss and optimizer
# (as a sanity check, the loss should decrease during training)
```

```python
def train(emb, loss_fn, sigmoid, train_label, train_edge):
    epochs = 500
    learning_rate = 0.1
    optimizer = SGD(emb.parameters(), lr=learning_rate, momentum=0.9)
    for i in range(epochs):
        # Prevent weights from adding
        optimizer.zero_grad()
        train_node_emb=emb(train_edge)
        dot_product_result=train_node_emb[0].mul(train_node_emb[1])
        dot_product_result=torch.sum(dot_product_result,1)
        sigmoid_result=sigmoid(dot_product_result)
        loss_result=loss_fn(sigmoid_result,train_label)
        loss_result.backward()
        optimizer.step()

        print(loss_result)
        print(accuracy(sigmoid_result,train_label))


loss_fn = nn.BCELoss()
sigmoid = nn.Sigmoid()
print(pos_edge_index.shape)

# Generate the positive and negative labels
pos_label = torch.ones(pos_edge_index.shape[1], )
neg_label = torch.zeros(neg_edge_index.shape[1], )

# Concat positive and negative labels into one tensor
train_label = torch.cat([pos_label, neg_label], dim=0)

# Concat positive and negative edges into one tensor
# Since the network is very small, we do not split the edges into val/test sets
train_edge = torch.cat([pos_edge_index, neg_edge_index], dim=1)
print(train_edge.shape)
train(emb, loss_fn, sigmoid, train_label, train_edge)
```

```
torch.Size([2, 78])
torch.Size([2, 156])
tensor(2.0445, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(2.0304, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(2.0037, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.9662, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.9195, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.8652, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.8046, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.7394, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.6707, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.5999, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.5279, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.4560, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.3848, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.3153, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
```

tensor(1.2480, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.1834, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.1221, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.0644, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(1.0103, grad_fn=<BinaryCrossEntropyBackward0>)
0.5
tensor(0.9601, grad_fn=<BinaryCrossEntropyBackward0>)
0.5064
tensor(0.9137, grad_fn=<BinaryCrossEntropyBackward0>)
0.5064
tensor(0.8712, grad_fn=<BinaryCrossEntropyBackward0>)
0.5064
tensor(0.8323, grad_fn=<BinaryCrossEntropyBackward0>)
0.5128
tensor(0.7969, grad_fn=<BinaryCrossEntropyBackward0>)
0.5192
tensor(0.7648, grad_fn=<BinaryCrossEntropyBackward0>)
0.5192
tensor(0.7358, grad_fn=<BinaryCrossEntropyBackward0>)
0.5321
tensor(0.7097, grad_fn=<BinaryCrossEntropyBackward0>)
0.5449
tensor(0.6861, grad_fn=<BinaryCrossEntropyBackward0>)
0.5513
tensor(0.6650, grad_fn=<BinaryCrossEntropyBackward0>)
0.5705
tensor(0.6460, grad_fn=<BinaryCrossEntropyBackward0>)
0.5962
tensor(0.6289, grad_fn=<BinaryCrossEntropyBackward0>)
0.5962
tensor(0.6135, grad_fn=<BinaryCrossEntropyBackward0>)
0.6154
tensor(0.5996, grad_fn=<BinaryCrossEntropyBackward0>)
0.6282
tensor(0.5871, grad_fn=<BinaryCrossEntropyBackward0>)
0.6538
tensor(0.5757, grad_fn=<BinaryCrossEntropyBackward0>)
0.6731
tensor(0.5654, grad_fn=<BinaryCrossEntropyBackward0>)
0.6859
tensor(0.5560, grad_fn=<BinaryCrossEntropyBackward0>)
0.6987
tensor(0.5475, grad_fn=<BinaryCrossEntropyBackward0>)
0.6987
tensor(0.5396, grad_fn=<BinaryCrossEntropyBackward0>)
0.7372
tensor(0.5324, grad_fn=<BinaryCrossEntropyBackward0>)
0.7692
tensor(0.5257, grad_fn=<BinaryCrossEntropyBackward0>)
0.7756
tensor(0.5195, grad_fn=<BinaryCrossEntropyBackward0>)
0.7821
tensor(0.5136, grad_fn=<BinaryCrossEntropyBackward0>)
0.7821
tensor(0.5082, grad_fn=<BinaryCrossEntropyBackward0>)
0.7821
tensor(0.5030, grad_fn=<BinaryCrossEntropyBackward0>)
0.7821
tensor(0.4981, grad_fn=<BinaryCrossEntropyBackward0>)
0.7756
tensor(0.4935, grad_fn=<BinaryCrossEntropyBackward0>)
0.7885
tensor(0.4891, grad_fn=<BinaryCrossEntropyBackward0>)
0.7885
tensor(0.4848, grad_fn=<BinaryCrossEntropyBackward0>)

0.8077
tensor(0.4807, grad_fn=<BinaryCrossEntropyBackward0>)
0.8141
tensor(0.4768, grad_fn=<BinaryCrossEntropyBackward0>)
0.8205
tensor(0.4729, grad_fn=<BinaryCrossEntropyBackward0>)
0.8205
tensor(0.4692, grad_fn=<BinaryCrossEntropyBackward0>)
0.8205
tensor(0.4655, grad_fn=<BinaryCrossEntropyBackward0>)
0.8205
tensor(0.4619, grad_fn=<BinaryCrossEntropyBackward0>)
0.8205
tensor(0.4584, grad_fn=<BinaryCrossEntropyBackward0>)
0.8205
tensor(0.4550, grad_fn=<BinaryCrossEntropyBackward0>)
0.8269
tensor(0.4516, grad_fn=<BinaryCrossEntropyBackward0>)
0.8333
tensor(0.4483, grad_fn=<BinaryCrossEntropyBackward0>)
0.8333
tensor(0.4449, grad_fn=<BinaryCrossEntropyBackward0>)
0.8333
tensor(0.4417, grad_fn=<BinaryCrossEntropyBackward0>)
0.8397
tensor(0.4384, grad_fn=<BinaryCrossEntropyBackward0>)
0.8397
tensor(0.4352, grad_fn=<BinaryCrossEntropyBackward0>)
0.8397
tensor(0.4320, grad_fn=<BinaryCrossEntropyBackward0>)
0.8397
tensor(0.4288, grad_fn=<BinaryCrossEntropyBackward0>)
0.8397
tensor(0.4257, grad_fn=<BinaryCrossEntropyBackward0>)
0.8462
tensor(0.4225, grad_fn=<BinaryCrossEntropyBackward0>)
0.8462
tensor(0.4194, grad_fn=<BinaryCrossEntropyBackward0>)
0.8526
tensor(0.4163, grad_fn=<BinaryCrossEntropyBackward0>)
0.859
tensor(0.4132, grad_fn=<BinaryCrossEntropyBackward0>)
0.859
tensor(0.4101, grad_fn=<BinaryCrossEntropyBackward0>)
0.859
tensor(0.4070, grad_fn=<BinaryCrossEntropyBackward0>)
0.859
tensor(0.4040, grad_fn=<BinaryCrossEntropyBackward0>)
0.859
tensor(0.4009, grad_fn=<BinaryCrossEntropyBackward0>)
0.8654
tensor(0.3979, grad_fn=<BinaryCrossEntropyBackward0>)
0.8718
tensor(0.3948, grad_fn=<BinaryCrossEntropyBackward0>)
0.8846
tensor(0.3918, grad_fn=<BinaryCrossEntropyBackward0>)
0.8846
tensor(0.3888, grad_fn=<BinaryCrossEntropyBackward0>)
0.8846
tensor(0.3857, grad_fn=<BinaryCrossEntropyBackward0>)
0.8846
tensor(0.3827, grad_fn=<BinaryCrossEntropyBackward0>)
0.8974
tensor(0.3797, grad_fn=<BinaryCrossEntropyBackward0>)
0.8974
tensor(0.3767, grad_fn=<BinaryCrossEntropyBackward0>)
0.8974
tensor(0.3737, grad_fn=<BinaryCrossEntropyBackward0>)
0.8974

tensor(0.3707, grad_fn=<BinaryCrossEntropyBackward0>)
0.9103
tensor(0.3678, grad_fn=<BinaryCrossEntropyBackward0>)
0.9103
tensor(0.3648, grad_fn=<BinaryCrossEntropyBackward0>)
0.9167
tensor(0.3619, grad_fn=<BinaryCrossEntropyBackward0>)
0.9167
tensor(0.3589, grad_fn=<BinaryCrossEntropyBackward0>)
0.9231
tensor(0.3560, grad_fn=<BinaryCrossEntropyBackward0>)
0.9231
tensor(0.3530, grad_fn=<BinaryCrossEntropyBackward0>)
0.9231
tensor(0.3501, grad_fn=<BinaryCrossEntropyBackward0>)
0.9231
tensor(0.3472, grad_fn=<BinaryCrossEntropyBackward0>)
0.9231
tensor(0.3443, grad_fn=<BinaryCrossEntropyBackward0>)
0.9359
tensor(0.3414, grad_fn=<BinaryCrossEntropyBackward0>)
0.9359
tensor(0.3385, grad_fn=<BinaryCrossEntropyBackward0>)
0.9359
tensor(0.3357, grad_fn=<BinaryCrossEntropyBackward0>)
0.9359
tensor(0.3328, grad_fn=<BinaryCrossEntropyBackward0>)
0.9487
tensor(0.3299, grad_fn=<BinaryCrossEntropyBackward0>)
0.9487
tensor(0.3271, grad_fn=<BinaryCrossEntropyBackward0>)
0.9551
tensor(0.3243, grad_fn=<BinaryCrossEntropyBackward0>)
0.9551
tensor(0.3215, grad_fn=<BinaryCrossEntropyBackward0>)
0.9551
tensor(0.3187, grad_fn=<BinaryCrossEntropyBackward0>)
0.9551
tensor(0.3159, grad_fn=<BinaryCrossEntropyBackward0>)
0.9551
tensor(0.3131, grad_fn=<BinaryCrossEntropyBackward0>)
0.9551
tensor(0.3103, grad_fn=<BinaryCrossEntropyBackward0>)
0.9615
tensor(0.3076, grad_fn=<BinaryCrossEntropyBackward0>)
0.9615
tensor(0.3049, grad_fn=<BinaryCrossEntropyBackward0>)
0.9679
tensor(0.3021, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2994, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2967, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2941, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2914, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2888, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2861, grad_fn=<BinaryCrossEntropyBackward0>)
0.9744
tensor(0.2835, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2809, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2783, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2758, grad_fn=<BinaryCrossEntropyBackward0>)

0.9808
tensor(0.2732, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2707, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2682, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2657, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2632, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2607, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2583, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2558, grad_fn=<BinaryCrossEntropyBackward0>)
0.9808
tensor(0.2534, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2510, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2486, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2463, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2439, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2416, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2393, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2370, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2348, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2325, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2303, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2280, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2258, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2237, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2215, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2194, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2172, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2151, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2130, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2110, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2089, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2069, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2049, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2029, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.2009, grad_fn=<BinaryCrossEntropyBackward0>)
0.9872
tensor(0.1989, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936

tensor(0.1970, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1951, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1931, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1913, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1894, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1875, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1857, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1839, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1821, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1803, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1785, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1767, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1750, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1733, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1716, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1699, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1682, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1666, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1650, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1633, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1617, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1601, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1586, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1570, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1555, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1540, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1525, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1510, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1495, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1480, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1466, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1452, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1438, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1424, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1410, grad_fn=<BinaryCrossEntropyBackward0>)

```
0.9936
tensor(0.1396, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1383, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1369, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1356, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1343, grad_fn=<BinaryCrossEntropyBackward0>)
0.9936
tensor(0.1330, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1317, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1305, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1292, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1280, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1267, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1255, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1243, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1232, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1220, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1208, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1197, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1186, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1174, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1163, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1152, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1142, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1131, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1120, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1110, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1100, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1089, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1079, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1069, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1060, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1050, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1040, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1031, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1021, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
```

```
tensor(0.1012, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.1003, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0994, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0985, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0976, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0967, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0959, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0950, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0942, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0933, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0925, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0917, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0909, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0901, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0893, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0885, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0878, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0870, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0863, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0855, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0848, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0841, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0833, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0826, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0819, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0812, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0806, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0799, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0792, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0786, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0779, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0773, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0766, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0760, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0754, grad_fn=<BinaryCrossEntropyBackward0>)
```

```
1.0
tensor(0.0748, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0742, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0736, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0730, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0724, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0718, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0712, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0707, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0701, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0696, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0690, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0685, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0679, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0674, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0669, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0664, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0659, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0654, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0649, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0644, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0639, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0634, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0629, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0625, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0620, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0615, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0611, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0606, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0602, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0597, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0593, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0589, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0585, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0580, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
```

```
tensor(0.0576, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0572, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0568, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0564, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0560, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0556, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0552, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0548, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0544, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0541, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0537, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0533, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0530, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0526, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0522, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0519, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0515, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0512, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0509, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0505, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0502, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0498, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0495, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0492, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0489, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0486, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0482, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0479, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0476, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0473, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0470, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0467, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0464, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0461, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0458, grad_fn=<BinaryCrossEntropyBackward0>)
```

```
1.0
tensor(0.0455, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0453, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0450, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0447, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0444, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0442, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0439, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0436, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0433, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0431, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0428, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0426, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0423, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0421, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0418, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0416, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0413, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0411, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0408, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0406, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0404, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0401, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0399, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0397, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0394, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0392, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0390, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0388, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0386, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0383, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0381, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0379, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0377, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0375, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
```

tensor(0.0373, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0371, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0369, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0367, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0365, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0363, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0361, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0359, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0357, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0355, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0353, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0351, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0349, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0348, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0346, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0344, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0342, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0340, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0339, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0337, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0335, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0333, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0332, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0330, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0328, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0327, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0325, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0323, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0322, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0320, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0319, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0317, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0315, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0314, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0312, grad_fn=<BinaryCrossEntropyBackward0>)

```
1.0
tensor(0.0311, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0309, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0308, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0306, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0305, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0303, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0302, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0300, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0299, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0298, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0296, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0295, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0293, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0292, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0291, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0289, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0288, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0287, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0285, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0284, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0283, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0281, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0280, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0279, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0278, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0276, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0275, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0274, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0273, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0272, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0270, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0269, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0268, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0267, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
```

tensor(0.0266, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0264, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0263, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0262, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0261, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0260, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0259, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0258, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0257, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0255, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0254, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0253, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0252, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0251, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0250, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0249, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0248, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0247, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0246, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0245, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0244, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0243, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0242, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0241, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0240, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0239, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0238, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0237, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0236, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0235, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0234, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0233, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0232, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0231, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0231, grad_fn=<BinaryCrossEntropyBackward0>)

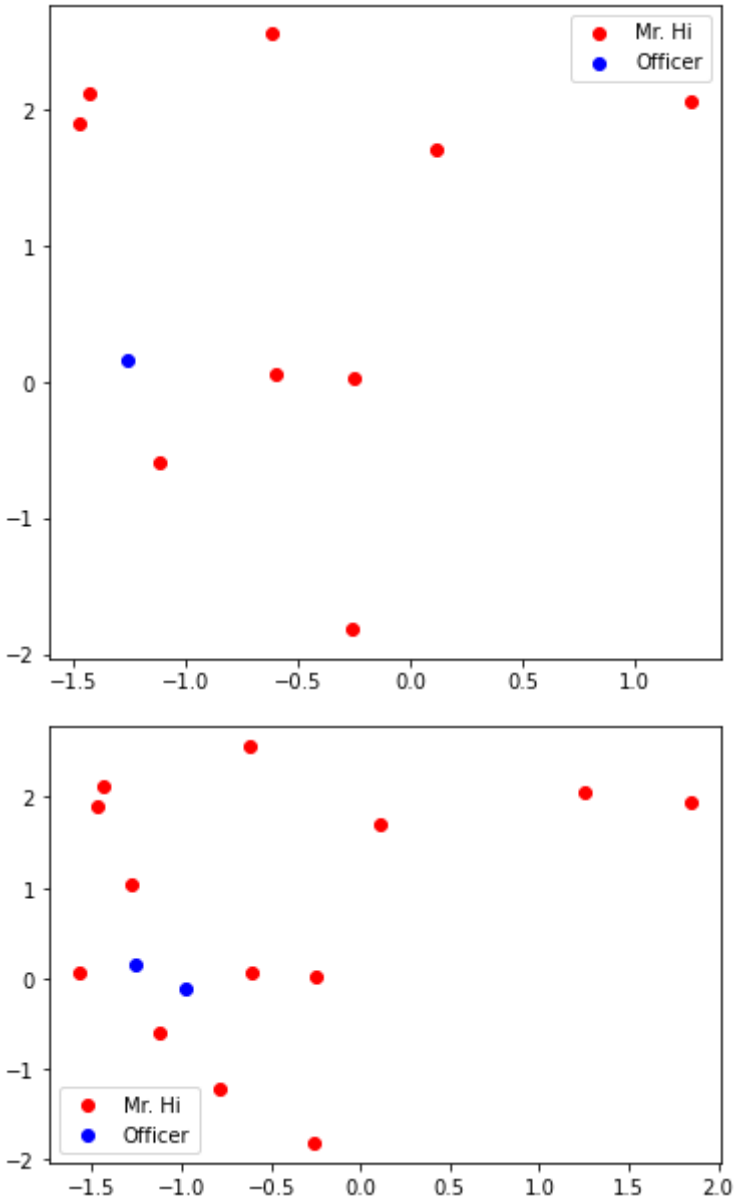
```
1.0
tensor(0.0230, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0229, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0228, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0227, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0226, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0225, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0224, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0223, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0223, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0222, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0221, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0220, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0219, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0218, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0218, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0217, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0216, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0215, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0214, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0214, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0213, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0212, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0211, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0210, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0210, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0209, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0208, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0207, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0207, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0206, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0205, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0204, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0204, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0203, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
```

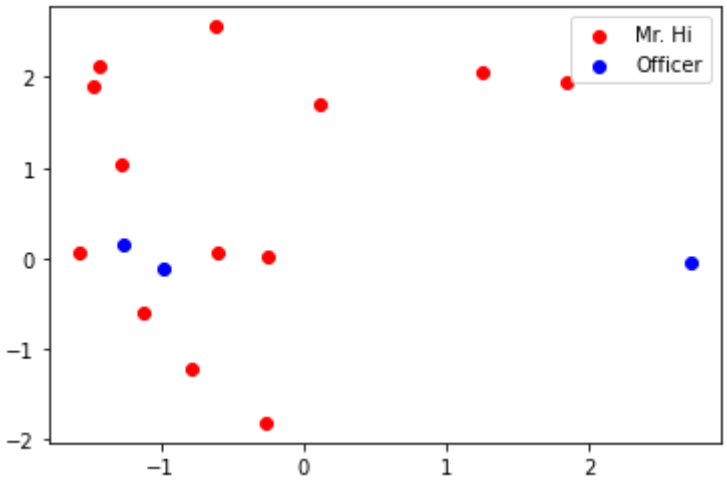
```
tensor(0.0202, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0201, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
tensor(0.0201, grad_fn=<BinaryCrossEntropyBackward0>)
1.0
```

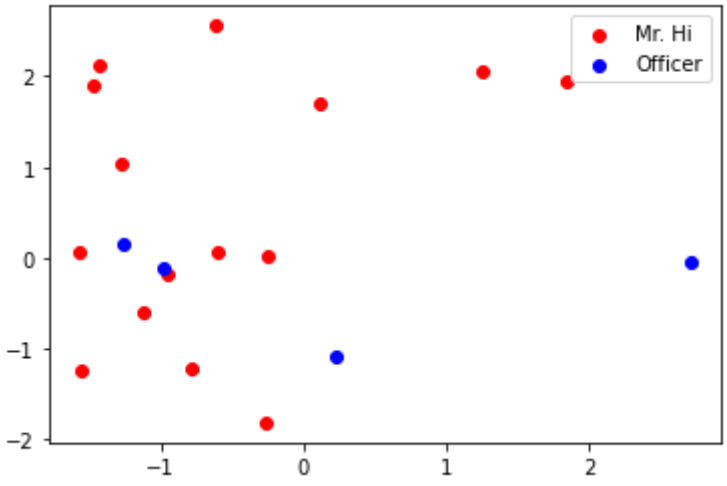
Visualize the final node embeddings

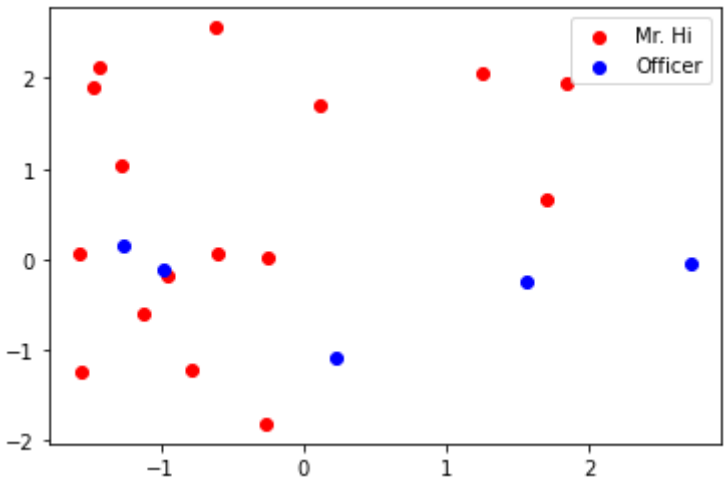
In [74]:

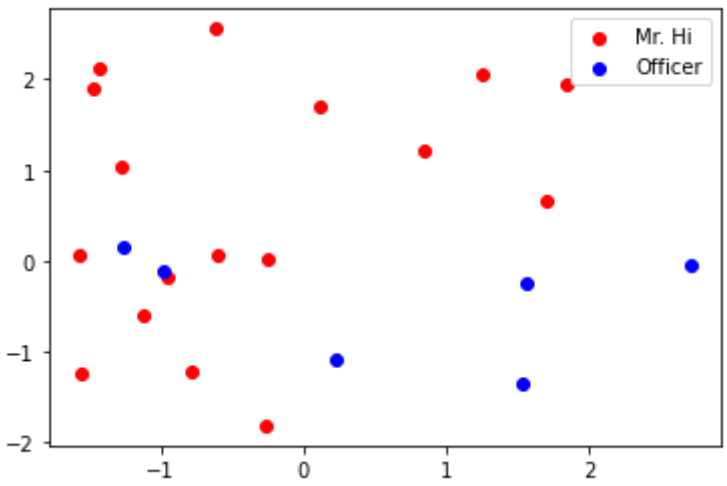
```
# Visualize the final learned embedding
visualize_emb(emb)
```

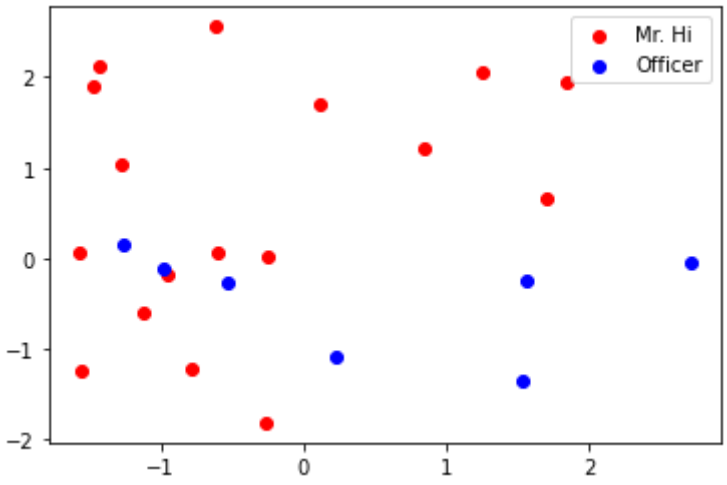


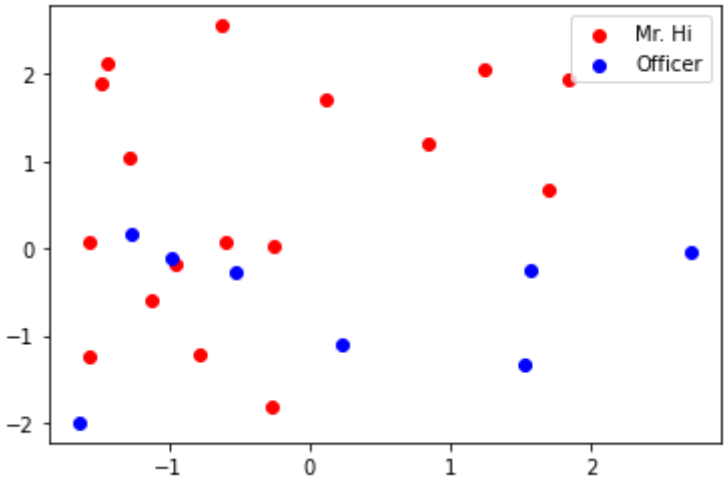

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

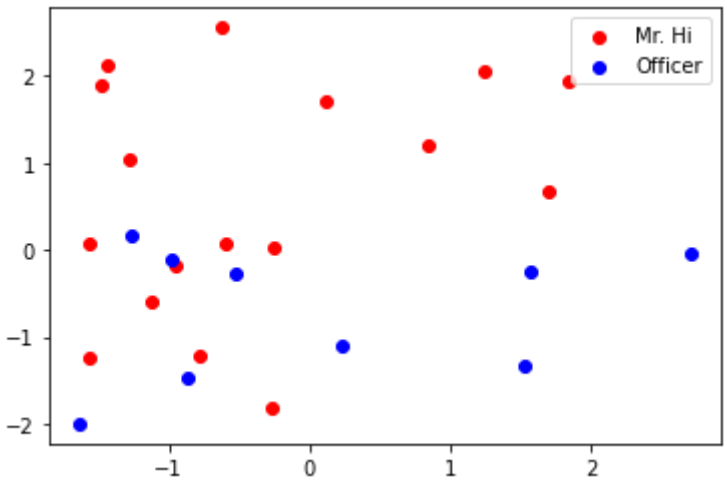

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

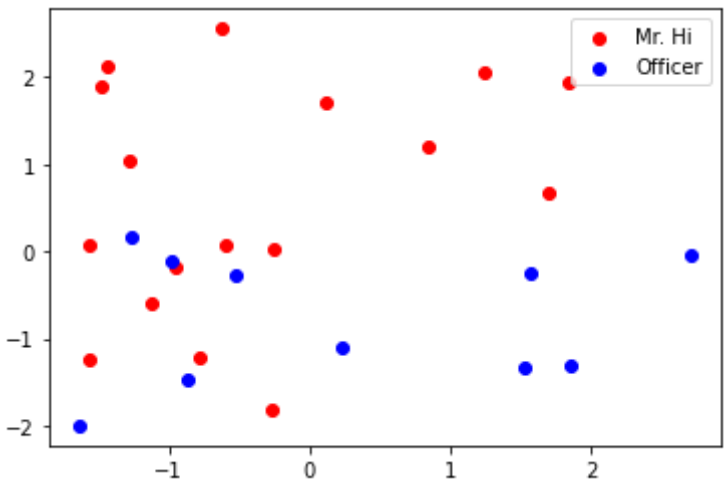

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

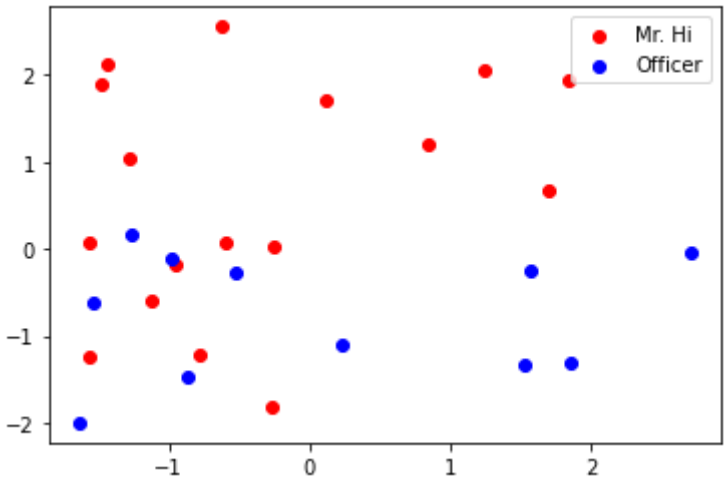

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

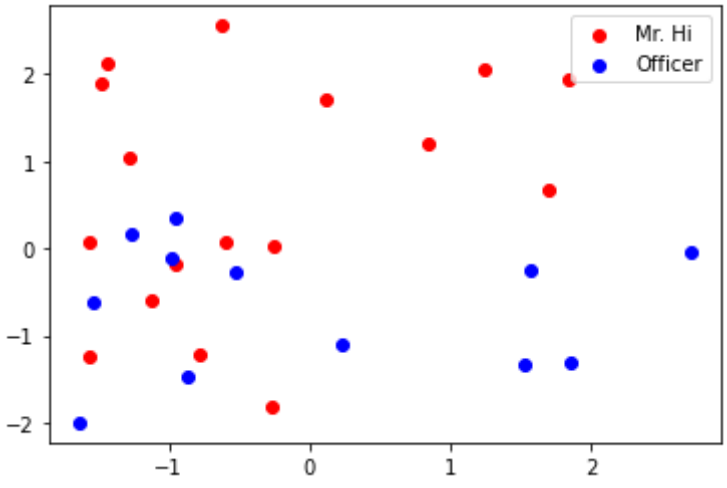

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

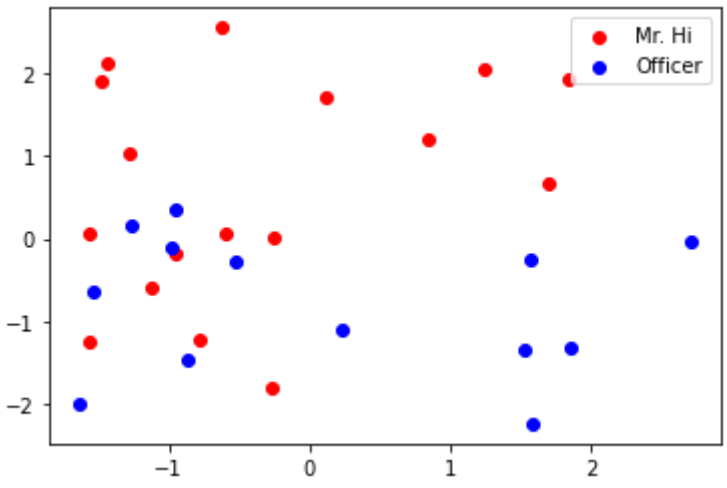

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

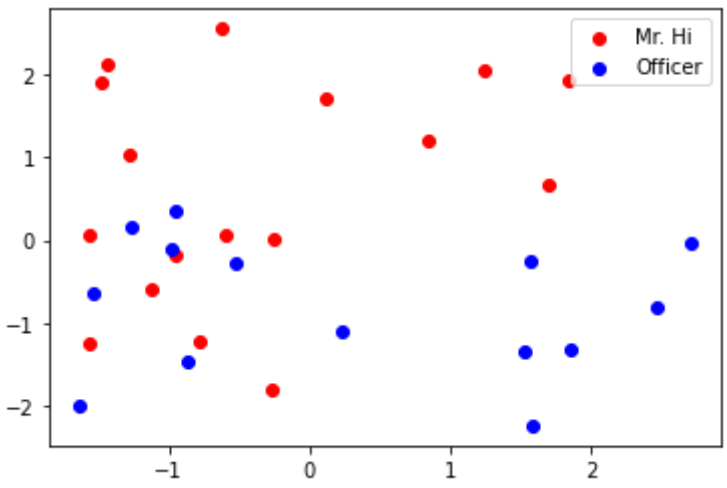

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

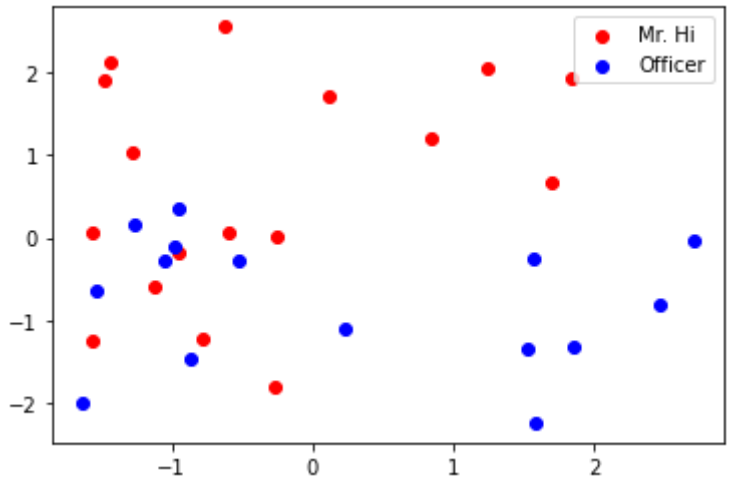

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

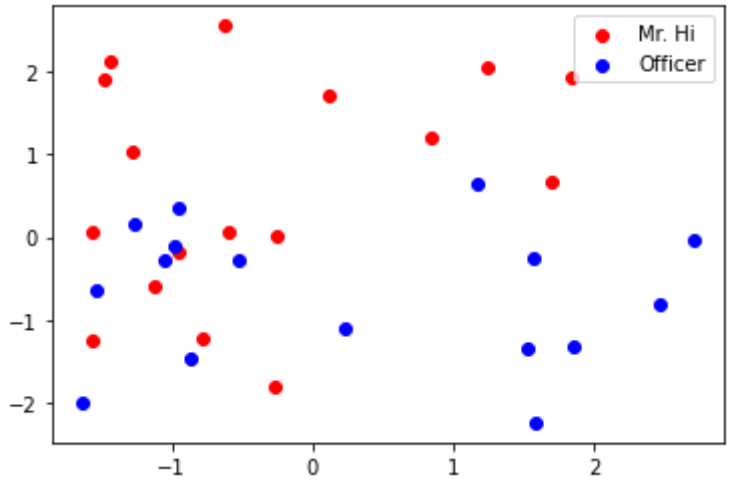

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2

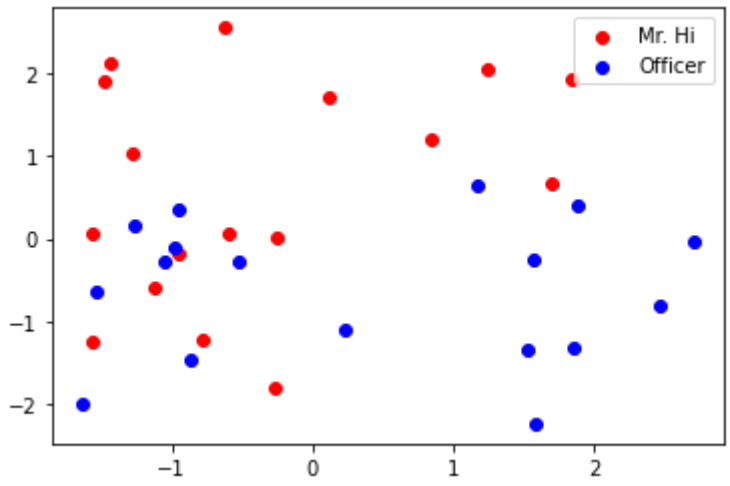

Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2


Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2


Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2


Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2


Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2


Mr. Hi
Officer
2
1
0
−1
−2
−1
0
1
2