Q2

November 2, 2023

[1]:

```python
import numpy as np
import sys
import pdb
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score, f1_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import OneHotEncoder
import random
```

[2]:

```python
def get_data(x_path, y_path):
    '''
    Args:
        x_path: path to x file
        y_path: path to y file
    Returns:
        x: np array of [NUM_OF_SAMPLES x n]
        y: np array of [NUM_OF_SAMPLES]
    '''
    x = np.load(x_path)
    y = np.load(y_path)

    y = y.astype('float')
    x = x.astype('float')

    #normalize x:
    x = 2*(0.5 - x/255)
    return x, y
```

[3]:

```python
def get_metric(y_true, y_pred):
    '''
    Args:
        y_true: np array of [NUM_SAMPLES x r] (one hot)
                or np array of [NUM_SAMPLES]
        y_pred: np array of [NUM_SAMPLES x r] (one hot)
                or np array of [NUM_SAMPLES]

    '''
    results = classification_report(y_pred, y_true)
    print(results)
```

Preparing the X_train, y_train, X_test, y_test

[4]:
```
x_train_path = "/home/tkarthikeyan/IIT Delhi/COL774-Machine Learning/Assignment␣
 ↪3/Data_b/x_train.npy"
y_train_path = "/home/tkarthikeyan/IIT Delhi/COL774-Machine Learning/Assignment␣
 ↪3/Data_b/y_train.npy"

X_train, y_train = get_data(x_train_path, y_train_path)

x_test_path = "/home/tkarthikeyan/IIT Delhi/COL774-Machine Learning/Assignment␣
 ↪3/Data_b/x_test.npy"
y_test_path = "/home/tkarthikeyan/IIT Delhi/COL774-Machine Learning/Assignment␣
 ↪3/Data_b/y_test.npy"

X_test, y_test = get_data(x_test_path, y_test_path)

#you might need one hot encoded y in part a,b,c,d,e
label_encoder = OneHotEncoder(sparse_output = False)
label_encoder.fit(np.expand_dims(y_train, axis = -1))

y_train_onehot = label_encoder.transform(np.expand_dims(y_train, axis = -1))
y_test_onehot = label_encoder.transform(np.expand_dims(y_test, axis = -1))
```

[5]:
```
class NeuralNetwork:
    def __init__(self, n, n_hidden_nodes, r, M):
        #Number of nodes in the architecture
        self.n = n
        self.n_hidden_nodes = n_hidden_nodes
        self.r = r

        #Mini batch size
        self.M = M

        #Weights and biases
        self.W = dict()
        self.b = dict()

    def initialize_weights_and_biases(self):
        n_nodes = [self.n] + self.n_hidden_nodes + [self.r]

        #Initialize weights
        for i in range(1,len(n_nodes)):
            self.W[str(i)] = np.random.uniform(low=-0.1, high=0.1,␣
 ↪size=(n_nodes[i], n_nodes[i-1]))
```

```
        #Initialize biases
        for i in range(1,len(n_nodes)):
            self.b[str(i)] = np.zeros((n_nodes[i],1))

    @staticmethod
    def sigmoid(x, derivative = False):
        if derivative == False:
            return 1 / (1 + np.exp(-x))
        else:
            return NeuralNetwork.sigmoid(x, derivative = False) * (1 - NeuralNetwork.sigmoid(x, derivative = False))

    @staticmethod
    def relu(x, derivative = False):
        if derivative == True:
            return np.where(x > 0, 1, np.where(x < 0, 0, np.random.random_sample()))
        else:
            return np.where(x <= 0, 0, x)

    @staticmethod
    def softmax(Z):
        return np.exp(Z) / np.sum(np.exp(Z), axis=0)

    def train(self, X_train, y_train, epoch_mode = True, activation="sigmoid", EPOCHS = 200, alpha = 0.01, stopping_threshold = None, adaptive_learning=False, printafter=20):
        self.initialize_weights_and_biases()

        a = dict()
        z = dict()
        del_z = dict()
        del_b = dict()
        del_W = dict()

        if epoch_mode == True:
            for epoch in range(EPOCHS):
                for i in range(0, X_train.shape[0], self.M):
                    y_actual = y_train[i:i+self.M,:].T

                    #Forward
                    a["0"] = X_train[i:i+self.M,:].T

                    for j in range(1,len(self.n_hidden_nodes)+1):
                        z[str(j)] = np.matmul(self.W[str(j)], a[str(j-1)]) + self.b[str(j)]
                        if activation == "relu":
                            a[str(j)] = NeuralNetwork.relu(z[str(j)])
                        else:
                            a[str(j)] = NeuralNetwork.sigmoid(z[str(j)])

                    j += 1
                    z[str(j)] = np.matmul(self.W[str(j)], a[str(j-1)]) + self.
    ↪b[str(j)]
                    a[str(j)] = NeuralNetwork.softmax(z[str(j)])

                    #Backward
                    del_z[str(j)] = a[str(j)] - y_actual
                    del_b[str(j)] = np.sum(del_z[str(j)], axis = 1).
    ↪reshape(-1,1)
                    del_W[str(j)] = np.matmul(del_z[str(j)], a[str(j-1)].T)

                    for k in range(j-1,0,-1):
                        if activation == "relu":
                            del_z[str(k)] = np.matmul(self.W[str(k+1)].T,␣
    ↪del_z[str(k+1)])*(NeuralNetwork.relu(z[str(k)], derivative=True))
                        else:
                            del_z[str(k)] = np.matmul(self.W[str(k+1)].T,␣
    ↪del_z[str(k+1)])*(NeuralNetwork.sigmoid(z[str(k)], derivative=True))
                        del_b[str(k)] = np.sum(del_z[str(k)], axis = 1).
    ↪reshape(-1,1)
                        del_W[str(k)] = np.matmul(del_z[str(k)], a[str(k-1)].T)

                    #Update
                    for l in range(1,len(self.n_hidden_nodes)+2):
                        if adaptive_learning == False:
                            self.W[str(l)] = self.W[str(l)] - alpha *␣
    ↪del_W[str(l)]
                            self.b[str(l)] = self.b[str(l)] - alpha *␣
    ↪del_b[str(l)]
                        else:
                            self.W[str(l)] = self.W[str(l)] - (alpha/np.
    ↪sqrt(epoch)) * del_W[str(l)]
                            self.b[str(l)] = self.b[str(l)] - (alpha/np.
    ↪sqrt(epoch)) * del_b[str(l)]

                y_pred, softmax_output = NN.predict(X_train,␣
    ↪activation=activation)
                softmax_loss = NeuralNetwork.
    ↪compute_softmax_loss(softmax_output, y_train_onehot)
                if epoch%printafter==0:
                    print(f"epoch {epoch}")
```

```
                    print("accuracy on train data:␣
↪",accuracy_score(y_train_onehot, y_pred))
                    if adaptive_learning:
                        print("learning rate: ",(alpha/np.sqrt(epoch)))
                    print("softmax loss: ",softmax_loss)
                    print("\n")

        else:
            epoch = 1
            window = 5
            loss_avg = 0
            while(True):
                for i in range(0, X_train.shape[0], self.M):
                    y_actual = y_train[i:i+self.M,:].T

                    #Forward
                    a["0"] = X_train[i:i+self.M,:].T

                    for j in range(1,len(self.n_hidden_nodes)+1):
                        z[str(j)] = np.matmul(self.W[str(j)], a[str(j-1)]) +␣
↪self.b[str(j)]
                        if activation == "relu":
                            a[str(j)] = NeuralNetwork.relu(z[str(j)])
                        else:
                            a[str(j)] = NeuralNetwork.sigmoid(z[str(j)])

                    j += 1
                    z[str(j)] = np.matmul(self.W[str(j)], a[str(j-1)]) + self.
↪b[str(j)]
                    a[str(j)] = NeuralNetwork.softmax(z[str(j)])

                    #Backward
                    del_z[str(j)] = a[str(j)] - y_actual
                    del_b[str(j)] = np.sum(del_z[str(j)], axis = 1).
↪reshape(-1,1)
                    del_W[str(j)] = np.matmul(del_z[str(j)], a[str(j-1)].T)

                    for k in range(j-1,0,-1):
                        if activation == "relu":
                            del_z[str(k)] = np.matmul(self.W[str(k+1)].T,␣
↪del_z[str(k+1)])*(NeuralNetwork.relu(z[str(k)], derivative=True))
                        else:
                            del_z[str(k)] = np.matmul(self.W[str(k+1)].T,␣
↪del_z[str(k+1)])*(NeuralNetwork.sigmoid(z[str(k)], derivative=True))
                        del_b[str(k)] = np.sum(del_z[str(k)], axis = 1).
↪reshape(-1,1)
```

```
                    del_W[str(k)] = np.matmul(del_z[str(k)], a[str(k-1)].T)

                #Update
                for l in range(1,len(self.n_hidden_nodes)+2):
                    if adaptive_learning == False:
                        self.W[str(l)] = self.W[str(l)] - alpha *␣
↪del_W[str(l)]
                        self.b[str(l)] = self.b[str(l)] - alpha *␣
↪del_b[str(l)]
                    else:
                        self.W[str(l)] = self.W[str(l)] - (alpha/np.
↪sqrt(epoch)) * del_W[str(l)]
                        self.b[str(l)] = self.b[str(l)] - (alpha/np.
↪sqrt(epoch)) * del_b[str(l)]

            y_pred, softmax_output = NN.predict(X_train,␣
↪activation=activation)
            softmax_loss = NeuralNetwork.
↪compute_softmax_loss(softmax_output, y_train_onehot)
            if epoch%printafter==0:
                print(f"epoch {epoch}")
                print("accuracy on train data:␣
↪",accuracy_score(y_train_onehot, y_pred))
                if adaptive_learning:
                    print("learning rate: ",(alpha/np.sqrt(epoch)))
                print("softmax loss: ",softmax_loss)
                print("\n")

            if epoch <= window:
                loss_avg += softmax_loss
                if epoch == window:
                    loss_avg /= window

            #End the training
            if epoch > window:
                new_loss_avg = ((window - 1)*loss_avg + softmax_loss)/window
                diff_avg_loss = abs(new_loss_avg - loss_avg)
                #print("diff avg loss:",diff_avg_loss)
                if diff_avg_loss < stopping_threshold or epoch > EPOCHS:
                    print("Convergence criteria satisfied!")
                    print(f"epoch {epoch}")
                    print("accuracy on train data:␣
↪",accuracy_score(y_train_onehot, y_pred))
                    if adaptive_learning:
                        print("learning rate: ",(alpha/np.sqrt(epoch)))
                    print("softmax loss: ",softmax_loss)
```

```
                        print("\n")
                        break

                epoch += 1

    def predict(self, X_test, activation="sigmoid"):
        y_pred = np.zeros((X_test.shape[0], self.r))
        softmax_output = np.zeros((X_test.shape[0], self.r))
        z = dict()
        a = dict()

        for i in range(X_test.shape[0]):
            a[str(0)] = X_test[i:i+1,:].T

            for j in range(1,len(self.n_hidden_nodes)+1):
                z[str(j)] = np.matmul(self.W[str(j)], a[str(j-1)]) + self.
 ↪b[str(j)]
                if activation == "relu":
                    a[str(j)] = NeuralNetwork.relu(z[str(j)])
                else:
                    a[str(j)] = NeuralNetwork.sigmoid(z[str(j)])

            j += 1
            z[str(j)] = np.matmul(self.W[str(j)], a[str(j-1)]) + self.b[str(j)]
            a[str(j)] = NeuralNetwork.softmax(z[str(j)])

            softmax_output[i] = a[str(j)].flatten()
            y_pred[i][np.argmax(a[str(j)])] = 1

        return y_pred, softmax_output

    @staticmethod
    def compute_softmax_loss(softmax_output, y_pred):
        #Softmax loss
        sm_loss = 0
        for i in range(y_pred.shape[0]):
            sm_loss = -1*np.log2(softmax_output[i][np.argmax(y_pred[i])])

        return sm_loss/(y_pred.shape[0])
```

[28]:

```
NN = NeuralNetwork(n = 1024, n_hidden_nodes = [100,50] , r = 5, M = 32)
NN.train(X_train, y_train_onehot, activation="sigmoid", EPOCHS=100, alpha=0.
 ↪001, printafter=10)
y_pred, softmax_output = NN.predict(X_test, activation="sigmoid")
print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred))
get_metric(y_test_onehot, y_pred)
```

```
epoch 0
accuracy on train data:  0.2091
softmax loss:  0.00023525478839202835


epoch 10
accuracy on train data:  0.5923
softmax loss:  0.00011264389363753989


epoch 20
accuracy on train data:  0.6572
softmax loss:  5.831073565883006e-05


epoch 30
accuracy on train data:  0.6702
softmax loss:  4.656056662492289e-05


epoch 40
accuracy on train data:  0.6758
softmax loss:  3.9645451044889796e-05


epoch 50
accuracy on train data:  0.6803
softmax loss:  3.4488996828750714e-05


epoch 60
accuracy on train data:  0.6869
softmax loss:  3.0593861550354343e-05


epoch 70
accuracy on train data:  0.6908
softmax loss:  2.8066258449785907e-05


epoch 80
accuracy on train data:  0.6976
softmax loss:  2.7373041902283903e-05


epoch 90
accuracy on train data:  0.7037
softmax loss:  2.8412086233645623e-05
```

```
accuracy on test data:  0.714
              precision    recall  f1-score   support

           0       0.93      0.95      0.94       224
           1       0.75      0.76      0.76       196
           2       0.56      0.64      0.60       176
           3       0.49      0.52      0.51       177
           4       0.80      0.66      0.72       227

   micro avg       0.71      0.71      0.71      1000
   macro avg       0.71      0.70      0.70      1000
weighted avg       0.72      0.71      0.72      1000
 samples avg       0.71      0.71      0.71      1000
```

[34]:

```
print("f1_score: ",f1_score(y_test_onehot, y_pred, average="macro"))
```

```
f1_score:  0.7029924239467376
```

Experimenting with single hidden layer

[41]:

```
hidden_layers = [[1],[5],[10],[50],[100]]
number_of_hidden_units = [1,5,10,50,100]
f1_score_train = []
f1_score_test = []
for hidden_layer in hidden_layers:
    print(f"Hidden layer: {hidden_layer}")
    NN = NeuralNetwork(n = 1024, n_hidden_nodes = hidden_layer , r = 5, M = 32)
    NN.train(X_train, y_train_onehot, epoch_mode= False, activation="sigmoid",␣
 ↪alpha = 0.01, stopping_threshold = 1.0e-06, printafter=50)
    y_pred_train, _ = NN.predict(X_train)
    y_pred_test, _ = NN.predict(X_test)

    print("accuracy on train data: ",accuracy_score(y_train_onehot,␣
 ↪y_pred_train))
    print("metrics for train data: ")
    get_metric(y_train_onehot, y_pred_train)
    f1_score_train.append(f1_score(y_train_onehot, y_pred_train,␣
 ↪average="macro"))

    print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred_test))
    print("metrics for test data: ")
    get_metric(y_test_onehot, y_pred_test)
    f1_score_test.append(f1_score(y_test_onehot, y_pred_test, average="macro"))

    print("\n")
```

```
Hidden layer: [1]
epoch 50
accuracy on train data:  0.4706
softmax loss:  0.00014958531153011078


epoch 100
accuracy on train data:  0.2091
softmax loss:  0.00023287816358262217


epoch 150
accuracy on train data:  0.2091
softmax loss:  0.0002328781641601466


epoch 200
accuracy on train data:  0.2091
softmax loss:  0.00023287816494873916


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.2091
softmax loss:  0.00023287816496746151


accuracy on train data:  0.2091
metrics for train data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.21      0.35     10000

   micro avg       0.21      0.21      0.21     10000
   macro avg       0.20      0.04      0.07     10000
weighted avg       1.00      0.21      0.35     10000
 samples avg       0.21      0.21      0.21     10000

accuracy on test data:  0.187
metrics for test data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.19      0.32      1000

   micro avg       0.19      0.19      0.19      1000
   macro avg       0.20      0.04      0.06      1000
weighted avg       1.00      0.19      0.32      1000
 samples avg       0.19      0.19      0.19      1000
```

```
Hidden layer: [5]
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
epoch 50
accuracy on train data:  0.6775
softmax loss:  5.1494534443521053e-05


epoch 100
accuracy on train data:  0.6858
softmax loss:  3.220427274554847e-05


epoch 150
accuracy on train data:  0.6952
softmax loss:  2.7774495226855474e-05


epoch 200
accuracy on train data:  0.6982
softmax loss:  2.603425353508207e-05


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.6977
softmax loss:  2.6049241749681784e-05
```

```
accuracy on train data:  0.6977
metrics for train data:
              precision    recall  f1-score   support

           0       0.95      0.86      0.90      2176
           1       0.64      0.77      0.70      1652
           2       0.73      0.56      0.63      2544
           3       0.41      0.56      0.47      1447
           4       0.77      0.73      0.75      2181

   micro avg       0.70      0.70      0.70     10000
   macro avg       0.70      0.70      0.69     10000
weighted avg       0.72      0.70      0.70     10000
 samples avg       0.70      0.70      0.70     10000

accuracy on test data:  0.689
metrics for test data:
              precision    recall  f1-score   support

           0       0.94      0.87      0.91       246
           1       0.63      0.76      0.69       163
           2       0.72      0.58      0.64       248
           3       0.41      0.49      0.45       158
           4       0.69      0.70      0.69       185

   micro avg       0.69      0.69      0.69      1000
   macro avg       0.68      0.68      0.68      1000
weighted avg       0.71      0.69      0.69      1000
 samples avg       0.69      0.69      0.69      1000



Hidden layer: [10]
epoch 50
accuracy on train data:  0.6885
softmax loss:  2.3303944233466556e-05


epoch 100
accuracy on train data:  0.7189
softmax loss:  2.3564423526932722e-05


epoch 150
accuracy on train data:  0.7241
softmax loss:  2.151708934034692e-05
```

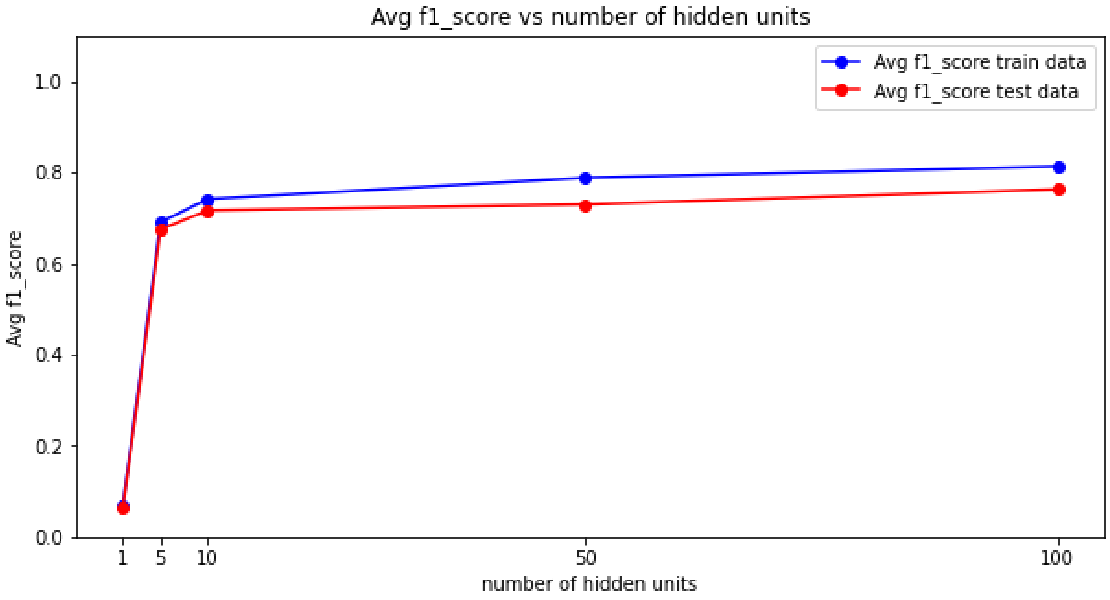
```
epoch 200
accuracy on train data:  0.7438
softmax loss:  1.7535741967859166e-05


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.7436
softmax loss:  1.7259669594328303e-05


accuracy on train data:  0.7436
metrics for train data:
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      1930
           1       0.75      0.81      0.78      1852
           2       0.67      0.66      0.66      2000
           3       0.51      0.59      0.55      1752
           4       0.85      0.72      0.78      2466

   micro avg       0.74      0.74      0.74     10000
   macro avg       0.74      0.74      0.74     10000
weighted avg       0.75      0.74      0.75     10000
 samples avg       0.74      0.74      0.74     10000

accuracy on test data:  0.724
metrics for test data:
              precision    recall  f1-score   support

           0       0.89      0.96      0.92       212
           1       0.73      0.76      0.75       191
           2       0.65      0.64      0.65       204
           3       0.51      0.55      0.53       175
           4       0.80      0.69      0.74       218

   micro avg       0.72      0.72      0.72      1000
   macro avg       0.72      0.72      0.72      1000
weighted avg       0.73      0.72      0.72      1000
 samples avg       0.72      0.72      0.72      1000



Hidden layer: [50]
epoch 50
accuracy on train data:  0.7392
softmax loss:  2.6987471937410535e-05
```

```
epoch 100
accuracy on train data:  0.7839
softmax loss:  1.7227472892700306e-05


epoch 150
accuracy on train data:  0.7706
softmax loss:  1.6560629492585795e-05


epoch 200
accuracy on train data:  0.7877
softmax loss:  8.237749203355322e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.788
softmax loss:  7.399416514018198e-06


accuracy on train data:  0.788
metrics for train data:
              precision    recall  f1-score   support

           0       0.94      0.98      0.96      1878
           1       0.79      0.88      0.84      1783
           2       0.64      0.72      0.68      1734
           3       0.65      0.62      0.64      2106
           4       0.91      0.76      0.83      2499

   micro avg       0.79      0.79      0.79     10000
   macro avg       0.79      0.79      0.79     10000
weighted avg       0.79      0.79      0.79     10000
 samples avg       0.79      0.79      0.79     10000

accuracy on test data:  0.735
metrics for test data:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96       215
           1       0.76      0.82      0.79       183
           2       0.55      0.65      0.60       168
           3       0.63      0.52      0.57       229
           4       0.76      0.70      0.73       205
   micro avg       0.73      0.73      0.73      1000
   macro avg       0.73      0.74      0.73      1000
weighted avg       0.73      0.73      0.73      1000
 samples avg       0.73      0.73      0.73      1000



Hidden layer: [100]
epoch 50
accuracy on train data:  0.733
softmax loss:  4.538836493085465e-05


epoch 100
accuracy on train data:  0.7818
softmax loss:  1.8238639023683118e-05


epoch 150
accuracy on train data:  0.8035
softmax loss:  9.2365944713612e-06


epoch 200
accuracy on train data:  0.8139
softmax loss:  9.666839084236848e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8133
softmax loss:  9.72563184360032e-06


accuracy on train data:  0.8133
metrics for train data:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      2010
           1       0.82      0.92      0.87      1766
           2       0.74      0.77      0.75      1883
           3       0.64      0.67      0.65      1914
           4       0.88      0.76      0.82      2427

   micro avg       0.81      0.81      0.81     10000
   macro avg       0.81      0.82      0.81     10000
weighted avg       0.82      0.81      0.81     10000
 samples avg       0.81      0.81      0.81     10000
```

```
accuracy on test data:  0.77
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       228
           1       0.74      0.84      0.79       174
           2       0.65      0.68      0.67       191
           3       0.63      0.59      0.61       197
           4       0.82      0.73      0.78       210

   micro avg       0.77      0.77      0.77      1000
   macro avg       0.76      0.77      0.76      1000
weighted avg       0.77      0.77      0.77      1000
 samples avg       0.77      0.77      0.77      1000
```

[43]:

```
#Plot avg f1_scores for different number of hidden units
plt.figure(figsize=(10,5))
plt.plot(number_of_hidden_units, f1_score_train, marker='o', markersize=6,␣
 ↪color='blue', label='Avg f1_score train data')
plt.plot(number_of_hidden_units, f1_score_test, marker='o', markersize=6,␣
 ↪color='red', label='Avg f1_score test data')

plt.title('Avg f1_score vs number of hidden units')
plt.xlabel('number of hidden units')
plt.ylabel('Avg f1_score')
plt.xticks(number_of_hidden_units)
plt.ylim(0,1.1)
plt.legend()
plt.show()
```



Experimenting with hidden layers

[44]:

```
hidden_layers = [[512], [512,256], [512,256,128], [512,256,128,64]]
network_depth = [1,2,3,4]
f1_score_train = []
f1_score_test = []
for hidden_layer in hidden_layers:
    print(f"Hidden layer: {hidden_layer}")
    NN = NeuralNetwork(n = 1024, n_hidden_nodes = hidden_layer , r = 5, M = 32)
    NN.train(X_train, y_train_onehot, epoch_mode= False, activation="sigmoid",␣
 ↪alpha = 0.01, stopping_threshold = 1.0e-06, printafter=50)
    y_pred_train, _ = NN.predict(X_train)
    y_pred_test, _ = NN.predict(X_test)

    print("accuracy on train data: ",accuracy_score(y_train_onehot,␣
 ↪y_pred_train))
    print("metrics for train data: ")
    get_metric(y_train_onehot, y_pred_train)
    f1_score_train.append(f1_score(y_train_onehot, y_pred_train,␣
 ↪average="macro"))

    print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred_test))
    print("metrics for test data: ")
    get_metric(y_test_onehot, y_pred_test)
    f1_score_test.append(f1_score(y_test_onehot, y_pred_test, average="macro"))

    print("\n")
```

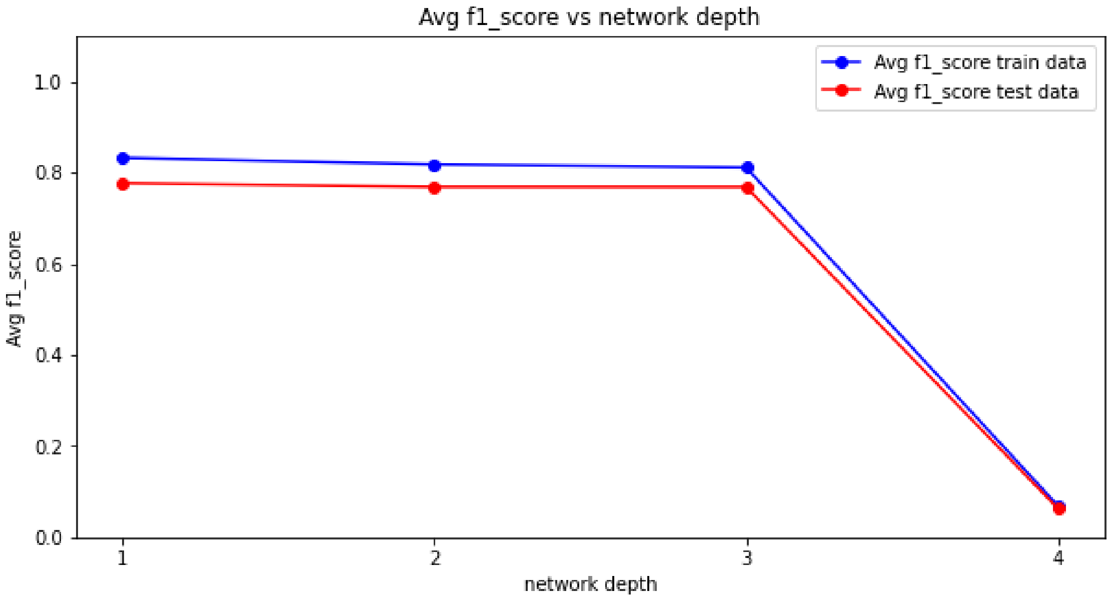
```
Hidden layer: [512]
epoch 50
accuracy on train data:  0.7327
softmax loss:  3.9801766116042114e-05


epoch 100
accuracy on train data:  0.7921
softmax loss:  5.780218764128989e-06


epoch 150
accuracy on train data:  0.8118
softmax loss:  3.7589816123345803e-06


epoch 200
accuracy on train data:  0.8299
softmax loss:  1.2992431023633802e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.831
softmax loss:  1.239104246690159e-06


accuracy on train data:  0.831
metrics for train data:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1933
           1       0.87      0.92      0.89      1863
           2       0.70      0.82      0.75      1669
           3       0.75      0.66      0.70      2286
           4       0.87      0.81      0.84      2249

   micro avg       0.83      0.83      0.83     10000
   macro avg       0.83      0.84      0.83     10000
weighted avg       0.83      0.83      0.83     10000
 samples avg       0.83      0.83      0.83     10000

accuracy on test data:  0.782
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       223
           1       0.81      0.88      0.84       181
           2       0.58      0.72      0.64       162
           3       0.73      0.56      0.64       241
           4       0.80      0.77      0.78       193

   micro avg       0.78      0.78      0.78      1000
   macro avg       0.78      0.79      0.78      1000
weighted avg       0.78      0.78      0.78      1000
 samples avg       0.78      0.78      0.78      1000



Hidden layer: [512, 256]
epoch 50
accuracy on train data:  0.7299
softmax loss:  2.5517614437811225e-05


epoch 100
accuracy on train data:  0.7836
softmax loss:  6.656917643055116e-06


epoch 150
accuracy on train data:  0.8126
softmax loss:  5.07376437203254e-06


epoch 200
accuracy on train data:  0.8213
softmax loss:  6.25927649641769e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8206
softmax loss:  6.134507568326737e-06


accuracy on train data:  0.8206
metrics for train data:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1943
           1       0.91      0.90      0.90      1997
           2       0.71      0.83      0.76      1661
           3       0.58      0.68      0.63      1733
           4       0.93      0.73      0.82      2666
   micro avg       0.82      0.82      0.82     10000
   macro avg       0.82      0.82      0.82     10000
weighted avg       0.84      0.82      0.82     10000
 samples avg       0.82      0.82      0.82     10000

accuracy on test data:  0.78
metrics for test data:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       226
           1       0.88      0.84      0.86       206
           2       0.59      0.76      0.66       154
           3       0.57      0.57      0.57       186
           4       0.85      0.70      0.77       228

   micro avg       0.78      0.78      0.78      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.79      0.78      0.78      1000
 samples avg       0.78      0.78      0.78      1000



Hidden layer: [512, 256, 128]
epoch 50
accuracy on train data:  0.7228
softmax loss:  3.62084087252217e-05


epoch 100
accuracy on train data:  0.7799
softmax loss:  1.917585498575037e-05


epoch 150
accuracy on train data:  0.8019
softmax loss:  1.001024824261873e-05


epoch 200
accuracy on train data:  0.8182
softmax loss:  5.357968292509009e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8138
softmax loss:  6.170754618629236e-06
```

```
accuracy on train data:  0.8138
metrics for train data:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1996
           1       0.88      0.94      0.90      1853
           2       0.68      0.80      0.74      1651
           3       0.58      0.64      0.61      1809
           4       0.94      0.73      0.82      2691

   micro avg       0.81      0.81      0.81     10000
   macro avg       0.81      0.82      0.81     10000
weighted avg       0.83      0.81      0.82     10000
 samples avg       0.81      0.81      0.81     10000

accuracy on test data:  0.778
metrics for test data:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       230
           1       0.82      0.89      0.85       182
           2       0.58      0.72      0.65       160
           3       0.60      0.56      0.58       201
           4       0.86      0.70      0.77       227

   micro avg       0.78      0.78      0.78      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.79      0.78      0.78      1000
 samples avg       0.78      0.78      0.78      1000



Hidden layer: [512, 256, 128, 64]
Convergence criteria satisfied!
epoch 25
accuracy on train data:  0.2091
softmax loss:  0.0002359463199165115


accuracy on train data:  0.2091
metrics for train data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.21      0.35     10000

   micro avg       0.21      0.21      0.21     10000
   macro avg       0.20      0.04      0.07     10000
weighted avg       1.00      0.21      0.35     10000
 samples avg       0.21      0.21      0.21     10000

accuracy on test data:  0.187
metrics for test data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.19      0.32      1000

   micro avg       0.19      0.19      0.19      1000
   macro avg       0.20      0.04      0.06      1000
weighted avg       1.00      0.19      0.32      1000
 samples avg       0.19      0.19      0.19      1000
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[45]:

```
#Plot avg f1_scores for different depth
plt.figure(figsize=(10,5))
plt.plot(network_depth, f1_score_train, marker='o', markersize=6, color='blue',␣
 ↪label='Avg f1_score train data')
plt.plot(network_depth, f1_score_test, marker='o', markersize=6, color='red',␣
 ↪label='Avg f1_score test data')

plt.title('Avg f1_score vs network depth')
plt.xlabel('network depth')
plt.ylabel('Avg f1_score')
plt.xticks(network_depth)
plt.ylim(0,1.1)
plt.legend()
plt.show()
```



Adaptive learning

[46]:
```
hidden_layers = [[512], [512,256], [512,256,128], [512,256,128,64]]
network_depth = [1,2,3,4]
f1_score_train = []
f1_score_test = []
for hidden_layer in hidden_layers:
    print(f"Hidden layer: {hidden_layer}")
    NN = NeuralNetwork(n = 1024, n_hidden_nodes = hidden_layer , r = 5, M = 32)
    NN.train(X_train, y_train_onehot, epoch_mode= False, activation="sigmoid",␣
 ↪adaptive_learning = True, alpha = 0.01, stopping_threshold = 5.0e-06,␣
 ↪printafter=50)
    y_pred_train, _ = NN.predict(X_train)
    y_pred_test, _ = NN.predict(X_test)

    print("accuracy on train data: ",accuracy_score(y_train_onehot,␣
 ↪y_pred_train))
    print("metrics for train data: ")
    get_metric(y_train_onehot, y_pred_train)
    f1_score_train.append(f1_score(y_train_onehot, y_pred_train,␣
 ↪average="macro"))

    print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred_test))
    print("metrics for test data: ")
```

```python
    get_metric(y_test_onehot, y_pred_test)
    f1_score_test.append(f1_score(y_test_onehot, y_pred_test, average="macro"))

    print("\n")
```

```
Hidden layer: [512]
epoch 50
accuracy on train data:  0.701
learning rate:  0.001414213562373095
softmax loss:  2.4384530256167004e-05


epoch 100
accuracy on train data:  0.7294
learning rate:  0.001
softmax loss:  2.2590405138106635e-05


epoch 150
accuracy on train data:  0.7535
learning rate:  0.0008164965809277261
softmax loss:  2.4823077935786723e-05


epoch 200
accuracy on train data:  0.7701
learning rate:  0.0007071067811865475
softmax loss:  2.5202653816610038e-05


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.7705
learning rate:  0.0007053456158585983
softmax loss:  2.518159706339217e-05


accuracy on train data:  0.7705
metrics for train data:
              precision    recall  f1-score   support

           0       0.97      0.94      0.95      2031
           1       0.81      0.83      0.82      1937
           2       0.65      0.70      0.68      1809
           3       0.57      0.62      0.59      1849
           4       0.85      0.75      0.80      2374
```

```
   micro avg       0.77      0.77      0.77     10000
   macro avg       0.77      0.77      0.77     10000
weighted avg       0.78      0.77      0.77     10000
 samples avg       0.77      0.77      0.77     10000

accuracy on test data:  0.769
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       232
           1       0.80      0.81      0.81       194
           2       0.61      0.69      0.65       176
           3       0.63      0.59      0.61       199
           4       0.80      0.75      0.77       199

   micro avg       0.77      0.77      0.77      1000
   macro avg       0.76      0.76      0.76      1000
weighted avg       0.77      0.77      0.77      1000
 samples avg       0.77      0.77      0.77      1000



Hidden layer: [512, 256]
epoch 50
accuracy on train data:  0.6913
learning rate:  0.001414213562373095
softmax loss:  2.829604797009096e-05


epoch 100
accuracy on train data:  0.7283
learning rate:  0.001
softmax loss:  2.5478154341155774e-05


epoch 150
accuracy on train data:  0.7609
learning rate:  0.0008164965809277261
softmax loss:  3.3394030775105174e-05


epoch 200
accuracy on train data:  0.7788
learning rate:  0.0007071067811865475
softmax loss:  2.511949838152347e-05


Convergence criteria satisfied!
```

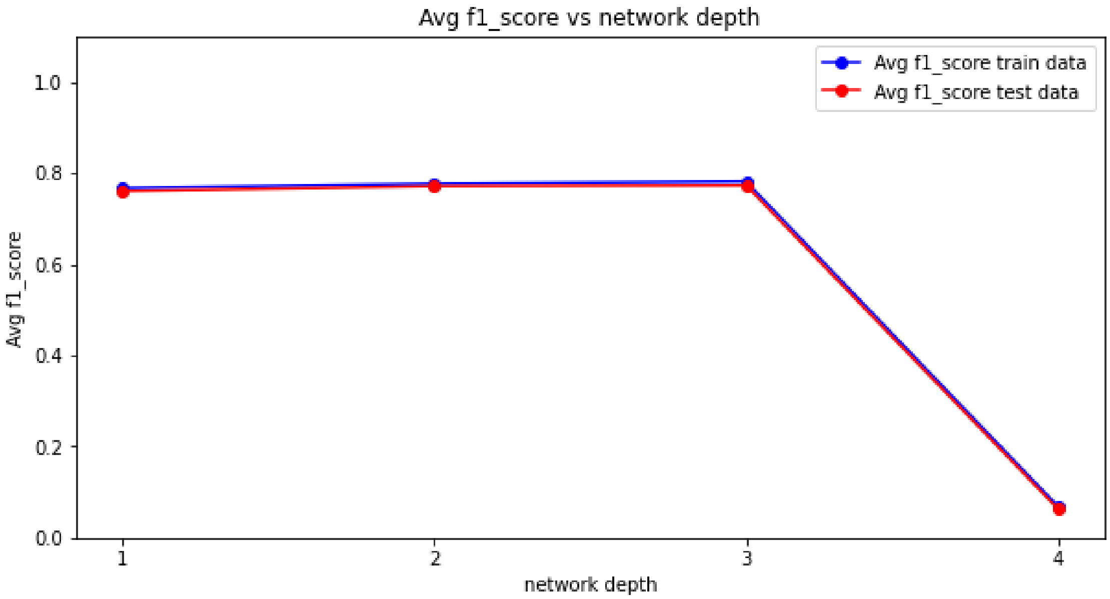
```
epoch 201
accuracy on train data:  0.779
learning rate:  0.0007053456158585983
softmax loss:  2.4884937458219116e-05


accuracy on train data:  0.779
metrics for train data:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1996
           1       0.84      0.86      0.85      1951
           2       0.66      0.72      0.69      1801
           3       0.56      0.61      0.58      1850
           4       0.85      0.74      0.79      2402

   micro avg       0.78      0.78      0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.79      0.78      0.78     10000
 samples avg       0.78      0.78      0.78     10000

accuracy on test data:  0.78
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       228
           1       0.83      0.83      0.83       199
           2       0.63      0.71      0.67       177
           3       0.62      0.60      0.61       193
           4       0.80      0.74      0.77       203

   micro avg       0.78      0.78      0.78      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.78      0.78      0.78      1000
 samples avg       0.78      0.78      0.78      1000



Hidden layer: [512, 256, 128]
epoch 50
accuracy on train data:  0.6908
learning rate:  0.001414213562373095
softmax loss:  2.5470862546841987e-05


epoch 100
accuracy on train data:  0.734
learning rate:  0.001
```

```
softmax loss:  3.018135410329875e-05


epoch 150
accuracy on train data:  0.7671
learning rate:  0.0008164965809277261
softmax loss:  3.57361083756996e-05


epoch 200
accuracy on train data:  0.7839
learning rate:  0.0007071067811865475
softmax loss:  1.8145684474911947e-05


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.7838
learning rate:  0.0007053456158585983
softmax loss:  1.784769632453523e-05


accuracy on train data:  0.7838
metrics for train data:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      1991
           1       0.85      0.87      0.86      1923
           2       0.70      0.72      0.71      1914
           3       0.54      0.62      0.58      1762
           4       0.85      0.73      0.79      2410

   micro avg       0.78      0.78      0.78     10000
   macro avg       0.78      0.78      0.78     10000
weighted avg       0.79      0.78      0.79     10000
 samples avg       0.78      0.78      0.78     10000

accuracy on test data:  0.783
metrics for test data:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       229
           1       0.83      0.85      0.84       194
           2       0.68      0.71      0.70       192
           3       0.57      0.60      0.58       176
           4       0.80      0.72      0.76       209

   micro avg       0.78      0.78      0.78      1000
   macro avg       0.77      0.77      0.77      1000
weighted avg       0.79      0.78      0.78      1000
 samples avg       0.78      0.78      0.78      1000



Hidden layer: [512, 256, 128, 64]
Convergence criteria satisfied!
epoch 6
accuracy on train data:  0.2091
learning rate:  0.004082482904638631
softmax loss:  0.000232068269599483


accuracy on train data:  0.2091
metrics for train data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.21      0.35     10000

   micro avg       0.21      0.21      0.21     10000
   macro avg       0.20      0.04      0.07     10000
weighted avg       1.00      0.21      0.35     10000
 samples avg       0.21      0.21      0.21     10000

accuracy on test data:  0.187
metrics for test data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.19      0.32      1000

   micro avg       0.19      0.19      0.19      1000
   macro avg       0.20      0.04      0.06      1000
weighted avg       1.00      0.19      0.32      1000
 samples avg       0.19      0.19      0.19      1000



/home/tkarthikeyan/.local/lib/python3.8/site-
```

```
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[47]:

```
#Plot avg f1_scores for different depth
plt.figure(figsize=(10,5))
plt.plot(network_depth, f1_score_train, marker='o', markersize=6, color='blue',␣
 ↪label='Avg f1_score train data')
plt.plot(network_depth, f1_score_test, marker='o', markersize=6, color='red',␣
 ↪label='Avg f1_score test data')

plt.title('Avg f1_score vs network depth')
plt.xlabel('network depth')
plt.ylabel('Avg f1_score')
plt.xticks(network_depth)
plt.ylim(0,1.1)
plt.legend()
plt.show()
```



Relu activation function

[48]:

```
hidden_layers = [[512], [512,256], [512,256,128], [512,256,128,64]]
network_depth = [1,2,3,4]
f1_score_train = []
f1_score_test = []
for hidden_layer in hidden_layers:
    print(f"Hidden layer: {hidden_layer}")
    NN = NeuralNetwork(n = 1024, n_hidden_nodes = hidden_layer , r = 5, M = 32)
    NN.train(X_train, y_train_onehot, epoch_mode= False, activation="relu",␣
 ↪adaptive_learning = True, alpha = 0.01, stopping_threshold = 5.0e-06,␣
 ↪printafter=50)
    y_pred_train, _ = NN.predict(X_train, activation="relu")
    y_pred_test, _ = NN.predict(X_test, activation="relu")

    print("accuracy on train data: ",accuracy_score(y_train_onehot,␣
 ↪y_pred_train))
    print("metrics for train data: ")
    get_metric(y_train_onehot, y_pred_train)
    f1_score_train.append(f1_score(y_train_onehot, y_pred_train,␣
 ↪average="macro"))

    print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred_test))
    print("metrics for test data: ")
    get_metric(y_test_onehot, y_pred_test)
    f1_score_test.append(f1_score(y_test_onehot, y_pred_test, average="macro"))

    print("\n")
```

```
Hidden layer: [512]
Convergence criteria satisfied!
epoch 6
accuracy on train data:  0.2091
learning rate:  0.004082482904638631
softmax loss:  0.00023208783009197323


accuracy on train data:  0.2091
metrics for train data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.21      0.35     10000

   micro avg       0.21      0.21      0.21     10000
   macro avg       0.20      0.04      0.07     10000
```

```
weighted avg       1.00      0.21      0.35     10000
 samples avg       0.21      0.21      0.21     10000

accuracy on test data:  0.187
metrics for test data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.19      0.32      1000

   micro avg       0.19      0.19      0.19      1000
   macro avg       0.20      0.04      0.06      1000
weighted avg       1.00      0.19      0.32      1000
 samples avg       0.19      0.19      0.19      1000
```

```
Hidden layer: [512, 256]
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Convergence criteria satisfied!
epoch 6
accuracy on train data:  0.2091
learning rate:  0.004082482904638631
softmax loss:  0.00023044619233218206
```

```
accuracy on train data:  0.2091
metrics for train data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.21      0.35     10000
```

```
   micro avg       0.21      0.21      0.21     10000
   macro avg       0.20      0.04      0.07     10000
weighted avg       1.00      0.21      0.35     10000
 samples avg       0.21      0.21      0.21     10000

accuracy on test data:  0.187
metrics for test data:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0
           4       1.00      0.19      0.32      1000

   micro avg       0.19      0.19      0.19      1000
   macro avg       0.20      0.04      0.06      1000
weighted avg       1.00      0.19      0.32      1000
 samples avg       0.19      0.19      0.19      1000



Hidden layer: [512, 256, 128]

/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

epoch 50
accuracy on train data:  0.7289
learning rate:  0.001414213562373095
softmax loss:  3.145936113878991e-05


epoch 100
accuracy on train data:  0.7583
learning rate:  0.001
softmax loss:  3.272657747861718e-05


epoch 150
```

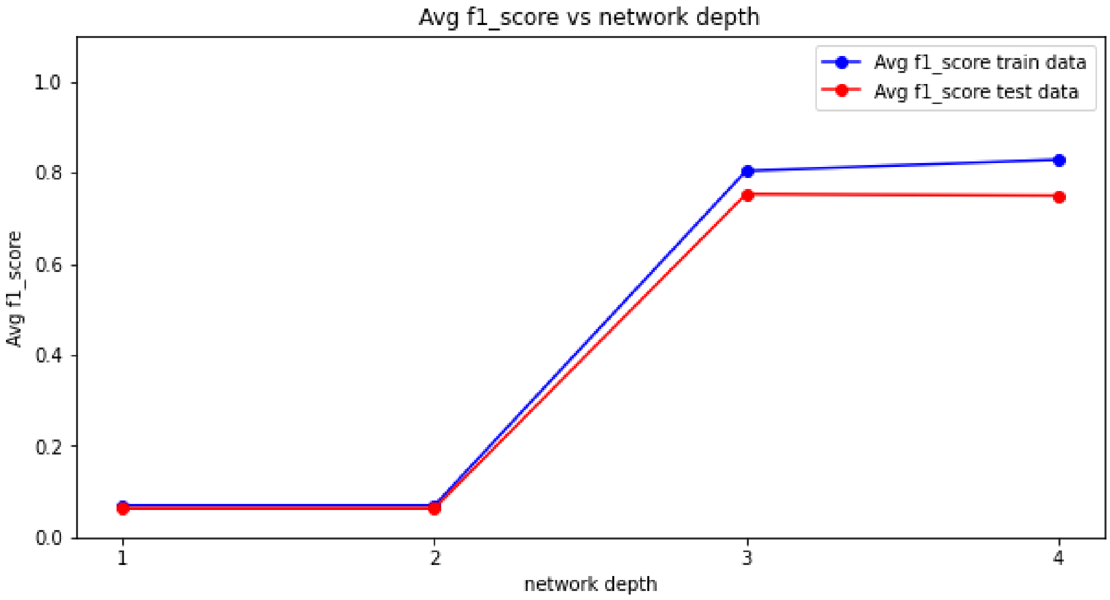
```
accuracy on train data:  0.7812
learning rate:  0.0008164965809277261
softmax loss:  1.1503443467968143e-05


epoch 200
accuracy on train data:  0.7962
learning rate:  0.0007071067811865475
softmax loss:  1.7943773712320744e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8045
learning rate:  0.0007053456158585983
softmax loss:  2.1482919875104044e-06


accuracy on train data:  0.8045
metrics for train data:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1966
           1       0.87      0.93      0.90      1845
           2       0.66      0.80      0.72      1624
           3       0.59      0.61      0.60      1932
           4       0.91      0.72      0.80      2633

   micro avg       0.80      0.80      0.80     10000
   macro avg       0.80      0.81      0.80     10000
weighted avg       0.82      0.80      0.81     10000
 samples avg       0.80      0.80      0.80     10000

accuracy on test data:  0.762
metrics for test data:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       226
           1       0.80      0.88      0.84       179
           2       0.57      0.70      0.63       164
           3       0.57      0.54      0.55       199
           4       0.85      0.69      0.76       232

   micro avg       0.76      0.76      0.76      1000
   macro avg       0.75      0.76      0.75      1000
weighted avg       0.77      0.76      0.76      1000
 samples avg       0.76      0.76      0.76      1000
```

```
Hidden layer: [512, 256, 128, 64]
epoch 50
accuracy on train data:  0.7216
learning rate:  0.001414213562373095
softmax loss:  1.9969842319606146e-05


epoch 100
accuracy on train data:  0.7709
learning rate:  0.001
softmax loss:  1.154033885296016e-05


epoch 150
accuracy on train data:  0.8061
learning rate:  0.0008164965809277261
softmax loss:  9.192962043513032e-07


epoch 200
accuracy on train data:  0.8028
learning rate:  0.0007071067811865475
softmax loss:  4.717764805687347e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.832
learning rate:  0.0007053456158585983
softmax loss:  9.181468735261794e-07


accuracy on train data:  0.832
metrics for train data:
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1988
           1       0.93      0.91      0.92      2034
           2       0.79      0.80      0.80      1932
           3       0.57      0.71      0.63      1602
           4       0.88      0.76      0.81      2444

   micro avg       0.83      0.83      0.83     10000
   macro avg       0.83      0.83      0.83     10000
weighted avg       0.85      0.83      0.84     10000
 samples avg       0.83      0.83      0.83     10000
```

```
accuracy on test data:  0.761
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      0.98      0.97       225
           1       0.86      0.81      0.83       210
           2       0.62      0.68      0.65       182
           3       0.52      0.56      0.54       174
           4       0.80      0.71      0.75       209

   micro avg       0.76      0.76      0.76      1000
   macro avg       0.75      0.75      0.75      1000
weighted avg       0.77      0.76      0.76      1000
 samples avg       0.76      0.76      0.76      1000
```

[49]:

```
#Plot avg f1_scores for different depth
plt.figure(figsize=(10,5))
plt.plot(network_depth, f1_score_train, marker='o', markersize=6, color='blue',␣
 ↪label='Avg f1_score train data')
plt.plot(network_depth, f1_score_test, marker='o', markersize=6, color='red',␣
 ↪label='Avg f1_score test data')

plt.title('Avg f1_score vs network depth')
plt.xlabel('network depth')
plt.ylabel('Avg f1_score')
plt.xticks(network_depth)
plt.ylim(0,1.1)
plt.legend()
plt.show()
```



Relu activation function (with learning rate = 0.001)

```
[6]: hidden_layers = [[512], [512,256], [512,256,128], [512,256,128,64]]
     network_depth = [1,2,3,4]
     f1_score_train = []
     f1_score_test = []
     for hidden_layer in hidden_layers:
         print(f"Hidden layer: {hidden_layer}")
         NN = NeuralNetwork(n = 1024, n_hidden_nodes = hidden_layer , r = 5, M = 32)
         NN.train(X_train, y_train_onehot, epoch_mode= False, activation="relu",␣
      ↪adaptive_learning = True, alpha = 0.001, stopping_threshold = 5.0e-06,␣
      ↪printafter=50)
         y_pred_train, _ = NN.predict(X_train, activation="relu")
         y_pred_test, _ = NN.predict(X_test, activation="relu")

         print("accuracy on train data: ",accuracy_score(y_train_onehot,␣
      ↪y_pred_train))
         print("metrics for train data: ")
         get_metric(y_train_onehot, y_pred_train)
         f1_score_train.append(f1_score(y_train_onehot, y_pred_train,␣
      ↪average="macro"))

         print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred_test))
         print("metrics for test data: ")
         get_metric(y_test_onehot, y_pred_test)
         f1_score_test.append(f1_score(y_test_onehot, y_pred_test, average="macro"))
```

```
    print("\n")
```

```
Hidden layer: [512]
epoch 50
accuracy on train data:  0.6984
learning rate:  0.0001414213562373095
softmax loss:  3.2206525277455114e-05


epoch 100
accuracy on train data:  0.7177
learning rate:  0.0001
softmax loss:  2.8674372883269804e-05


epoch 150
accuracy on train data:  0.7355
learning rate:  8.164965809277261e-05
softmax loss:  2.6139527954960384e-05


epoch 200
accuracy on train data:  0.7488
learning rate:  7.071067811865475e-05
softmax loss:  2.3554841256167726e-05


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.7493
learning rate:  7.053456158585983e-05
softmax loss:  2.3523313554780805e-05


accuracy on train data:  0.7493
metrics for train data:
              precision    recall  f1-score   support

           0       0.95      0.91      0.93      2050
           1       0.78      0.77      0.78      1990
           2       0.64      0.66      0.65      1904
           3       0.54      0.62      0.58      1748
           4       0.84      0.76      0.80      2308

   micro avg       0.75      0.75      0.75     10000
   macro avg       0.75      0.74      0.75     10000
weighted avg       0.76      0.75      0.75     10000
```

```
 samples avg       0.75      0.75      0.75     10000

accuracy on test data:  0.734
metrics for test data:
              precision    recall  f1-score   support

           0       0.94      0.95      0.95       228
           1       0.77      0.76      0.76       200
           2       0.60      0.63      0.61       189
           3       0.57      0.56      0.56       189
           4       0.75      0.73      0.74       194

   micro avg       0.73      0.73      0.73      1000
   macro avg       0.73      0.72      0.73      1000
weighted avg       0.74      0.73      0.73      1000
 samples avg       0.73      0.73      0.73      1000



Hidden layer: [512, 256]
epoch 50
accuracy on train data:  0.7485
learning rate:  0.0001414213562373095
softmax loss:  2.3803159130456424e-05


epoch 100
accuracy on train data:  0.7995
learning rate:  0.0001
softmax loss:  1.7794089613993414e-05


epoch 150
accuracy on train data:  0.8238
learning rate:  8.164965809277261e-05
softmax loss:  1.1009632399852487e-05


epoch 200
accuracy on train data:  0.842
learning rate:  7.071067811865475e-05
softmax loss:  8.309603697956972e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8419
learning rate:  7.053456158585983e-05
```

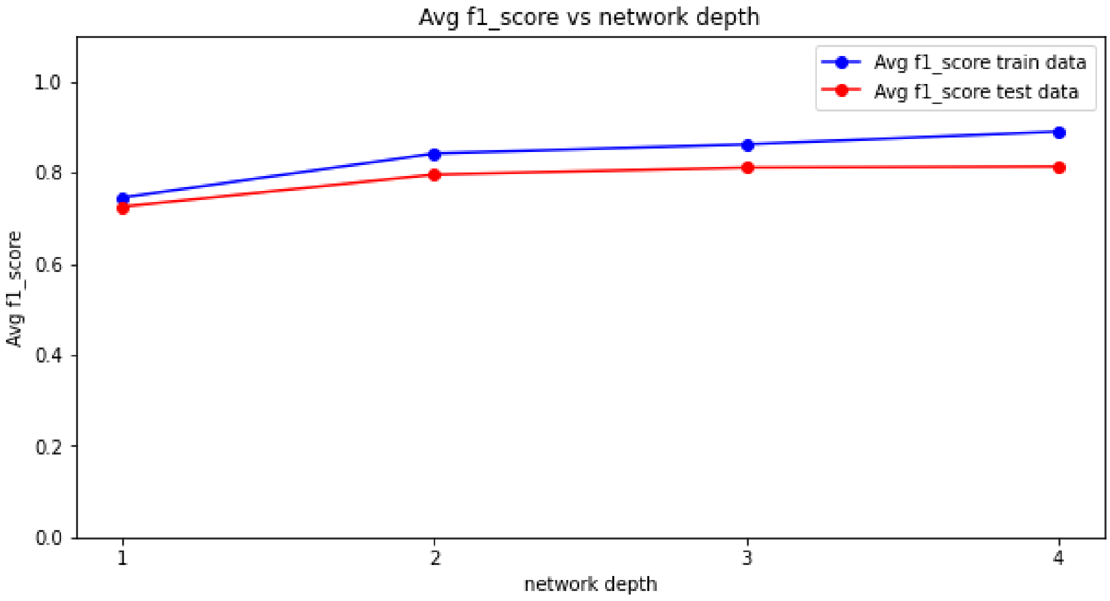
```
softmax loss:  8.155999959208848e-06


accuracy on train data:  0.8419
metrics for train data:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98      1956
           1       0.88      0.91      0.90      1910
           2       0.77      0.81      0.78      1857
           3       0.69      0.71      0.70      1963
           4       0.89      0.80      0.84      2314

   micro avg       0.84      0.84      0.84     10000
   macro avg       0.84      0.84      0.84     10000
weighted avg       0.84      0.84      0.84     10000
 samples avg       0.84      0.84      0.84     10000

accuracy on test data:  0.803
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       224
           1       0.85      0.86      0.86       196
           2       0.69      0.75      0.72       182
           3       0.64      0.62      0.63       191
           4       0.83      0.75      0.79       207

   micro avg       0.80      0.80      0.80      1000
   macro avg       0.80      0.80      0.80      1000
weighted avg       0.80      0.80      0.80      1000
 samples avg       0.80      0.80      0.80      1000



Hidden layer: [512, 256, 128]
epoch 50
accuracy on train data:  0.7774
learning rate:  0.0001414213562373095
softmax loss:  2.9403732478028548e-05


epoch 100
accuracy on train data:  0.8168
learning rate:  0.0001
softmax loss:  1.2094483311416297e-05
```

```
epoch 150
accuracy on train data:  0.847
learning rate:  8.164965809277261e-05
softmax loss:  8.964043796841776e-06


epoch 200
accuracy on train data:  0.8617
learning rate:  7.071067811865475e-05
softmax loss:  5.319551939104827e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8618
learning rate:  7.053456158585983e-05
softmax loss:  5.11851412273001e-06


accuracy on train data:  0.8618
metrics for train data:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1922
           1       0.88      0.93      0.91      1872
           2       0.82      0.83      0.82      1927
           3       0.71      0.76      0.74      1893
           4       0.92      0.81      0.86      2386

   micro avg       0.86      0.86      0.86     10000
   macro avg       0.86      0.86      0.86     10000
weighted avg       0.86      0.86      0.86     10000
 samples avg       0.86      0.86      0.86     10000

accuracy on test data:  0.816
metrics for test data:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       224
           1       0.83      0.89      0.86       186
           2       0.70      0.76      0.73       182
           3       0.72      0.64      0.67       211
           4       0.83      0.79      0.81       197

   micro avg       0.82      0.82      0.82      1000
   macro avg       0.81      0.81      0.81      1000
weighted avg       0.81      0.82      0.81      1000
 samples avg       0.82      0.82      0.82      1000
```

```
Hidden layer: [512, 256, 128, 64]
epoch 50
accuracy on train data:  0.7796
learning rate:  0.0001414213562373095
softmax loss:  2.1823853735152896e-05


epoch 100
accuracy on train data:  0.8303
learning rate:  0.0001
softmax loss:  9.080570686102212e-06


epoch 150
accuracy on train data:  0.8567
learning rate:  8.164965809277261e-05
softmax loss:  5.53839396842743e-06


epoch 200
accuracy on train data:  0.8847
learning rate:  7.071067811865475e-05
softmax loss:  5.457597634399534e-06


Convergence criteria satisfied!
epoch 201
accuracy on train data:  0.8908
learning rate:  7.053456158585983e-05
softmax loss:  3.781473057701032e-06


accuracy on train data:  0.8908
metrics for train data:
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      1990
           1       0.93      0.95      0.94      1941
           2       0.85      0.87      0.86      1903
           3       0.74      0.81      0.78      1832
           4       0.94      0.84      0.89      2334

   micro avg       0.89      0.89      0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

```
 samples avg       0.89      0.89      0.89     10000

accuracy on test data:  0.82
metrics for test data:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       232
           1       0.87      0.91      0.89       191
           2       0.71      0.80      0.76       177
           3       0.67      0.63      0.65       201
           4       0.81      0.76      0.78       199

   micro avg       0.82      0.82      0.82      1000
   macro avg       0.81      0.82      0.81      1000
weighted avg       0.82      0.82      0.82      1000
 samples avg       0.82      0.82      0.82      1000
```

```
[7]: #Plot avg f1_scores for different depth
     plt.figure(figsize=(10,5))
     plt.plot(network_depth, f1_score_train, marker='o', markersize=6, color='blue',␣
      ↪label='Avg f1_score train data')
     plt.plot(network_depth, f1_score_test, marker='o', markersize=6, color='red',␣
      ↪label='Avg f1_score test data')

     plt.title('Avg f1_score vs network depth')
     plt.xlabel('network depth')
     plt.ylabel('Avg f1_score')
     plt.xticks(network_depth)
     plt.ylim(0,1.1)
     plt.legend()
     plt.show()
```



Neural Networks using scikit learn

[19]:

```
hidden_layers = [[512], [512,256], [512,256,128], [512,256,128,64]]
network_depth = [1,2,3,4]
f1_score_train = []
f1_score_test = []
for hidden_layer in hidden_layers:
    print(f"Hidden layer: {hidden_layer}")
    clf = MLPClassifier(activation="relu", solver="sgd", alpha = 0,␣
 ↪batch_size=32, hidden_layer_sizes=np.array(hidden_layer),␣
 ↪learning_rate="invscaling", tol=5e-6, n_iter_no_change=5, verbose=True,␣
 ↪learning_rate_init=0.01).fit(X_train, y_train_onehot)
    y_pred_train = clf.predict(X_train)
    y_pred_test = clf.predict(X_test)

    print("accuracy on train data: ",accuracy_score(y_train_onehot,␣
 ↪y_pred_train))
    print("metrics for train data: ")
    get_metric(y_train_onehot, y_pred_train)
    f1_score_train.append(f1_score(y_train_onehot, y_pred_train,␣
 ↪average="macro"))

    print("accuracy on test data: ",accuracy_score(y_test_onehot, y_pred_test))
    print("metrics for test data: ")
    get_metric(y_test_onehot, y_pred_test)
    f1_score_test.append(f1_score(y_test_onehot, y_pred_test, average="macro"))
    print("\n")
```

```
Hidden layer: [512]
Iteration 1, loss = 1.82100347
Iteration 2, loss = 1.71391782
Iteration 3, loss = 1.52315981
Iteration 4, loss = 1.51474497
Iteration 5, loss = 1.51029548
Iteration 6, loss = 1.50750493
Iteration 7, loss = 1.50565037
Iteration 8, loss = 1.50367111
Iteration 9, loss = 1.50203666
Iteration 10, loss = 1.50045418
Iteration 11, loss = 1.49933494
Iteration 12, loss = 1.49815999
Iteration 13, loss = 1.49698195
Iteration 14, loss = 1.49620743
Iteration 15, loss = 1.49550628
Iteration 16, loss = 1.49454919
Iteration 17, loss = 1.49363666
Iteration 18, loss = 1.49279152
Iteration 19, loss = 1.49200207
Iteration 20, loss = 1.49143824
Iteration 21, loss = 1.49060290
Iteration 22, loss = 1.48995872
Iteration 23, loss = 1.48938653
Iteration 24, loss = 1.48872835
Iteration 25, loss = 1.48810307
Iteration 26, loss = 1.48745704
Iteration 27, loss = 1.48675391
Iteration 28, loss = 1.48601222
Iteration 29, loss = 1.48555051
Iteration 30, loss = 1.48489258
Iteration 31, loss = 1.48442929
Iteration 32, loss = 1.48394169
Iteration 33, loss = 1.48350804
Iteration 34, loss = 1.48292553
Iteration 35, loss = 1.48245257
Iteration 36, loss = 1.48207413
Iteration 37, loss = 1.48157765
Iteration 38, loss = 1.48119172
Iteration 39, loss = 1.48078643
Iteration 40, loss = 1.48028934
Iteration 41, loss = 1.47994385
Iteration 42, loss = 1.47956204
Iteration 43, loss = 1.47913680
Iteration 44, loss = 1.47881800
Iteration 45, loss = 1.47835719
Iteration 46, loss = 1.47806096
Iteration 47, loss = 1.47762217
Iteration 48, loss = 1.47733798
Iteration 49, loss = 1.47684794
Iteration 50, loss = 1.47658435
Iteration 51, loss = 1.47617737
Iteration 52, loss = 1.47584291
Iteration 53, loss = 1.47550371
Iteration 54, loss = 1.47513914
Iteration 55, loss = 1.47476276
Iteration 56, loss = 1.47438837
Iteration 57, loss = 1.47401940
Iteration 58, loss = 1.47368000
Iteration 59, loss = 1.47319879
Iteration 60, loss = 1.47285898
Iteration 61, loss = 1.47256885
Iteration 62, loss = 1.47207112
Iteration 63, loss = 1.47172472
Iteration 64, loss = 1.47138364
Iteration 65, loss = 1.47103137
Iteration 66, loss = 1.47073728
Iteration 67, loss = 1.47036312
Iteration 68, loss = 1.47003571
Iteration 69, loss = 1.46979185
Iteration 70, loss = 1.46951244
Iteration 71, loss = 1.46925222
Iteration 72, loss = 1.46899571
Iteration 73, loss = 1.46865432
Iteration 74, loss = 1.46840627
Iteration 75, loss = 1.46819481
Iteration 76, loss = 1.46778235
Iteration 77, loss = 1.46766729
Iteration 78, loss = 1.46732930
Iteration 79, loss = 1.46709043
Iteration 80, loss = 1.46691234
Iteration 81, loss = 1.46659407
Iteration 82, loss = 1.46647751
Iteration 83, loss = 1.46619232
Iteration 84, loss = 1.46591034
Iteration 85, loss = 1.46570945
Iteration 86, loss = 1.46540203
Iteration 87, loss = 1.46522251
Iteration 88, loss = 1.46499896
Iteration 89, loss = 1.46479823
Iteration 90, loss = 1.46448519
Iteration 91, loss = 1.46429983
Iteration 92, loss = 1.46408573
Iteration 93, loss = 1.46382916
Iteration 94, loss = 1.46363847
Iteration 95, loss = 1.46335830
Iteration 96, loss = 1.46317960
Iteration 97, loss = 1.46292187
Iteration 98, loss = 1.46274574
Iteration 99, loss = 1.46254516
Iteration 100, loss = 1.46228647
Iteration 101, loss = 1.46209452
Iteration 102, loss = 1.46192999
Iteration 103, loss = 1.46172054
Iteration 104, loss = 1.46138636
Iteration 105, loss = 1.46131177
Iteration 106, loss = 1.46108355
Iteration 107, loss = 1.46087098
Iteration 108, loss = 1.46064470
Iteration 109, loss = 1.46041485
Iteration 110, loss = 1.46027431
Iteration 111, loss = 1.46008784
Iteration 112, loss = 1.45988650
Iteration 113, loss = 1.45972962
Iteration 114, loss = 1.45952104
Iteration 115, loss = 1.45931163
Iteration 116, loss = 1.45909816
Iteration 117, loss = 1.45895664
Iteration 118, loss = 1.45877721
Iteration 119, loss = 1.45861600
Iteration 120, loss = 1.45835821
Iteration 121, loss = 1.45818034
Iteration 122, loss = 1.45801117
Iteration 123, loss = 1.45784181
Iteration 124, loss = 1.45764247
Iteration 125, loss = 1.45748175
Iteration 126, loss = 1.45728778
Iteration 127, loss = 1.45711550
Iteration 128, loss = 1.45688891
Iteration 129, loss = 1.45679229
Iteration 130, loss = 1.45655648
Iteration 131, loss = 1.45639470
Iteration 132, loss = 1.45618799
Iteration 133, loss = 1.45606132
Iteration 134, loss = 1.45590004
Iteration 135, loss = 1.45568361
Iteration 136, loss = 1.45551304
Iteration 137, loss = 1.45533163
Iteration 138, loss = 1.45524969
Iteration 139, loss = 1.45502739
Iteration 140, loss = 1.45485948
Iteration 141, loss = 1.45471283
Iteration 142, loss = 1.45452669
Iteration 143, loss = 1.45431788
Iteration 144, loss = 1.45420511
Iteration 145, loss = 1.45401596
Iteration 146, loss = 1.45389316
Iteration 147, loss = 1.45370787
Iteration 148, loss = 1.45356289
Iteration 149, loss = 1.45343819
Iteration 150, loss = 1.45324188
Iteration 151, loss = 1.45308210
Iteration 152, loss = 1.45292444
Iteration 153, loss = 1.45276024
Iteration 154, loss = 1.45260947
Iteration 155, loss = 1.45243541
Iteration 156, loss = 1.45227674
Iteration 157, loss = 1.45219223
Iteration 158, loss = 1.45195802
Iteration 159, loss = 1.45185917
Iteration 160, loss = 1.45166933
Iteration 161, loss = 1.45155882
Iteration 162, loss = 1.45138742
Iteration 163, loss = 1.45124925
Iteration 164, loss = 1.45108788
Iteration 165, loss = 1.45094660
Iteration 166, loss = 1.45077949
Iteration 167, loss = 1.45066910
Iteration 168, loss = 1.45050262
Iteration 169, loss = 1.45032493
Iteration 170, loss = 1.45023156
Iteration 171, loss = 1.45006452
Iteration 172, loss = 1.44994194
Iteration 173, loss = 1.44978832
Iteration 174, loss = 1.44964848
Iteration 175, loss = 1.44951194
Iteration 176, loss = 1.44936269
Iteration 177, loss = 1.44922681
Iteration 178, loss = 1.44907058
Iteration 179, loss = 1.44894591
Iteration 180, loss = 1.44877779
Iteration 181, loss = 1.44865781
Iteration 182, loss = 1.44852393
Iteration 183, loss = 1.44839336
Iteration 184, loss = 1.44824172
Iteration 185, loss = 1.44808943
Iteration 186, loss = 1.44795043
Iteration 187, loss = 1.44782687
Iteration 188, loss = 1.44768827
Iteration 189, loss = 1.44756264
Iteration 190, loss = 1.44745318
Iteration 191, loss = 1.44731446
Iteration 192, loss = 1.44718501
Iteration 193, loss = 1.44703131
Iteration 194, loss = 1.44688879
Iteration 195, loss = 1.44674095
Iteration 196, loss = 1.44659639
Iteration 197, loss = 1.44651981
Iteration 198, loss = 1.44630971
Iteration 199, loss = 1.44626250
Iteration 200, loss = 1.44608502
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
accuracy on train data:  0.3589
metrics for train data:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88      1990
           1       0.21      0.75      0.33       557
           2       0.03      0.58      0.07       117
           3       0.00      0.00      0.00         0
           4       0.66      0.73      0.69      1890

   micro avg       0.36      0.79      0.49      4554
   macro avg       0.36      0.59      0.39      4554
weighted avg       0.68      0.79      0.71      4554
 samples avg       0.36      0.36      0.36      4554

accuracy on test data:  0.363
metrics for test data:
              precision    recall  f1-score   support

           0       0.88      0.89      0.89       226
           1       0.20      0.71      0.31        56
           2       0.05      0.60      0.08        15
           3       0.00      0.00      0.00         0
           4       0.60      0.69      0.65       163

   micro avg       0.36      0.79      0.50       460
   macro avg       0.35      0.58      0.39       460
weighted avg       0.67      0.79      0.71       460
 samples avg       0.36      0.36      0.36       460
```

```
Hidden layer: [512, 256]
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Iteration 1, loss = 1.75599291
Iteration 2, loss = 1.50284355
Iteration 3, loss = 1.39129507
Iteration 4, loss = 1.38382212
Iteration 5, loss = 1.37945388
Iteration 6, loss = 1.37674041
Iteration 7, loss = 1.37485859
Iteration 8, loss = 1.37307523
Iteration 9, loss = 1.37166307
Iteration 10, loss = 1.37024469
Iteration 11, loss = 1.36924451
Iteration 12, loss = 1.36834121
Iteration 13, loss = 1.36742281
Iteration 14, loss = 1.36596267
Iteration 15, loss = 1.36591150
Iteration 16, loss = 1.36515085
Iteration 17, loss = 1.36496215
Iteration 18, loss = 1.36400063
Iteration 19, loss = 1.36335653
Iteration 20, loss = 1.36291710
Iteration 21, loss = 1.36243440
Iteration 22, loss = 1.36167984
Iteration 23, loss = 1.36167079
Iteration 24, loss = 1.36108227
Iteration 25, loss = 1.36048485
Iteration 26, loss = 1.35995516
Iteration 27, loss = 1.35961524
Iteration 28, loss = 1.35919002
Iteration 29, loss = 1.35920701
Iteration 30, loss = 1.35863334
Iteration 31, loss = 1.35818692
Iteration 32, loss = 1.35783203
Iteration 33, loss = 1.35752371
Iteration 34, loss = 1.35744524
Iteration 35, loss = 1.35693297
Iteration 36, loss = 1.35671225
Iteration 37, loss = 1.35632659
Iteration 38, loss = 1.35607923
Iteration 39, loss = 1.35565661
Iteration 40, loss = 1.35532491
Iteration 41, loss = 1.35524164
Iteration 42, loss = 1.35503213
Iteration 43, loss = 1.35473587
Iteration 44, loss = 1.35456664
Iteration 45, loss = 1.35412737
Iteration 46, loss = 1.35385597
Iteration 47, loss = 1.35384404
Iteration 48, loss = 1.35366097
Iteration 49, loss = 1.35333122
Iteration 50, loss = 1.35296687
Iteration 51, loss = 1.35278991
Iteration 52, loss = 1.35252206
Iteration 53, loss = 1.35240748
Iteration 54, loss = 1.35215867
Iteration 55, loss = 1.35190118
Iteration 56, loss = 1.35157876
Iteration 57, loss = 1.35146040
Iteration 58, loss = 1.35123069
Iteration 59, loss = 1.35098145
Iteration 60, loss = 1.35093349
Iteration 61, loss = 1.35068915
Iteration 62, loss = 1.35031236
Iteration 63, loss = 1.35013024
Iteration 64, loss = 1.35012739
Iteration 65, loss = 1.35003691
Iteration 66, loss = 1.34982555
Iteration 67, loss = 1.34947631
Iteration 68, loss = 1.34929738
Iteration 69, loss = 1.34929669
Iteration 70, loss = 1.34913114
Iteration 71, loss = 1.34883042
Iteration 72, loss = 1.34868134
Iteration 73, loss = 1.34864712
Iteration 74, loss = 1.34836011
Iteration 75, loss = 1.34807917
Iteration 76, loss = 1.34797559
Iteration 77, loss = 1.34787465
Iteration 78, loss = 1.34777258
Iteration 79, loss = 1.34758600
Iteration 80, loss = 1.34750898
Iteration 81, loss = 1.34723970
Iteration 82, loss = 1.34714625
Iteration 83, loss = 1.34700567
Iteration 84, loss = 1.34683882
Iteration 85, loss = 1.34662978
Iteration 86, loss = 1.34655793
Iteration 87, loss = 1.34639897
Iteration 88, loss = 1.34631632
Iteration 89, loss = 1.34605786
Iteration 90, loss = 1.34599202
Iteration 91, loss = 1.34590211
Iteration 92, loss = 1.34566993
Iteration 93, loss = 1.34562009
Iteration 94, loss = 1.34530316
Iteration 95, loss = 1.34535392
Iteration 96, loss = 1.34511840
Iteration 97, loss = 1.34508257
Iteration 98, loss = 1.34492088
Iteration 99, loss = 1.34477977
Iteration 100, loss = 1.34469043
Iteration 101, loss = 1.34450016
Iteration 102, loss = 1.34451317
Iteration 103, loss = 1.34436293
Iteration 104, loss = 1.34416883
Iteration 105, loss = 1.34412784
Iteration 106, loss = 1.34393020
Iteration 107, loss = 1.34386049
Iteration 108, loss = 1.34368240
Iteration 109, loss = 1.34351548
Iteration 110, loss = 1.34346316
Iteration 111, loss = 1.34343423
Iteration 112, loss = 1.34330118
Iteration 113, loss = 1.34307329
Iteration 114, loss = 1.34295323
Iteration 115, loss = 1.34266210
Iteration 116, loss = 1.34288639
Iteration 117, loss = 1.34263167
Iteration 118, loss = 1.34253032
Iteration 119, loss = 1.34240821
Iteration 120, loss = 1.34237846
Iteration 121, loss = 1.34209611
Iteration 122, loss = 1.34211200
Iteration 123, loss = 1.34199165
Iteration 124, loss = 1.34185707
Iteration 125, loss = 1.34172529
Iteration 126, loss = 1.34168345
Iteration 127, loss = 1.34150643
Iteration 128, loss = 1.34146673
Iteration 129, loss = 1.34139252
Iteration 130, loss = 1.34123276
Iteration 131, loss = 1.34114062
Iteration 132, loss = 1.34104560
Iteration 133, loss = 1.34099703
Iteration 134, loss = 1.34089745
Iteration 135, loss = 1.34064735
Iteration 136, loss = 1.34067901
Iteration 137, loss = 1.34052816
Iteration 138, loss = 1.34047595
Iteration 139, loss = 1.34022566
Iteration 140, loss = 1.34035068
Iteration 141, loss = 1.34006524
Iteration 142, loss = 1.34015107
Iteration 143, loss = 1.34002622
Iteration 144, loss = 1.33980957
Iteration 145, loss = 1.33978189
Iteration 146, loss = 1.33974959
Iteration 147, loss = 1.33956550
Iteration 148, loss = 1.33959023
Iteration 149, loss = 1.33943153
Iteration 150, loss = 1.33934654
Iteration 151, loss = 1.33923359
Iteration 152, loss = 1.33923656
Iteration 153, loss = 1.33907638
Iteration 154, loss = 1.33897517
Iteration 155, loss = 1.33882380
Iteration 156, loss = 1.33879743
Iteration 157, loss = 1.33873184
Iteration 158, loss = 1.33852695
Iteration 159, loss = 1.33854091
Iteration 160, loss = 1.33844815
Iteration 161, loss = 1.33841766
Iteration 162, loss = 1.33822323
Iteration 163, loss = 1.33813612
Iteration 164, loss = 1.33808318
Iteration 165, loss = 1.33807003
Iteration 166, loss = 1.33790580
Iteration 167, loss = 1.33785206
Iteration 168, loss = 1.33775579
Iteration 169, loss = 1.33763462
Iteration 170, loss = 1.33757735
Iteration 171, loss = 1.33748661
Iteration 172, loss = 1.33754250
Iteration 173, loss = 1.33731174
Iteration 174, loss = 1.33731091
Iteration 175, loss = 1.33705106
Iteration 176, loss = 1.33725832
Iteration 177, loss = 1.33692821
Iteration 178, loss = 1.33708370
Iteration 179, loss = 1.33692998
Iteration 180, loss = 1.33672309
Iteration 181, loss = 1.33667763
Iteration 182, loss = 1.33663205
Iteration 183, loss = 1.33659152
Iteration 184, loss = 1.33656869
Iteration 185, loss = 1.33644886
Iteration 186, loss = 1.33631704
Iteration 187, loss = 1.33624661
Iteration 188, loss = 1.33621864
Iteration 189, loss = 1.33608505
Iteration 190, loss = 1.33600338
Iteration 191, loss = 1.33588587
Iteration 192, loss = 1.33582098
Iteration 193, loss = 1.33584034
Iteration 194, loss = 1.33574212
Iteration 195, loss = 1.33565059
Iteration 196, loss = 1.33556211
Iteration 197, loss = 1.33560856
Iteration 198, loss = 1.33546104
Iteration 199, loss = 1.33536024
Iteration 200, loss = 1.33527121
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

```
accuracy on train data:  0.5267
metrics for train data:
              precision    recall  f1-score   support

           0       0.89      0.88      0.89      2008
           1       0.64      0.70      0.67      1806
           2       0.35      0.59      0.44      1168
           3       0.06      0.56      0.11       227
           4       0.68      0.73      0.70      1967
```

```
   micro avg       0.53      0.73      0.61      7176
   macro avg       0.53      0.69      0.56      7176
weighted avg       0.66      0.73      0.68      7176
 samples avg       0.53      0.53      0.53      7176

accuracy on test data:  0.521
metrics for test data:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       229
           1       0.62      0.69      0.65       178
           2       0.34      0.61      0.44       112
           3       0.06      0.50      0.11        22
           4       0.61      0.70      0.65       164

   micro avg       0.52      0.74      0.61       705
   macro avg       0.51      0.68      0.55       705
weighted avg       0.65      0.74      0.68       705
 samples avg       0.52      0.52      0.52       705



Hidden layer: [512, 256, 128]
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Iteration 1, loss = 1.76380936
Iteration 2, loss = 1.38822441
Iteration 3, loss = 1.37395561
Iteration 4, loss = 1.36949873
Iteration 5, loss = 1.36609622
Iteration 6, loss = 1.36384879
Iteration 7, loss = 1.36170042
Iteration 8, loss = 1.36104174
Iteration 9, loss = 1.35993561
Iteration 10, loss = 1.35869727
Iteration 11, loss = 1.35788140
Iteration 12, loss = 1.35715549
Iteration 13, loss = 1.35655835
Iteration 14, loss = 1.35555626
Iteration 15, loss = 1.35546035
Iteration 16, loss = 1.35481342
Iteration 17, loss = 1.35427231
Iteration 18, loss = 1.35355794
Iteration 19, loss = 1.35328877
Iteration 20, loss = 1.35279381
Iteration 21, loss = 1.35206872
Iteration 22, loss = 1.35187235
Iteration 23, loss = 1.35125906
Iteration 24, loss = 1.35109893
Iteration 25, loss = 1.35066431
Iteration 26, loss = 1.35031724
Iteration 27, loss = 1.34988212
Iteration 28, loss = 1.34967275
Iteration 29, loss = 1.34915685
Iteration 30, loss = 1.34871293
Iteration 31, loss = 1.34865636
Iteration 32, loss = 1.34782173
Iteration 33, loss = 1.34799764
Iteration 34, loss = 1.34749533
Iteration 35, loss = 1.34747387
Iteration 36, loss = 1.34707925
Iteration 37, loss = 1.34701910
Iteration 38, loss = 1.34669471
Iteration 39, loss = 1.34637387
Iteration 40, loss = 1.34622964
Iteration 41, loss = 1.34576911
Iteration 42, loss = 1.34561575
Iteration 43, loss = 1.34543691
Iteration 44, loss = 1.34494329
Iteration 45, loss = 1.34484796
Iteration 46, loss = 1.34442891
Iteration 47, loss = 1.34450047
Iteration 48, loss = 1.34425555
Iteration 49, loss = 1.34391610
Iteration 50, loss = 1.34371431
Iteration 51, loss = 1.34373249
Iteration 52, loss = 1.34320654
Iteration 53, loss = 1.34325544
Iteration 54, loss = 1.34297260
Iteration 55, loss = 1.34286864
Iteration 56, loss = 1.34254717
Iteration 57, loss = 1.34214021
Iteration 58, loss = 1.34238252
Iteration 59, loss = 1.34202606
Iteration 60, loss = 1.34194018
Iteration 61, loss = 1.34165714
Iteration 62, loss = 1.34165045
Iteration 63, loss = 1.34128502
Iteration 64, loss = 1.34113178
Iteration 65, loss = 1.34106945
Iteration 66, loss = 1.34079202
Iteration 67, loss = 1.34061909
Iteration 68, loss = 1.34038140
Iteration 69, loss = 1.34046859
Iteration 70, loss = 1.34008239
Iteration 71, loss = 1.33991804
Iteration 72, loss = 1.33994966
Iteration 73, loss = 1.33973863
Iteration 74, loss = 1.33955967
Iteration 75, loss = 1.33927927
Iteration 76, loss = 1.33915318
Iteration 77, loss = 1.33905393
Iteration 78, loss = 1.33881603
Iteration 79, loss = 1.33875910
Iteration 80, loss = 1.33869055
Iteration 81, loss = 1.33849121
Iteration 82, loss = 1.33841044
Iteration 83, loss = 1.33829843
Iteration 84, loss = 1.33780279
Iteration 85, loss = 1.33802149
Iteration 86, loss = 1.33781639
Iteration 87, loss = 1.33781684
Iteration 88, loss = 1.33736824
Iteration 89, loss = 1.33746816
Iteration 90, loss = 1.33722624
Iteration 91, loss = 1.33701026
Iteration 92, loss = 1.33701021
Iteration 93, loss = 1.33679836
Iteration 94, loss = 1.33670708
Iteration 95, loss = 1.33646210
Iteration 96, loss = 1.33652772
Iteration 97, loss = 1.33630714
Iteration 98, loss = 1.33620470
Iteration 99, loss = 1.33623657
Iteration 100, loss = 1.33598109
Iteration 101, loss = 1.33585048
Iteration 102, loss = 1.33569540
Iteration 103, loss = 1.33551924
Iteration 104, loss = 1.33556123
Iteration 105, loss = 1.33533158
Iteration 106, loss = 1.33514170
Iteration 107, loss = 1.33500388
Iteration 108, loss = 1.33476624
Iteration 109, loss = 1.33489067
Iteration 110, loss = 1.33468848
Iteration 111, loss = 1.33461397
Iteration 112, loss = 1.33423274
Iteration 113, loss = 1.33441666
Iteration 114, loss = 1.33423589
Iteration 115, loss = 1.33388573
Iteration 116, loss = 1.33426413
Iteration 117, loss = 1.33396289
Iteration 118, loss = 1.33379433
Iteration 119, loss = 1.33395570
Iteration 120, loss = 1.33367703
Iteration 121, loss = 1.33348642
Iteration 122, loss = 1.33339904
Iteration 123, loss = 1.33335125
Iteration 124, loss = 1.33330120
Iteration 125, loss = 1.33314045
Iteration 126, loss = 1.33295453
Iteration 127, loss = 1.33279385
Iteration 128, loss = 1.33279488
Iteration 129, loss = 1.33266766
Iteration 130, loss = 1.33263197
Iteration 131, loss = 1.33244764
Iteration 132, loss = 1.33238185
Iteration 133, loss = 1.33233309
Iteration 134, loss = 1.33211635
Iteration 135, loss = 1.33206129
Iteration 136, loss = 1.33191412
Iteration 137, loss = 1.33193500
Iteration 138, loss = 1.33174531
Iteration 139, loss = 1.33164275
Iteration 140, loss = 1.33157977
Iteration 141, loss = 1.33158093
Iteration 142, loss = 1.33150074
Iteration 143, loss = 1.33117412
Iteration 144, loss = 1.33116357
Iteration 145, loss = 1.33111280
Iteration 146, loss = 1.33101557
Iteration 147, loss = 1.33097566
Iteration 148, loss = 1.33084151
Iteration 149, loss = 1.33060484
Iteration 150, loss = 1.33070459
Iteration 151, loss = 1.33060298
Iteration 152, loss = 1.33041688
Iteration 153, loss = 1.33034815
Iteration 154, loss = 1.33019880
Iteration 155, loss = 1.33029594
Iteration 156, loss = 1.33010516
Iteration 157, loss = 1.32992872
Iteration 158, loss = 1.32984731
Iteration 159, loss = 1.32972096
Iteration 160, loss = 1.32968075
Iteration 161, loss = 1.32959529
Iteration 162, loss = 1.32949004
Iteration 163, loss = 1.32923991
Iteration 164, loss = 1.32951352
Iteration 165, loss = 1.32934075
Iteration 166, loss = 1.32920078
Iteration 167, loss = 1.32911943
Iteration 168, loss = 1.32898543
Iteration 169, loss = 1.32894605
Iteration 170, loss = 1.32869095
Iteration 171, loss = 1.32877422
Iteration 172, loss = 1.32867062
Iteration 173, loss = 1.32861083
Iteration 174, loss = 1.32855360
Iteration 175, loss = 1.32839278
Iteration 176, loss = 1.32834345
Iteration 177, loss = 1.32824340
Iteration 178, loss = 1.32808126
Iteration 179, loss = 1.32813627
Iteration 180, loss = 1.32815307
Iteration 181, loss = 1.32796064
Iteration 182, loss = 1.32788895
Iteration 183, loss = 1.32779912
Iteration 184, loss = 1.32772729
Iteration 185, loss = 1.32754699
Iteration 186, loss = 1.32747568
Iteration 187, loss = 1.32727646
Iteration 188, loss = 1.32744426
Iteration 189, loss = 1.32735706
Iteration 190, loss = 1.32732263
Iteration 191, loss = 1.32716804
Iteration 192, loss = 1.32695542
Iteration 193, loss = 1.32709863
Iteration 194, loss = 1.32680616
Iteration 195, loss = 1.32680788
Iteration 196, loss = 1.32691740
Iteration 197, loss = 1.32686565
Iteration 198, loss = 1.32667633
Iteration 199, loss = 1.32653160
Iteration 200, loss = 1.32654795
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
accuracy on train data:  0.5642
metrics for train data:
              precision    recall  f1-score   support

           0       0.89      0.88      0.88      1985
           1       0.67      0.70      0.68      1889
           2       0.42      0.59      0.49      1384
           3       0.14      0.54      0.23       531
           4       0.71      0.71      0.71      2067

   micro avg       0.56      0.72      0.63      7856
   macro avg       0.56      0.68      0.60      7856
weighted avg       0.65      0.72      0.68      7856
 samples avg       0.56      0.56      0.56      7856

accuracy on test data:  0.562
metrics for test data:
              precision    recall  f1-score   support

           0       0.90      0.91      0.90       225
           1       0.64      0.68      0.66       184
           2       0.40      0.61      0.48       130
           3       0.18      0.54      0.27        63
           4       0.63      0.68      0.66       173

   micro avg       0.56      0.73      0.63       775
   macro avg       0.55      0.69      0.59       775
weighted avg       0.63      0.73      0.67       775
 samples avg       0.56      0.56      0.56       775
```

```
Hidden layer: [512, 256, 128, 64]
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
Iteration 1, loss = 1.81368436
Iteration 2, loss = 1.44544842
Iteration 3, loss = 1.38917601
Iteration 4, loss = 1.38138721
Iteration 5, loss = 1.37689060
Iteration 6, loss = 1.37331294
Iteration 7, loss = 1.37047256
Iteration 8, loss = 1.36867094
Iteration 9, loss = 1.36701830
Iteration 10, loss = 1.36629762
Iteration 11, loss = 1.36428958
Iteration 12, loss = 1.36411254
Iteration 13, loss = 1.36277406
Iteration 14, loss = 1.36201276
Iteration 15, loss = 1.36122268
Iteration 16, loss = 1.36026491
Iteration 17, loss = 1.35992904
Iteration 18, loss = 1.35911954
Iteration 19, loss = 1.35855620
Iteration 20, loss = 1.35809020
Iteration 21, loss = 1.35724215
Iteration 22, loss = 1.35677803
Iteration 23, loss = 1.35652282
Iteration 24, loss = 1.35581300
Iteration 25, loss = 1.35539375
Iteration 26, loss = 1.35481140
Iteration 27, loss = 1.35403886
Iteration 28, loss = 1.35396309
Iteration 29, loss = 1.35331822
Iteration 30, loss = 1.35319095
Iteration 31, loss = 1.35287776
Iteration 32, loss = 1.35243008
Iteration 33, loss = 1.35148415
Iteration 34, loss = 1.35169121
Iteration 35, loss = 1.35138144
Iteration 36, loss = 1.35108106
Iteration 37, loss = 1.35077123
Iteration 38, loss = 1.35049022
Iteration 39, loss = 1.35019795
Iteration 40, loss = 1.34961928
Iteration 41, loss = 1.34928929
Iteration 42, loss = 1.34920142
Iteration 43, loss = 1.34881146
Iteration 44, loss = 1.34820406
Iteration 45, loss = 1.34864597
Iteration 46, loss = 1.34793955
Iteration 47, loss = 1.34757309
Iteration 48, loss = 1.34729816
Iteration 49, loss = 1.34697372
Iteration 50, loss = 1.34694880
Iteration 51, loss = 1.34656289
Iteration 52, loss = 1.34625051
Iteration 53, loss = 1.34602164
Iteration 54, loss = 1.34617614
Iteration 55, loss = 1.34566605
Iteration 56, loss = 1.34559899
Iteration 57, loss = 1.34530104
Iteration 58, loss = 1.34506928
Iteration 59, loss = 1.34490077
Iteration 60, loss = 1.34454417
Iteration 61, loss = 1.34442064
Iteration 62, loss = 1.34423867
Iteration 63, loss = 1.34389345
Iteration 64, loss = 1.34411251
Iteration 65, loss = 1.34349359
Iteration 66, loss = 1.34326719
Iteration 67, loss = 1.34327519
Iteration 68, loss = 1.34291595
Iteration 69, loss = 1.34269664
Iteration 70, loss = 1.34257381
Iteration 71, loss = 1.34238822
Iteration 72, loss = 1.34225701
Iteration 73, loss = 1.34217715
Iteration 74, loss = 1.34199308
Iteration 75, loss = 1.34156418
Iteration 76, loss = 1.34145023
Iteration 77, loss = 1.34130128
Iteration 78, loss = 1.34107888
Iteration 79, loss = 1.34097266
Iteration 80, loss = 1.34051020
Iteration 81, loss = 1.34031797
Iteration 82, loss = 1.34050280
Iteration 83, loss = 1.34022977
Iteration 84, loss = 1.34014982
Iteration 85, loss = 1.33990893
Iteration 86, loss = 1.33985747
Iteration 87, loss = 1.33962806
Iteration 88, loss = 1.33948541
Iteration 89, loss = 1.33930273
Iteration 90, loss = 1.33923741
Iteration 91, loss = 1.33903170
Iteration 92, loss = 1.33890641
Iteration 93, loss = 1.33867310
Iteration 94, loss = 1.33863891
Iteration 95, loss = 1.33824019
Iteration 96, loss = 1.33817394
Iteration 97, loss = 1.33787108
Iteration 98, loss = 1.33795980
Iteration 99, loss = 1.33778427
Iteration 100, loss = 1.33762673
Iteration 101, loss = 1.33748697
Iteration 102, loss = 1.33726676
Iteration 103, loss = 1.33727729
Iteration 104, loss = 1.33722797
Iteration 105, loss = 1.33697452
Iteration 106, loss = 1.33682430
Iteration 107, loss = 1.33675692
Iteration 108, loss = 1.33645577
Iteration 109, loss = 1.33642302
Iteration 110, loss = 1.33632492
Iteration 111, loss = 1.33604307
Iteration 112, loss = 1.33606265
Iteration 113, loss = 1.33586589
Iteration 114, loss = 1.33588622
Iteration 115, loss = 1.33560642
Iteration 116, loss = 1.33555894
Iteration 117, loss = 1.33544376
Iteration 118, loss = 1.33528809
Iteration 119, loss = 1.33515806
Iteration 120, loss = 1.33508599
Iteration 121, loss = 1.33489823
Iteration 122, loss = 1.33480901
Iteration 123, loss = 1.33486307
Iteration 124, loss = 1.33455791
Iteration 125, loss = 1.33450993
Iteration 126, loss = 1.33431833
Iteration 127, loss = 1.33432848
Iteration 128, loss = 1.33407660
Iteration 129, loss = 1.33384337
Iteration 130, loss = 1.33399172
Iteration 131, loss = 1.33385179
Iteration 132, loss = 1.33338531
Iteration 133, loss = 1.33348674
Iteration 134, loss = 1.33337484
Iteration 135, loss = 1.33334390
Iteration 136, loss = 1.33319013
Iteration 137, loss = 1.33311663
Iteration 138, loss = 1.33289270
Iteration 139, loss = 1.33284942
Iteration 140, loss = 1.33269074
Iteration 141, loss = 1.33260482
Iteration 142, loss = 1.33249308
Iteration 143, loss = 1.33238653
Iteration 144, loss = 1.33241869
Iteration 145, loss = 1.33235419
Iteration 146, loss = 1.33195410
Iteration 147, loss = 1.33200446
Iteration 148, loss = 1.33184600
Iteration 149, loss = 1.33173865
Iteration 150, loss = 1.33163852
Iteration 151, loss = 1.33162311
Iteration 152, loss = 1.33157170
Iteration 153, loss = 1.33124986
Iteration 154, loss = 1.33114563
Iteration 155, loss = 1.33117362
Iteration 156, loss = 1.33111677
Iteration 157, loss = 1.33091936
Iteration 158, loss = 1.33059599
Iteration 159, loss = 1.33069087
Iteration 160, loss = 1.33072471
Iteration 161, loss = 1.33060597
Iteration 162, loss = 1.33048843
Iteration 163, loss = 1.33034758
Iteration 164, loss = 1.33026873
Iteration 165, loss = 1.32985650
Iteration 166, loss = 1.33006810
Iteration 167, loss = 1.33002381
Iteration 168, loss = 1.33001853
Iteration 169, loss = 1.32981868
Iteration 170, loss = 1.32967690
Iteration 171, loss = 1.32967191
Iteration 172, loss = 1.32964922
Iteration 173, loss = 1.32915260
Iteration 174, loss = 1.32933370
Iteration 175, loss = 1.32919457
Iteration 176, loss = 1.32925131
Iteration 177, loss = 1.32907655
Iteration 178, loss = 1.32889414
Iteration 179, loss = 1.32900540
Iteration 180, loss = 1.32868908
Iteration 181, loss = 1.32884859
Iteration 182, loss = 1.32870995
Iteration 183, loss = 1.32838215
Iteration 184, loss = 1.32848429
Iteration 185, loss = 1.32849773
Iteration 186, loss = 1.32830025
Iteration 187, loss = 1.32789618
Iteration 188, loss = 1.32815089
Iteration 189, loss = 1.32800030
Iteration 190, loss = 1.32774316
Iteration 191, loss = 1.32808793
Iteration 192, loss = 1.32764538
Iteration 193, loss = 1.32781256
Iteration 194, loss = 1.32763628
Iteration 195, loss = 1.32751946
Iteration 196, loss = 1.32739331
Iteration 197, loss = 1.32729238
Iteration 198, loss = 1.32733673
Iteration 199, loss = 1.32719312
Iteration 200, loss = 1.32718437
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/neural_network/_multilayer_perceptron.py:691:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

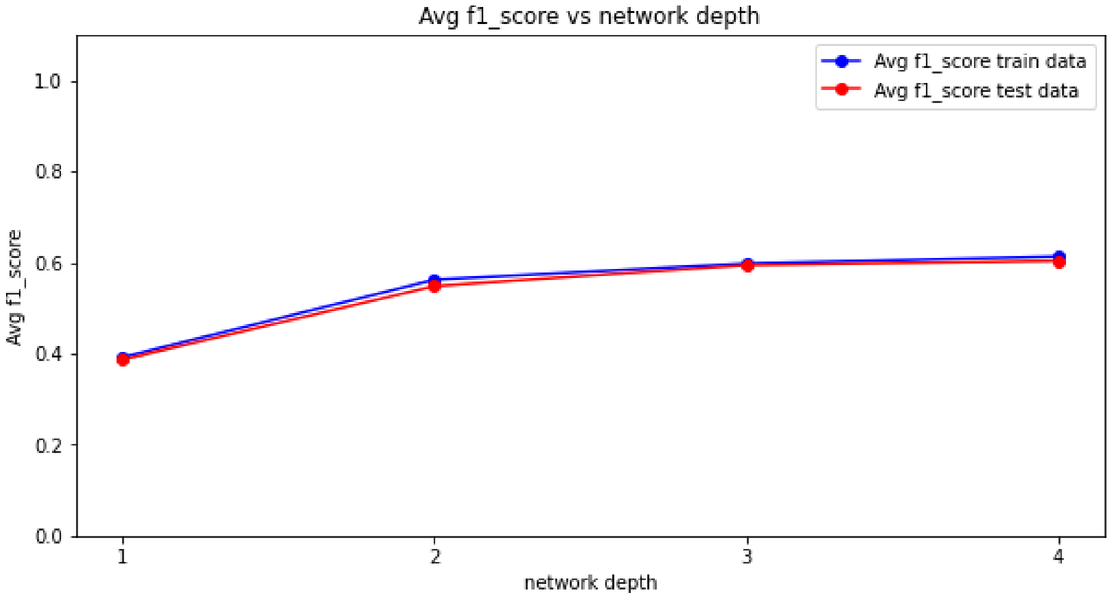
```
accuracy on train data:  0.5846
metrics for train data:
              precision    recall  f1-score   support

           0       0.89      0.88      0.88      1997
           1       0.68      0.69      0.69      1958
           2       0.46      0.58      0.51      1568
           3       0.18      0.53      0.27       688
           4       0.71      0.71      0.71      2064

   micro avg       0.58      0.71      0.64      8275
   macro avg       0.58      0.68      0.61      8275
weighted avg       0.66      0.71      0.67      8275
 samples avg       0.58      0.58      0.58      8275

accuracy on test data:  0.581
metrics for test data:
              precision    recall  f1-score   support

           0       0.90      0.90      0.90       228
           1       0.66      0.68      0.67       194
           2       0.44      0.59      0.51       149
           3       0.20      0.45      0.28        85
           4       0.64      0.69      0.66       172

   micro avg       0.58      0.70      0.64       828
   macro avg       0.57      0.66      0.60       828
weighted avg       0.64      0.70      0.66       828
 samples avg       0.58      0.58      0.58       828
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

```
/home/tkarthikeyan/.local/lib/python3.8/site-
packages/sklearn/metrics/_classification.py:1469: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[20]:

```
#Plot avg f1_scores for different depth
plt.figure(figsize=(10,5))
plt.plot(network_depth, f1_score_train, marker='o', markersize=6, color='blue',␣
 ↪label='Avg f1_score train data')
plt.plot(network_depth, f1_score_test, marker='o', markersize=6, color='red',␣
 ↪label='Avg f1_score test data')

plt.title('Avg f1_score vs network depth')
plt.xlabel('network depth')
plt.ylabel('Avg f1_score')
plt.xticks(network_depth)
plt.ylim(0,1.1)
plt.legend()
plt.show()
```



[ ]: