MSSV: B2017028
Họ và tên: Lê Hoàng Dũng

Bài 3
import thư viện và viết các hàm predict, loss

```
import tensorflow as tf

@tf.function
def layer1(X, W, B):
  return tf.nn.relu(X@W + B)

def layer2(X, W, B):
  return tf.nn.softmax(X@W + B)


@tf.function
def predict(X, W1, B1, W2, B2):
  return layer2(layer1(X, W1, B1), W2, B2)

@tf.function
def L(y, y_hat):
  return - tf.reduce_mean(tf.reduce_sum(y * tf.math.log(y_hat + 0.00001), axis=1))
```

Khởi tạo tham số và training

```python
X = tf.constant([
    [0.0, 0],
    [0, 1],
    [1, 0],
    [1, 1]
])
y = tf.constant([
    [1.0, 0, 0],
    [0, 1, 0],
    [0, 1, 0],
    [0, 0, 1]
])

n = 2
W1 = tf.Variable(tf.random.normal(shape=(n, 2)))
B1 = tf.Variable(tf.random.normal(shape=(2,)))
W2 = tf.Variable(tf.random.normal(shape=(2, 3)))
B2 = tf.Variable(tf.random.normal(shape=(3,)))

alpha = 0.1
epochs = 1000

for i in range(epochs):
  with tf.GradientTape() as t:
    current_loss = L(y, predict(X, W1, B1, W2, B2))
  print("epoch ", i, " ", current_loss)

  dw1, db1, dw2, db2 = t.gradient(current_loss, [W1, B1, W2, B2])
  W1.assign_sub(alpha*dw1)
  B1.assign_sub(alpha*db1)
  W2.assign_sub(alpha*dw2)
  B2.assign_sub(alpha*db2)
```

kết quả khi sau khi traning:

```
epoch  961   tf.Tensor(0.07921183, shape=(), dtype=float32)
epoch  962   tf.Tensor(0.079105884, shape=(), dtype=float32
epoch  963   tf.Tensor(0.07900016, shape=(), dtype=float32)
epoch  964   tf.Tensor(0.078894645, shape=(), dtype=float32
epoch  965   tf.Tensor(0.078789435, shape=(), dtype=float32
epoch  966   tf.Tensor(0.078684375, shape=(), dtype=float32
epoch  967   tf.Tensor(0.07857971, shape=(), dtype=float32)
epoch  968   tf.Tensor(0.0784752, shape=(), dtype=float32)
epoch  969   tf.Tensor(0.07837099, shape=(), dtype=float32)
epoch  970   tf.Tensor(0.07826699, shape=(), dtype=float32)
epoch  971   tf.Tensor(0.0781632, shape=(), dtype=float32)
epoch  972   tf.Tensor(0.078059725, shape=(), dtype=float32
epoch  973   tf.Tensor(0.077956446, shape=(), dtype=float32
epoch  974   tf.Tensor(0.07785349, shape=(), dtype=float32)
epoch  975   tf.Tensor(0.077750765, shape=(), dtype=float32
epoch  976   tf.Tensor(0.0776482, shape=(), dtype=float32)
epoch  977   tf.Tensor(0.077545926, shape=(), dtype=float32
epoch  978   tf.Tensor(0.07744389, shape=(), dtype=float32)
epoch  979   tf.Tensor(0.07734206, shape=(), dtype=float32)
epoch  980   tf.Tensor(0.0772405, shape=(), dtype=float32)
epoch  981   tf.Tensor(0.07713921, shape=(), dtype=float32)
epoch  982   tf.Tensor(0.077038094, shape=(), dtype=float32
epoch  983   tf.Tensor(0.076937236, shape=(), dtype=float32
epoch  984   tf.Tensor(0.076836646, shape=(), dtype=float32
epoch  985   tf.Tensor(0.076736234, shape=(), dtype=float32
epoch  986   tf.Tensor(0.07663607, shape=(), dtype=float32)
epoch  987   tf.Tensor(0.076536186, shape=(), dtype=float32
epoch  988   tf.Tensor(0.07643655, shape=(), dtype=float32)
epoch  989   tf.Tensor(0.0763371, shape=(), dtype=float32)
epoch  990   tf.Tensor(0.07623788, shape=(), dtype=float32)
epoch  991   tf.Tensor(0.07613893, shape=(), dtype=float32)
epoch  992   tf.Tensor(0.076040104, shape=(), dtype=float32
epoch  993   tf.Tensor(0.07594164, shape=(), dtype=float32)
epoch  994   tf.Tensor(0.075843275, shape=(), dtype=float32
epoch  995   tf.Tensor(0.075745285, shape=(), dtype=float32
epoch  996   tf.Tensor(0.0756474, shape=(), dtype=float32)
epoch  997   tf.Tensor(0.07554981, shape=(), dtype=float32)
epoch  998   tf.Tensor(0.07545238, shape=(), dtype=float32)
epoch  999   tf.Tensor(0.07535525, shape=(), dtype=float32)
```

Bài 4:

import thư viện, dọc tập dữ liệu và viets các hàm cần thiết

```python
import tensorflow as tf
from sklearn.preprocessing import LabelEncoder
import pandas as pd
from sklearn.model_selection import train_test_split
import numpy as np

df = pd.read_csv('/content/data.csv')

X = df.iloc[:, :-1]
y = df.iloc[:, -1:]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

X_train = tf.cast(X_train, dtype=tf.float32)
X_test = tf.cast(X_test, dtype=tf.float32)
y_train = tf.cast(y_train, dtype=tf.float32)
y_test = tf.cast(y_test, dtype=tf.float32)

@tf.function
def layer1(X, W, B):
  return tf.nn.relu(X@W + B)

def layer2(X, W, B):
  return tf.nn.sigmoid(X@W + B)


@tf.function
def predict(X, W1, B1, W2, B2):
  return layer2(layer1(X, W1, B1), W2, B2)

@tf.function
def L(y, y_hat):
  return -1 * tf.reduce_mean(((y * tf.math.log(y_hat)) + (1-y)*tf.math.log(1-y_hat)))
```

Khởi tạo tham só và tranining:

```python
n = 2
W1 = tf.Variable(tf.random.normal(shape=(n, 2)))
B1 = tf.Variable(tf.random.normal(shape=(2,)))
W2 = tf.Variable(tf.random.normal(shape=(2, 1)))
B2 = tf.Variable(tf.random.normal(shape=(1,)))

alpha = 0.1
epochs = 2000

for i in range(epochs):
  with tf.GradientTape() as t:
    current_loss = L(y_train, predict(X_train, W1, B1, W2, B2))
  print("epoch ", i, " ", current_loss)

  dw1, db1, dw2, db2 = t.gradient(current_loss, [W1, B1, W2, B2])
  W1.assign_sub(alpha*dw1)
  B1.assign_sub(alpha*db1)
  W2.assign_sub(alpha*dw2)
  B2.assign_sub(alpha*db2)
```

```
epoch  1975   tf.Tensor(0.6552928, shape=(), dtype=float32)
epoch  1976   tf.Tensor(0.6552917, shape=(), dtype=float32)
epoch  1977   tf.Tensor(0.65529054, shape=(), dtype=float32)
epoch  1978   tf.Tensor(0.6552894, shape=(), dtype=float32)
epoch  1979   tf.Tensor(0.6552882, shape=(), dtype=float32)
epoch  1980   tf.Tensor(0.655287, shape=(), dtype=float32)
epoch  1981   tf.Tensor(0.65528595, shape=(), dtype=float32)
epoch  1982   tf.Tensor(0.6552848, shape=(), dtype=float32)
epoch  1983   tf.Tensor(0.65528375, shape=(), dtype=float32)
epoch  1984   tf.Tensor(0.6552826, shape=(), dtype=float32)
epoch  1985   tf.Tensor(0.6552815, shape=(), dtype=float32)
epoch  1986   tf.Tensor(0.6552804, shape=(), dtype=float32)
epoch  1987   tf.Tensor(0.6552793, shape=(), dtype=float32)
epoch  1988   tf.Tensor(0.65527815, shape=(), dtype=float32)
epoch  1989   tf.Tensor(0.655277, shape=(), dtype=float32)
epoch  1990   tf.Tensor(0.655276, shape=(), dtype=float32)
epoch  1991   tf.Tensor(0.6552749, shape=(), dtype=float32)
epoch  1992   tf.Tensor(0.65527374, shape=(), dtype=float32)
epoch  1993   tf.Tensor(0.65527266, shape=(), dtype=float32)
epoch  1994   tf.Tensor(0.65527165, shape=(), dtype=float32)
epoch  1995   tf.Tensor(0.6552706, shape=(), dtype=float32)
epoch  1996   tf.Tensor(0.65526944, shape=(), dtype=float32)
epoch  1997   tf.Tensor(0.65526843, shape=(), dtype=float32)
epoch  1998   tf.Tensor(0.65526736, shape=(), dtype=float32)
epoch  1999   tf.Tensor(0.65526634, shape=(), dtype=float32)
```

viets hàm dự đoán:

```
@tf.function
def predict_label(X, W1,B1,W2, B2, threshold=0.5):
  preds = predict(X, W1,B1,W2, B2)
  return tf.where(preds>=threshold, 1.0,0.0)
```

+ Code + Text

```
[ ] @tf.function
    def evaluate(y_pred, y_true):
      return (len(y_true) - tf.reduce_sum(tf.math.abs(tf.math.subtract(y_pred, y_true)))) / len(y_true)
```

```
[ ] y_preds = predict(X_test, W1, B1, W2, B2)
    y_label = predict_label(X_test, W1, B1, W2, B2)
    print(evaluate(y_label, y_test).numpy())

    0.8

[ ]
```

Bài 5

import thư viên và đọc tập dữ liệu:

```
[ ] import scipy.io
    from sklearn.model_selection import train_test_split
    import tensorflow as tf
    import numpy as np
    from sklearn.preprocessing import OneHotEncoder
    from sklearn.metrics import accuracy_score
```

Follow link (ctrl + click)

```
!wget https://archive.ics.uci.edu/ml/machine-learning-databases/character-trajectories/mixoutALL_shifted.mat
mat_data = scipy.io.loadmat('mixoutALL_shifted.mat')
constants = mat_data['consts'][0][0]
constants_charlabels = constants[4][0] - 1
mixout = mat_data['mixout'][0]
```

```
--2023-09-06 02:41:55--  https://archive.ics.uci.edu/ml/machine-learning-databases/character-trajectories/mixoutALL_shifted.mat
Resolving archive.ics.uci.edu (archive.ics.uci.edu)... 128.195.10.252
Connecting to archive.ics.uci.edu (archive.ics.uci.edu)|128.195.10.252|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified
Saving to: ‘mixoutALL_shifted.mat.5’

mixoutALL_shifted.m     [ <=>                ]   7.54M  43.0MB/s    in 0.2s

2023-09-06 02:41:56 (43.0 MB/s) - ‘mixoutALL_shifted.mat.5’ saved [7912437]
```

xử lí tập dữ liệu

```
def load_dataset():
  global mixout
  maximum_sample = max([len(sample[0]) for sample in mixout])
  # building the data dictionary
  dict1=[]
  for i,samp in enumerate(mixout):
    dict2=[]
    for j,channel in enumerate(samp):
        dict3=[]
        for k in range(maximum_sample):
            if k < len(channel):
                dict3.append(channel[k])
            else:
                dict3.append(0)
        dict2.append(dict3)
    dict1.append(dict2)
  mixout = np.array(dict1)
  inputs = mixout
  labels = constants_charlabels
  return inputs, labels
```

```
[ ] X, y = load_dataset()
```

```
[ ] X.shape
```

```
(2858, 3, 205)
```

```
[ ] y.shape
```

```
(2858,)
```

```
[ ] X = np.reshape(X, (X.shape[0], -1))
```

```
X.shape
```

```
(2858, 615)
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

```
[ ] enc = OneHotEncoder(sparse=False)
    enc.fit(y)
    y_train = enc.transform(y_train)
    y_test = enc.transform(y_test)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/preprocessing/_encoders.py:868: FutureWarning: `s
  warnings.warn(
```

```
[ ] X_train = tf.cast(X_train, dtype=tf.float32)
    X_test = tf.cast(X_test, dtype=tf.float32)
    y_train = tf.cast(y_train, dtype=tf.float32)
    y_test = tf.cast(y_test, dtype=tf.float32)
```

```
[ ] X_train.shape
```

```
TensorShape([2286, 615])
```

thử các kiến trúc và training

```
@tf.function
def layer1(X, W, B):
  return tf.nn.relu(X@W + B)

def layer2(X, W, B):
  return tf.nn.softmax(X@W + B)


@tf.function
def predict(X, W1, B1, W2, B2, W3, B3, W4, B4):
  return layer2(layer1(layer1(layer1(X, W1, B1), W2, B2), W3, B3), W4, B4)

# @tf.function
# def predict(X, W1, B1, W2, B2, W3, B3, W4, B4, W5, B5, W6, B6, W7, B7, W8, B8, W9, B9, W10, B10):
#   return layer2(layer1(layer1(layer1(layer1(layer1(layer1(layer1(layer1(X, W1, B1), W2, B2), W3, B3), W4, B4), W5, B5), W6, B6), W7, B7), W8, B8), W9, B9), W10, B10)

@tf.function
def L(y, y_hat):
  return - tf.reduce_mean(tf.reduce_sum(y * tf.math.log(y_hat + 0.00001), axis=1))

n = 615
W1 = tf.Variable(tf.random.normal(shape=(n, 250)))
B1 = tf.Variable(tf.random.normal(shape=(250,)))
W2 = tf.Variable(tf.random.normal(shape=(250, 200)))
B2 = tf.Variable(tf.random.normal(shape=(200,)))
W3 = tf.Variable(tf.random.normal(shape=(200, 66)))
B3 = tf.Variable(tf.random.normal(shape=(66,)))
W4 = tf.Variable(tf.random.normal(shape=(66, 20)))
B4 = tf.Variable(tf.random.normal(shape=(20,)))

# n = 615
# W1 = tf.Variable(tf.random.normal(shape=(n, 50)))
# B1 = tf.Variable(tf.random.normal(shape=(50,)))
# W2 = tf.Variable(tf.random.normal(shape=(50, 50)))
# B2 = tf.Variable(tf.random.normal(shape=(50,)))
# W3 = tf.Variable(tf.random.normal(shape=(50, 40)))
# B3 = tf.Variable(tf.random.normal(shape=(40,)))
# W4 = tf.Variable(tf.random.normal(shape=(40, 40)))
# B4 = tf.Variable(tf.random.normal(shape=(40,)))
# W5 = tf.Variable(tf.random.normal(shape=(40, 40)))
```

```
B4 = tf.Variable(tf.random.normal(shape=(20,)))

# n = 615
# W1 = tf.Variable(tf.random.normal(shape=(n, 50)))
# B1 = tf.Variable(tf.random.normal(shape=(50,)))
# W2 = tf.Variable(tf.random.normal(shape=(50, 50)))
# B2 = tf.Variable(tf.random.normal(shape=(50,)))
# W3 = tf.Variable(tf.random.normal(shape=(50, 40)))
# B3 = tf.Variable(tf.random.normal(shape=(40,)))
# W4 = tf.Variable(tf.random.normal(shape=(40, 40)))
# B4 = tf.Variable(tf.random.normal(shape=(40,)))
# W5 = tf.Variable(tf.random.normal(shape=(40, 40)))
# B5 = tf.Variable(tf.random.normal(shape=(40,)))
# W6 = tf.Variable(tf.random.normal(shape=(40, 30)))
# B6 = tf.Variable(tf.random.normal(shape=(30,)))
# W7 = tf.Variable(tf.random.normal(shape=(30, 30)))
# B7 = tf.Variable(tf.random.normal(shape=(30,)))
# W8 = tf.Variable(tf.random.normal(shape=(30, 30)))
# B8 = tf.Variable(tf.random.normal(shape=(30,)))
# W9 = tf.Variable(tf.random.normal(shape=(30, 30)))
# B9 = tf.Variable(tf.random.normal(shape=(30,)))
# W10 = tf.Variable(tf.random.normal(shape=(30, 20)))
# B10 = tf.Variable(tf.random.normal(shape=(20,)))


alpha = 0.01
epochs = 500

for i in range(epochs):
  with tf.GradientTape() as t:
    current_loss = L(y_train, predict(X_train, W1, B1, W2, B2, W3, B3, W4, B4))
  print("epoch ", i, " ", current_loss)

# for i in range(epochs):
#   with tf.GradientTape() as t:
#     current_loss = L(y_train, predict(X_train, W1, B1, W2, B2, W3, B3, W4, B4, W5, B5, W6, B6, W7, B7, W8, B8, W9, B9, W10, B10))
#   print("epoch ", i, " ", current_loss)

  dw1, db1, dw2, db2, dw3, db3, dw4, db4 = t.gradient(current_loss, [W1, B1, W2, B2, W3, B3, W4, B4])
  W1.assign_sub(alpha*dw1)
```

```
# W9 = tf.Variable(tf.random.normal(shape=(30, 30)))
# B9 = tf.Variable(tf.random.normal(shape=(30,)))
# W10 = tf.Variable(tf.random.normal(shape=(30, 20)))
# B10 = tf.Variable(tf.random.normal(shape=(20,)))


alpha = 0.01
epochs = 500

for i in range(epochs):
  with tf.GradientTape() as t:
    current_loss = L(y_train, predict(X_train, W1, B1, W2, B2, W3, B3, W4, B4))
  print("epoch ", i, " ", current_loss)

# for i in range(epochs):
#   with tf.GradientTape() as t:
#     current_loss = L(y_train, predict(X_train, W1, B1, W2, B2, W3, B3, W4, B4, W5, B5, W6, B6, W7, B7, W8, B8, W9, B9, W10, B10))
#   print("epoch ", i, " ", current_loss)

  dw1, db1, dw2, db2, dw3, db3, dw4, db4 = t.gradient(current_loss, [W1, B1, W2, B2, W3, B3, W4, B4])
  W1.assign_sub(alpha*dw1)
  B1.assign_sub(alpha*db1)
  W2.assign_sub(alpha*dw2)
  B2.assign_sub(alpha*db2)
  W3.assign_sub(alpha*dw3)
  B3.assign_sub(alpha*db3)
  W4.assign_sub(alpha*dw4)
  B4.assign_sub(alpha*db4)
  # W5.assign_sub(alpha*dw5)
  # B5.assign_sub(alpha*db5)

  # dw1, db1, dw2, db2, dw3, db3, dw4, db4, dw5, db5, dw6, db6, dw7, db7, dw8, db8, dw9,db9, dw10, db10 = t.gradient(current_loss, [W1, B1, W2, B2, W3, B3, W4, B4, W5, B5, W6,
  # W1.assign_sub(alpha*dw1)
  # B1.assign_sub(alpha*db1)
  # W2.assign_sub(alpha*dw2)
  # B2.assign_sub(alpha*db2)
  # W3.assign_sub(alpha*dw3)
  # B3.assign_sub(alpha*db3)
  # W4.assign_sub(alpha*dw4)
  # B4.assign_sub(alpha*db4)
```

```
epoch  105   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  106   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  107   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  108   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  109   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  110   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  111   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  112   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  113   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  114   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  115   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  116   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  117   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  118   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  119   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  120   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  121   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  122   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  123   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  124   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  125   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  126   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  127   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  128   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  129   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  130   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  131   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  132   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  133   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  134   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  135   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  136   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  137   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  138   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  139   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  140   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  141   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  142   tf.Tensor(10.802819, shape=(), dtype=float32)
epoch  143   tf.Tensor(10.802819, shape=(), dtype=float32)
```

```
[ ] @tf.function
    def predict_label(X, W1,B1, W2, B2, W3, B3, W4, B4):
```

hàm dự đoán:

```
@tf.function
def predict_label(X, W1,B1, W2, B2, W3, B3, W4, B4):
  preds = predict(X, W1,B1, W2, B2, W3, B3, W4, B4)
  return tf.argmax(preds, axis=1)
```

```
[ ] y_test_enc = tf.argmax(y_test, axis=1)
```

```
[ ] y_preds = predict_label(X_test, W1, B1, W2, B2, W3, B3, W4, B4)
```

```
[ ] print(y_test_enc.shape, y_preds.shape)
```

```
(572,) (572,)
```