proj1.py

```
import tensorflow as tf
import numpy as np

# set the random seeds to make sure your results are reproducible
from numpy.random import seed
seed(1)
from tensorflow import set_random_seed
set_random_seed(1)

# specify path to training data and testing data

train_x_location = "x_train.csv"
train_y_location = "y_train.csv"
test_x_location = "x_test.csv"
test_y_location = "y_test.csv"

print("Reading training data")
x_train = np.loadtxt(train_x_location, dtype="uint8", delimiter=",")
y_train = np.loadtxt(train_y_location, dtype="uint8", delimiter=",")

m, n = x_train.shape # m training examples, each with n features
m_labels,  = y_train.shape # m2 examples, each with k labels
l_min = y_train.min()

assert m_labels == m, "x_train and y_train should have same length."
assert l_min == 0, "each label should be in the range 0 - k-1."
k = y_train.max()+1

print(m, "examples,", n, "features,", k, "categiries.")


# print("Pre processing x of training data")
# x_train = x_train / 1.0

# define the training model
model = tf.keras.models.Sequential([
    # input_shape should be specified in the first layer
    tf.keras.layers.Dense(5, activation=tf.keras.activations.relu,
                          input_shape=(n,)),
    # another layer
    tf.keras.layers.Dense(5, activation=tf.keras.activations.relu),
    # another layer with l2 regularization
    tf.keras.layers.Dense(5, activation=tf.nn.relu,
                          kernel_regularizer=tf.keras.regularizers.l2(0.001)),
    # dropouts layer
    tf.keras.layers.Dropout(0.2),
    # last layer is softmax
    tf.keras.layers.Dense(k, activation=tf.nn.softmax)
])
# loss='categorical_entropy' expects input to be one-hot encoded
# loss='sparse_categorical_entropy' expects input to be the category as a number
# options for optimizer: 'sgd' and 'adam'. sgd is stochastic gradient descent
```

```
53 model.compile(optimizer='adam',
54               loss='sparse_categorical_crossentropy',
55               metrics=['accuracy'])
56
57 print("train")
58 model.fit(x_train, y_train, epochs=500, batch_size=32)
59 # default batch size is 32
60
61
62 print("Reading␣testing␣data")
63 x_test = np.loadtxt(test_x_location, dtype="uint8", delimiter=",")
64 y_test = np.loadtxt(test_y_location, dtype="uint8", delimiter=",")
65
66 m_test, n_test = x_test.shape
67 m_test_labels,  = y_test.shape
68 l_min = y_train.min()
69
70 assert m_test_labels == m_test, "x_test␣and␣y_test␣should␣have␣same␣length."
71 assert n_test == n, "train␣and␣x_test␣should␣have␣same␣number␣of␣features."
72
73 print(m_test, "test␣examples.")
74
75 # print("Pre processing testing data")
76 # x_test = x_test / 1.0
77
78
79 print("evaluate")
80 model.evaluate(x_test, y_test)
```