linear_train_test.py

```
import tensorflow as tf
import numpy as np

# specify path to training data and testing data

train_x_location = "dataset_x_train.csv"
train_y_location = "dataset_y_train.csv"
test_x_location = "dataset_x_train.csv"
test_y_location = "dataset_y_train.csv"

print("Reading␣training␣data")
# each instance is stored as a row of 2 values. entire training is a 2d matrix
x_train_1d = np.loadtxt(train_x_location, dtype="uint8", delimiter=",")
print(x_train_1d.shape)
x_train_3d = x_train_1d.reshape(-1,2,1)
print(x_train_3d.shape)
y_train = np.loadtxt(train_y_location, dtype="uint8", delimiter=",")

# define the training model
model = tf.keras.models.Sequential([
    # input_shape should be specified in the first layer
    tf.keras.layers.Dense(1,
                          activation=tf.keras.activations.linear,
                          use_bias=False,
                          input_shape=(2,1)),
])

# options for optimizer: 'sgd' and 'adam'. sgd is stochastic gradient descent
# loss='mean_squared_error'
model.compile(optimizer='sgd',
              loss='mean_squared_error',
              metrics=['accuracy'])

print("train")
print(x_train_3d)
print(y_train)
model.fit(x_train_3d, y_train, epochs=100, batch_size=1)

print("Reading␣testing␣data")
x_test_1d = np.loadtxt(test_x_location, dtype="uint8", delimiter=",")
x_test_3d = x_test_1d.reshape(-1,2,1)
y_test = np.loadtxt(test_y_location, dtype="uint8", delimiter=",")

print("evaluate")
model.evaluate(x_test_3d, y_test, batch_size=1)
```