1区域房价回归预测

In [1]: import tensorflow as tf

import numpy as np

import matplotlib.pyplot as plt

In [2]: boston\_housing = tf.keras.datasets.boston\_housing

(train\_x, train\_y), (test\_x, test\_y) = boston\_housing.load\_data( )

In [3]: train\_x. shape, train\_y. shape

Out[3]:

In [4]: test\_x. shape, test\_y. shape

Out[4]:

In [5]: num\_train=len(train\_x)

num\_test=len(test\_x)

In [6]: x\_train=(train\_x-train\_x.min(axis=0))/(train\_x.max(axis=0)- train\_x.min(axis=0))

y\_train=train\_y

x\_test=(test\_x-test\_x.min(axis=0))/(test\_x.max(axis=0)- test\_x.min(axis=0))

y\_test=test\_y

In [7]: x0\_train = np.ones(num\_train). reshape(-1, 1)

x0\_test = np, ones(num\_test), reshape(-1, 1)

In [8]: X\_train=tf.cast(tf.concat([x0\_train,x\_train],axis=1),tf.float32)

X\_test=tf.cast(tf.concat({x0\_test, x\_test}, axis=1), tf.float32)

In [9]: X\_train. shape, X\_test. shape

Out[9]: : (TensorShape([404, 14]), TensorShape([102, 14]))

In [10]: Y\_train=tf.constant(y\_train.reshape(-1, 1), tf.float32)

Y\_test=tf.constant(y\_test.reshape(-1, 1), tf.float32)

In [11]: Y\_train. shape, Y\_test. shape

Out[11]: (TensorShape([404, 1]), TensorShape([102, 1]))

In [12]: learn\_rate = 0.01

iter= 2000

display\_step =200

In [13]: np. random. seed (612)

W = tf. Variable(np. random. randn(14, 1), dtype=tf.float32)

mse\_train=[ ]

mse\_test=[ ]

for i in range :

with tf.GradientTape( ) as tape:

PRED\_train=tf. natmul(X\_train, W)

Loss\_train=0.5\* tf. reduce\_mean(tf.square(Y\_train-PRED\_train))

PRED\_test=tf.matmul(X\_test, W)

Loss\_test=0.5\* tf. reduce\_mean(tf.square(Y\_test-PRED\_test))

mse\_train. append (Loss\_train)

mse\_test. append (Loss\_test)

dL\_dW= tape. gradient (Loss\_train,W)

W. assign\_sub(learn\_rate\*dL\_dW)

if i % display\_step == 0 :

print("i: %i, Train Loss: %f, Test Loss: %f " % (i, Loss\_train,Loss\_test))

plt. subplot (131)

plt.ylabel("MSE")

plt.plot(mse\_train,color="blue",linewidth=3)

plt.plot(mse\_test, color="red", linewidth=1.5)

plt.subplot (132)

plt.plot(y\_train, color="blue", marker="o", label="true\_price")

plt.plot(PRED\_train,color="red",marker=".",label="predict")

plt.legend( )

plt.ylabel("Price")

plt. subplot (133)

plt.plot(y\_test, color="blue", marker="o", label="true\_price")

plt.plot(PRED\_test, color="red", marker=".", label="predict")

2.服装消费趋势分析

import matplotlib as mpl

import matplotlib.pyplot as plt

imatplotlib inline

import numpy as np

import sklearn

import pandas as pd

import os

import sys

import time

import tensorflow as tf

from tensorflow import keras

model = keras.models.Sequential( )

model.add(keras.layers.Flatten(input\_shape=[28, 28]))

model.add(keras.layers.Dense(300,activation="relu"))

model.add(keras.layers.Dense(100, activation="relu"))

model.add(keras.layers.Dense(10, activation="softmax"))

model.compile(loss="sparse\_categorical\_crossentropy",

optimizer = "sgd",

metrics = ["accuracy"])

model = keras.models.Sequential([

keras.layers.Flatten(input\_shape=[28, 28]),

keras.layers.Dense(300,activation=relu),

keras.layers.Dense(100,activation=relu),

keras.layers.Dense(10, activation=softmax)

])

type(history)

tensorflow.python.keras.callbacks.History

def plot\_learning\_curves(history):

pd.DataFrame(history.history).plot(figsize=(8, 5))

plt.grid(True)

plt.gca( ).set\_ylim(0, 1)

plt.show( )

plot\_learning\_curves(history)

print(np.max(x\_train\_scaled), np.min(x\_train\_scaled)

2.023144 -0.8105139

history = model.fit(x\_train\_scaled, y\_train, epochs=10,

validation\_data=(x\_valid\_scaled, y\_valid))

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler( )

x\_train\_scaled = scaler.fit\_transform(

x\_train.astype(np.float32).reshape(-1, 1)).reshape(-1, 28, 28)

x\_valid\_scaled = scaler.transform(

x\_valid.astype(np.float32).reshape(-1, 1)).reshape(-1, 28, 28)

x\_test\_scaled = scaler.transform(

x\_test.astype(np.float32).reshape(-1, 1)).reshape(-1, 28, 28)

print(np.max(x\_train\_scaled), np.min(x\_train\_scaled))

2.023144 -0.8105139

history = model.fit(x\_train\_scaled, y\_train, epochs=10,

validation\_data=(x\_valid\_scaled, y\_valid))