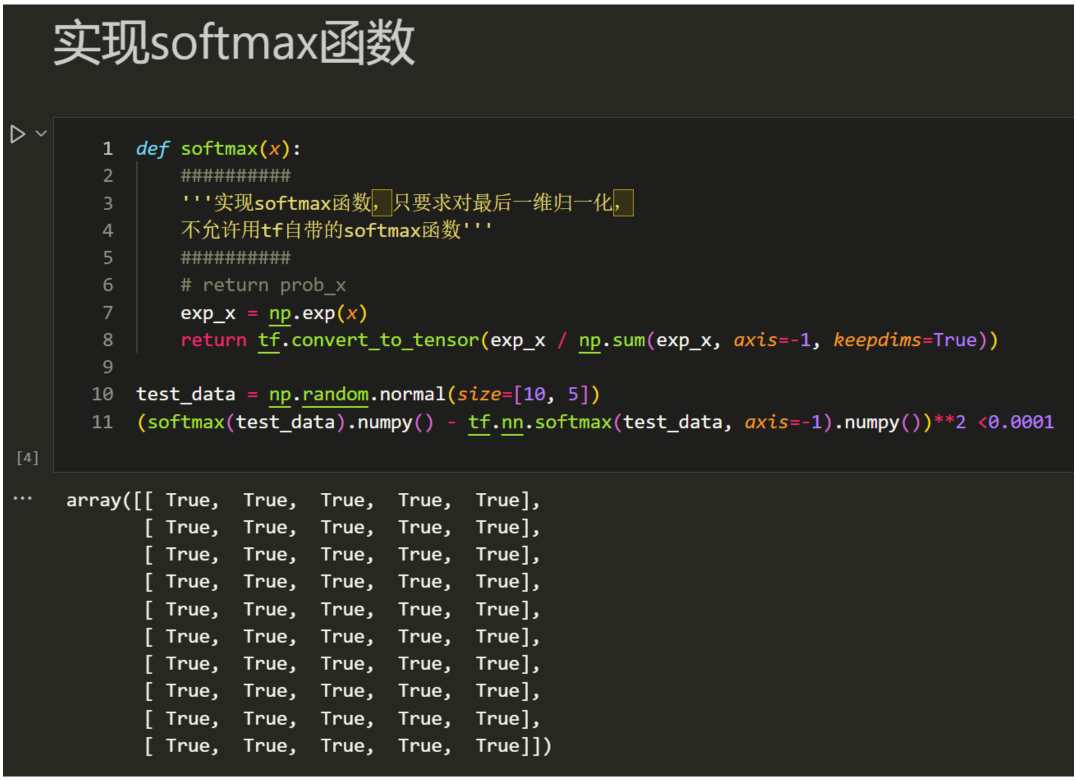
Chap4 报告

21051923 邵屹杰

（代码环境在文末说明）

tf2.0-exercise 代码与结果

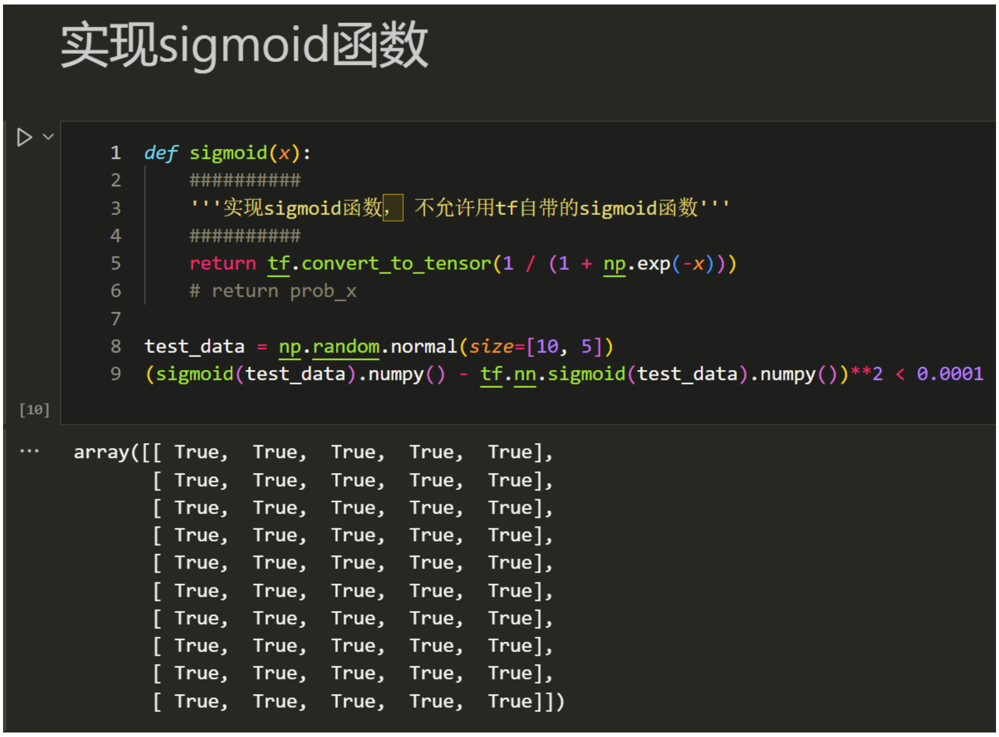


实现softmax函数

```python
def softmax(x):
    ##########
    '''实现softmax函数，只要求对最后一维归一化，
    不允许用tf自带的softmax函数'''
    ##########
    # return prob_x
    exp_x = np.exp(x)
    return tf.convert_to_tensor(exp_x / np.sum(exp_x, axis=-1, keepdims=True))

test_data = np.random.normal(size=[10, 5])
(softmax(test_data).numpy() - tf.nn.softmax(test_data, axis=-1).numpy())**2 <0.0001
```

[4]

```
array([[ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

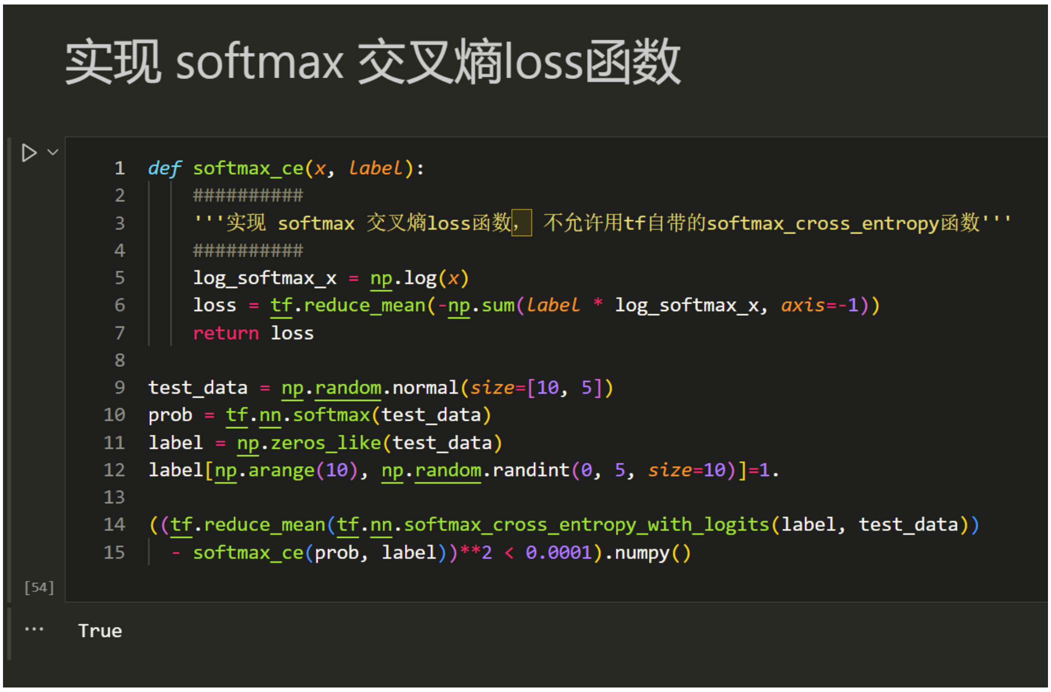


实现sigmoid函数

```python
def sigmoid(x):
    ##########
    '''实现sigmoid函数， 不允许用tf自带的sigmoid函数'''
    ##########
    return tf.convert_to_tensor(1 / (1 + np.exp(-x)))
    # return prob_x

test_data = np.random.normal(size=[10, 5])
(sigmoid(test_data).numpy() - tf.nn.sigmoid(test_data).numpy())**2 < 0.0001
```

[10]

```
array([[ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

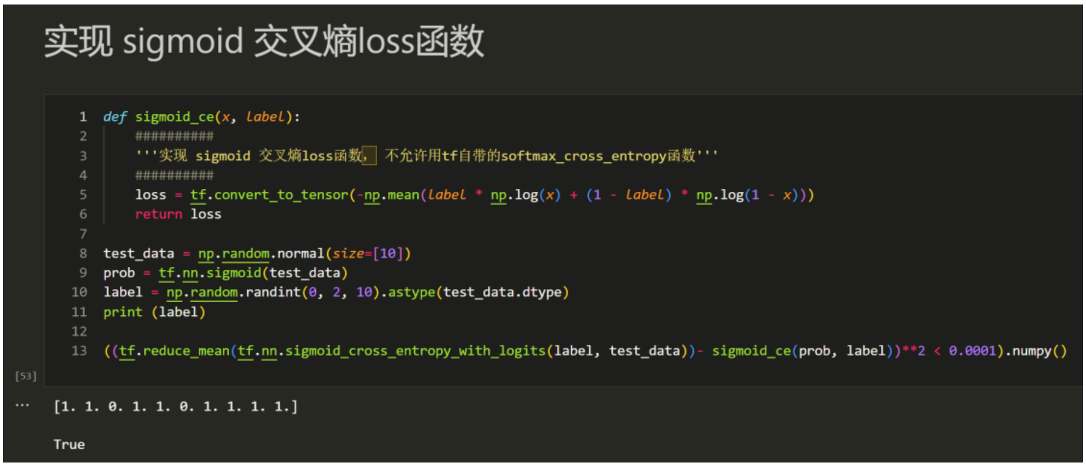


实现 softmax 交叉熵loss函数

```python
def softmax_ce(x, label):
    ##########
    '''实现 softmax 交叉熵loss函数， 不允许用tf自带的softmax_cross_entropy函数'''
    ##########
    log_softmax_x = np.log(x)
    loss = tf.reduce_mean(-np.sum(label * log_softmax_x, axis=-1))
    return loss

test_data = np.random.normal(size=[10, 5])
prob = tf.nn.softmax(test_data)
label = np.zeros_like(test_data)
label[np.arange(10), np.random.randint(0, 5, size=10)]=1.

((tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(label, test_data))
  - softmax_ce(prob, label))**2 < 0.0001).numpy()
```

[54]

```
True
```



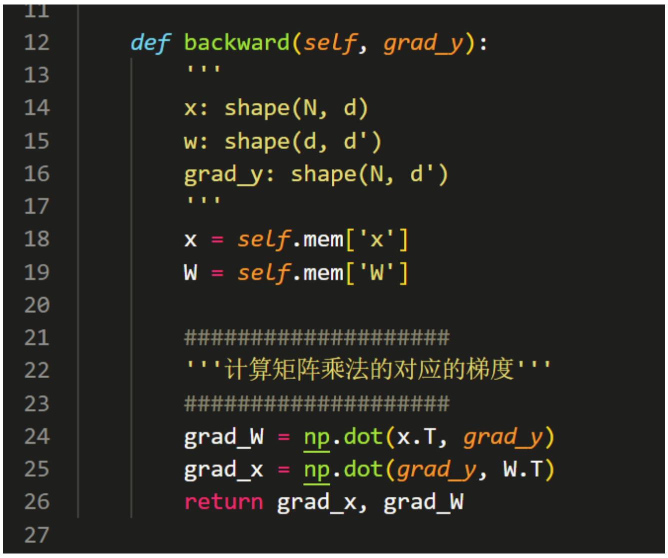
实现 sigmoid 交叉熵loss函数

```python
def sigmoid_ce(x, label):
    ##########
    '''实现 sigmoid 交叉熵loss函数， 不允许用tf自带的softmax_cross_entropy函数'''
    ##########
    loss = tf.convert_to_tensor(-np.mean(label * np.log(x) + (1 - label) * np.log(1 - x)))
    return loss

test_data = np.random.normal(size=[10])
prob = tf.nn.sigmoid(test_data)
label = np.random.randint(0, 2, 10).astype(test_data.dtype)
print (label)

((tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(label, test_data))- sigmoid_ce(prob, label))**2 < 0.0001).numpy()
```

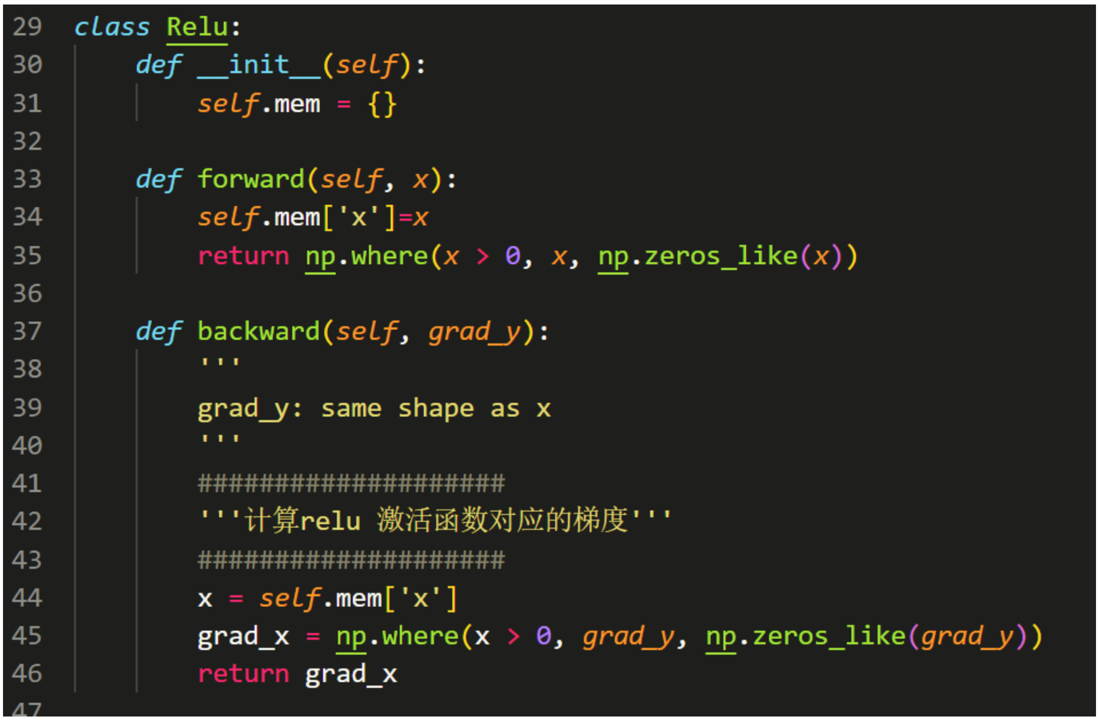
[53]

```
[1. 1. 0. 1. 1. 0. 1. 1. 1. 1.]

True
```

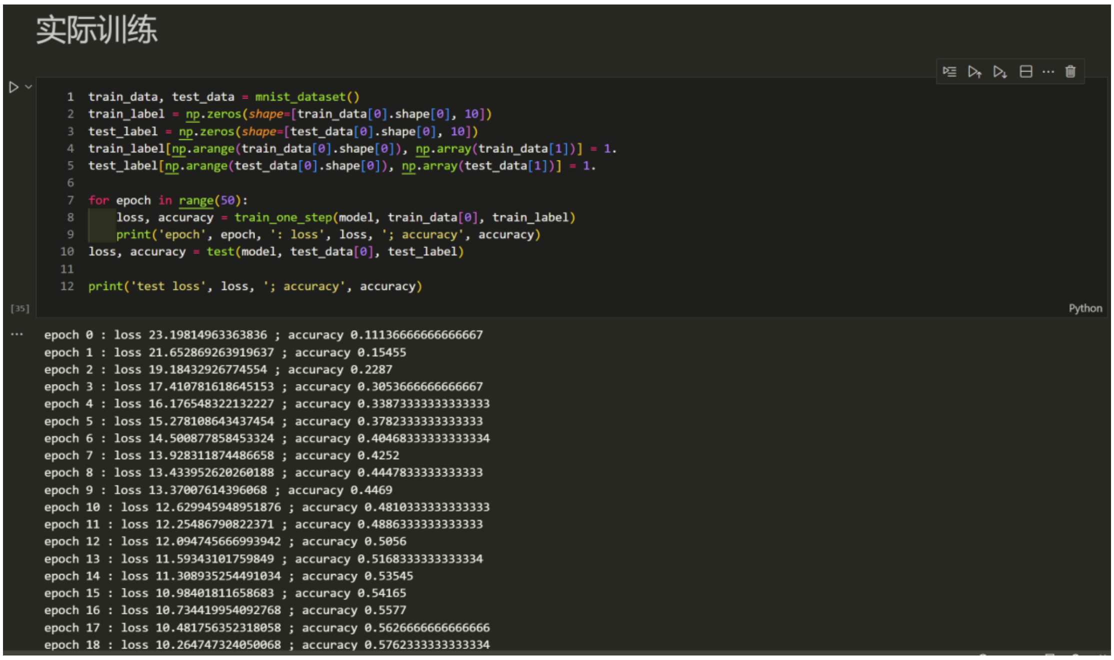
tutorial_minst_fnn-numpy-exercise 代码填空部分与结果



```python
    def backward(self, grad_y):
        '''
        x: shape(N, d)
        w: shape(d, d')
        grad_y: shape(N, d')
        '''
        x = self.mem['x']
        W = self.mem['W']

        ####################
        '''计算矩阵乘法的对应的梯度'''
        ####################
        grad_W = np.dot(x.T, grad_y)
        grad_x = np.dot(grad_y, W.T)
        return grad_x, grad_W

```



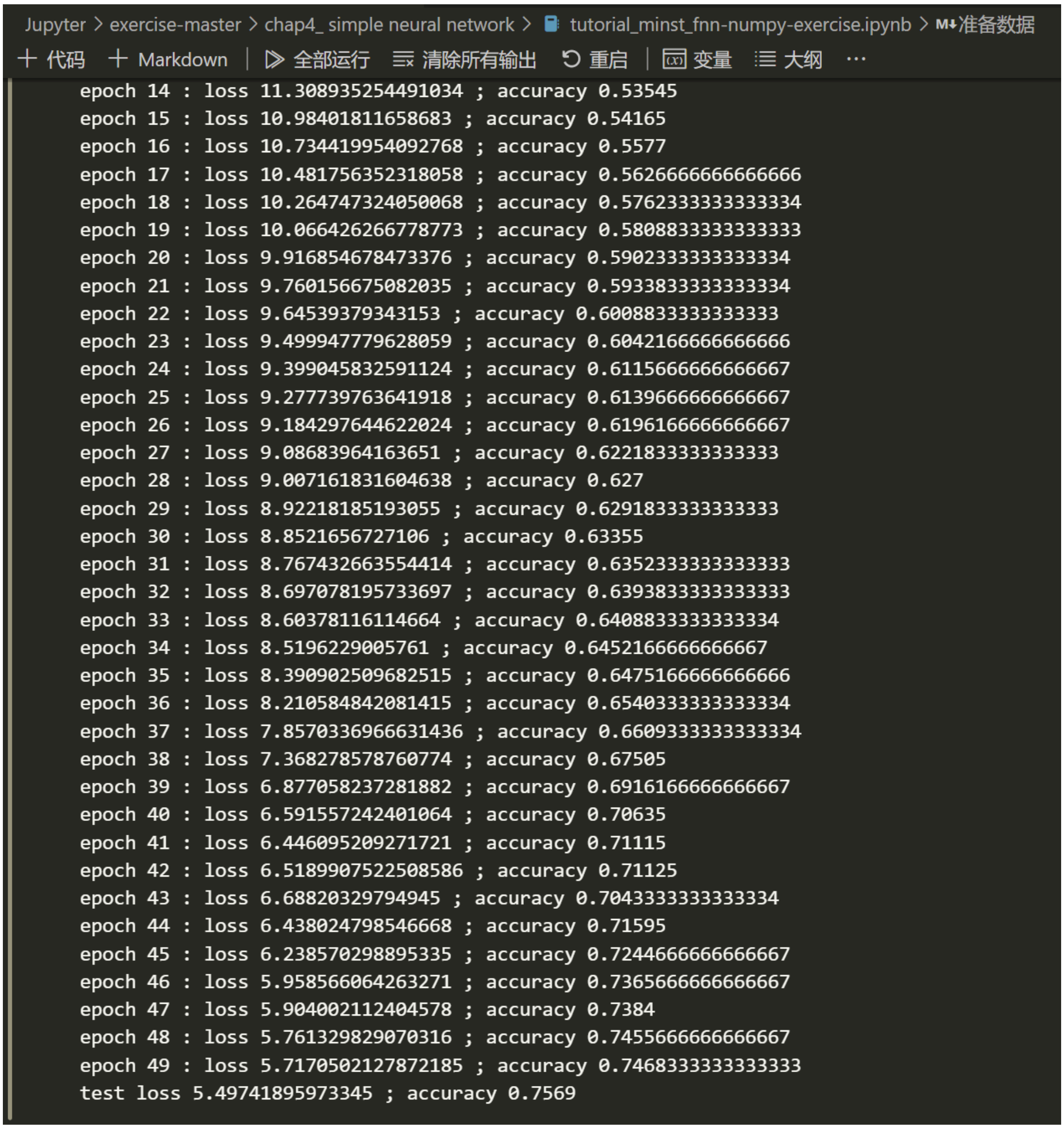
```python
class Relu:
    def __init__(self):
        self.mem = {}

    def forward(self, x):
        self.mem['x']=x
        return np.where(x > 0, x, np.zeros_like(x))

    def backward(self, grad_y):
        '''
        grad_y: same shape as x
        '''
        #####################
        '''计算relu 激活函数对应的梯度'''
        #####################
        x = self.mem['x']
        grad_x = np.where(x > 0, grad_y, np.zeros_like(grad_y))
        return grad_x
```



实际训练

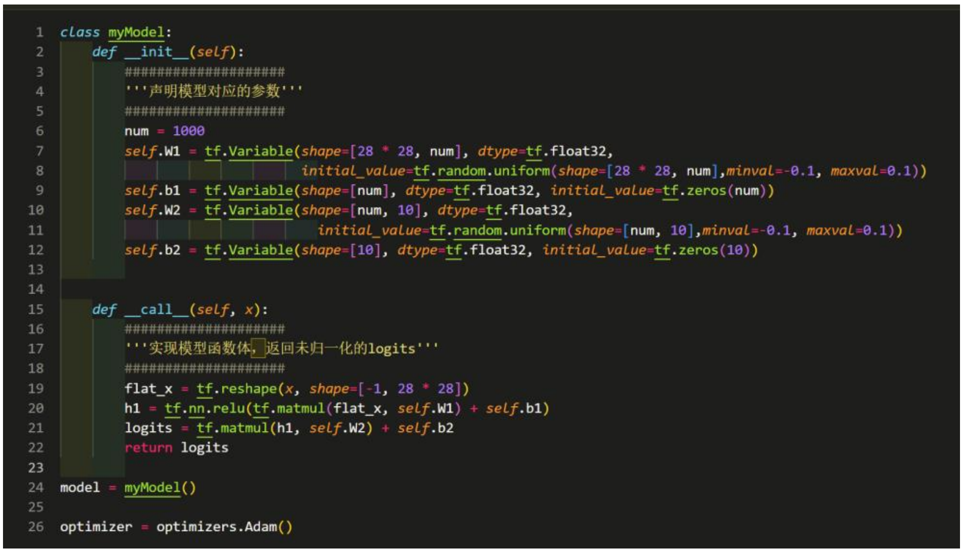
```python
train_data, test_data = mnist_dataset()
train_label = np.zeros(shape=[train_data[0].shape[0], 10])
test_label = np.zeros(shape=[test_data[0].shape[0], 10])
train_label[np.arange(train_data[0].shape[0]), np.array(train_data[1])] = 1.
test_label[np.arange(test_data[0].shape[0]), np.array(test_data[1])] = 1.

for epoch in range(50):
    loss, accuracy = train_one_step(model, train_data[0], train_label)
    print('epoch', epoch, ': loss', loss, '; accuracy', accuracy)
loss, accuracy = test(model, test_data[0], test_label)

print('test loss', loss, '; accuracy', accuracy)
```

```
epoch 0 : loss 23.19814963363836 ; accuracy 0.11136666666666667
epoch 1 : loss 21.652869263919637 ; accuracy 0.15455
epoch 2 : loss 19.18432926774554 ; accuracy 0.2287
epoch 3 : loss 17.410781618645153 ; accuracy 0.30536666666666667
epoch 4 : loss 16.176548322132227 ; accuracy 0.33873333333333333
epoch 5 : loss 15.278108643437454 ; accuracy 0.3782333333333333
epoch 6 : loss 14.500877858453324 ; accuracy 0.40468333333333334
epoch 7 : loss 13.928311874486658 ; accuracy 0.4252
epoch 8 : loss 13.433952620260188 ; accuracy 0.4447833333333333
epoch 9 : loss 13.37007614396068 ; accuracy 0.4469
epoch 10 : loss 12.629945948951876 ; accuracy 0.48103333333333333
epoch 11 : loss 12.25486790822371 ; accuracy 0.4886333333333333
epoch 12 : loss 12.094745666993942 ; accuracy 0.5056
epoch 13 : loss 11.59343101759849 ; accuracy 0.5168333333333334
epoch 14 : loss 11.308935254491034 ; accuracy 0.53545
epoch 15 : loss 10.98401811658683 ; accuracy 0.54165
epoch 16 : loss 10.734419954092768 ; accuracy 0.5577
epoch 17 : loss 10.481756352318058 ; accuracy 0.5626666666666666
epoch 18 : loss 10.264747324050068 ; accuracy 0.57623333333333334
```

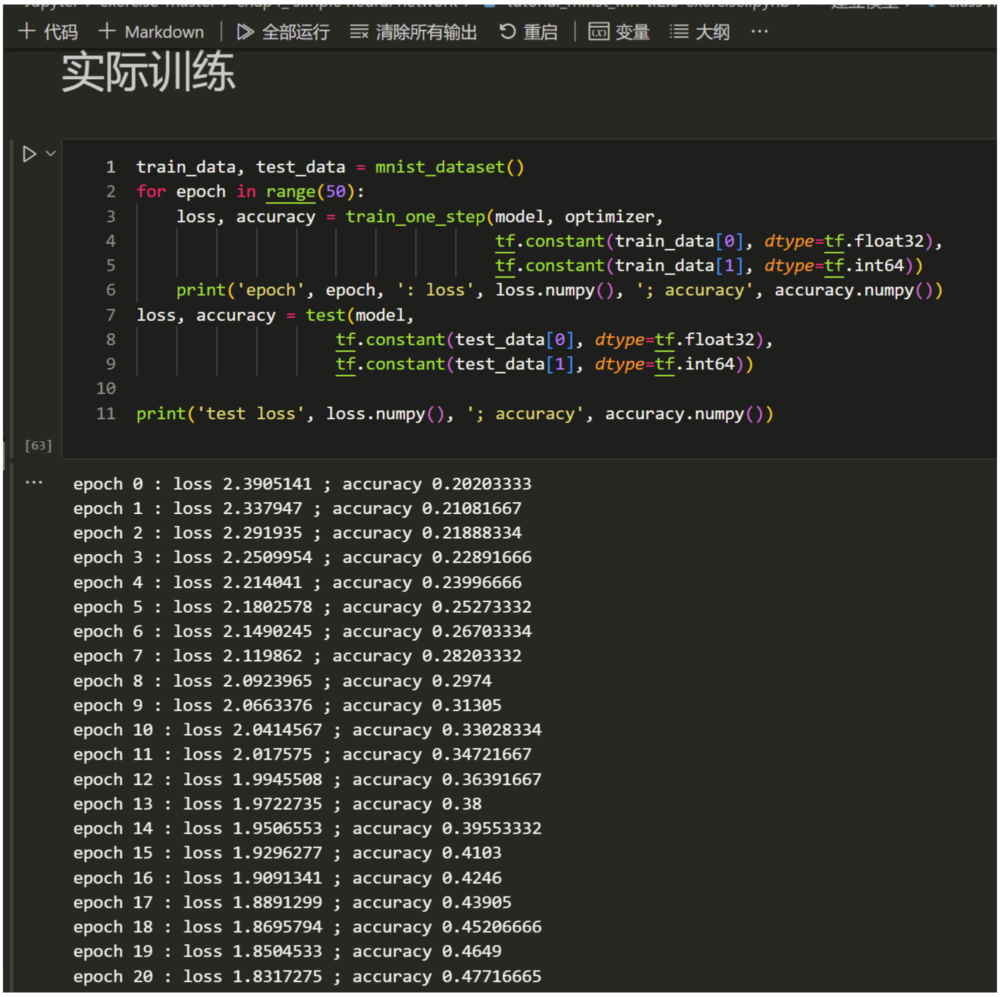

Jupyter > exercise-master > chap4_ simple neural network > tutorial_minst_fnn-numpy-exercise.ipynb > M↓准备数据
＋ 代码
＋ Markdown
▷ 全部运行
清除所有输出
重启
变量
大纲
···

```
epoch 14 : loss 11.308935254491034 ; accuracy 0.53545
epoch 15 : loss 10.98401811658683 ; accuracy 0.54165
epoch 16 : loss 10.734419954092768 ; accuracy 0.5577
epoch 17 : loss 10.481756352318058 ; accuracy 0.5626666666666666
epoch 18 : loss 10.264747324050068 ; accuracy 0.5762333333333334
epoch 19 : loss 10.066426266778773 ; accuracy 0.5808833333333333
epoch 20 : loss 9.916854678473376 ; accuracy 0.5902333333333334
epoch 21 : loss 9.760156675082035 ; accuracy 0.5933833333333334
epoch 22 : loss 9.64539379343153 ; accuracy 0.6008833333333333
epoch 23 : loss 9.499947779628059 ; accuracy 0.6042166666666666
epoch 24 : loss 9.399045832591124 ; accuracy 0.6115666666666667
epoch 25 : loss 9.277739763641918 ; accuracy 0.6139666666666667
epoch 26 : loss 9.184297644622024 ; accuracy 0.6196166666666667
epoch 27 : loss 9.08683964163651 ; accuracy 0.6221833333333333
epoch 28 : loss 9.007161831604638 ; accuracy 0.627
epoch 29 : loss 8.92218185193055 ; accuracy 0.6291833333333333
epoch 30 : loss 8.8521656727106 ; accuracy 0.63355
epoch 31 : loss 8.767432663554414 ; accuracy 0.6352333333333333
epoch 32 : loss 8.697078195733697 ; accuracy 0.6393833333333333
epoch 33 : loss 8.60378116114664 ; accuracy 0.6408833333333334
epoch 34 : loss 8.5196229005761 ; accuracy 0.6452166666666667
epoch 35 : loss 8.390902509682515 ; accuracy 0.6475166666666666
epoch 36 : loss 8.210584842081415 ; accuracy 0.6540333333333334
epoch 37 : loss 7.8570336966631436 ; accuracy 0.6609333333333334
epoch 38 : loss 7.368278578760774 ; accuracy 0.67505
epoch 39 : loss 6.877058237281882 ; accuracy 0.6916166666666667
epoch 40 : loss 6.591557242401064 ; accuracy 0.70635
epoch 41 : loss 6.446095209271721 ; accuracy 0.71115
epoch 42 : loss 6.5189907522508586 ; accuracy 0.71125
epoch 43 : loss 6.68820329794945 ; accuracy 0.7043333333333334
epoch 44 : loss 6.438024798546668 ; accuracy 0.71595
epoch 45 : loss 6.238570298895335 ; accuracy 0.7244666666666667
epoch 46 : loss 5.958566064263271 ; accuracy 0.7365666666666667
epoch 47 : loss 5.904002112404578 ; accuracy 0.7384
epoch 48 : loss 5.761329829070316 ; accuracy 0.7455666666666667
epoch 49 : loss 5.7170502127872185 ; accuracy 0.7468333333333333
test loss 5.49741895973345 ; accuracy 0.7569
```

tutorial_minst_fnn-tf2.0-exercise 代码填空与结果



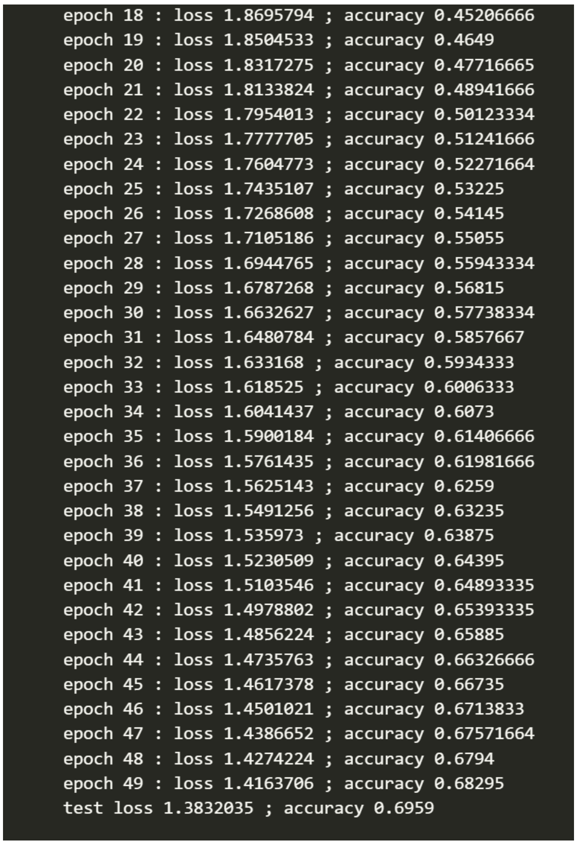
```python
class myModel:
    def __init__(self):
        ####################
        '''声明模型对应的参数'''
        ####################
        num = 1000
        self.W1 = tf.Variable(shape=[28 * 28, num], dtype=tf.float32,
                                initial_value=tf.random.uniform(shape=[28 * 28, num],minval=-0.1, maxval=0.1))
        self.b1 = tf.Variable(shape=[num], dtype=tf.float32, initial_value=tf.zeros(num))
        self.W2 = tf.Variable(shape=[num, 10], dtype=tf.float32,
                                 initial_value=tf.random.uniform(shape=[num, 10],minval=-0.1, maxval=0.1))
        self.b2 = tf.Variable(shape=[10], dtype=tf.float32, initial_value=tf.zeros(10))


    def __call__(self, x):
        ####################
        '''实现模型函数体，返回未归一化的logits'''
        ####################
        flat_x = tf.reshape(x, shape=[-1, 28 * 28])
        h1 = tf.nn.relu(tf.matmul(flat_x, self.W1) + self.b1)
        logits = tf.matmul(h1, self.W2) + self.b2
        return logits

model = myModel()

optimizer = optimizers.Adam()
```



实际训练

```python
train_data, test_data = mnist_dataset()
for epoch in range(50):
    loss, accuracy = train_one_step(model, optimizer,
                                    tf.constant(train_data[0], dtype=tf.float32),
                                    tf.constant(train_data[1], dtype=tf.int64))
    print('epoch', epoch, ': loss', loss.numpy(), '; accuracy', accuracy.numpy())
loss, accuracy = test(model,
                    tf.constant(test_data[0], dtype=tf.float32),
                    tf.constant(test_data[1], dtype=tf.int64))

print('test loss', loss.numpy(), '; accuracy', accuracy.numpy())
```

```
epoch 0 : loss 2.3905141 ; accuracy 0.20203333
epoch 1 : loss 2.337947 ; accuracy 0.21081667
epoch 2 : loss 2.291935 ; accuracy 0.21888334
epoch 3 : loss 2.2509954 ; accuracy 0.22891666
epoch 4 : loss 2.214041 ; accuracy 0.23996666
epoch 5 : loss 2.1802578 ; accuracy 0.25273332
epoch 6 : loss 2.1490245 ; accuracy 0.26703334
epoch 7 : loss 2.119862 ; accuracy 0.28203332
epoch 8 : loss 2.0923965 ; accuracy 0.2974
epoch 9 : loss 2.0663376 ; accuracy 0.31305
epoch 10 : loss 2.0414567 ; accuracy 0.33028334
epoch 11 : loss 2.017575 ; accuracy 0.34721667
epoch 12 : loss 1.9945508 ; accuracy 0.36391667
epoch 13 : loss 1.9722735 ; accuracy 0.38
epoch 14 : loss 1.9506553 ; accuracy 0.39553332
epoch 15 : loss 1.9296277 ; accuracy 0.4103
epoch 16 : loss 1.9091341 ; accuracy 0.4246
epoch 17 : loss 1.8891299 ; accuracy 0.43905
epoch 18 : loss 1.8695794 ; accuracy 0.45206666
epoch 19 : loss 1.8504533 ; accuracy 0.4649
epoch 20 : loss 1.8317275 ; accuracy 0.47716665
```



```
epoch 18 : loss 1.8695794 ; accuracy 0.45206666
epoch 19 : loss 1.8504533 ; accuracy 0.4649
epoch 20 : loss 1.8317275 ; accuracy 0.47716665
epoch 21 : loss 1.8133824 ; accuracy 0.48941666
epoch 22 : loss 1.7954013 ; accuracy 0.50123334
epoch 23 : loss 1.7777705 ; accuracy 0.51241666
epoch 24 : loss 1.7604773 ; accuracy 0.52271664
epoch 25 : loss 1.7435107 ; accuracy 0.53225
epoch 26 : loss 1.7268608 ; accuracy 0.54145
epoch 27 : loss 1.7105186 ; accuracy 0.55055
epoch 28 : loss 1.6944765 ; accuracy 0.55943334
epoch 29 : loss 1.6787268 ; accuracy 0.56815
epoch 30 : loss 1.6632627 ; accuracy 0.57738334
epoch 31 : loss 1.6480784 ; accuracy 0.5857667
epoch 32 : loss 1.633168 ; accuracy 0.5934333
epoch 33 : loss 1.618525 ; accuracy 0.6006333
epoch 34 : loss 1.6041437 ; accuracy 0.6073
epoch 35 : loss 1.5900184 ; accuracy 0.61406666
epoch 36 : loss 1.5761435 ; accuracy 0.61981666
epoch 37 : loss 1.5625143 ; accuracy 0.6259
epoch 38 : loss 1.5491256 ; accuracy 0.63235
epoch 39 : loss 1.535973 ; accuracy 0.63875
epoch 40 : loss 1.5230509 ; accuracy 0.64395
epoch 41 : loss 1.5103546 ; accuracy 0.64893335
epoch 42 : loss 1.4978802 ; accuracy 0.65393335
epoch 43 : loss 1.4856224 ; accuracy 0.65885
epoch 44 : loss 1.4735763 ; accuracy 0.66326666
epoch 45 : loss 1.4617378 ; accuracy 0.66735
epoch 46 : loss 1.4501021 ; accuracy 0.6713833
epoch 47 : loss 1.4386652 ; accuracy 0.67571664
epoch 48 : loss 1.4274224 ; accuracy 0.6794
epoch 49 : loss 1.4163706 ; accuracy 0.68295
test loss 1.3832035 ; accuracy 0.6959
```

运行环境 mytensor

```
Package                      Version
---------------------------- ---------
absl-py                      1.4.0
astroid                      2.15.0
astunparse                   1.6.3
attrs                        22.2.0
autopep8                     1.6.0
backcall                     0.2.0
cached-property              1.5.2
cachetools                   5.3.0
certifi                      2022.12.7
charset-normalizer           3.0.1
chex                         0.1.5
colorama                     0.4.6
cycler                       0.11.0
debugpy                      1.5.1
decorator                    5.1.1
dill                         0.3.6
dm-pix                       0.4.0
dm-tree                      0.1.8
docstring-to-markdown        0.11
entrypoints                  0.4
etils                        0.9.0
exceptiongroup               1.1.0
flake8                       5.0.4
flatbuffers                  23.3.3
flax                         0.6.4
fonttools                    4.38.0
gast                         0.4.0
gin-config                   0.5.0
google-auth                  2.16.2
google-auth-oauthlib         0.4.6
google-pasta                 0.2.0
grpcio                       1.51.3
h5py                         3.8.0
idna                         3.4
importlib-metadata           6.0.0
importlib-resources          5.12.0
iniconfig                    2.0.0
ipykernel                    6.15.2
ipython                      7.34.0
jax                          0.3.25
```

```
jaxlib                         0.3.25
jedi                           0.18.2
jupyter_client                 7.4.9
jupyter_core                   4.11.1
keras                          2.11.0
kiwisolver                     1.4.4
libclang                       15.0.6.1
Markdown                       3.4.1
markdown-it-py                 2.2.0
MarkupSafe                     2.1.2
matplotlib                     3.5.3
matplotlib-inline              0.1.6
mccabe                         0.7.0
mdurl                          0.1.2
mediapy                        1.1.2
msgpack                        1.0.4
nest-asyncio                   1.5.6
numpy                          1.21.6
oauthlib                       3.2.2
opencv-contrib-python          4.7.0.72
opencv-python                  4.7.0.72
opt-einsum                     3.3.0
optax                          0.1.4
orbax                          0.1.0
packaging                      23.0
parso                          0.8.3
pickleshare                    0.7.5
Pillow                         9.4.0
pip                            22.3.1
platformdirs                   3.1.1
pluggy                         1.0.0
prompt-toolkit                 3.0.38
protobuf                       3.19.6
psutil                         5.9.0
pyasn1                         0.4.8
pyasn1-modules                 0.2.8
pycodestyle                    2.10.0
pydocstyle                     6.2.3
pyflakes                       3.0.1
Pygments                       2.14.0
pylint                         2.17.0
pyparsing                      3.0.9
pytest                         7.2.2
python-dateutil                2.8.2
```

```
python-lsp-jsonrpc            1.0.0
python-lsp-server             1.7.1
pytoolconfig                  1.2.5
pywin32                       305.1
PyYAML                        6.0
pyzmq                         23.2.0
rawpy                         0.18.0
requests                      2.28.2
requests-oauthlib             1.3.1
rich                          13.3.1
rope                          1.7.0
rsa                           4.9
scipy                         1.7.3
setuptools                    65.6.3
six                           1.16.0
snowballstemmer               2.2.0
tensorboard                   2.11.2
tensorboard-data-server       0.6.1
tensorboard-plugin-wit        1.8.1
tensorflow                    2.11.0
tensorflow-estimator          2.11.0
tensorflow-intel              2.11.0
tensorflow-io-gcs-filesystem 0.31.0
tensorstore                   0.1.28
termcolor                     2.2.0
tomli                         2.0.1
tomlkit                       0.11.6
toolz                         0.12.0
tornado                       6.2
traitlets                     5.9.0
typed-ast                     1.5.4
typing_extensions             4.5.0
ujson                         5.7.0
urllib3                       1.26.14
wcwidth                       0.2.6
Werkzeug                      2.2.3
whatthepatch                  1.0.4
wheel                         0.38.4
wincertstore                  0.2
wrapt                         1.15.0
yapf                          0.32.0
zipp                          3.15.0
```