1

June 3, 2020

1 2

1.0.1 30

[16]:
```
from keras.datasets import mnist
from keras import models, layers
from keras.utils import to_categorical

(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# 2            (Dense)
network = models.Sequential() #
network.add(layers.Dense(512, activation='relu', input_shape=(28*28,)))
network.add(layers.Dense(10, activation='softmax'))

#        =                    ,
network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])

train_images = train_images.reshape((60000, 28*28))
#                    [0, 255] (uint8)   [0, 1] (float32).
train_images = train_images.astype('float32') / 255

test_images = test_images.reshape((10000, 28*28))
#                    [0, 255] (uint8)   [0, 1] (float32).
test_images = test_images.astype('float32') / 255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)


network.fit(train_images, train_labels, epochs=5, batch_size=128)
```

```
WARNING:tensorflow:From /opt/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:422: The name tf.global_variables
is deprecated. Please use tf.compat.v1.global_variables instead.
```

```
Epoch 1/5
60000/60000 [==============================] - 5s 87us/step - loss: 0.2580 -
accuracy: 0.9245
Epoch 2/5
60000/60000 [==============================] - 6s 92us/step - loss: 0.1037 -
accuracy: 0.9690
Epoch 3/5
60000/60000 [==============================] - 5s 86us/step - loss: 0.0672 -
accuracy: 0.9796
Epoch 4/5
60000/60000 [==============================] - 5s 87us/step - loss: 0.0486 -
accuracy: 0.9850
Epoch 5/5
60000/60000 [==============================] - 5s 88us/step - loss: 0.0378 -
accuracy: 0.9888
```

[16]:
```
<keras.callbacks.callbacks.History at 0x1a5647f290>
```

[17]:
```
test_loss, test_acc = network.evaluate(test_images, test_labels)
print('test acc: ', test_acc)
```

```
10000/10000 [==============================] - 1s 73us/step
test acc:  0.9763000011444092
```

2

[19]:
```
import numpy as np

#     -
x = np.array(12)
x.ndim
```

[19]:
```
0
```

[20]:
```
#     -
x = np.array([12, 3, 6, 14])
x.ndim
```

[20]:
```
1
```

[21]:
```
#      -      -    ,
x = np.array([[5, 78, 2, 34, 0],
              [6, 79, 3, 35, 1],
              [7, 80, 4, 36, 2]])
x.ndim
```

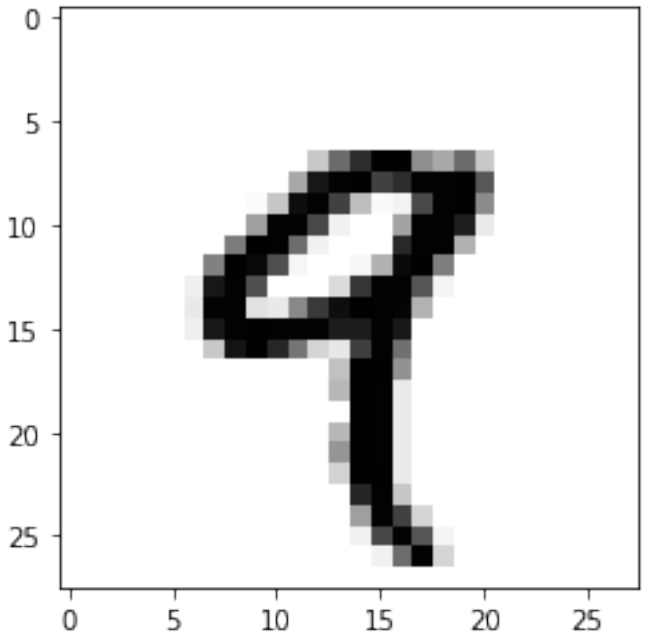
[21]:
```
2
```

[22]:
```
#            -
x = np.array([[[5, 78, 2, 34, 0],
               [6, 79, 3, 35, 1],
               [7, 80, 4, 36, 2]],
              [[5, 78, 2, 34, 0],
               [6, 79, 3, 35, 1],
               [7, 80, 4, 36, 2]],
              [[5, 78, 2, 34, 0],
               [6, 79, 3, 35, 1],
               [7, 80, 4, 36, 2]]])
x.ndim
```

[22]: 3

[29]:
```
import matplotlib.pyplot as plt


(train_images, train_labels), (test_images, test_labels) = mnist.load_data()
digit = train_images[4]
plt.imshow(digit, cmap=plt.cm.binary)
plt.show()
```



[ ]:

[ ]:
```
def naive_relu(x):
    '''            relu
               z = np.maximum(z, 0.)
    '''

    assert len(x.shape) == 2

    x = x.copy()   #
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i, j] = max(x[i, j], 0)
    return x


def naive_add(x, y):
    '''
               z = x + y
             ,                  :
                   (            =  _       _ _1)!
    '''

    assert len(x.shape) == 2   #      ,    x -           numpy
    assert len(y.shape) == 1   #      ,    y -     numpy
    assert x.shape[1] == y.shape[0]

    x = x.copy()
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            x[i, j] += y[i, j]
    return x
```

[35]:
```
"""                        maximum
                    """

# x =         (64, 3, 32, 10)
x = np.random.random((64, 3, 32, 10))
# y =         (32, 10)
y = np.random.random((32, 10))

#
z = np.maximum(x, y)
```

2.0.1

[60]:
```
import numpy as np

x = np.array([1, 2, 3, 4])
y = np.array([5, 6, 7, 8])
z = np.dot(x, y)
z
```

[60]: 70

[61]:
```
def naive_vector_dot(x, y):
    """

    """
    assert x.ndim == 1
    assert y.ndim == 1
    assert x.shape[0] == y.shape[0]

    z = 0.
    for i in range(x.shape[0]):
        #            : [1, 2, 3, 4]   [5, 6, 7, 8]
        #          : 0 += 1*5 + 2*6 + 3*7 + 4*8,
        z += x[i]*y[i]
    return z
```

[62]:
```
naive_vector_dot(x, y)
```

[62]: 70.0

[70]:
```
import numpy as np

x = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8]])
y = np.array([9, 10, 11, 12])



def naive_matrix_vector_dot(x, y):
    """
    """

    assert x.ndim == 2
    assert y.ndim == 1
    #                                x                          y
    assert x.shape[1] == y.shape[0]

    z = np.zeros(x.shape[0])
```

```
    for i in range(x.shape[0]):
        for j in range(x.shape[1]):
            z[i] += x[i,j] * y[j]
    return z


naive_matrix_vector_dot(x, y)
```

[70]:
```
array([110., 278.])
```

[72]:
```
import numpy as np


def naive_matrix_dot(x, y):
    """
    """

    assert x.ndim == 2   #      ,    x -
    assert y.ndim == 2
    assert x.shape[1] == x.shape[0]

    z = np.zeros((x.shape[0], y.shape[1]))
    for i in range(x.shape[0]):
        for j in range(y.shape[1]):
            row_x = x[i, :]
            column_y = y[:, j]
            z[i, j] = naive_vector_dot(row_x, column_y)
    return z
```

[ ]:

[ ]: