tf2.0-exercise

October 24, 2025

1 Tensorflow2.0

[2]:
```python
import tensorflow as tf
import numpy as np
```

1.1 softmax

[3]:
```python
def softmax(x):
    ##########
    ''' softmax
      tf  softmax '''
    ##########

    #    TensorFlow      float32
    x = tf.cast(x, tf.float32)

    #
    exp_x = tf.exp(x)

    #
    #
    sum_exp_x = tf.reduce_sum(exp_x, axis=-1, keepdims=True)

    #  softmax
    prob_x = exp_x / sum_exp_x

    return prob_x

#
test_data = np.random.normal(size=[10, 5]).astype(np.float32)
print(test_data)
print(softmax(test_data).numpy())
# softmax     tf.nn.softmax
(softmax(test_data).numpy() - tf.nn.softmax(test_data, axis=-1).numpy())**2 <0.
  ↪0001
```

```
[[ 0.7522302  -0.05871538 -0.3995748   1.4157422  -0.05660807]
 [-0.20385897  1.3567791   0.19437975 -0.04637642  0.13372248]
 [-0.016101   -0.57322943  2.3795023  -0.8478688  -0.70687836]
 [ 1.9659184   1.6274889  -0.2850514   1.556407   -0.74753094]
 [ 0.5989454  -1.785647    1.4455488  -0.95527136 -1.2994574 ]
 [ 0.7322035  -0.5830047   1.963724   -0.9338021  -0.4387086 ]
 [ 0.05598401  0.5450637  -0.9637479  -0.12449894 -0.41151586]
 [ 1.0257028   0.89750886 -0.8318968  -1.5604948  -0.1657166 ]
 [ 1.1550524  -1.385352    0.7867837  -0.3040725   1.0288265 ]
 [ 1.1559253  -1.4780195   0.2581899   0.36149567 -1.1963102 ]]
[[0.2411104  0.10715853 0.07620674 0.46813974 0.10738458]
 [0.10180011 0.4847577  0.15160067 0.11916316 0.14267832]
 [0.07416123 0.04248339 0.81390595 0.03228084 0.03716859]
 [0.3923926  0.27973235 0.04131778 0.2605387  0.02601865]
 [0.2642006  0.02433988 0.6160401  0.05584009 0.03957928]
 [0.1925321  0.05167916 0.6596989  0.03638867 0.05970113]
 [0.22457077 0.36623326 0.08100078 0.18748668 0.14070858]
 [0.4141087  0.36428428 0.06462032 0.0311848  0.12580198]
 [0.346666   0.02732925 0.23986904 0.08057891 0.30555683]
 [0.49351493 0.03543175 0.2011031  0.22298923 0.04696101]]
```

```
[3]: 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]])
```

1.2 sigmoid

```
[4]: def sigmoid(x):
         ##########
         ''' sigmoid     tf  sigmoid '''
         ##########

         #    TensorFlow      float32
         x = tf.cast(x, tf.float32)

         #sigmoid(x) = 1 / (1 + exp(-x))
         prob_x = 1.0 / (1.0 + tf.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
```

[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]])
```

1.3 softmax loss

[7]:
```python
def softmax_ce(x, label):
    ##########
    '''  softmax  loss    tf  softmax_cross_entropy '''
    ##########

    # x      softmax label  one-hot
    x = tf.cast(x, tf.float32)
    label = tf.cast(label, tf.float32)

    #    -sum(label * log(x))
    #ce_per_example
    ce_per_example = -tf.reduce_sum(label * tf.math.log(x), axis=-1)
    #           batch
    loss = tf.reduce_mean(ce_per_example)

    return loss

test_data = np.random.normal(size=[10, 5]).astype(np.float32)
prob = tf.nn.softmax(test_data)
#   one-hot
label = np.zeros_like(test_data, dtype=np.float32)
label[np.arange(10), np.random.randint(0, 5, size=10)] = 1.0

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

[7]:
```
np.True_
```

1.4 sigmoid loss

[9]:
```python
def sigmoid_ce(x, label):
    ##########
    '''  sigmoid  loss    tf  sigmoid_cross_entropy '''
    ##########
    # x    ŷ   (0,1) label   [0,1]  0/1
    x = tf.cast(x, tf.float32)
    label = tf.cast(label, tf.float32)

    #      L = -[ y*log(ŷ) + (1-y)*log(1-ŷ) ]
    ce = -(label * tf.math.log(x) + (1.0 - label) * tf.math.log(1.0 - x))

    #
    ce_per_example = tf.reduce_sum(ce, axis=-1) if len(ce.shape) > 1 else ce

    #   batch
    loss = tf.reduce_mean(ce_per_example)
    return loss

test_data = np.random.normal(size=[10]).astype(np.float32)
prob = tf.nn.sigmoid(test_data)
label = np.random.randint(0, 2, 10).astype(np.float32)
print (label)

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

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

[9]:
```
np.True_
```