



TensorFlow

前向传播（张量）实战

主讲：龙良曲

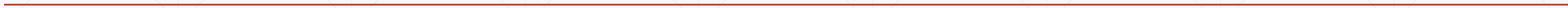
What we have learned

- create tensor
 - indexing and slices
 - reshape and broadcasting
 - math operations
-

Recap

- $out = \text{relu}\{\text{relu}\{\text{relu}[X@W_1 + b_1]@W_2 + b_2\}@W_3 + b_3\}$
 - $pred = \text{argmax}(out)$
 - $loss = \text{MSE}(out, label)$
 - minimize $loss$
 - $[W'_1, b'_1, W'_2, b'_2, W'_3, b'_3]$
-

**JUST
DO
IT.**

The text "JUST DO IT." is rendered in a bold, black, sans-serif font. The letters are heavily textured with a splatter or ink-blot effect, giving them a gritty, dynamic appearance. The background is a light gray with a subtle, repeating diamond-shaped grid pattern. The overall composition is centered and minimalist.

$$\text{relu}[X@W_1 + b_1]$$

```

# 784 => 512
w1, b1 = tf.Variable(tf.random.truncated_normal([784, 512], stddev=0.1)),
tf.Variable(tf.zeros([512]))

for step, (x,y) in enumerate(train_db):
    # [b, 28, 28] => [b, 784]
    x = tf.reshape(x, (-1, 784))
    with tf.GradientTape() as tape:

        # layer1.
        h1 = x @ w1 + b1
        h1 = tf.nn.relu(h1)
```

$$\{relu\{relu[X@W_1 + b_1]@W_2 + b_2\}$$


```
● ● ●  
  
# 784 => 512  
w1, b1 = tf.Variable(tf.random.truncated_normal([784, 512], stddev=0.1)),  
tf.Variable(tf.zeros([512]))  
# 512 => 256  
w2, b2 = tf.Variable(tf.random.truncated_normal([512, 256], stddev=0.1)),  
tf.Variable(tf.zeros([256]))  
  
for step, (x,y) in enumerate(train_db):  
    # [b, 28, 28] => [b, 784]  
    x = tf.reshape(x, (-1, 784))  
    with tf.GradientTape() as tape:  
  
        # layer1.  
        h1 = x @ w1 + b1  
        h1 = tf.nn.relu(h1)  
        # layer2  
        h2 = h1 @ w2 + b2  
        h2 = tf.nn.relu(h2)
```

$$out = \text{relu}\{\text{relu}\{\text{relu}[X@W_1 + b_1]@W_2 + b_2\}@W_3 + b_3\}$$

```
# 784 => 512
w1, b1 = tf.Variable(tf.random.truncated_normal([784, 512], stddev=0.1)),
tf.Variable(tf.zeros([512]))
# 512 => 256
w2, b2 = tf.Variable(tf.random.truncated_normal([512, 256], stddev=0.1)),
tf.Variable(tf.zeros([256]))
# 256 => 10
w3, b3 = tf.Variable(tf.random.truncated_normal([256, 10], stddev=0.1)),
tf.Variable(tf.zeros([10]))
for step, (x,y) in enumerate(train_db):
    # [b, 28, 28] => [b, 784]
    x = tf.reshape(x, (-1, 784))
    with tf.GradientTape() as tape:

        # layer1.
        h1 = x @ w1 + b1
        h1 = tf.nn.relu(h1)
        # layer2
        h2 = h1 @ w2 + b2
        h2 = tf.nn.relu(h2)
        # output
        out = h2 @ w3 + b3
```

Compute loss



```
# compute loss
# [b, 10] - [b, 10]
loss = tf.square(y-out)
# [b, 10] => [b]
loss = tf.reduce_mean(loss, axis=1)
# [b] => scalar
loss = tf.reduce_mean(loss)
```

Compute gradient and update w

```
# compute gradient
grads = tape.gradient(loss, [w1, b1, w2, b2, w3, b3])
# for g in grads:
#     print(tf.norm(g))
# update w' = w - lr*grad
optimizer.apply_gradients(zip(grads, [w1, b1, w2, b2, w3, b3]))
```

下一课时

合并与切割

Thank You.
