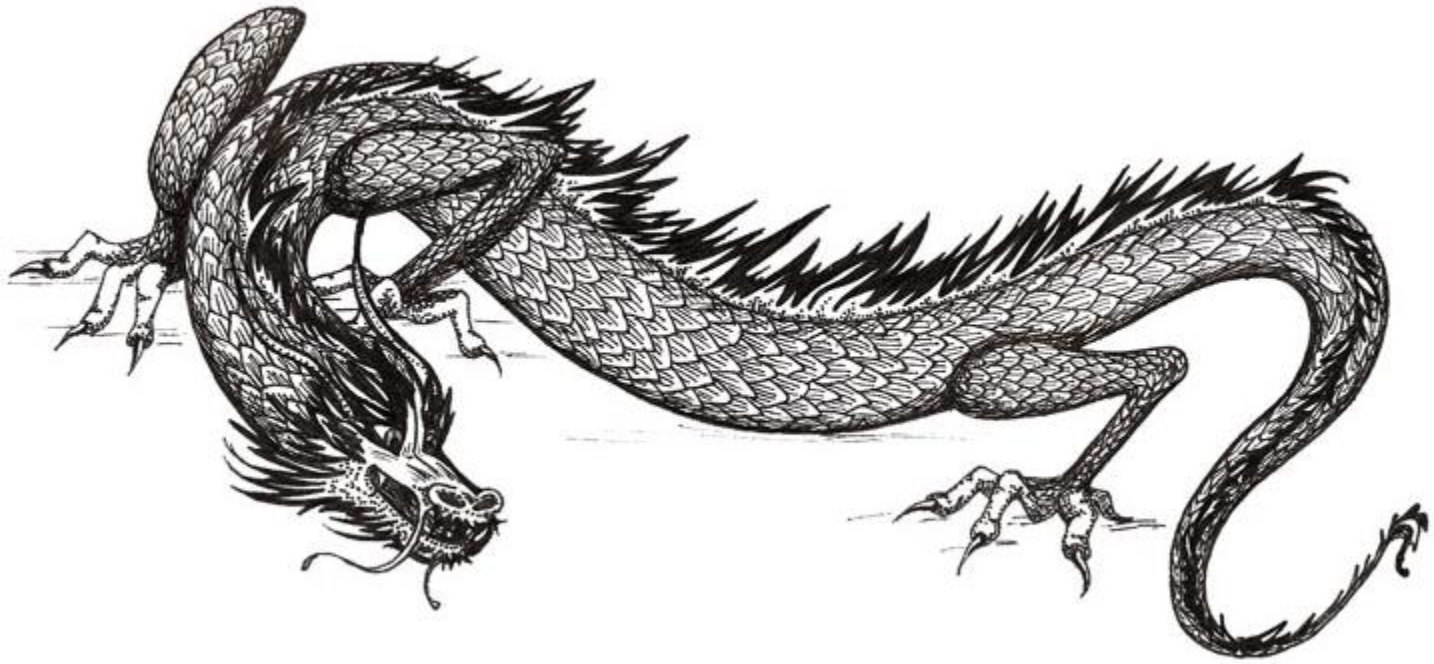


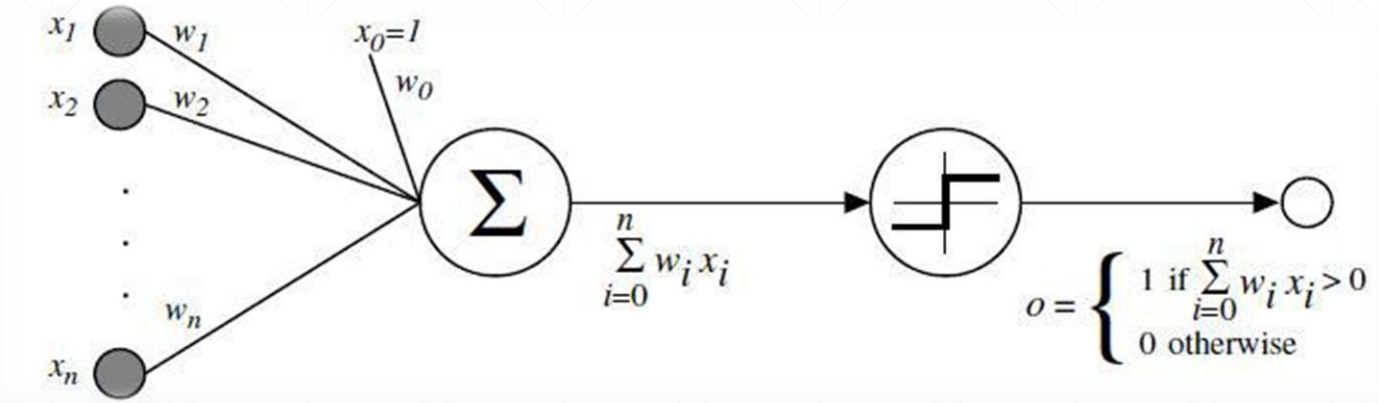
单输出感知机及其梯度

主讲：龙良曲

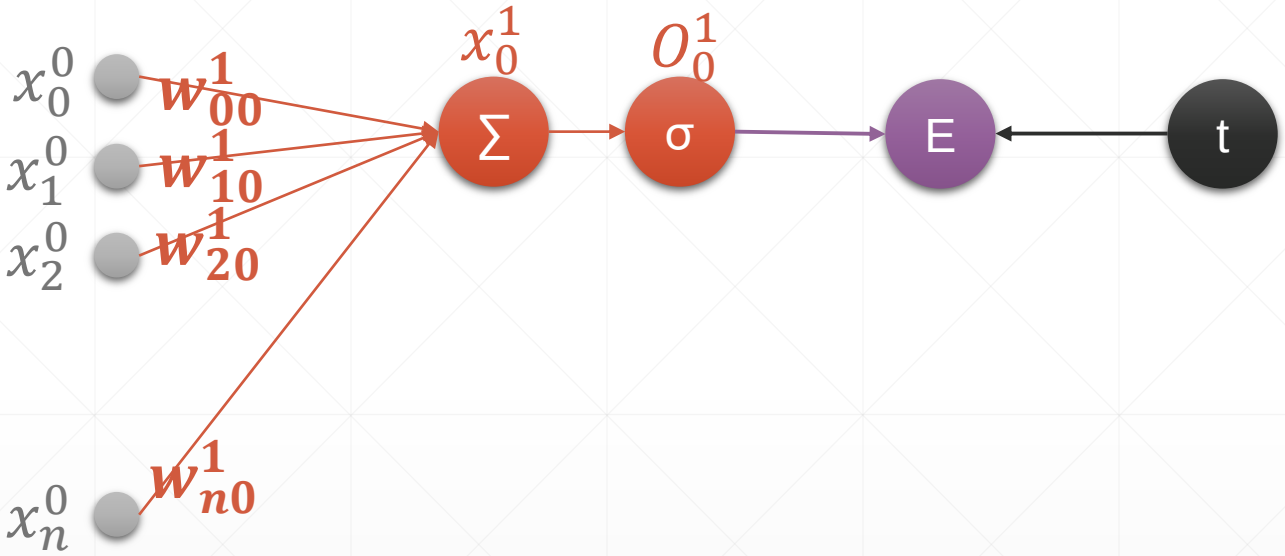
recap

- $y = XW + b$

- $y = \sum x_i * w_i + b$



Perceptron with Sigmoid+MSE



Derivative

$$E = \frac{1}{2}(O_0^1 - t)^2$$

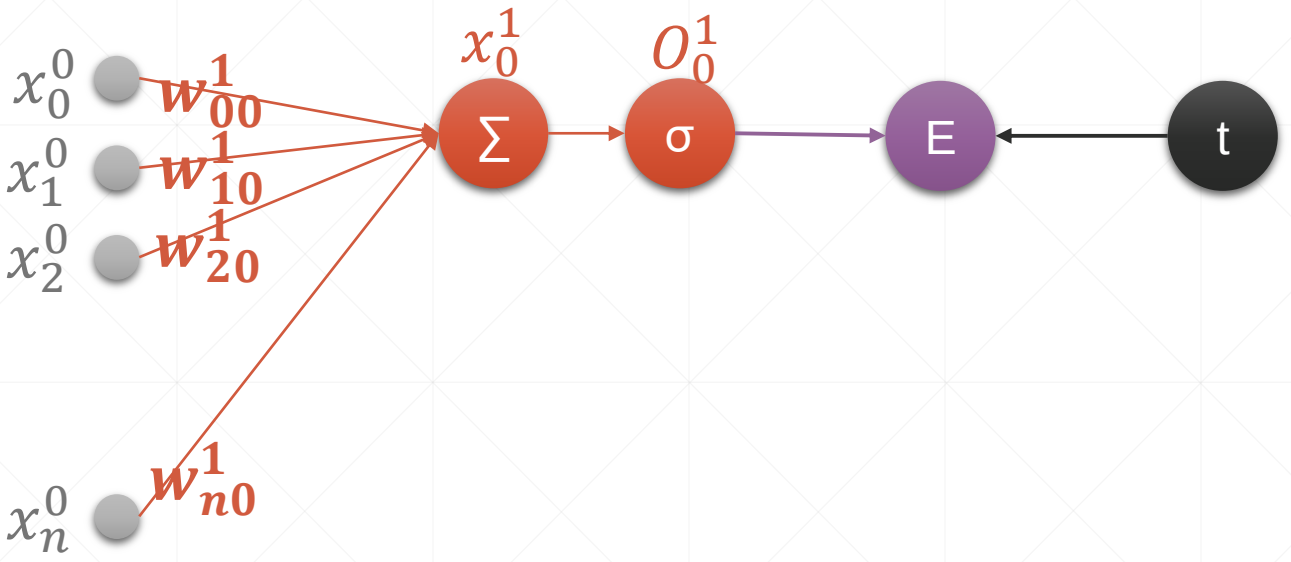
$$\frac{\partial E}{\partial w_{j0}} = (O_0 - t)\frac{\partial O_0}{\partial w_{j0}}$$

$$\frac{\partial E}{\partial w_{j0}} = (O_0 - t)\frac{\partial \sigma(x_0)}{\partial w_{j0}}$$

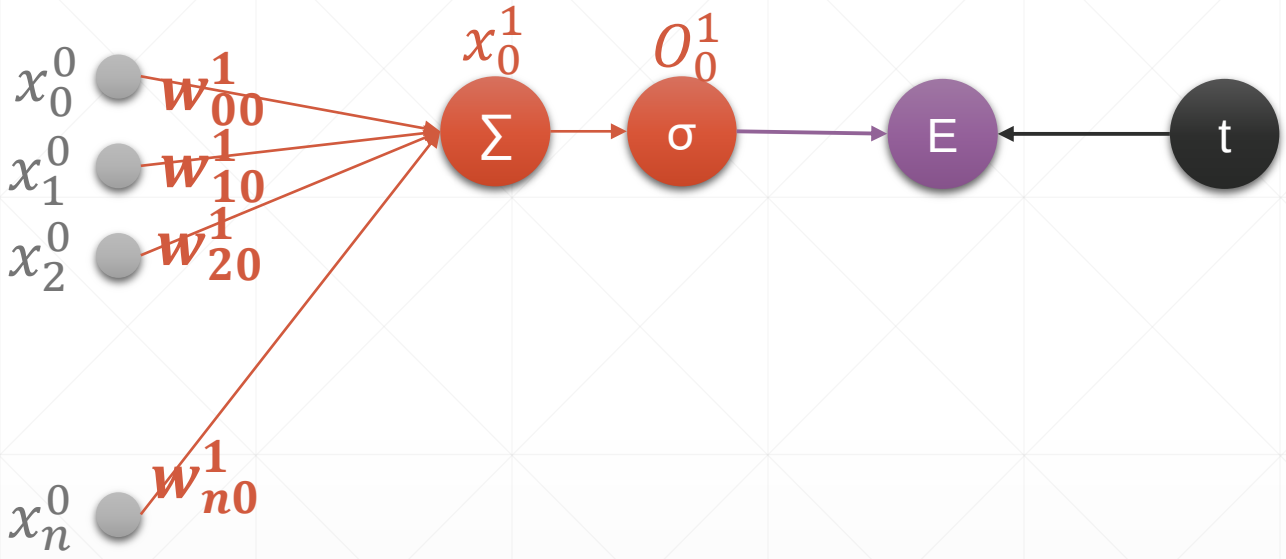
$$\frac{\partial E}{\partial w_{j0}} = (O_0 - t)\,\sigma(x_0)(1 - \sigma(x_0))\frac{\partial x_0^1}{\partial w_{j0}}$$

$$\frac{\partial E}{\partial w_{j0}} = (O_0 - t)O_0\,(1 - O_0)\frac{\partial x_0^1}{\partial w_{j0}}$$

$$\frac{\partial E}{\partial w_{j0}} = (O_0 - t)O_0\,(1 - O_0)\,x_j^0$$



Perceptron



$$\frac{\partial E}{\partial w_{j0}} = (O_0 - t) O_0 (1 - O_0) x_j^0$$

```python
x=tf.random.normal([1,3])
w=tf.ones([3,1])
b=tf.ones([1])
y = tf.constant([1])

with tf.GradientTape() as tape:
    tape.watch([w, b])
    logits = tf.sigmoid(x@w+b)
    loss = tf.reduce_mean(tf.losses.MSE(y, logits))

grads = tape.gradient(loss, [w, b])
w grad: tf.Tensor(
[[-0.00478814]
 [-0.00588211]
 [ 0.00186196]], shape=(3, 1), dtype=float32)
b grad: tf.Tensor([-0.00444918], shape=(1,), dtype=float32)
```

下一课时

多层感知机梯度

Thank You.