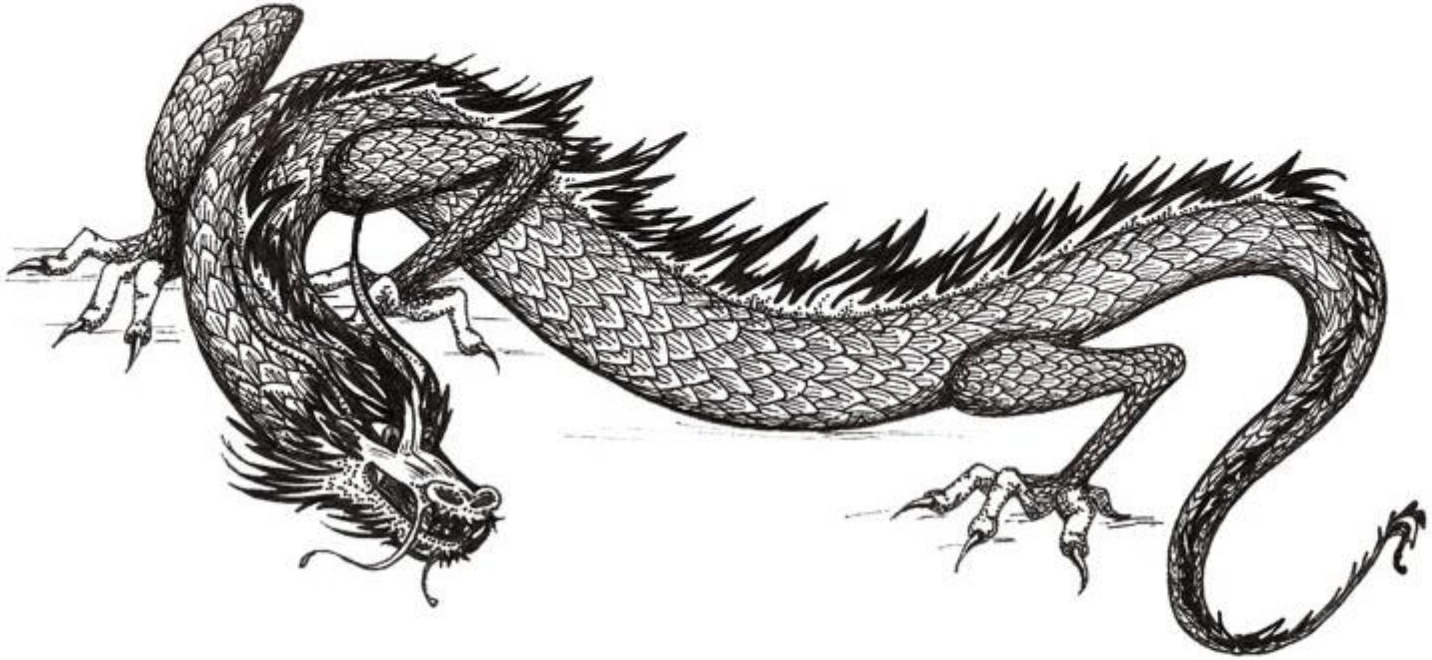


Loss及其梯度

主讲：龙良曲

Outline

- Mean Squared Error
- Cross Entropy Loss

MSE

- $\text{loss} = \sum[y - f_\theta(x)]^2$

- $\frac{\nabla loss}{\nabla \theta} = 2\sum[y - f_\theta(x)] * \frac{\nabla f_\theta(x)}{\nabla \theta}$

- $f_\theta(x) = sigmoid(XW + b)$

- $f_\theta(x) = relu(XW + b)$

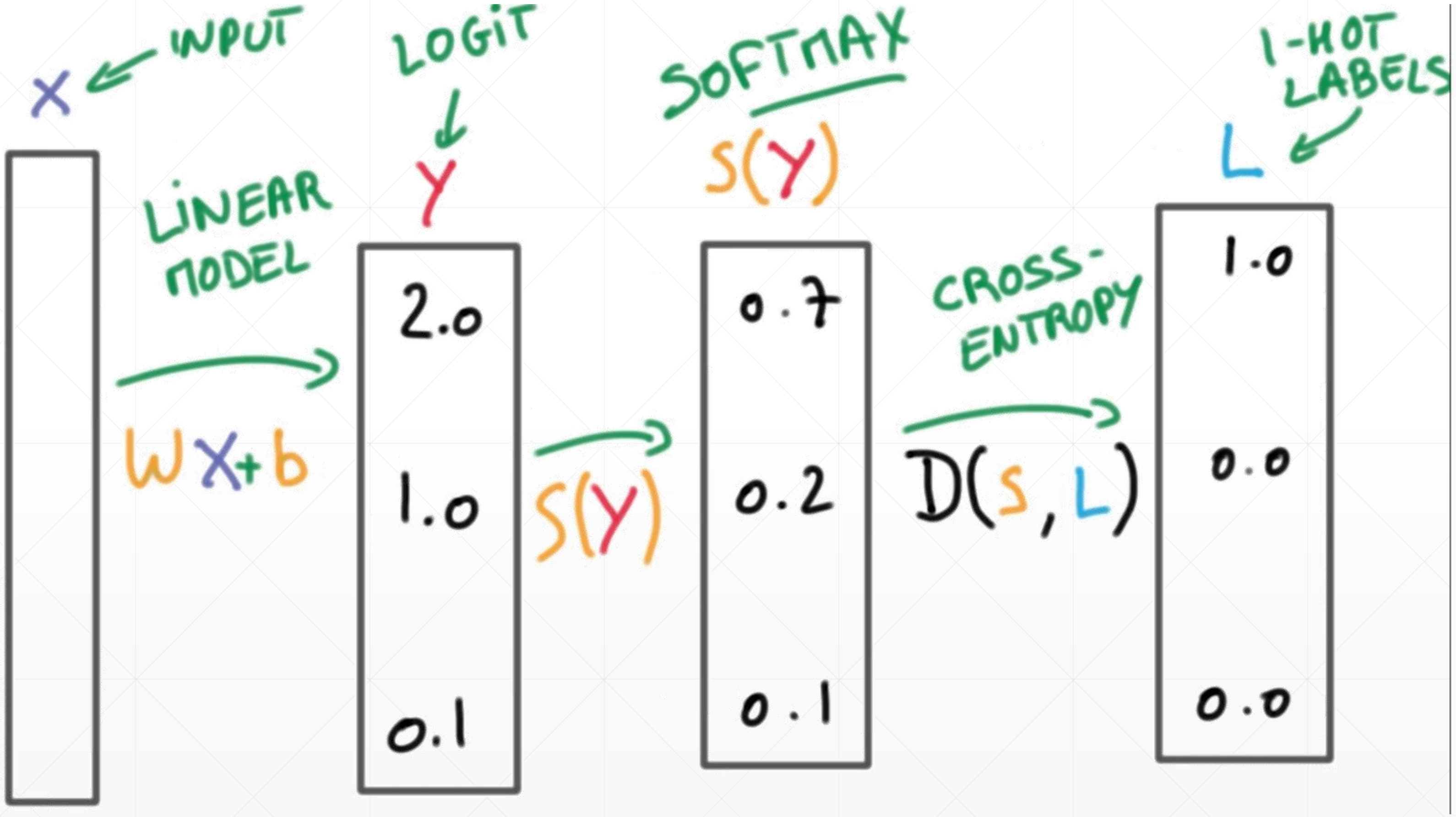
MSE Gradient

```
In [3]: x=tf.random.normal([2,4])
In [4]: w=tf.random.normal([4,3])
In [5]: b=tf.zeros([3])
In [6]: y=tf.constant([2,0])

In [9]: with tf.GradientTape() as tape:
   ...:     tape.watch([w,b])
   ...:     prob = tf.nn.softmax(x@w+b, axis=1)
   ...:     loss = tf.reduce_mean(tf.losses.MSE(tf.one_hot(y,depth=3), prob))

In [10]: grads = tape.gradient(loss, [w,b])
In [11]: grads[0]
[[-0.00967887, -0.00335512,  0.01303399],
       [-0.04446869,  0.06194263, -0.01747394],
       [-0.04530644,  0.01043231,  0.03487412],
       [ 0.02006017, -0.03638988,  0.0163297 ]]
In [12]: grads[1] # [-0.02585024,  0.06217915, -0.03632889]
```

Cross Entropy Loss



CrossEntropy

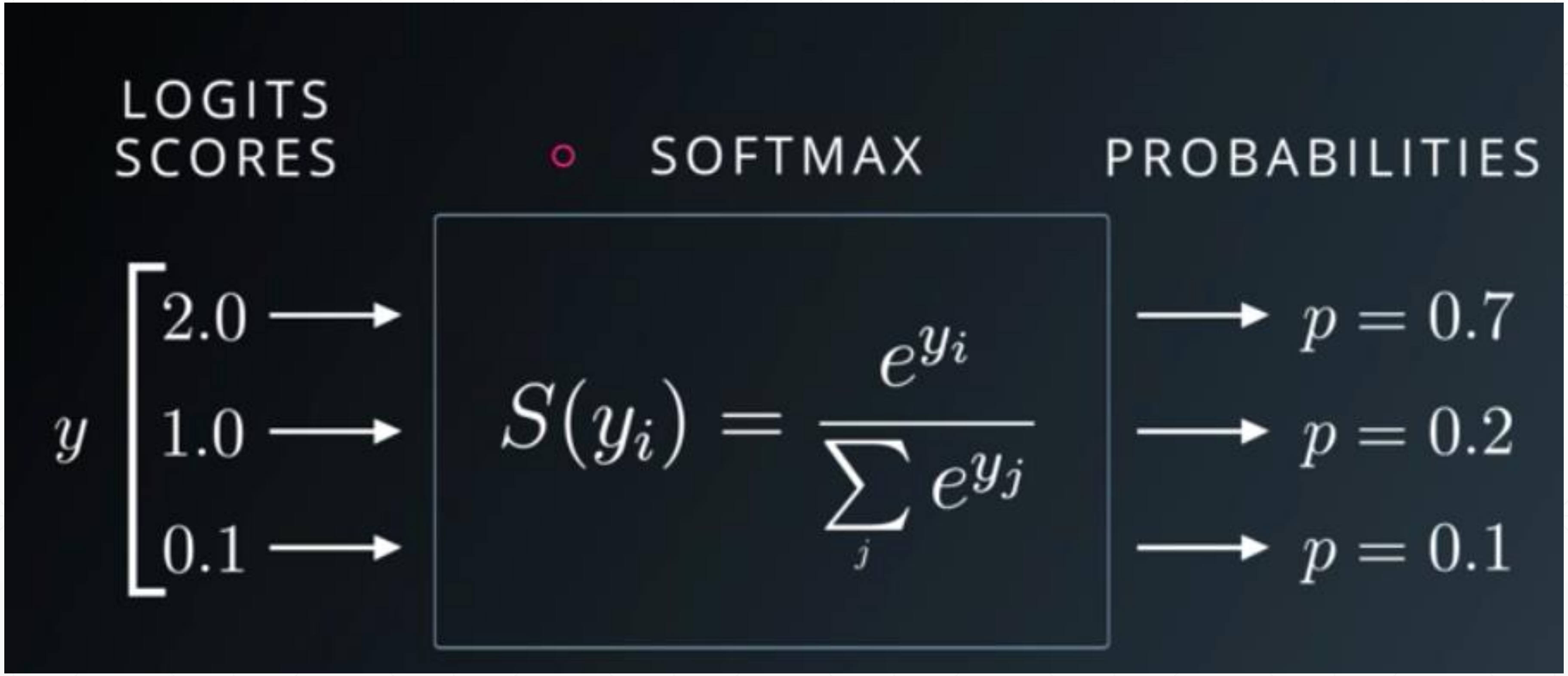
- $H([0,1,0],[p_0,p_1,p_2]) = D_{KL}(p|q) = -1\log p_1$

$$\frac{d}{dx}\log_2(x) = \frac{1}{x \cdot \ln(2)}$$

- p =softmax(*logits*)

Softmax

- soft version of max



Derivative

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$

$$\frac{\partial p_i}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j}$$

$$f(x) = \frac{g(x)}{h(x)}$$

$$f'(x) = \frac{g\prime(x)h(x) - h\prime(x)g(x)}{h(x)^2}$$

$$g(x) = e^{a_i}$$

$$h(x) = \sum_{k=1}^{N} e^{a_k}$$

when $i = j$

$$\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} = \frac{e^{a_i} \sum_{k=1}^{N} e^{a_k} - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2}$$

$$= \frac{e^{a_i} \left(\sum_{k=1}^{N} e^{a_k} - e^{a_j}\right)}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2}$$

$$= \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \times \frac{\left(\sum_{k=1}^{N} e^{a_k} - e^{a_j}\right)}{\sum_{k=1}^{N} e^{a_k}}$$

$$= p_i(1 - p_j)$$

Derivative

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$

$$\frac{\partial p_i}{\partial a_j} = \frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j}$$

$$f(x) = \frac{g(x)}{h(x)}$$

$$f'(x) = \frac{g\prime(x)h(x) - h\prime(x)g(x)}{h(x)^2}$$

$$g(x) = e^{a_i}$$

$$h(x) = \sum_{k=1}^{N} e^{a_k}$$

when $i \neq j$

$$\frac{\partial \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}}{\partial a_j} = \frac{0 - e^{a_j} e^{a_i}}{\left(\sum_{k=1}^{N} e^{a_k}\right)^2}$$

$$= \frac{-e^{a_j}}{\sum_{k=1}^{N} e^{a_k}} \times \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}}$$

$$= -p_j \cdot p_i$$

Derivative

$$\frac{\partial p_i}{\partial a_j} = \begin{cases} p_i(1-p_j) & if \quad i=j \\ -p_j.p_i & if \quad i \neq j \end{cases}$$

Or using Kronecker delta $\delta ij = \begin{cases} 1 & if \quad i=j \\ 0 & if \quad i \neq j \end{cases}$

$$\frac{\partial p_i}{\partial a_j} = p_i(\delta_{ij} - p_j)$$

Crossentropy gradient

```
In [3]: x=tf.random.normal([2,4])
In [4]: w=tf.random.normal([4,3])
In [5]: b=tf.zeros([3])
In [6]: y=tf.constant([2,0])

In [14]: with tf.GradientTape() as tape:
    ...:     tape.watch([w,b])
    ...:     logits = x@w+b
    ...:     loss =
 tf.reduce_mean(tf.losses.categorical_crossentropy(tf.one_hot(y,depth=3), logits,
                 from_logits=True))

In [15]: grads = tape.gradient(loss, [w,b])
<tf.Tensor: id=163, shape=(4, 3), dtype=float32, numpy=
array([[-0.08729011, -0.10937974,  0.19666985],
       [-0.22951077,  0.36995798, -0.14044718],
       [-0.3506433 , -0.2172048 ,  0.56784815],
       [ 0.08480322, -0.26216313,  0.17735994]], dtype=float32)>
In [17]: grads[1] # [-0.07538486,  0.51023775, -0.4348529 ]
```

下一课时

单输出感知机梯度

Thank You.