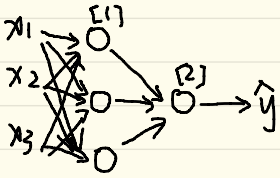
1.3 Shallow Neural Network

1.3 video 1

Neural Networks Overview



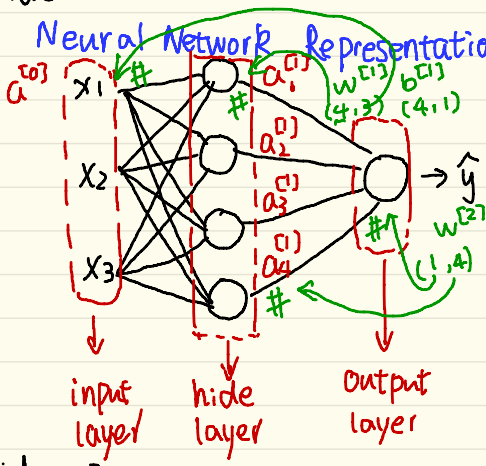
[ ]表示神经网络层数

$z^{[1]}$ $a^{[1]}$

$z^{[2]}$ $a^{[2]}$

1.3 video 2

Neural Network Representation



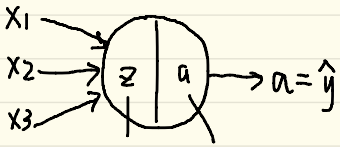
$$a^{[1]}=\begin{bmatrix}a_1^{[1]}\\a_2^{[1]}\\\vdots\\a_4^{[1]}\end{bmatrix}$$

惯例上 input layer不算正式层

∴这是个2层的NN

1.3 video 3

Computing a NN's Output



$z(x)=w^Tx+b;\ \sigma(z)=\frac{1}{1+e^{-z}}=a=\hat{y}$

$a_i^{[\Omega]}$ Ω: layer

i: node in layer

$$\begin{cases}z_1^{[1]}=w_1^{[1]T}\cdot x+b_1^{[1]};\ a_1^{[1]}=\sigma(z_1^{[1]})\\z_2^{[1]}=w_2^{[1]T}\cdot x+b_2^{[1]};\ a_2^{[1]}=\sigma(z_2^{[1]})\\z_3^{[1]}=w_3^{[1]T}\cdot x+b_3^{[1]};\ a_3^{[1]}=\sigma(z_3^{[1]})\\z_4^{[1]}=w_4^{[1]T}\cdot x+b_4^{[1]};\ a_4^{[1]}=\sigma(z_4^{[1]})\end{cases}$$

向量化:

$$z^{[1]}=w^{[1]}\cdot x+b^{[1]}$$

$$a^{[1]}=\sigma(z^{[1]})$$

給 x

$$\Rightarrow \begin{aligned} z^{[1]} &= W^{[1]} x + b^{[1]} \quad (4\times1 = 4\times3 \cdot 3\times1 + 4\times1) \\ a^{[1]} &= \sigma(z^{[1]}) \quad (4\times1) \\ z^{[2]} &= W^{[2]} x + b^{[2]} \quad (1\times1 = 1\times4 \cdot 4\times1 + 1\times1) \\ a^{[2]} &= \sigma(z^{[2]}) \quad (1\times1) \end{aligned}$$

$x = a^{[0]}$

$x = a^{[1]}$

4步算出最終 $y = a^{[2]}$

$$w^{[2]} = w^T$$

$$b^{[2]} = b$$

1.3 Video 4

Vectorizing across multiple examples

$a^{[2](i)}$ → example, a → layer

for i to m:

$$z^{[1](i)} = W^{[1]} x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

$$z^{[2](i)} = W^{[2]} x^{(i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma(z^{[2](i)})$$

$$\Rightarrow X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(i)} \\ | & | & & | \end{bmatrix}$$

$n_x \times m$

$$Z^{[1]} = \begin{bmatrix} | & | & \\ z^{[1](1)} & z^{[1](2)} & \cdots \\ | & | & \end{bmatrix}$$

hide nodes × m

$x = a^{[0]}$

$$\therefore \begin{aligned} Z^{[1]} &= W^{[1]} X + b^{[1]} \\ A^{[1]} &= \sigma(Z^{[1]}) \\ Z^{[2]} &= W^{[2]} A^{[1]} + b^{[2]} \\ A^{[2]} &= \sigma(Z^{[2]}) \end{aligned}$$

1.3 Video 5

Explanation for Vectorized Implemenation.

忽略 $b^{[1]} = 0$

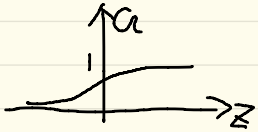
$$Z^{[1]} = W^{[1]} \cdot X \rightarrow (\text{input nodes}, m)$$

(Chidenodes, m) ↙ ↓ (hide nodes, input) nodes) ↗ ×

1.3 Video 6

Activation functions

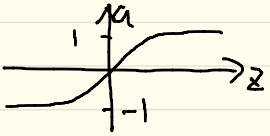
✗ $\sigma(z) = \frac{1}{1+e^{-z}}$



⇕

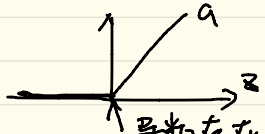
✗ $g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$

$= \tanh(z)$



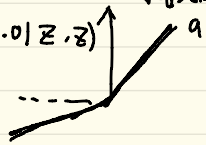
✓ $Relu(z) = \max(0, z)$

线性整流，快



导数存在 0.000…00

✓ $Leaky\ Relu(z) = \max(0.01z, z)$



1.3.Video7: 但不能用 linear(z)，线性激活函数只能用于 y 是实数 0~100…00 机器学习中估值房价之类的模型中

1.3 video 8

Derivatives of activation functions:

$\sigma'(z) = \sigma(z)(1-\sigma(z))$ $z=0$ 时 $\sigma'=0.5$

$a' = a(1-a)$ $z=\pm\infty$ 时 $\sigma'=0$

$g'(z) = 1-(g(z))^2$ $z=0$ 时 $g'=1$

$a' = 1-a^2$ $z=\pm\infty$ 时 $\sigma'=0$

$ReLU(z) = \max(0, z)$

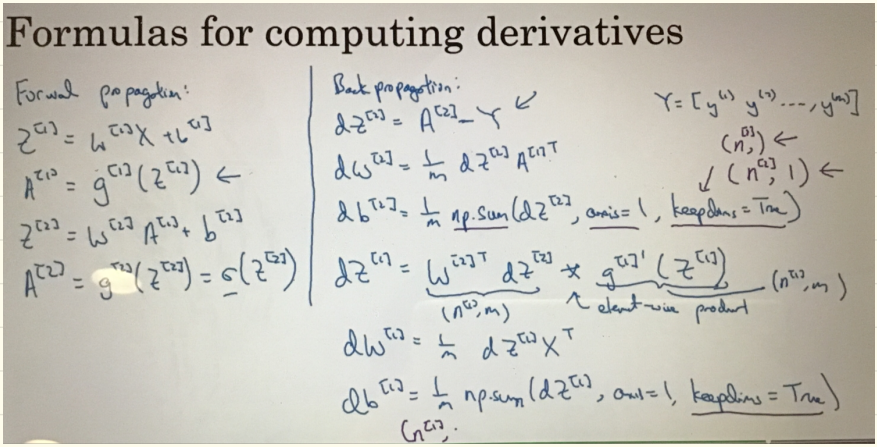
$ReLU'(z) = \begin{cases} 0 & z<0 \\ 1 & z\geq 0 \end{cases}$

$Leaky\ ReLU(z) = \max(0.01z, z)$

$Leaky\ ReLU'(z) = \begin{cases} 0.01 & z<0 \\ 1 & z\geq 0 \end{cases}$

1.3 video 9

Gradient descent for NN



1.3 video 11

Random Initialization

Set initial = 0 $w^{[1]} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ .. $w^{[1]} = \begin{bmatrix} \cdots \\ \vdots \\ \cdots \end{bmatrix}$ 对称

∴ 要随机初始化: $w^{[1]} = np.random.randn((2,2)) * 0.01$

$b^{[1]} = np.zero((2,1))$

初始值设小，以防斜率直为0梯度学习很慢

1.4