神经网络基础

1、二分类问题

- 样本 (x, y)，其中 $x \in R^{n_x}$，$y \in \{0, 1\}$
- 样本数量为 m，$\{(x^{(1)}, y^{(1)}\}, \{x^{(2)}, y^{(2)}\}, \cdots, \{x^{(m)}, y^{(m)}\}$
- 数据的矩阵情况

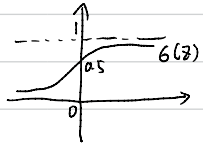
$$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{m} \Big\} n^x \qquad y = \underset{(R^{1\times m})}{[y^{(1)}, y^{(2)}, \cdots y^{(m)}]}$$

2、Logistic Regression

$\hat{y}$ 表示预测的结果，$\hat{y} = P(y=1|X)$

输入：$X \in R^{n_x}$，$w \in R^{n_x}$，$b \in R$　　输出：$\hat{y} = w^T x + b$

引入 Sigmoid 函数 $\hat{y} = Sigmoid(w^T x + b)$，$Sigmoid(z) = \frac{1}{1+e^{-z}}$　图像为



存在偏置 b，所以设置 $x_0 = 1$，即 $x \in R^{n_x+1}$

参数 θ 也修改为 $\theta = [\underbrace{\theta_0}_{b}, \theta_1, \cdots, \theta_{n_x}]$

3、Logistic Regression Cost function

loss function：

一般用 squared error 表示 loss function：$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

但在 LR 中很少使用，因为它是 non-convex，容易陷入 local optima

LR 的 loss function 为 $L(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$

- if $y=1$，$L(\hat{y}, y) = -\log \hat{y}$ ⟵ $\hat{y}$ 越大越好
- if $y=0$，$L(\hat{y}, y) = -\log(1-\hat{y})$ ⟵ $\hat{y}$ 越小越好
- 目标是最小化 loss function

cost function

全部样本的 loss function ⟶ cost function

$$J(w, b) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log(1-\hat{y}^{(i)})]$$

4. Gradient Descent

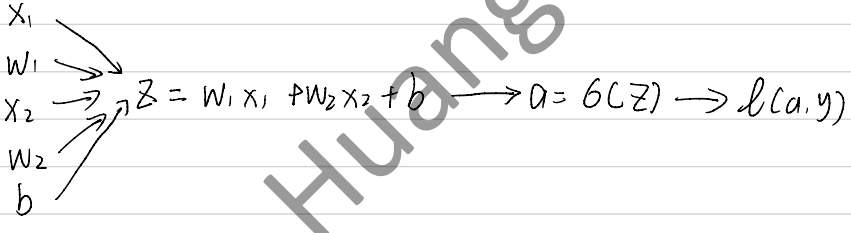
$$w := w - \alpha \frac{dJ(w)}{dw} \quad \left(\frac{\partial J(w,b)}{\partial w}\right) \qquad b := b - \alpha \frac{dJ(w)}{db} \quad \left(\frac{\partial J(w,b)}{\partial b}\right)$$

5. Gradient Descent in Logistic Regression

对于一个样本：$z = w^T x + b$

$\hat{y} = a = \sigma(z)$

$\mathcal{L}(a, y) = -(y \log a + (1-y) \log(1-a))$



对 dz 求导：

$$dz = \frac{\partial \mathcal{L}}{\partial z} = \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z} = \left(-\frac{y}{a} + \frac{1-y}{1-a}\right) \cdot a(1-a) = a - y$$

对 w_1, w_2 和 b 求导

$$dw_1 = \frac{\partial \mathcal{L}}{\partial w_1} = \frac{\partial \mathcal{L}}{\partial z} \cdot \frac{\partial z}{\partial w_1} = (a-y) x_1$$

$dw_2 = (a-y) x_2$, $db = a - y$

更新参数

$$w_1 := w_1 - \alpha dw_1 \qquad w_2 := w_2 - \alpha dw_2 \qquad b := b - \alpha db$$

6. m 个样本的 Gradient Descent

$a^{(i)} = \hat{y}^{(i)} = \sigma(z^{(i)}) = \sigma(w^T x^{(i)} + b)$

$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$

$dw_1 = \frac{1}{m} \sum_{i=1}^{m} x_1^{(i)} (a^{(i)} - y^{(i)})$ $\qquad dw_2 = \frac{1}{m} \sum_{i=1}^{m} x_2^{(i)} (a^{(i)} - y^{(i)})$

$db = \frac{1}{m} \sum_{i=1}^{m} (a^{(i)} - y^{(i)})$

7. Vectorization

$X: R^{n_x \times m}$ $W: R^{n_x \times 1}$ $b: R$ $Z: R^{1 \times m}$

$Z = W^T X + b$

$db = \frac{1}{m} \sum_{i=1}^{m} dz^{(i)} = \frac{1}{m}\, np.sum(dZ)$

$dW = \frac{1}{m} X dZ^T$

8. Logistic Regression cost function 的解释

预测输出: $\hat{y} = \sigma(W^T x + b)$，其中 $\sigma(z) = \frac{1}{1+e^{-z}}$

给定预测样本 x 下 $y=1$ 的概率 $\hat{y} = P(y=1 \mid x)$

if $y=1$: $P(y|x) = \hat{y}$

if $y=0$: $P(y|x) = 1-\hat{y}$

整合以上情况

$$P(y|x) = \hat{y}^{y}(1-\hat{y})^{(1-y)}$$

由于 log 函数严格单调，上述等式两边取 log

$$\log(P(y|x)) = \log \hat{y}^{y}(1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y)\log(1-\hat{y})$$

我们希望 $P(y|x)$ 越大越好，loss function 越小越好

$$L(\hat{y}, y) = -\log(P(y|x)) = -[y \log \hat{y} + (1-y)\log(1-\hat{y})]$$

假设样本是 IID 的，我们希望结果的联合概率越大越好

$$\max \prod_{i=1}^{m} P(y^{(i)}|x^{(i)})$$

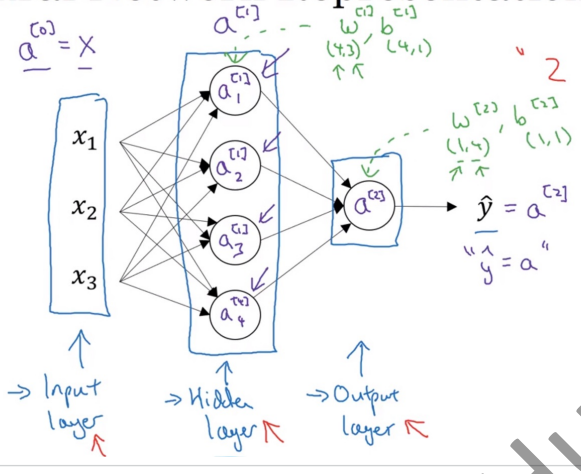
引入 log 函数，上式整化

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})]$$

浅层神经网络

1、简单双层 NN

Neural Network Representation



$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$, $a_1^{[1]} = \sigma(z_1^{[1]})$

$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$, $a_2^{[1]} = \sigma(z_2^{[1]})$

$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}$, $a_3^{[1]} = \sigma(z_3^{[1]})$

$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}$, $a_4^{[1]} = \sigma(z_4^{[1]})$

$$\begin{bmatrix} — & w_1^{[1]} & — \\ — & w_2^{[1]} & — \\ — & w_3^{[1]} & — \\ — & w_4^{[1]} & — \end{bmatrix} \begin{bmatrix} x_1^{[1]} \\ x_2^{[1]} \\ x_3^{[1]} \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

$W^{[1]}$ (4×3) $b^{[1]}$ (4×1)

$$a^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

为了得到最终结果，只需

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

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

$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$

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

2、多样本向量化

$x \longrightarrow a^{[2]} = \hat{y}$

$x^{(1)} \longrightarrow a^{[2](1)} = \hat{y}^{(1)}$

$\vdots$

$x^{(m)} \longrightarrow a^{[2](m)} = \hat{y}^{(m)}$

for i = 1 to m:

$z^{[1](i)} = w^{[1]} x^{(i)} + b^{[1]}$, $a^{[1](i)} = \sigma(z^{[1](i)})$,

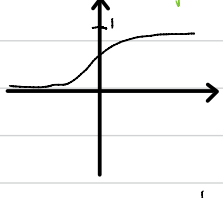
$z^{[2](i)} = w^{[2]} a^{[1](i)} + b^{[2]}$, $a^{[2](i)} = \sigma(z^{[2](i)})$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \quad Z^{[1]} = \begin{bmatrix} | & | & & | \\ z^{[1](1)} & z^{[1](2)} & \cdots & z^{[1](m)} \\ | & | & & | \end{bmatrix} \quad W^{[1]} = \begin{bmatrix} — \\ — \\ — \\ — \end{bmatrix}$$

$$W^{[1]}X^{(1)} = \begin{bmatrix} | \\ z^{[1](1)} \\ | \end{bmatrix} \quad W^{[1]}X^{(2)} = \begin{bmatrix} | \\ z^{[1](2)} \\ | \end{bmatrix} \quad \cdots \quad W^{[1]}X^{(m)} = \begin{bmatrix} | \\ z^{[1](m)} \\ | \end{bmatrix}$$

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

3. Activation functions

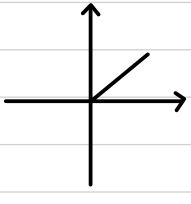


Sigmoid: $a = \frac{1}{1+e^{-z}}$

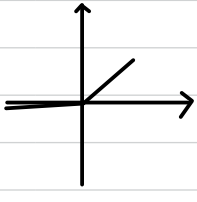
$a' = a(1-a)$

tanh: $a = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$

$a' = 1-a^{2}$



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



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

Sigmoid: $[0,1]$，一般只用于二元分类

tanh: $[-1,1]$，表现优于 Sigmoid，能起到归一化的作用

尽量选择 ReLU: Sigmoid 与 tanh 在 $|z|$ 很大时，梯度较小，而 ReLU 克服了这点

Leaky ReLU: $z<0$ 时，梯度下降仍不为 0

为什么需要非线性 Activation function？

如果不使用，那么神经网络只是一个单纯的线性组合，不加去掉隐藏层

4、NN的梯度下降

参数：$W^{[1]}$，$b^{[1]}$，$W^{[2]}$，$b^{[2]}$

输入层特征数量：$n_x = n^{[0]}$

隐藏层单元个数：$n^{[1]}$

输出层单元个数：$n^{[2]} = 1$

$W^{[1]}$的维度$(n^{[1]}, n^{[0]})$　　$b^{[1]}$的维度$(n^{[1]}, 1)$

$W^{[2]}$的维度$(n^{[2]}, n^{[1]})$　　$b^{[2]}$的维度$(n^{[2]}, 1)$

Cost function：$J(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}, y)$，$\hat{y} = a^{[2]}$

Back Propagation：（逐元素乘积）

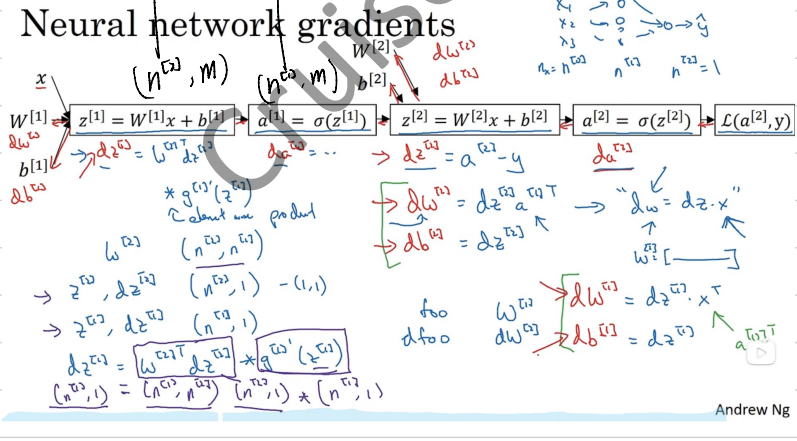
$dz^{[2]} = A^{[2]} - Y$　　$dW^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$

$db^{[2]} = \frac{1}{m}\, np.sum(dz^{[2]}, axis=1, keepdims=True)$

$dz^{[1]} = W^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$　　$dW^{[1]} = \frac{1}{m} dz^{[1]} x^T$

$db^{[1]} = \frac{1}{m}\, np.sum(dz^{[1]}, axis=1, keepdims=True)$

$(n^{[1]}, m)$　$(n^{[2]}, m)$



5、随机初始化

如果将神经元的W全部设置相同，那么每个神经元所计算出的梯度也相同。因

此无论经过多少次迭代，它们仍然是对称的。那么这将毫无意义。

$W^{[1]} = np.random.randn((2,2)) * 0.01$

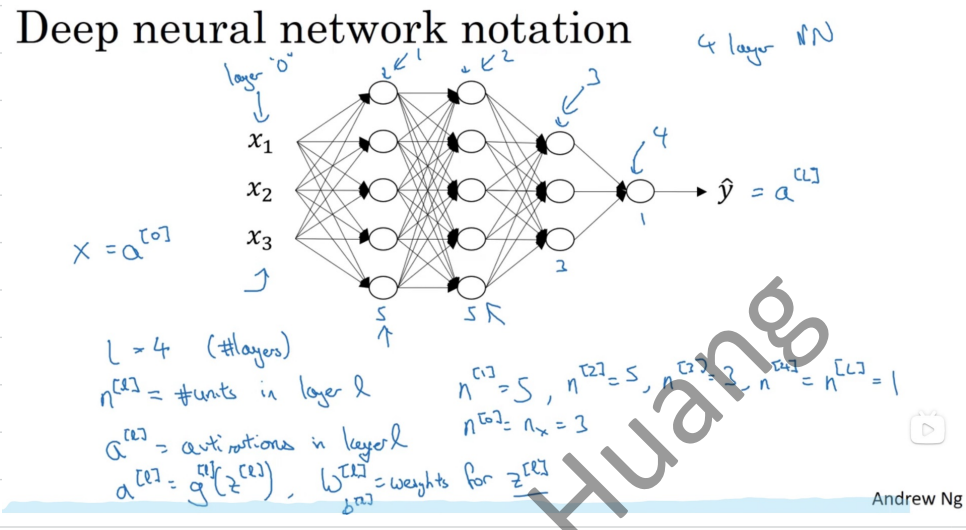
$b^{[1]} = np.zeros((2,1))$

乘0.01是因为如果使用 Sigmoid / tanh，使用较小的W值可以使Z位于0附近，得到较大的梯度，加快收敛速度。

使用 ReLU / Leaky ReLU 则不存在这一问题。

深层神经网络

1、DNN的符号



2、计算矩阵的维度

$W^{[l]}: (n^{[l]}, n^{[l-1]})$　　$b^{[l]}: (n^{[l]}, 1)$

$dW^{[l]}: (n^{[l]}, n^{[l-1]})$　　$db^{[l]}: (n^{[l]}, 1)$

$Z^{[l]}: (n^{[l]}, 1)$　　$A^{[l]}: (n^{[l]}, 1)$

} 对于单个样本

$(n^{[l]},1)$ $(n^{[l]},1)$ $(n^{[l]},n^{[l-1]})$ $(n^{[l]},1)$

对于单个样本：$A^{[l]} = g(Z^{[l]}) = W^{[l]}A^{[l-1]} + b^{[l]}$

对于多个样本：$A^{[l]} = g(Z^{[l]}) = W^{[l]}A^{[l-1]} + b^{[l]}$

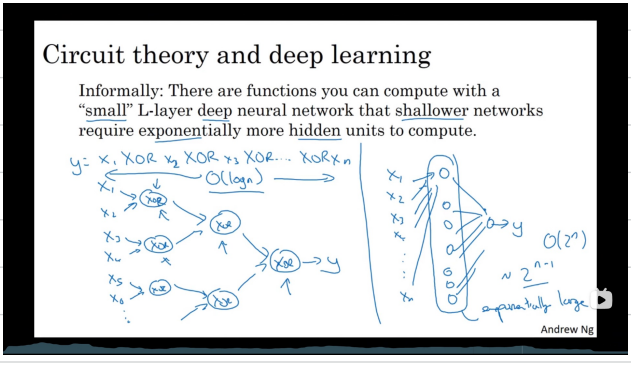
$(n^{[l]},m)$ $(n^{[l]},m)$ $(n^{[l]},n^{[l-1]})$ $(n^{[l-1]},m)$ $(n^{[l]},1)$ ← 使用了广播

3、为什么使用深层表示

对于人脸识别，不同层的神经网络可以提取或者组合不同的特征，可以实现从局部到整体，由简单到复杂的模型学习。

对于语音识别，不同层可以分别学习音调、音素、单词等信息。

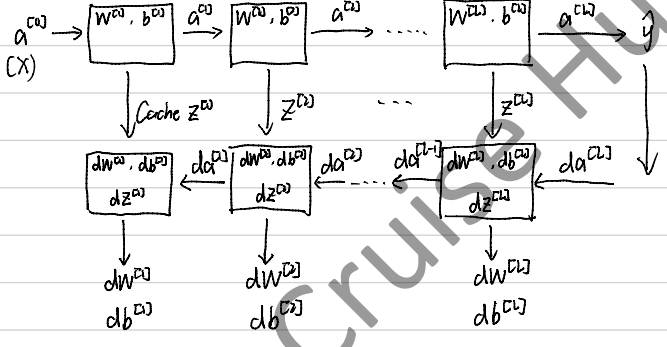
对于电路逻辑运算



左边：$1+2+\cdots+2^{\log_2(n)-1}=n-1$

右边：2^{n-1}

4. 构建 DNN



5. 参数和超参数

参数：$W^{[1]}$，$b^{[1]}$，$W^{[2]}$，……

超参数：控制参数的输出值

Learning rate：α

iteration：N

hidden layer：L

hidden Units：$n^{[1]}$，$n^{[2]}$，…

choice of activation function：Sigmoid，ReLU，tanh …