

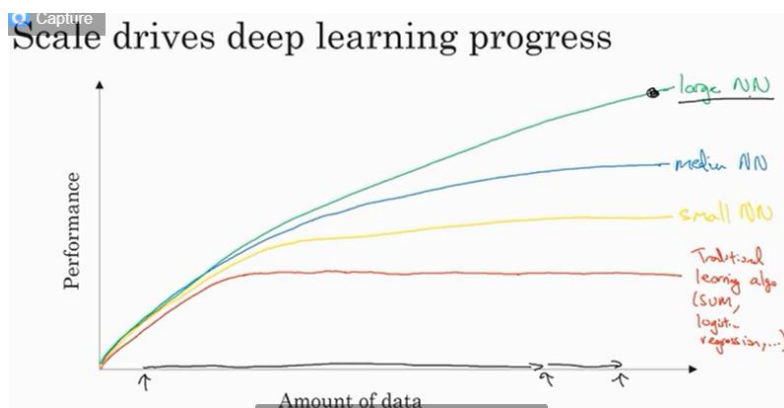
Part 1 Neural Network

1. Always connected with supervised learning
2. Application & network type

Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,...,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation
Image, Radar info	Position of other cars	Autonomous driving

Handwritten notes: Standard NN, CNN, RNN, Custom/Hybrid, (standard nn: fc nn)

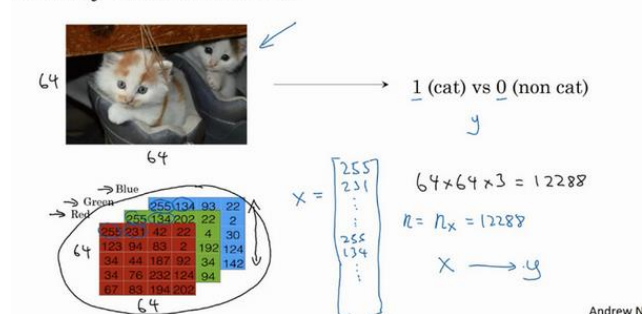
3. Data type:
 - Structured data: columns of features
 - Unstructured data: audio/image/text -> harder
4. Advantage of nn



Data are easier to collect nowadays
 More data & bigger nn → better performance
 (small amount of data -> little difference or even worse)

5. Activation function: from 'sigmoid' to 'relu' ---- sigmoid: small gradient -> training process slow
6. Image data → 特征向量

Binary Classification



7. Loss function:
 - 一般只找得到局部最优
 - Logistic regression: 一般不用目标与实际的差的平方和，损失函数如下：

$L(\hat{y}, y) = -y \log(\hat{y}) - (1-y) \log(1-\hat{y})$ (损失函数: 针对单个样本, $y = \text{sigmoid}(wx+b)$)

$$J(w, b) = \frac{1}{n} \sum_{i=1}^n L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{n} \sum_{i=1}^n (-y^{(i)} \log \hat{y}^{(i)} - (1-y^{(i)}) \log(1-\hat{y}^{(i)}))$$
 (代价函数: 针对训练集)

求导:

$$z = wx + b \rightarrow a = \sigma(z) \rightarrow L = -y \log(a) - (1-y) \log(1-a) \rightarrow J = \frac{1}{n} \sum L$$

单样本: $\frac{\partial L}{\partial a} = -\frac{y}{a} + \frac{1-y}{1-a}$
 $\frac{\partial a}{\partial z} = -\left(\frac{1}{1+e^{-z}}\right)^2 (1-e^{-z}) = a^2 e^{-z}$
 $\frac{\partial z}{\partial wx} = x$
 $\frac{\partial z}{\partial b} = 1$

多样本: $\frac{\partial J}{\partial L^{(i)}} = \frac{1}{n}$
 $\frac{\partial L^{(i)}}{\partial a^{(i)}} = -\frac{y^{(i)}}{a^{(i)}} + \frac{1-y^{(i)}}{1-a^{(i)}}$
 $\frac{\partial L^{(i)}}{\partial z^{(i)}} = a^{(i)2} e^{-z^{(i)}}$
 $\frac{\partial z^{(i)}}{\partial wx^{(i)}} = x^{(i)}$
 $\frac{\partial z^{(i)}}{\partial b} = 1$

$$\frac{\partial J}{\partial wx^{(i)}} = \frac{1}{n} \sum_i \frac{\partial L^{(i)}}{\partial wx^{(i)}} = \frac{1}{n} \sum_i \left[\left(-\frac{y^{(i)}}{a^{(i)}} + \frac{1-y^{(i)}}{1-a^{(i)}} \right) a^{(i)2} e^{-z^{(i)}} x^{(i)} \right]$$

$$\frac{\partial J}{\partial b} \text{ 同理}$$

用矩阵表示

$$\frac{\partial J}{\partial W} = \frac{1}{n} \text{sum} \left(\left(-\frac{Y}{A} + \frac{1-Y}{1-A} \right) A^2 e^{-Z} X, \text{axis}=0 \right)$$

$$\frac{\partial J}{\partial b} = \frac{1}{n} \text{sum} \left(\left(-\frac{Y}{A} + \frac{1-Y}{1-A} \right) A^2 e^{-Z}, \text{axis}=0 \right)$$

原理:

逻辑回归损失函数与代价函数原理分析

目标: 最大化 $P(y=1|x)$ 和 $P(y=0|x)$

① 令: $P(y=1|x) = y$
 $P(y=0|x) = 1-y$ } \Rightarrow 合并 $P(y|x) = y^y (1-y)^{1-y}$ } $y=0 \rightarrow 1-y$
 $y=1 \rightarrow y$

② 由于 $\log(x)$ 是单增函数 \Rightarrow 最大化 $P(y|x) \Leftrightarrow$ 最大化 $\log P(y|x)$
 $\log P(y|x) = \log(y^y (1-y)^{1-y}) = y \log y + (1-y) \log(1-y)$

③ 由于最大化 P 与最小化 loss 对应 \Rightarrow 增加负号
 $loss = -(\text{---})$

④ 对于多个样本 \Rightarrow 最大化整体概率 $\Rightarrow \pi P(y|x)$
 $\log \pi P(y|x) = \sum \log P(y|x)$

⑤ 增加负号并进一步缩放

8. 避免 bug 技巧:

一维数组(n,) 行/列向量(n,1)/(1,n)

尽量避免一维数组的使用

对于不确定结构的数据最好都是用 reshape

9. 神经网络表示技巧:

N 层神经网络: n 不包含输入层

第 n 层第 i 个参数: n-上标, i-下标

Eg: 2 层神经网络中隐藏层四个神经元的激活值

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}$$

10. 激活函数:

函数	表达式	导数	产生	函数特点	缺点	优点	使用情景
Sigmoid	$\sigma(z) = \frac{1}{1+e^{-z}}$	$a(1-a)$	和生物学相似	(0,1) 经过(0,0.5)	1. Z 过大/过小时导数过小->训练慢 2. 输出都大于 0, 容易出现 zigzag 现象	1. 满足输出结果在[0,1]之间的条件, 常常用于输出层	一般不用了
Tanh	$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$1-a^2$	Sigmoid 平移缩放得到	(-1,1) 经过(0,0)	1. 和 sigmoid 一样容易出现导数过小现象	1. 训练效果好于 sigmoid 2. 均值为 0, 可以更好进行优化	隐藏层, 效果都挺好
ReLU	$\text{Max}(0, z)$	$z>0:1$ $z=0:$ 未定义 $z<0:0$		非饱和	1. 容易出现神经元坏死现象 ($z<0$ 时) \rightarrow lr 不要太大 2. 均值非 0	1. 不会出现 tanh 和 sigmoid 中饱和后梯度过小的问题 2. $z<0$ 时为 0, 可以增加稀疏程度, 避免过拟合 3. 计算复杂度低	默认激活函数, 不知道用什么的时候可以用
Leaky ReLU	$\text{Max}(az, z)$ (a 是参数)	$z>0:1$ $z=0:$ 未定义 $z<0:a$	ReLU 变形得到	$z<0$ 时有较小的斜度		1. 都存在斜率	很少使用

(有关 sigmoid 的 zigzag 解释:

https://www.zhihu.com/question/50396271?from=profile_question_card)

线性激活函数与非线性激活函数:

线性激活函数: 使得多层神经网络丧失意义

证明:

线性激活函数.

若 $g(z) = z$:

对于多层神经网络: $a^{[n]} = W^{[n]} a^{[n-1]} + b^{[n]}$

$$\begin{aligned}
 a^{[n]} &= W^{[n]} a^{[n-1]} + b^{[n]} \\
 &= W^{[n]} (W^{[n-1]} a^{[n-2]} + b^{[n-1]}) + b^{[n]} \\
 &= W^{[n]} W^{[n-1]} a^{[n-2]} + W^{[n]} b^{[n-1]} + b^{[n]} \\
 &= \underline{W^{[n]} W^{[n-1]}} a^{[n-2]} + \underline{W^{[n]} b^{[n-1]} + b^{[n]}} \\
 &= \underline{W'} a^{[n-2]} + \underline{b'}
 \end{aligned}$$

多层的意义, 与单层等价

使用情况: 不可在隐藏层使用, 只能在输出层使用。对于预测问题 (结果并非在 0, 1 之间的可以使用)

激活函数的导数的推导:

激活函数的导数推导

① sigmoid $a = g(z) = \frac{1}{1+e^{-z}}$

$$g'(z) = -\frac{1}{(1+e^{-z})^2} (-e^{-z}) = a(1-a)$$

② tanh $a = g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = -1 + \frac{2e^z}{e^z + e^{-z}} = -1 + \frac{2}{1+e^{-2z}}$

$$\begin{aligned}
 g'(z) &= -\frac{2}{(1+e^{-2z})^2} (-2e^{-2z}) = \frac{4e^{-2z}}{(1+e^{-2z})^2} = \frac{4e^{-2z}}{1+2e^{-2z}+e^{-4z}} = 1 - \frac{(1-e^{-2z})^2}{(1+e^{-2z})^2} \\
 &= 1 - \frac{1^2 - 2e^{-2z} + e^{-4z}}{1^2 + 2e^{-2z} + e^{-4z}} = 1 - a^2
 \end{aligned}$$

11. 参数的初始化

W:

不可全部初始化为 0 \rightarrow 所有的神经元相同, 输出相同, 更新相同

使用 $a * \text{np.random.randn}$ \rightarrow 高斯分布, a 一般为一个较小的数, 如 0.01 (避免 W 过大导致波动过大/在 tanh 和 sigmoid 中达到饱和 \rightarrow 如果没有使用这两个激活函数可以大一点)

B: 可以全部直接初始化为 0

12. 前向传播和后向传播:

前向传播: $\text{input}: a^{[n-1]} \rightarrow \text{output}: a^{[n]}, \text{cache}(z^{[n]})(w^{[n]}, b^{[n]})$

Debug 技巧:

核对导数与原矩阵维度

13. 深度学习的理解:

1. 深度(deep and shallow)增加的作用 \gg 神经元(small and big)数量的增加 (深度的增加 \Rightarrow 神经元成指数增加, 通过增加深度能够更容易实现复杂函数的学习)

2. 学习内容:

基本单元 \rightarrow 逐渐组合 \rightarrow 查找学习更大单元 \rightarrow 得到结果

Eg: 人脸识别: edges \rightarrow 面部器官 \rightarrow 部分脸 \rightarrow 全脸

3. 越深 \neq 越好:

4.

14. 超参数:

a) 什么是超参数? 影响了最后 W 与 b 的结果

lr、iterations、隐藏层数量 L 、隐藏单元数量 n 、activation function 激活函数的选择

b) 如何寻找最佳超参数? 尝试尝试再尝试