

神经网络与深度学习

1652613白皓月 ilab暑期实习笔记

2.1 二分类

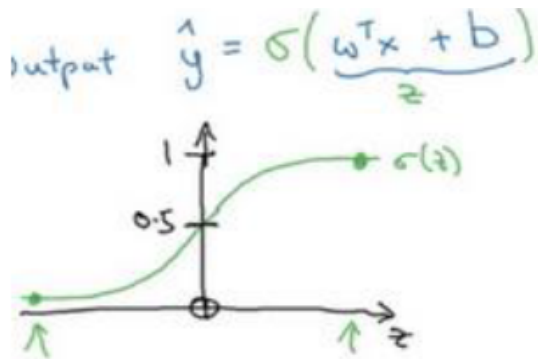
逻辑回归是用于二分类的算法，识别是否是猫咪为二分类

主要介绍了一些符号的说明

X表示所有训练数据的输入值，Y表示所有训练数据集的输出值

2.2 逻辑回归

\hat{y} 表示的是这是一只猫的概率多大



$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

If z large $\sigma(z) \approx \frac{1}{1+0}$

If z large negative number

$$\sigma(z) = \frac{1}{1 + e^{-z}} \approx \frac{1}{1 + \text{large number}} \approx 0$$

任务是去让机器学习w和b，使得对y=1有很好的估计

Given $\{(\underline{x^{(1)}}, \underline{y^{(1)}}), \dots, (\underline{x^{(m)}}, \underline{y^{(m)}})\}$, want $\underline{\hat{y}^{(i)}} \approx \underline{y^{(i)}}$.

可以使用额外的特征 x_0 来使只有一个参数

2.3 逻辑回归的代价函数

通过训练代价函数来获得参数

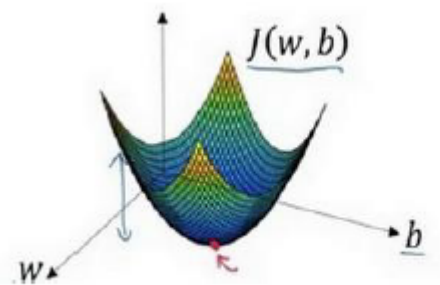
逻辑回归中不是使用平方差，而是使用另一个和对数有关的函数

损失函数 (loss function) 是在单个训练样本定义，代价函数是对m个样本的损失函数取平均

逻辑回归是个非常小的神经网络

2.4 梯度下降法

在测试集上，通过最小化代价函数来训练参数w和b



26

代价函数必须为凸函数

逻辑回归的代价函数（成本函数） $J(w, b)$ 是含有两个参数的。

$$w := w - \alpha \frac{\partial J(w, b)}{\partial w} \quad b := b - \alpha \frac{\partial J(w, b)}{\partial b}$$

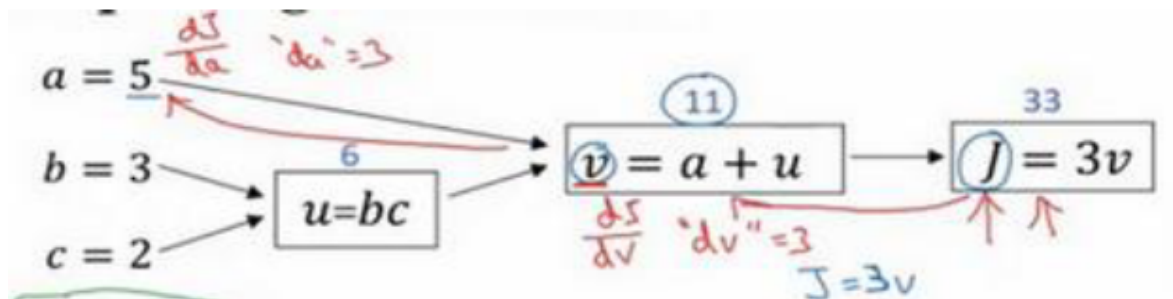
round表示求偏导

2.7

前向过程计算网络输出，反向传输计算梯度或者导数

计算图用于处理代价函数？

2.8



利用链式法则，看某个变量增加0.001时，输出变量增加多少

dvar表示最终关心的变量的导数，其他比如dj/dv就用dv表示

一步一步，先计算dv，计算da也要通过dv

2.9逻辑回归梯度下降

使用计算图实现逻辑回归中的梯度下降

对于单个样本来说，计算出dz后，通过 $x_1 * dz$ 计算 dw_1 ，然后更新 $w_1 = w - \alpha dw_1$

2.10 m个样本的梯度下降

微分求平均

向量化摆脱for循环

2.11, 2.12, 2.13向量化

np.dot算是,其他np.function也有，避免写循环

np.dot可以算矩阵乘法np.dot(A,v)

还有exp, log, abs, max等函数

消除有关多个特征值的循环：只定义一个dw

算z时n个训练集的循环也可以消除： $Z=np.dot(w.T,X)+b$ ，得到行向量

b也会自动扩展

2.14向量化逻辑回归的梯度输出

```
J = 0, dw1 = 0, dw2 = 0, db = 0
for i = 1 to m:
    z(i) = wTx(i) + b
    a(i) = σ(z(i))
    J += -[y(i) log a(i) + (1 - y(i)) log(1 - a(i))]
    dz(i) = a(i) - y(i)
    dw1 += x1(i) dz(i)
    dw2 += x2(i) dz(i)
    db += dz(i)
J = J/m, dw1 = dw1/m, dw2 = dw2/m
db = db/m
```

我们的目标是不使用 **for** 循环，而是向量，我们可以这么做：

$$Z = w^T X + b = np.dot(w.T, X) + b$$

$$A = \sigma(Z)$$

$$dZ = A - Y$$

$$dw = \frac{1}{m} * X * dz^T$$

$$db = \frac{1}{m} * np.sum(dZ)$$

$$w := w - a * dw$$

$$b := b - a * db$$

推荐一次迭代一次梯度下降

2.15广播

A.sum(axis=0)表示运算按列执行

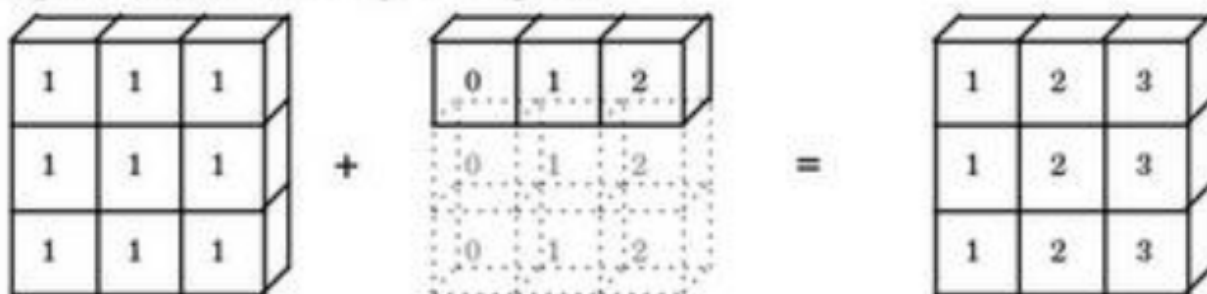
写代码不确定维度时可以使用reshape

两数组后缘维度轴长度相符或者其中一方轴长度为1时可以广播

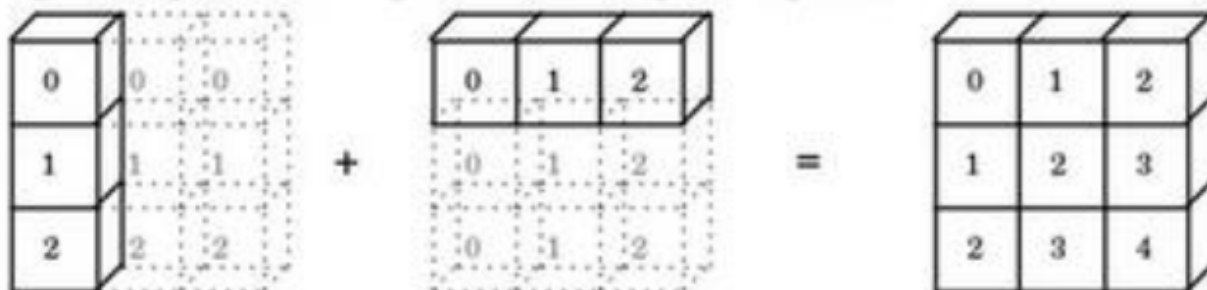
`np.arange(3) + 5`



`np.ones((3, 3)) + np.arange(3)`



`np.arange(3).reshape((3, 1)) + np.arange(3)`



`np.random.randn(5)`生成的是一维数组，不如 `(5,1)` 这样直接声明为向量表现清楚，先不使用，用列向量或者行向量。

可以用`assert`确保

3.2

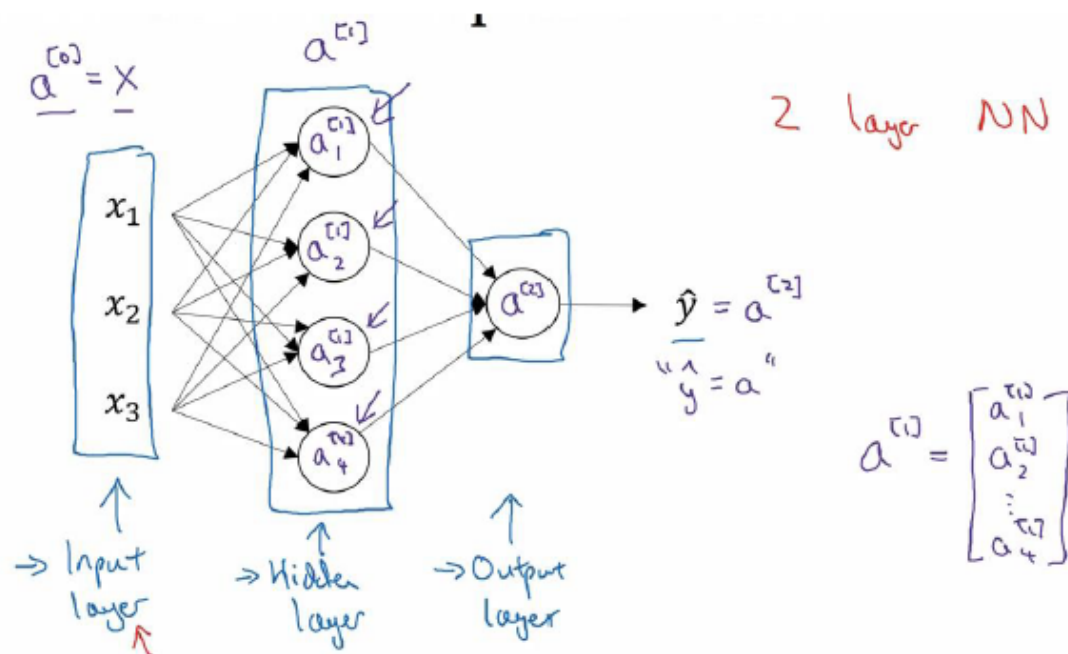


图 3.2.2

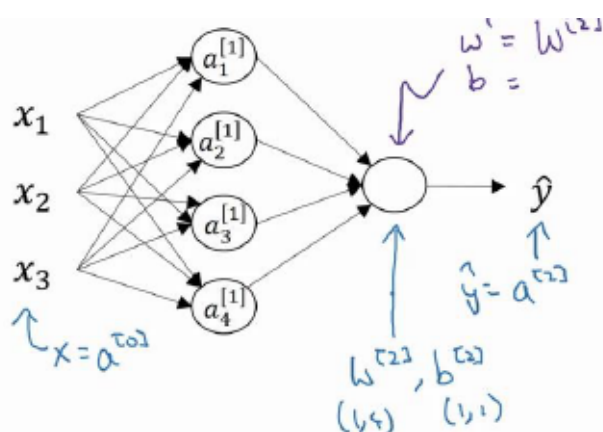
算作两层神经网络

分别有 w_1, b_1, w_2, b_2 这些参数

3.3 计算输出

逻辑回归和神经网络的不同体现在

这个神经网络知识将逻辑回归里的那个计算做了四遍



Given input x :

$$\rightarrow z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$\begin{matrix} (4,1) & (4,3) & (3,1) & (4,1) \end{matrix}$

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

$\begin{matrix} (4,1) & (4,1) \end{matrix}$

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

$\begin{matrix} (1,1) & (1,4) & (4,1) & (1,1) \end{matrix}$

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

$\begin{matrix} (1,1) & (1,1) \end{matrix}$

第一层可以理解，第二层是为了什么呢。以第一节例子来看，神经网络就是，基于逻辑回归重复使用了两次该模型得到上述例子的神经网络

3.4 多样本向量化

不同列表示的不同训练样本，不同行表示不同的隐藏单元

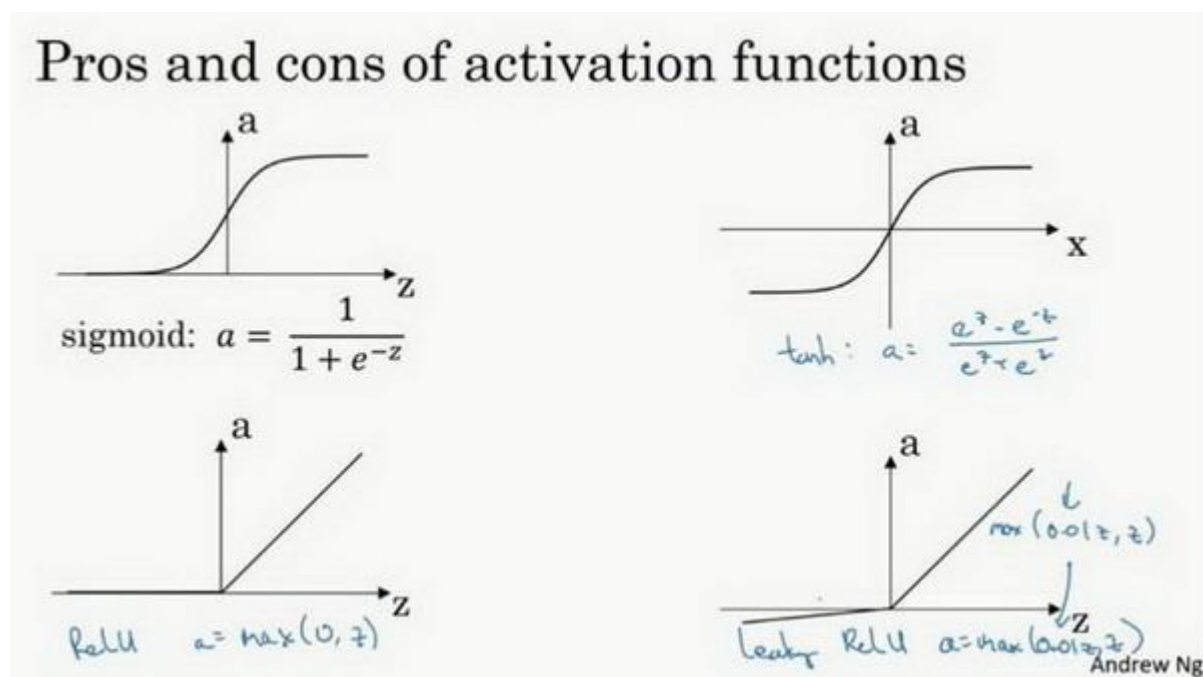
不同层的话就是不同矩阵了

3.5 向量化实现的解释

本质是用每一层输入 $A^{[i-1]}$ 计算 $A^{[i]}$ ，即使网络再深，基本还是重复这两步运算

3.6 激活函数

在神经网络的不同层中，激活函数可以不同



卷积神经网络

1.2 边缘检测示例

Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3	<u>0</u>	<u>1</u>	<u>2</u>	<u>7</u>	<u>4</u>
1	<u>5</u>	<u>8</u>	<u>9</u>	<u>3</u>	<u>1</u>
2	<u>7</u>	<u>2</u>	<u>5</u>	<u>1</u>	<u>3</u>
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6

"convolution"

1	0	-1
1	0	-1
1	0	-1

3x3
filter

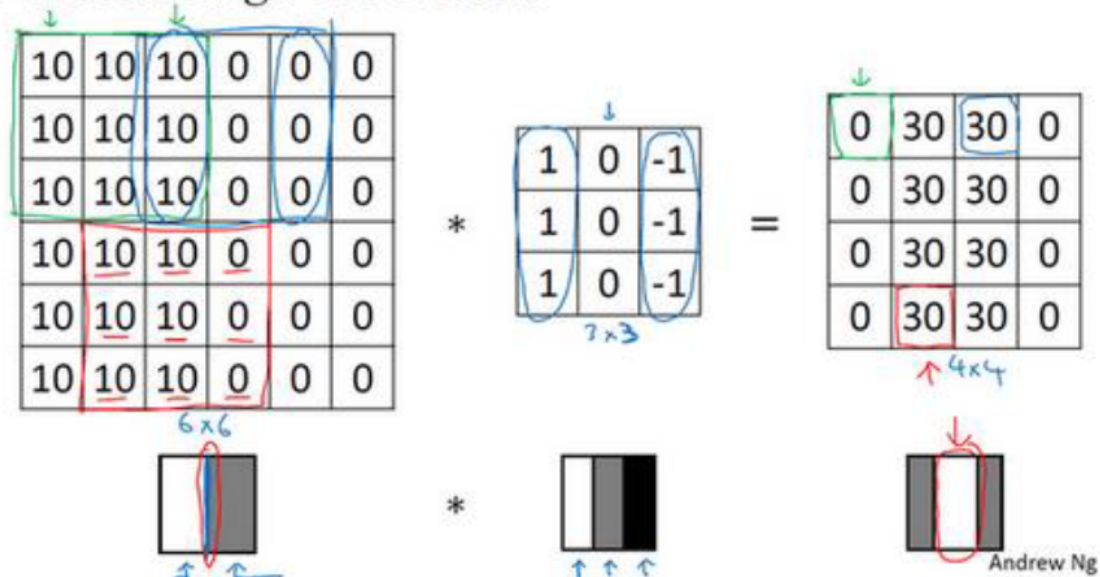
=

<u>-5</u>	<u>-4</u>	0	<u>8</u>
<u>-10</u>	-2	2	<u>3</u>
0	-2	-4	-7
-3	-2	-3	<u>-16</u>

4x4

卷积过程就是用过滤器套在左图上，将每个数相加

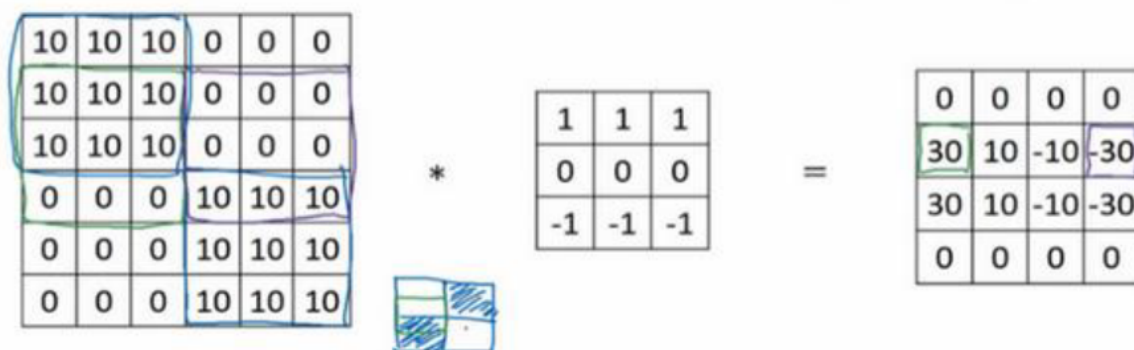
Vertical edge detection



垂直剪测的实例，只不过检测出来有点粗了

1.3更多边缘检测

看亮度变化



30表示正边，-30表示存在负边

可以利用神经网络去学习过滤器的这9个参数

1.4Padding

防止边缘信息丢失，四周填充像素

分为valid卷积和same卷积

valid不填充，same填充使输入和输出一样

习惯都是奇数维过滤器