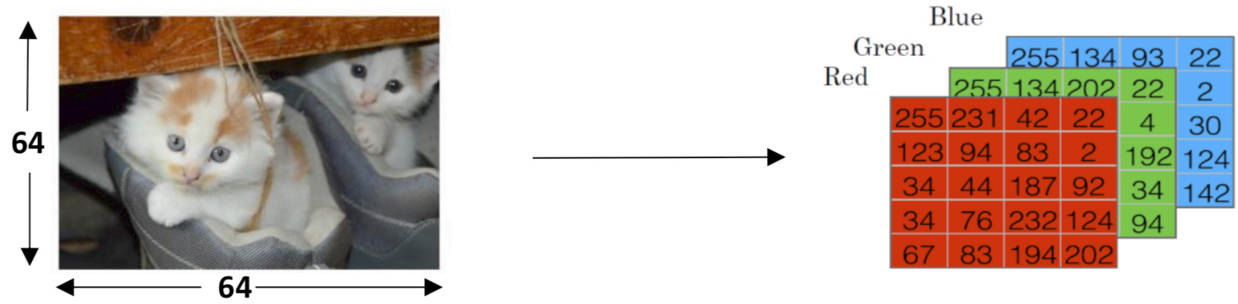
1. Binary Classification

一个典型的二分类问题，识别图像内容是否是一只猫：



符号说明：

1. 一个样本记为：$(x^{(i)},y^{(i)})$，其中$x^{(i)}\in\mathbb{R}^n$，$y\in\{0,1\}$
2. m个样本：$\{(x^{(1)},y^{(1)}),(x^{(2)},y^{(2)}),\cdots,(x^{(m)},y^{(m)})\}$
3. 所有样本的输入记为X：

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

   $X\in\mathbb{R}^{n\times m}$，和很多表达不一样，但只要在推导过程中前后统一就行。

4. 所有样本的输出记为Y：

$$Y=[y^{(1)},y^{(2)},\cdots,y^{(m)}]$$

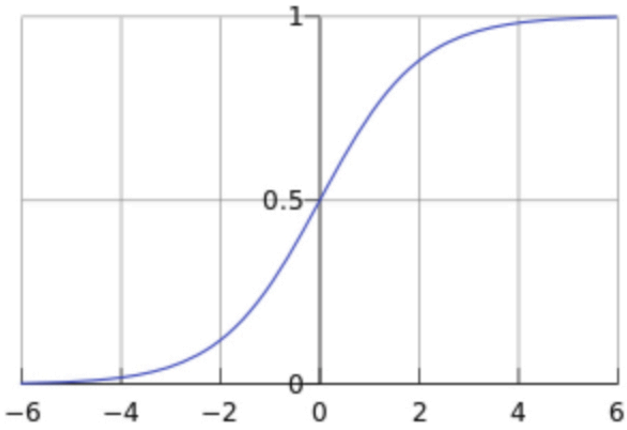
   $Y\in\mathbb{R}^{1\times m}$.

2. Logistic Regression

LR是一种用于二分类的算法，同时也可以将它看做是最小的神经单元。

2.1 Sigmoid函数及性质

$$\begin{align}
\sigma(z) &= \frac{1}{1+e^{-z}} \tag{1}\\
\sigma'(z) &= \sigma(z)(1-\sigma(z)) \tag{2}\\
\sigma(-z) &= 1-\sigma(z) \tag{3}\\
[\log\sigma(x)]' &= 1-\sigma(x) \tag{4}\\
[\log(1-\sigma(x))]' &= -\sigma(x) \tag{5}
\end{align}$$



2.2 LR and Cost Function

参数：$w \in \mathbb{R}^{n \times 1}, b \in \mathbb{R}$

输入：$x^{(i)}$

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

损失函数：

$$
\begin{aligned}
J(w,b) &= \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \\
&= -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]
\end{aligned}
$$

损失函数理解：

- 如果$y^{(i)} = 1$：$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\log\hat{y}^{(i)}$，此时$\hat{y}^{(i)}$应接近1；
- 如果$y^{(i)} = 0$：$\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\log(1-\hat{y}^{(i)})$，此时$\hat{y}^{(i)}$应接近0；

2.3 GD

符号定义：

$$
\begin{aligned}
\mathrm{d}w &= \frac{\partial J(w,b)}{\partial w} \\
\mathrm{d}b &= \frac{\partial J(w,b)}{\partial b}
\end{aligned}
$$

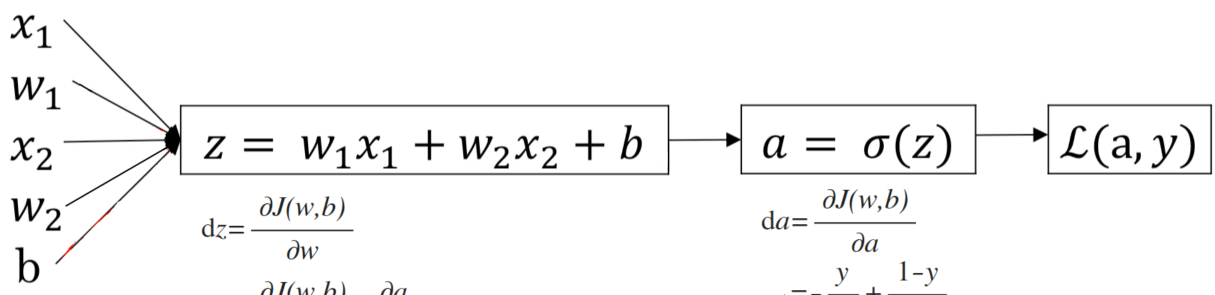
利用梯度下降更新参数时，设学习率(learning rate) 为α.

则参数更新公式为：

$$
\begin{aligned}
w &:= w - \alpha \cdot \mathrm{d}w \\
b &:= b - \alpha \cdot \mathrm{d}b
\end{aligned}
$$

2.4 Computation Graph

计算题示例：



$$dz=\frac{\partial J(w,b)}{\partial w}=\frac{\partial J(w,b)}{\partial a}\cdot\frac{\partial a}{\partial z}=a-y$$

$$da=\frac{\partial J(w,b)}{\partial a}=-\frac{y}{a}+\frac{1-y}{1-a}$$

$$\begin{cases} w_1:=w_1-\alpha\cdot dw_1 \\ w_1:=w_1-\alpha\cdot dw_1 \\ b\ :=b-\alpha\cdot db \end{cases}$$

$$dw_1=\frac{\partial J(w,b)}{\partial w_1}=x_1\cdot dz \qquad dw_2=\frac{\partial J(w,b)}{\partial w_2}=x_2\cdot dz \qquad db=\frac{\partial J(w,b)}{\partial b}=dz$$

2.5 LR on m examples

m个样本的训练过程：

$J=0;\ dw_1=0;\ dw_2=0;\ db=0$

$for\ \ iter=1\ \ to\ \ iter\_num:$ loop1

$\quad for\ \ i=1\ \ to\ \ m:$ loop2

$\qquad z^{(i)} = w^T x^{(i)} + b$

$\qquad a^{(i)} = \sigma(z^{(i)})$

$\qquad J \mathrel{+}= -[y^{(i)}\log a^{(i)}+(1-y^{(i)})\log(1-a^{(i)})]$

$\qquad dz^{(i)} = a^{(i)} - y^{(i)}$

$\qquad dw_1 \mathrel{+}= x_1^{(i)}\cdot dz^{(i)}$

$\qquad dw_2 \mathrel{+}= x_2^{(i)}\cdot dz^{(i)}$ loop3

$\qquad db \mathrel{+}= dz^{(i)}$

$\quad J \mathrel{/}= m$

$\quad dw_1 \mathrel{/}= m;\ dw_2 \mathrel{/}= m;\ db \mathrel{/}= m$

$\quad w_1 := w_1 - \alpha\cdot dw_1$

$\quad w_2 := w_2 - \alpha\cdot dw_2$

$\quad b := b - \alpha\cdot db$

缺点： 抛开最外层的迭代，里面也有两层循环，使用for循环处理，效率太低！

2.6 Vectorization

一种好的解决办法是将for循环改写成向量运算。

使用numpy计算向量内积时的一个小细节：

```
import numpy as np

X = np.array([[1, 2, 3]])
Y = np.array([[0, 1, 0]])

# 方式一
ret1 = np.dot(X, Y.T)

# 方式二
ret2 = np.sum(X * Y)
print(ret1)
print(ret2)
```

输出:

```
[[2]]
2
```

用向量/矩阵运算代替for循环：

$$
\begin{aligned}
&for\ \ iter=1\ \ to\ \ iter\_num: \\
&\qquad Z = w^{\mathrm{T}}X + b \\
&\qquad A = \sigma(Z) \\
&\qquad \mathrm{d}Z = A - Y \\
&\qquad \mathrm{d}w = \frac{1}{m}X\mathrm{d}Z^{\mathrm{T}} \\
&\qquad \mathrm{d}b = \frac{1}{m}np.sum(\mathrm{d}Z) \\
&\qquad w := w - \alpha\cdot\mathrm{d}w \\
&\qquad b := b - \alpha\cdot\mathrm{d}b
\end{aligned}
$$

3. 问题

3.1 LR的损失函数为什么用交叉熵而不是平方损失?

若用均方差误差来表示LR的损失，则$\mathcal{L}(w,b)$记作：

$$
\begin{aligned}
z &= w^T x + b && (1)\\
\hat{y} &= \sigma\left(z\right) && (2)\\
\mathcal{L}(w,b) &= \frac{1}{2}(\hat{y}-y)^2 && (3)
\end{aligned}
$$

则对w和b的偏导数记作：

$$
\begin{aligned}
\frac{\partial \mathcal{L}(w,b)}{\partial w} &= (\hat{y}-y)\sigma'(z)x && (4)\\
\frac{\partial \mathcal{L}(w,b)}{\partial b} &= (\hat{y}-y)\sigma'(z) && (5)
\end{aligned}
$$

可以看到w,b的更新速率与当前的预测值sigmoid函数的导数有关，由sigmoid函数图形可知，当预测值接近0或1时，$\sigma'(z)\approx 0$， 用梯度下降时，参数更新很慢！

而如果用交叉熵损失，

$$
\begin{aligned}
z &= w^T x + b && (1)\\
\hat{y} &= \sigma\left(z\right) && (2)\\
\mathcal{L}(w,b) &= -\frac{1}{m}\sum y\log\hat{y} + (1-y)\log(1-\hat{y}) && (3)
\end{aligned}
$$

对w求偏导：

$$
\frac{\partial \mathcal{L}(w,b)}{\partial w} = \frac{1}{m}\sum x\left(\sigma(z)-y\right)
$$

可以看到，没有收到σ导数的影响。