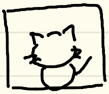
1.2 Course 1 Neural Networks and Deep Learning week2 Logistic Regression as a Neural Network

1.2 Video 1:

Binary Classification

- Example cat : 1 not cat : 0

64 
64

R G B 像素展开 $x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ 255 \\ \vdots \end{bmatrix}$

$64 \times 64 \times 3 = 12288$

$n_x = 12288$ 特征向量的维度

(x, y) $x \in \mathbb{R}^{n_x}$ $y \in \{0, 1\}$

m training example : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots (x^{(m)}, y^{(m)})\}$

M_{train} = # training examples ; M_{test} = # test examples

$$X = \begin{bmatrix} | & | & & | \\ x_1 & x_2 & \cdots & x_m \\ | & | & & | \end{bmatrix} \updownarrow n_x, \quad \xleftrightarrow{m}$$

$X \in \mathbb{R}^{n_x \times m}$ 向量维度 × 向量训练个数

$X.\text{shape}(n_x, m)$ python

$Y = [y^{(1)}, y^{(2)}, y^{(3)} \cdots y^{(m)}]$

$Y.\text{shape}(1, m)$ python

1.2 video 2:

Logistic Regression

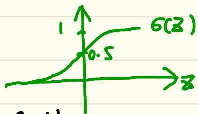
- Given x, want $\hat{y} = P(y=1|x)$ $0 \le \hat{y} \le 1$

Parameter : $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$

Output $\hat{y} = \sigma(\underbrace{w^T x + b}_{z})$

$\downarrow$

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



if $z \to \infty$
$\sigma(z) \to 1$
if $z \to -\infty$
$\sigma(z) \to 0$

另一种方式: $x_0 = 1$ $x \in \mathbb{R}^{n_x+1}$

$\hat{y} = \sigma(\theta^T x)$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \to b \\ \left.\begin{matrix} \\ \\ \\ \end{matrix}\right\} w \end{matrix}$$

1.2 Video 3:

Logistic Regression Cost Function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma = \frac{1}{1+e^{-z}}$ $z^{(i)} = w^T x^{(i)} + b$

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

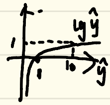
Loss (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y}-y)^2$ > 尽量小，衡量 $\hat{y}$ 与真实 y 的误差

$\mathcal{L}(\hat{y}, y) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$ small

error a single training example

If $y=1 \Rightarrow \mathcal{L}(\hat{y}, y) = -\log\hat{y} \rightarrow \log\hat{y}$ large $\rightarrow \hat{y}$ large

$\rightarrow \hat{y} \in \{0,1\} \rightarrow \hat{y} \rightarrow 1$



If $y=0 \Rightarrow \mathcal{L}(\hat{y}, y) = -\log(1-\hat{y}) \rightarrow \log(1-\hat{y})$ large $\rightarrow \hat{y}$ small

$\rightarrow \hat{y} \in \{0,1\} \rightarrow \hat{y} \rightarrow 0$

average of loss error

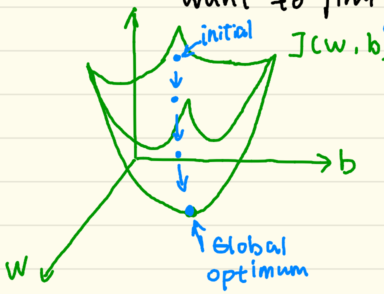
Cost fuction $= J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log\hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})]$

1.2 Video 4:

Gradient Descent

$J(w,b) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

want to find w, b to minimize $J(w,b)$



convex 凸函数：定义在某个向量空间的凸子集 C 上的实值函数 f，对任意 $x, y \in \mathrm{dom} f$，$0 \le \theta \le 1$ 有 $f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta)f(y)$

凸，有最优解　　非凸，无最优解

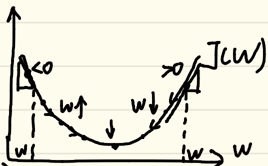
忽略b:

Repeat {

$w := w - \alpha\frac{dJ(w)}{dw}$ }

learning rate

迭代更新



Code use: "dw" 代表求导 $\Rightarrow w = w - \alpha dw$

考虑b:

Repeat {

$w := w - \alpha\frac{dJ(w,b)}{dw}$

$b := b - \alpha\frac{dJ(w,b)}{db}$

1.2 Video 5、6

Derivatives 微积分基础

1.2 Video 7

computation graph

$J(a,b,c) = 3(a+bc)$ $u=bc; v=a+u; J=3v$

$a=5 \longrightarrow v$

$b=3,\ c=2 \longrightarrow u=bc \longrightarrow v=a+u \longrightarrow J=3v$

正传播 Forward propagation

1.2 Video 8.

Derivatives with a Computation graph

$a=5 \longrightarrow v$

$b=3,\ c=2 \longrightarrow u=bc \longrightarrow v=a+u \longrightarrow J=3v$

backward propagation 反向传播：求导

$\frac{dJ}{dv}=3$

$\frac{dJ}{da} = \frac{dJ}{dv}\cdot\frac{dv}{da}=3$

代码中可用'dvar'代表 $\frac{d\,\text{Final output var}}{d\,\text{var}}$

1.2 video 9

Logistic Regression Gradient Descent

$x_1,\ w_1,\ x_2,\ w_2,\ b \longrightarrow z=w_1x_1+w_2x_2+b \longrightarrow \hat{y}=\sigma(z) \longrightarrow \mathcal{L}(\hat{y},y)$

$\hat{y}=a$

$\frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{d\hat{y}}\cdot\frac{d\hat{y}}{dz}$

$\frac{d\mathcal{L}}{d\hat{y}} = -\frac{y}{\hat{y}} + \frac{(1-y)}{(1-\hat{y})} = da$

$= \left[-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}\right] \times \hat{y}(1-\hat{y}) \leftarrow \left(\frac{1}{1+e^{-z}}\right)'$

$= \hat{y}-y = a-y = dz$

$dw_2 = x_2 \cdot dz$

$db = dz$

$dw_1 = \frac{d\mathcal{L}}{dz}\cdot\frac{dz}{dw_1}$

$= dz \cdot x_1$

$$w_1 = w_1 - \alpha dw_1$$
$$w_2 = w_2 - \alpha dw_2$$ update ...
$$b = b - \alpha db$$

1.2 video 10

Gradient Descent on m Examples

$$J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(a^{(i)}, y) \Rightarrow a^{(i)} = \sigma(w^T x^{(i)} + b)$$

$$\frac{\partial J(w,b)}{\partial w_1} = \frac{1}{m}\sum_{i=1}^{m} \frac{\partial}{\partial w_1}\mathcal{L}(a^{(i)}, y^{(i)})$$

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

$\boxed{\text{For}}$ $i=1$ to m 第1循环所有training example m

$$z^{(i)} = w^T x^{(i)} + b$$
$$a^{(i)} = \sigma(z^{(i)})$$
$$\Rightarrow \quad J_t = -[y^{(i)} \log a^{(i)} + (1-y^{(i)})\log(1-a^{(i)})]$$
$$dz^{(i)} = a^{(i)} - y^{(i)}$$

$\boxed{\text{For}}$ 第2循环所有特征n

$$\left.\begin{aligned} dw_1 &\mathrel{+}= dz^{(i)} \times x_1^{(i)} \\ dw_2 &\mathrel{+}= dz^{(i)} \times x_2^{(i)} \end{aligned}\right\} n=2$$
$$db \mathrel{+}= dz^{(i)}$$

$$\Rightarrow \quad J /= m$$
$$dw_1 /= m \quad dw_2 /= m \quad db /= m$$

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

但数据集过大，用For效率太低，所以用Vectorization加速运算

For iter in range(1000): 迭代1000次梯度下降

np.dot(a,b) ⇒ a,b为2个多维向量矩阵

去掉1st For: $\underset{1\times m}{Z} = np.dot(W.T, X) + b$ ☆ "broadcasting"

$$dZ = [dz^{(1)}, dz^{(2)}, \cdots dz^{(m)}]_{1\times m}$$

$$A = [a^{(1)}, a^{(2)}, \cdots, a^{(m)}] = \sigma(Z)$$

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

$$dZ = A - Y$$

去掉2nd For: $db = \frac{1}{m}\sum_{i=1}^{m} dz^{(i)} = \frac{1}{m} np.sum(dZ)$

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

$$= \frac{1}{m} \underset{n\times m}{\begin{bmatrix} | & & | \\ x_1 & \cdots & x_m \\ | & & | \end{bmatrix}} \times \underset{m\times 1}{\begin{bmatrix} -dz^{(1)}- \\ \vdots \\ -dz^{(m)}- \end{bmatrix}}$$

$$= \frac{1}{m}\underset{n\times 1}{[x^{(1)}dz^{(1)} + \cdots + x^{(m)}dz^{(m)}]}$$

Python - broadcasting: $A_{3\times 4} \;/\; B_{1\times 4}$

Python ∵ $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$

∴ $(m,n) \begin{matrix} \pm \\ \times \\ / \end{matrix} \begin{matrix} (1,n) \\ (m,1) \end{matrix} \leftrightarrow (m,n)$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 100 & 200 & 300 \\ 100 & 200 & 300 \end{bmatrix}$$

$$\begin{bmatrix} 2 & 4 & 6 \\ 1 & 3 & 5 \end{bmatrix} + \begin{bmatrix} 100 & 100 & 100 \\ 200 & 200 & 200 \end{bmatrix}$$

bug: ✗ a= np.random.randn(5) Don't use

a.shape=(5,) → 秩为1的数组而非矩阵

✓ → a=np.random.randn(5,1)

a.shape =(5,1)

或 assert(a.shape == (5,1))

或 a = a.reshape(5,1)

shift +Enter. / Cell Run

1.2 video 18

Explanation of logic regression cost function

If $y=1$ $p(y|x) = \hat{y}$

If $y=0$ $p(y|x)=1-\hat{y}$

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

$\log P(y|x)$ → 取log令它单调

$= y\log\hat{y} + (1-y)\log(1-\hat{y})$ ⇒ 最大似然估计

想使它最小，因此前加负号