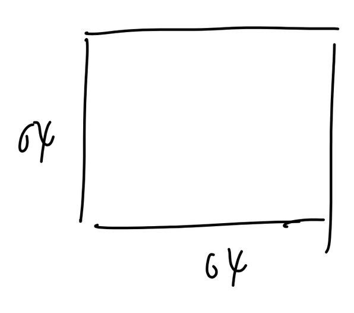
神经网络第二章

2022年7月10日 星期日　10:48

Logistic → Binary Classification



photos

use red / green / blue. 3.

then $x = \begin{bmatrix} \cdots \\ \cdots \\ \cdots \end{bmatrix}$

64 × 64 pixels

dimension = 64 × 64 × 3 = 12288

$x \longrightarrow y$

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

m training examples: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

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

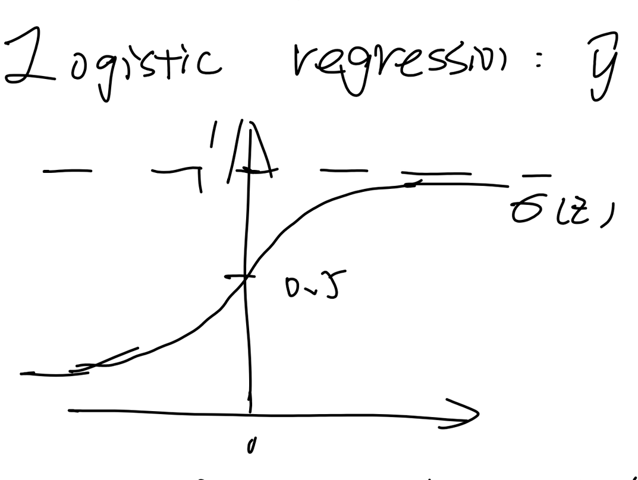
Given x, want $\hat{y} = P(y=1 \mid x)$

$x \in \mathbb{R}^{n_x}$ Parameters: $w \in \mathbb{R}^{n_x}$ $b \in \mathbb{R}$.

output: $\hat{y} = w^T x + b$, difficult to tell weather "0 or 1"

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

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



Given $\{(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})\}$

Want $\hat{y}^{(i)} \approx y^{(i)}$

$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ (loss function)

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

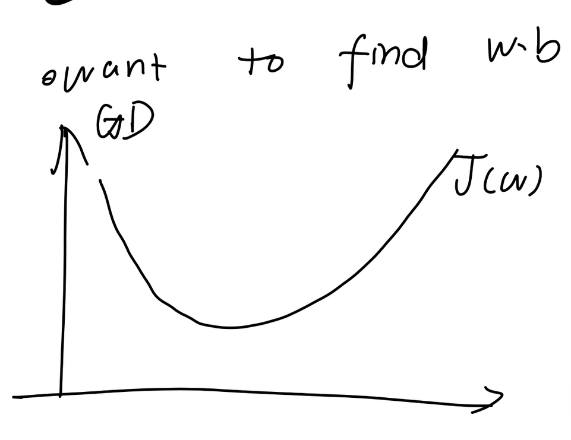
Cost function: $J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

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

{ loss function: 单个训练集
  cost function: 对于全体训练集

Gradient Descent

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

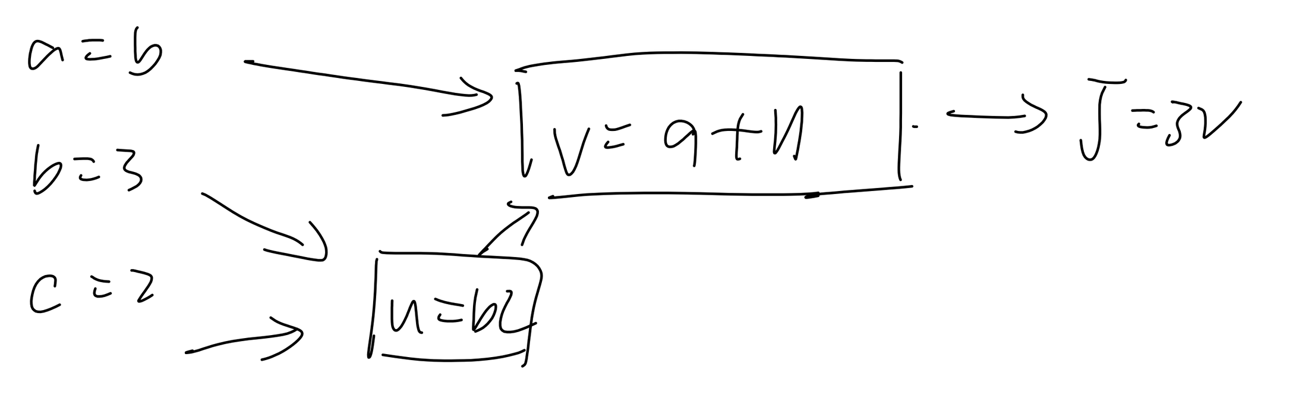


Repeat {

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

}

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



反向传播：求微分：derivatives

"dvar"：表示 Computing derivatives

$z = w^T x + b$　$L(a, y) = -(y\log(a) + (1-y)\log(1-a))$

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

$x_1, w_1, x_2, w_2, b \rightarrow$ [$z = w_1x_1 + w_2x_2 + b$] $\rightarrow \hat{y} = a = \sigma(z) \rightarrow L(a, y)$

① "da" $= \frac{dL(a,y)}{da}$　$[-(y\ln a + (1-y)\ln(1-a))]'$

$= -\frac{y}{a} + \frac{1-y}{1-a}$

② "dz" $= \frac{dL}{dz} = \frac{dL(a,y)}{dz}$

$= a - y$

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

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

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

Ways of Logistic regression on m examples.

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

For $i = 1$ to m. {

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

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

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

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

$dw_1 += x_1^{(i)} dz^{(i)}$

$dw_2 += x_2^{(i)} dz^{(i)}$

$db += dz^{(i)}$

$J /= m$

$dw_1 /= m;\ dw_2 /= m;\ db /= m$

$dw_1 = \frac{\partial J}{\partial w_1}$

$w_1 := w_1 - \alpha dw_1$

$w_2 := w_2 - \alpha dw_2$

$b := b - \alpha db$

}

Python 使用

z = np.dot(w, x) + b　(表示 $w^T x + b$)

avoid using "for" in a direct way

eg: $v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \rightarrow u \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$

```
import numpy as np
u = np.exp(v)
```

Logistic regression derivatives

$J = 0$, ~~$dw_1 = 0$, $dw_2 = 0$~~, $db = 0$ → dw = np.zeros(n_x, 1)

for i = 1 to m:

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

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

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

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

for j=1…n_x: dw_j += …

$dw_1 += x_1^{(i)} dz^{(i)}$　$n_x = 2$　→ $dw += x^{(i)} dz^{(i)}$

$dw_2 += x_2^{(i)} dz^{(i)}$

$db += dz^{(i)}$

$J = J/m$, ~~$dw_1 = dw_1/m$, $dw_2 = dw_2/m$~~, $db = db/m$

$dw /= m$

z = np.dot(w, x) + b

"Broadcasting"：自动将 b 补成了一个行向量 (广播)

$dz^{(1)} = a^{(1)} - y^{(1)}$　$dz^{(2)} = a^{(2)} - y^{(2)}$

$dz = [dz^{(1)}\ dz^{(2)} \ldots dz^{(m)}]$

$A = [a^{(1)} \ldots a^{(m)}]$　$Y = [y^{(1)} \ldots y^{(m)}]$

$dz = A - Y = [a^{(1)} - y^{(1)}\ \ a^{(2)} - y^{(2)} \ldots]$

$dw = 0$　　$db = 0$

$dw += x^{(1)} dz^{(1)}$　　$db += dz^{(1)}$

$dw += x^{(2)} dz^{(2)}$　　$db += dz^{(2)}$

⋮　　⋮

　　$db += dz^{(m)}$

$dw /= m$　　$db /= m$

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

$dw = \frac{1}{m} X dz^T = \frac{1}{m}[x^{(1)}dz^{(1)} + \ldots + x^{(m)}dz^{(m)}]$

$z = w^T x + b$

= np.dot(w, x) + b

$A = \sigma(z)$

$dz = A - Y$

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

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

$w := w - \alpha dw$

$b := b - \alpha db$.

Broadcasting in Python.

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

$= A_{(3,4)}$

cal = A.sum(axis = 0) 沿垂直求和，axis=1 则水平求和.　(按 col)

percentage = 100 * A / (cal.reshape(1, 4))

eg: $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + 100 = \begin{bmatrix} 101 \\ 102 \\ 103 \\ 104 \end{bmatrix}$ "broadcasting"

a.T 表示转置

a = np.random.randn(5)

Explaination of Logistic Regression

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

Interprot $\hat{y} = P(y=1 \mid x)$

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

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

log P(labels in training set) =

$\log \prod_{i=1}^{m} P(y^{(i)} \mid x^{(i)})$

$\log P(\ldots) = \sum_{i=1}^{m} \log P(y^{(i)} \mid x^{(i)})$

由最大似然估计…

max　min $J(w, b)$

↓ 最小

↳ 极大似然，最大！