Copyright Notice

Basics of Neural Network Programming

Binary Classification

deeplearning.ai

Binary Classification

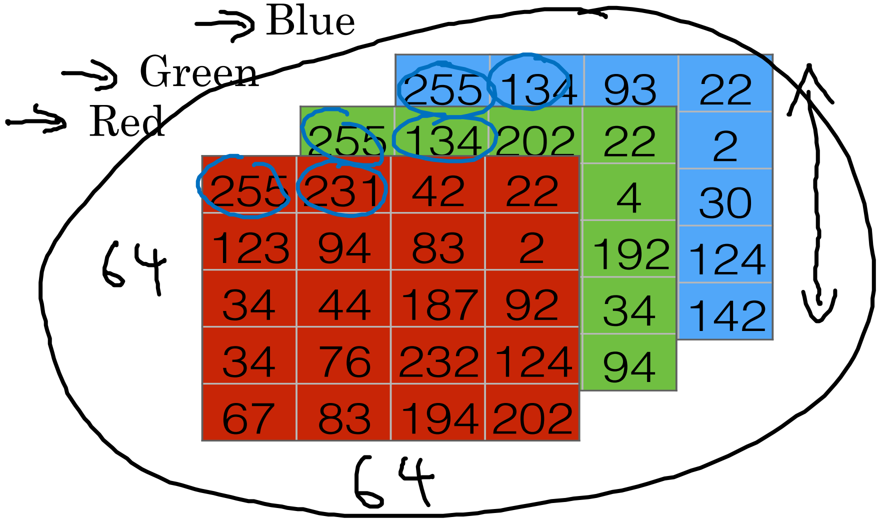
64

64

$\rightarrow$ 1 (cat) vs 0 (non cat)

y

viene definito un vettore x per l'immagine considerando i 3 livelli RBG



$$x = \begin{bmatrix} 255 \\ 231 \\ \vdots \\ \vdots \\ 255 \\ 134 \\ \vdots \end{bmatrix}$$

$$64 \times 64 \times 3 = 12288$$

$$n = n_x = 12288$$

$$x \longrightarrow y$$

Andrew Ng

Notation

un singolo set di addestramento è rappresentato da una coppia x,y dove:
- x è un vettore di più dimensioni
- y è uguale a 0 o a 1

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

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

$M = M_{train}$ $\qquad$ $M_{test} = \#\text{test examples}.$

colonne

$$X = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{m} \Big\updownarrow n_x \qquad \cancel{\begin{bmatrix} - & x^{(1)T} & - \\ & \vdots & \\ - & x^{(m)T} & - \end{bmatrix}}$$

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

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

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

$$X \in \mathbb{R}^{n_x \times m} \qquad X.\text{shape} = (n_x, m)$$

X = matrice dimensionale = nx per m, > la si implementa in python

Andrew Ng

Basics of Neural Network Programming

Logistic Regression

deeplearning.ai

Logistic Regression

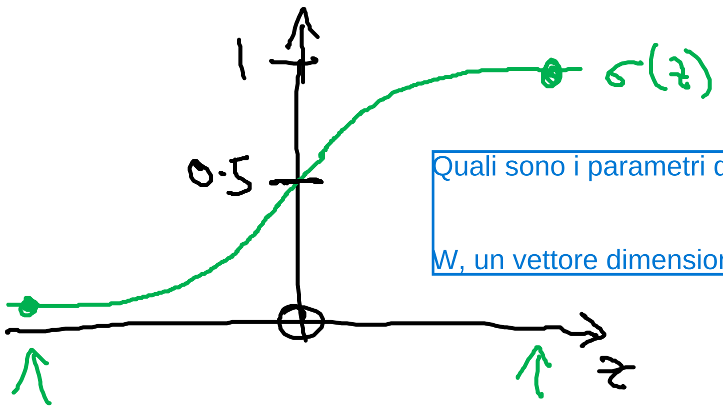
vuoi capire che dato X come immagine, questa immagine sia un gatto

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

$x \in \mathbb{R}^{n_x}$ $\qquad \underline{0 \leq \hat{y} \leq 1}$

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

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



Quali sono i parametri della regressione logistica?

W, un vettore dimensionale nx e b, un numero reale.

$x_0 = 1, \quad x \in \mathbb{R}^{n_x+1}$

extra feature

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

$$\underbrace{\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{matrix} \} b \leftarrow \\ \\ \left. \begin{matrix} \\ \\ \\ \end{matrix} \right\} w \leftarrow \end{matrix}}$$

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

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

If z large negative number

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

Andrew Ng

la regressione logistica può essere vista come una rete neurale molto piccola. perché è equivalente a una rete neurale con un solo neurone (senza strati nascosti): con:
Un solo neurone
Nessuno strato nascosto
Attivazione sigmoid

Basics of Neural Network Programming

deeplearning.ai

Logistic Regression cost function

La cost function della logistic regression è la funzione che misura l'errore tra le predizioni del modello e i valori reali. Il suo obiettivo è minimizzare questo errore per migliorare la precisione del modello.

La funzione di perdita calcola l'errore per un singolo esempio di formazione;
la funzione di costo è la media delle funzioni di perdita dell'intero set di formazione.

Logistic Regression cost function

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

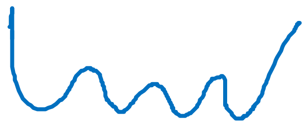
$z^{(i)} = w^T x^{(i)} + b$

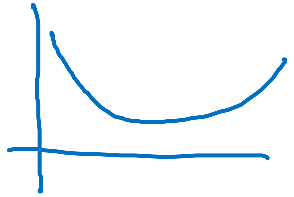
Given $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

$x^{(i)}$
$y^{(i)}$
$z^{(i)}$

i-th example.

$\boxed{\text{Loss}}$ (error) function: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$





$\boxed{\mathcal{L}(\hat{y}, y)} = -\left(\boxed{y \log \hat{y}} + (1-y)\log(1-\hat{y})\right) \leftarrow$

If $y=1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y} \leftarrow$ Want $\log \hat{y}$ large, want $\hat{y}$ large.

If $y=0$: $\mathcal{L}(\hat{y}, y) = -\log(1-\hat{y}) \leftarrow$ Want $\log 1-\hat{y}$ large .... want $\hat{y}$ small

$\boxed{\text{Cost}}$ function: $J(w,b) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log \hat{y}^{(i)} + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$

Andrew Ng

y^i è la probabilità predetta dal modello.
 è il valore reale.
m è il numero di esempi nel dataset.

Basics of Neural Network Programming

deeplearning.ai

Gradient Descent

Il Gradient Descent è un algoritmo di ottimizzazione utilizzato per trovare i valori ottimali dei parametri di un modello minimizzando una funzione di costo.

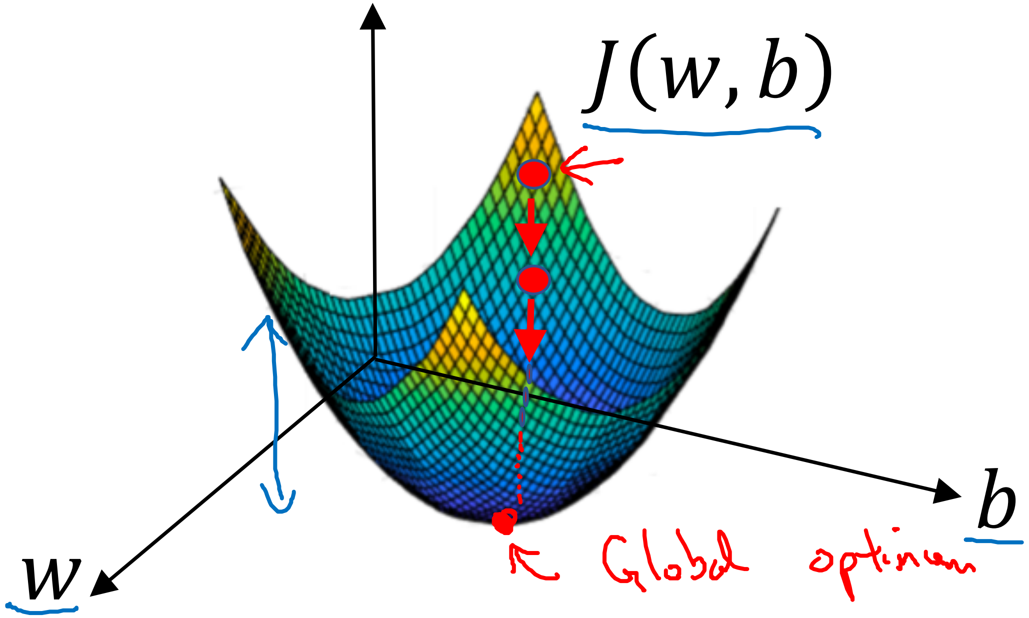
Gradient Descent

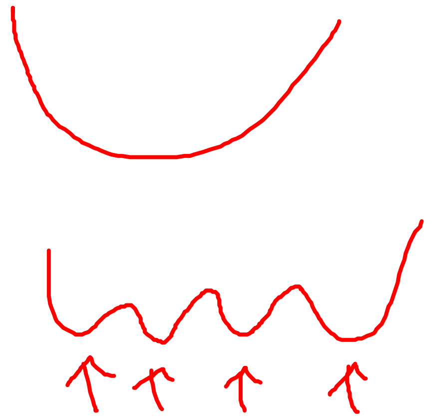
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \dfrac{1}{1+e^{-z}}$ ← algoritmo di reg logistica

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

funzione di costo

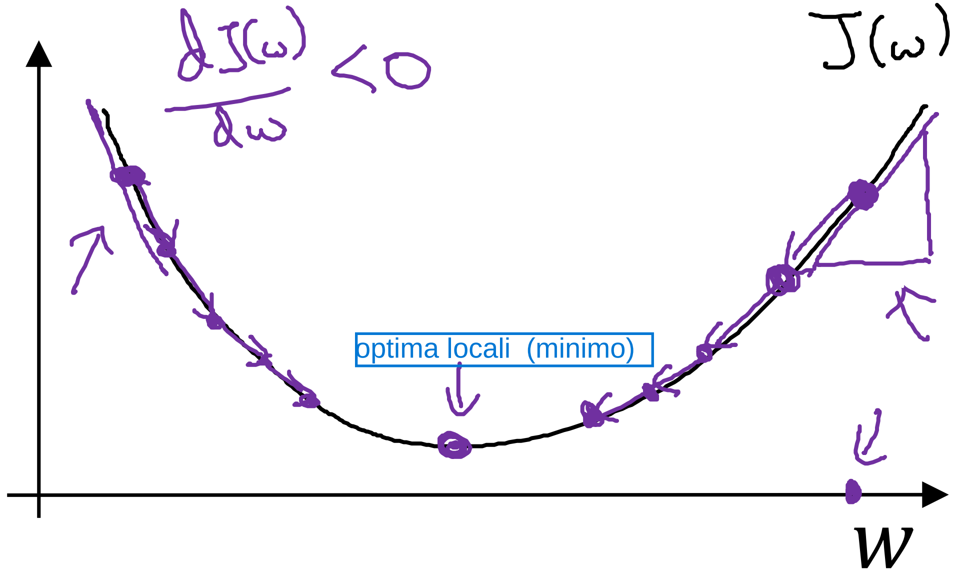
Want to find w, b that minimize $J(w,b)$





Andrew Ng

Gradient Descent



learning rate

Repeat {

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

}

"dw"

$$w := w - \alpha\, dw$$

$$\frac{dJ(w)}{dw} = ?$$

rappresenta la pendenza della funzione rispetto a una delle variabili

$J(w,b)$

$$w := w - \alpha \boxed{\frac{dJ(w,b)}{dw}} \qquad \boxed{\frac{\partial J(w,b)}{\partial w}} \rightarrow dw$$

$$b := b - \alpha \boxed{\frac{dJ(w,b)}{db}} \qquad \boxed{\frac{\partial J(w,b)}{\partial b}} \rightarrow db$$

∂ "partial derivative"

d J

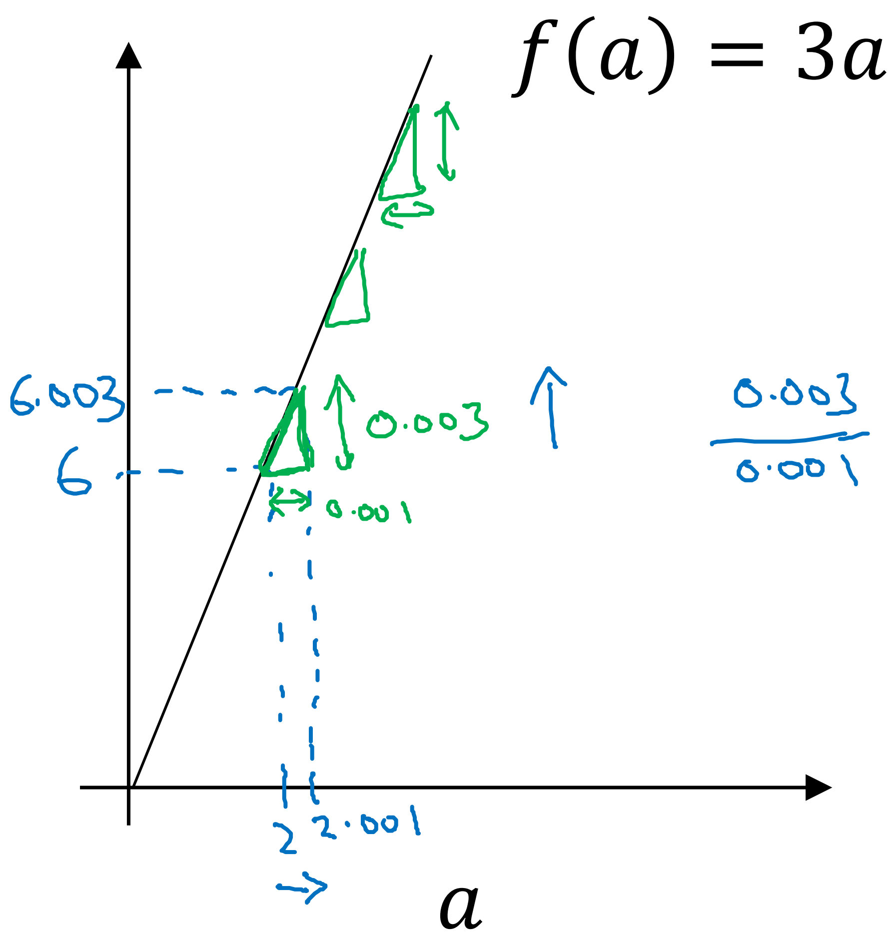
Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Derivatives

Intuition about derivatives



$\frac{0.003}{0.001}$ $\frac{\text{height}}{\text{width}}$

$\rightarrow a = 2$ $f(a) = 6$

$a = 2.001$ $f(a) = 6.003$

slope (derivative) of $f(a)$ at $a=2$ is 3

$\rightarrow a = 5$ $f(a) = 15$

$a = 5.001$ $f(a) = 15.003$

slope at $a=5$ is also 3

$$\frac{d f(a)}{da} = 3 = \frac{d}{da} f(a)$$

0.001

0.00000001

0.0000000001

Andrew Ng

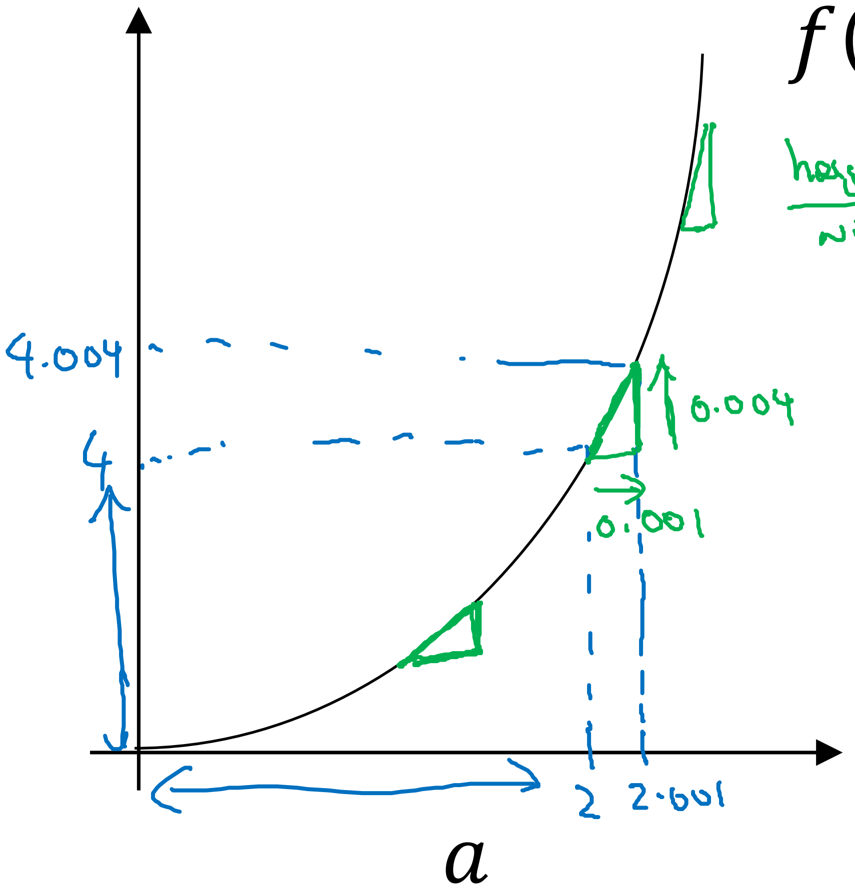
Basics of Neural Network Programming

More derivatives examples

deeplearning.ai

Intuition about derivatives

$$f(a) = a^2$$



$\frac{\text{height}}{\text{width}}$

$$\frac{d}{da} a^2 = 2a$$

0.001

$(2a) \times 0.001$

0.001

0.00000....01

$a = 2$ $\quad f(a) = 4$

$a = 2.001$ $\quad f(a) \approx 4.004$ $\quad (4.004\boxed{001})$

slope (derivative) of $f(a)$ at $a = 2$ is 4.

$\boxed{\frac{d}{da} f(a) = 4}$ when $\boxed{a = 2}$.

$a = 5$ $\quad f(a) = 25$

$a = 5.001$ $\quad f(a) \approx 25.010$

$\boxed{\frac{d}{da} f(a) = 10}$ when $\boxed{a = 5}$

$$\frac{d}{da} f(a) = \frac{d}{da} a^2 = \boxed{2a}$$

More derivative examples

$f(a) = a^2$

$\frac{d}{da} f(a) = \underbrace{2a}_{4}$

$a = 2$ $\quad f(a) = 4$

$a = 2.001$ $\quad f(a) \approx 4.004$

$f(a) = a^3$

$\frac{d}{da} f(a) = \underbrace{3a^2}_{3 \times 2^2 = 12}$

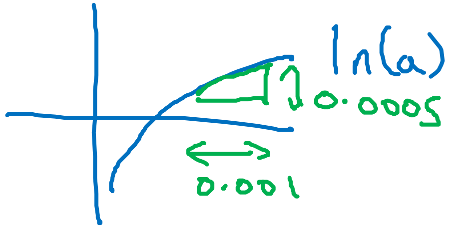
$a = 2$ $\quad f(a) = 8$

$a = \underline{2.001}$ $\quad f(a) \approx 8.\underline{012}$

$f(a) = \log_e(a)$

$\ln(a)$

$\frac{d}{da} f(a) = \frac{1}{a}$



$\frac{d}{da} f(a) = \boxed{\frac{1}{2}}$

$a = 2$ $\quad f(a) \approx 0.69315$

$a = 2.\underline{001}$ $\quad \underline{f(a)} \approx 0.69365$

0.0005 $\quad \underline{0.0005}$

deeplearning.ai

Basics of Neural Network Programming

Computation Graph

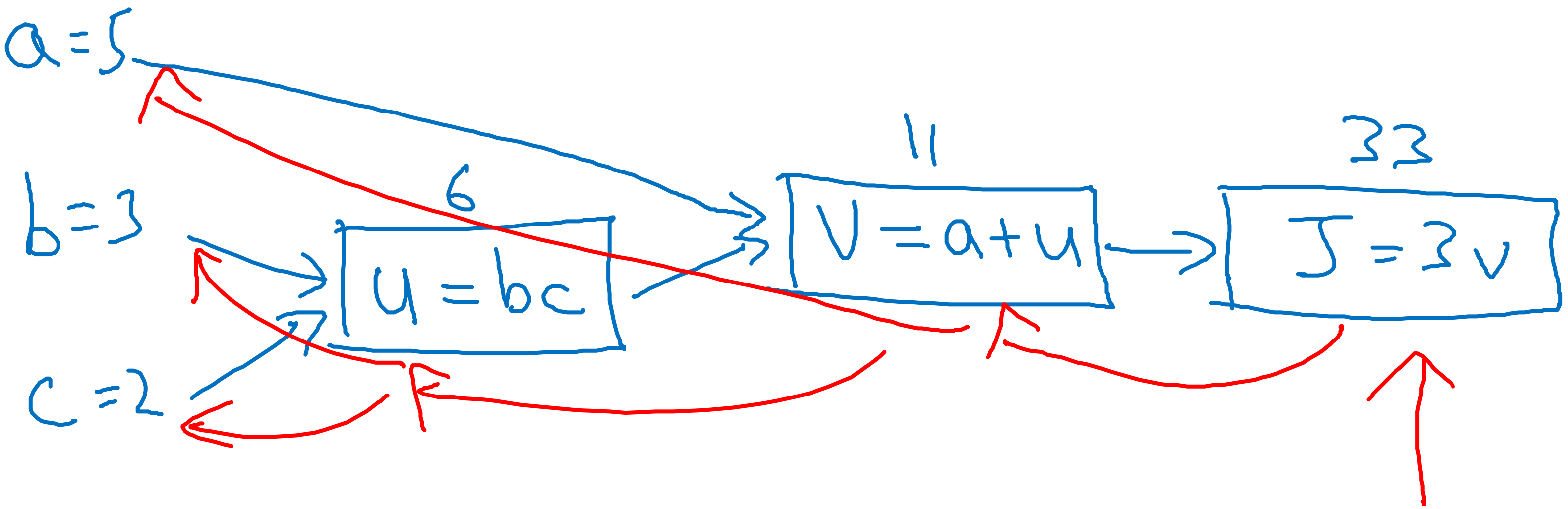
Computation Graph

$$J(a,b,c) = \underbrace{3(\underbrace{a+\underbrace{bc}_{u}}_{v})}_{J} = 3(5+3\times2) = 33$$

$u = bc$

$v = a+u$

$J = 3v$



Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives

$$\begin{tikzcd}
a = 5 \arrow[drr] & & \\
b = 3 \arrow[r] & \overset{6}{\boxed{u=bc}} \arrow[r] & \overset{11}{\boxed{v = a + u}} \arrow[r] & \overset{33}{\boxed{J = 3v}} \\
c = 2 \arrow[ur] & &
\end{tikzcd}$$

$\frac{dJ}{da}$ "da" = 3

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

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

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

3×1

$\frac{dv}{da} = 1$

$a \to v \to J$

$J = 3v$

$v = 11 \to 11.001$

$J = 33 \to 33.003$

$a = 5 \to 5.001$

$\to v = 11 \to 11.001$

$J = 33 \to 33.003$

$\frac{d\,\text{FinalOutputVar}}{d\,\text{var}}$

dJdvar

"dvar"

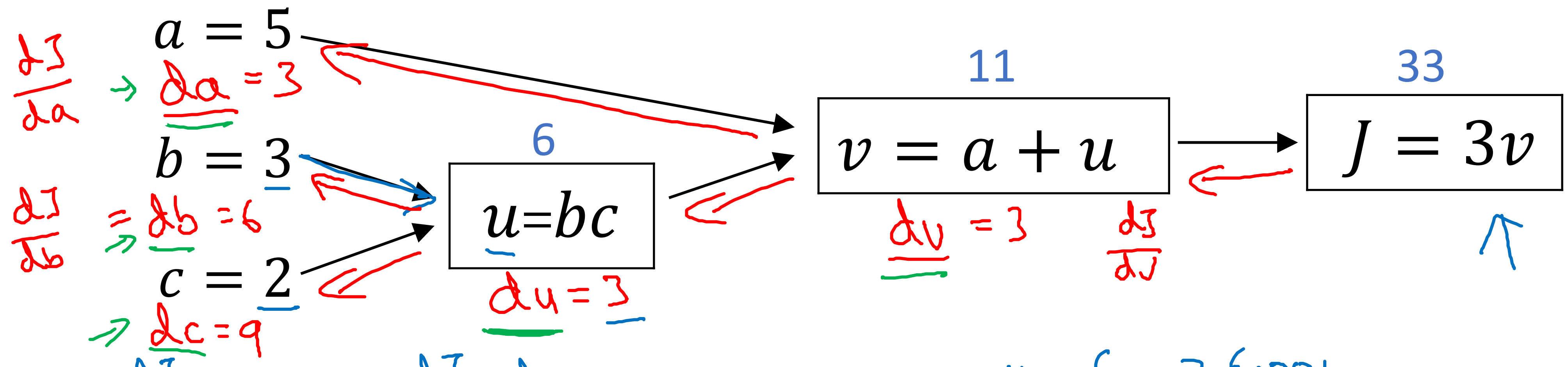
$f(a) = 3a$

$\frac{df(a)}{da} = \frac{df}{da} = 3$

$J = 3v$

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

Computing derivatives



$$\frac{dJ}{du} = 3 = \underbrace{\frac{dJ}{dv}}_{3} \cdot \underbrace{\frac{dv}{du}}_{1}$$

$$\frac{dJ}{db} = \underbrace{\boxed{\frac{dJ}{du}}}_{\to 3} \cdot \underbrace{\frac{du}{db}}_{=2} = 6$$

$$\frac{dJ}{dc} = \boxed{\frac{dJ}{du}} \cdot \frac{du}{dc} = 9$$

$$3 \times 3$$

u = 6 → 6.001
v = 11 → 11.001
J = 33 → 33.003

b = 3 → 3.001
u = b·c = 6 → 6.002
J = 33.006

c = 2
.006

v = 11.002
J = 3v

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

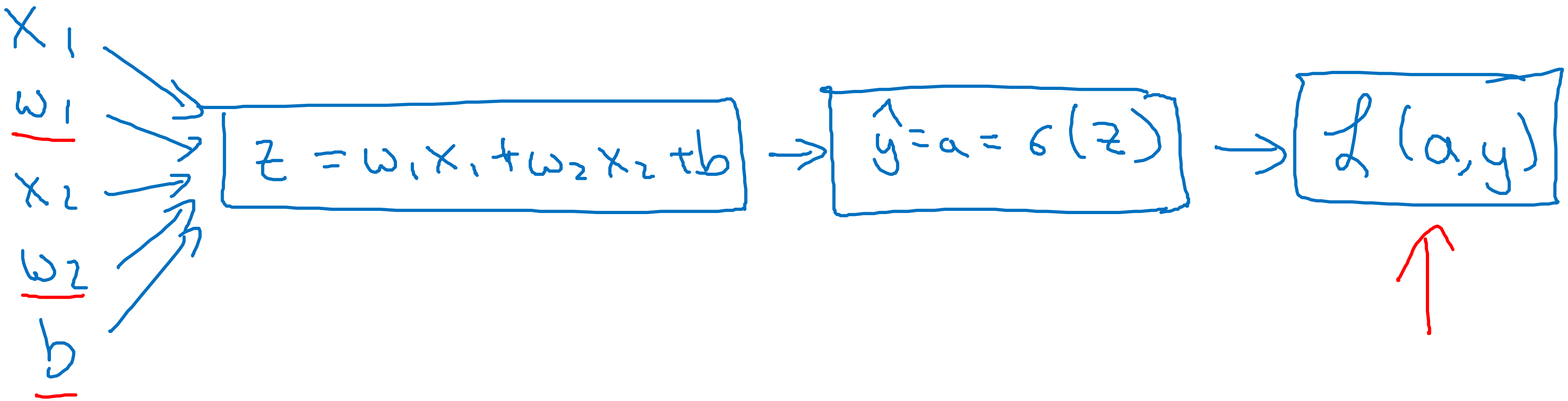
Logistic Regression Gradient descent

Logistic regression recap

$$z = w^T x + b$$

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

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



Logistic regression derivatives

x_1, w_1, x_2, w_2, b

$$\boxed{z = w_1x_1 + w_2x_2 + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{\mathcal{L}(a, y)}$$

$$dz = \frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}(a,y)}{dz} = a - y = \frac{d\mathcal{L}}{da} \cdot \frac{da}{dz}$$

$$\text{"}da\text{"} = \frac{d\mathcal{L}(a,y)}{da} = -\frac{y}{a} + \frac{1-y}{1-a}$$

$$a(1-a)$$

$$\frac{\partial \mathcal{L}}{\partial w_1} = \text{"}dw_1\text{"} = x_1 \cdot dz \qquad dw_2 = x_2 \cdot dz \qquad db = dz$$

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

deeplearning.ai

Basics of Neural Network Programming

Gradient descent on m examples

Logistic regression on m examples

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

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

$$(x^{(i)}, y^{(i)})$$

$$\underline{dw_1^{(i)}}, \underline{dw_2^{(i)}}, \underline{db^{(i)}}$$

$$\underline{\frac{\partial}{\partial w_1} J(w,b)} = \frac{1}{m}\sum_{i=1}^{m} \underbrace{\frac{\partial}{\partial w_1}\mathcal{L}(a^{(i)}, y^{(i)})}_{\underline{dw_1^{(i)}}}$$

$$\underline{dw_1^{(i)}} - (x^{(i)}, y^{(i)})$$

Andrew Ng

Logistic regression on m examples

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

For $i = 1$ to m

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

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

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

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

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

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

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

dw_3
$\vdots$
dw_n

$n = 2$

$J /= m$

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

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

$$w_1 := w_1 - \alpha\, dw_1$$

$$w_2 := w_2 - \alpha\, dw_2$$

$$b := b - \alpha\, db$$

Vectorization

Andrew Ng

Basics of Neural Network Programming

Vectorization

deeplearning.ai

What is vectorization?

$z = \underbrace{w^T x} + b$

$w = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $x = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix}$ $w \in \mathbb{R}^{n_x}$ $x \in \mathbb{R}^{n_x}$

Non-vectorized:

```
z = 0
for i in range(n-x):
    z += w[i] * x[i]
z += b
```

Vectorized

$z = \underbrace{np.\underline{dot}(w, x)}_{w^T x} + b$

→ GPU
→ CPU } SIMD - single instruction multiple data.

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$u = Av$

$u_i = \sum_i \sum_j A_{ij} v_j$

```
u = np.zeros((n,1))
for i ...
    for j ....
        u[i] += A[i][j] * v[j]
```

```
u = np.dot(A,v)
```

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

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

```
u = np.zeros((n,1))
for i in range(n):
    u[i]=math.exp(v[i])
```

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

np.log(v)
np.abs(v)
np.maximum(v, 0)
v**2        1/v
```

Logistic regression derivatives

$dw = np.zeros((n\_x, 1))$

J = 0, $\boxed{\cancel{\text{dw1 = 0, dw2 = 0}}}$, db = 0

for i = 1 to n:

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

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

$$J \mathrel{+}= -\left[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})\right]$$

$$dz^{(i)} = a^{(i)}(1 - a^{(i)})$$

for j = 1 … n_x

dw_j += …

$$\boxed{\cancel{\begin{aligned} dw_1 &\mathrel{+}= x_1^{(i)} dz^{(i)} \\ dw_2 &\mathrel{+}= x_2^{(i)} dz^{(i)} \end{aligned}}}$$

$n_x = 2$

$dw \mathrel{+}= x^{(i)} dz^{(i)}$

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

J = J/m, $\boxed{\cancel{dw_1 = dw_1/m,\ dw_2 = dw_2/m}}$, db = db/m

$dw /= m.$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

$$\boxed{z^{(1)} = w^T x^{(1)} + b \\ a^{(1)} = \sigma(z^{(1)})} \qquad \boxed{z^{(2)} = w^T x^{(2)} + b \\ a^{(2)} = \sigma(z^{(2)})} \qquad \boxed{z^{(3)} = w^T x^{(3)} + b \\ a^{(3)} = \sigma(z^{(3)})}$$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix} \qquad (n_x, m) \qquad \mathbb{R}^{n_x \times m}$$

$$[\;\text{———}\;] \; w^T \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}$$

$$Z = \boxed{[z^{(1)} \; z^{(2)} \; \cdots \; z^{(m)}]} = \underbrace{w^T X + \underbrace{[b \; b \cdots b]}_{1\times m}} = \Big[\, \underbrace{\boxed{w^T x^{(1)} + b} \quad \boxed{w^T x^{(2)} + b} \quad \cdots \quad w^T x^{(m)} + b}_{1\times m} \,\Big]$$

$$\rightarrow Z = \text{np.dot}(w.T, X) + b$$

$(1,1)$ $\mathbb{R}$ "Broadcasting"

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

deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

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

$$\boxed{dZ = [dz^{(1)} \; dz^{(2)} \ldots dz^{(m)}]} \leftarrow$$

$1 \times m$

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

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

$\rightarrow dw = 0$

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

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

$\vdots$

$dw /= m$

$\cancel{dw_1}$

$\cancel{dw_2}$

$\vdots$

$db = 0$

$db \mathrel{+}= dz^{(1)}$

$db \mathrel{+}= dz^{(2)}$

$\vdots$

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

$db /= m.$

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

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

$$= \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \ldots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix}$$

$$= \frac{1}{m} \left[x^{(1)} dz^{(1)} + \ldots + x^{(m)} dz^{(m)} \right]$$

$n \times 1$

Andrew Ng

Implementing Logistic Regression

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

$\boxed{\text{for } i = 1 \text{ to } m:}$

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

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

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

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

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

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

$dw \mathrel{+}= x^{(i)} * dz^{(i)}$

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

$J = J/m,\ dw_1 = dw_1/m,\ dw_2 = dw_2/m$

$db = db/m$

for iter in range(1000):

$Z = w^T X + b$

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

$A = \sigma(Z)$

$dZ = A - Y$

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

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

$w := w - \alpha dw$

$b := b - \alpha db$

Andrew Ng

deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

Calories from Carbs, Proteins, Fats in 100g of different foods:

$$
\begin{array}{r c}
 & \begin{array}{cccc} \text{Apples} & \text{Beef} & \text{Eggs} & \text{Potatoes} \end{array} \\
\begin{array}{r} \text{Carb} \\ \text{Protein} \\ \text{Fat} \end{array} &
\left[\begin{array}{cccc} 56.0 & 0.0 & 4.4 & 68.0 \\ 1.2 & 104.0 & 52.0 & 8.0 \\ 1.8 & 135.0 & 99.0 & 0.9 \end{array}\right]
\end{array}
= A
$$

(3,4)

59 cal

$\frac{56}{59} \approx 94.9\%$

Calculate % of calories from Carb, Protein, Fat. Can you do this without explicit for-loop?

```
cal = A.sum(axis = 0)
percentage = 100*A/(cal.reshape(1,4))
```

(3,4) / (1,4)

Broadcasting example

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

$$\underset{(m,n)\ (2,3)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \begin{bmatrix}100 & 200 & 300\\100 & 200 & 300\end{bmatrix}$$

$(1,n) \rightsquigarrow (m,n)$ $(2,3)$

$$\underset{(m,n)}{\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}} + \begin{bmatrix}100 & 100 & 100\\200 & 200 & 200\end{bmatrix} =$$

$(m,1)$

$\Downarrow$

(m,n)

General Principle

(m, n) matrix $\quad +,\ -,\ *,\ / \quad (1, n) \rightsquigarrow (m, n)$

$(m, 1) \rightsquigarrow (m, n)$

$(m, 1) \quad + \quad \mathbb{R}$

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + 100 = \begin{bmatrix} 101 \\ 102 \\ 103 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} + 100 = \begin{bmatrix} 101 & 102 & 103 \end{bmatrix}$$

Matlab/Octave: bsxfun

Basics of Neural Network Programming

deeplearning.ai

Explanation of logistic regression cost function (Optional)

Logistic regression cost function

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

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

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

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

Andrew Ng

Logistic regression cost function

$$\boxed{\text{If } \quad y = 1: \qquad p(y|x) = \hat{y}}$$
$$\boxed{\text{If } \quad y = 0: \qquad p(y|x) = 1 - \hat{y}}$$

$p(y|x)$

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

If $y=1$: $\underline{p(y|x) = \hat{y}}$ $\quad \underbrace{(1-\hat{y})^{0}}_{=1}$

If $\underline{y=0}$: $\underline{p(y|x)} = \underbrace{\hat{y}^{0}}_{=1} \quad (1-\hat{y})^{(1-y)} = 1 \times (1-\hat{y}) = \underline{1-\hat{y}}$

$$\uparrow \log \underline{p(y|x)} = \log \hat{y}^{y} (1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y) \log (1-\hat{y})$$
$$= -\underline{\mathcal{L}(\hat{y}, y)} \downarrow$$

Andrew Ng

Cost on m examples

$$\underline{\log p(\text{labels in training set})} = \log \prod_{i=1}^{m} p(y^{(i)} | x^{(i)})$$

$$\underline{\log p(\cdots)} = \sum_{i=1}^{m} \underbrace{\log p(y^{(i)} | x^{(i)})}_{-\mathcal{L}(\hat{y}^{(i)}, y^{(i)})}$$

$$= -\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Maximum likelihood estimation

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

Andrew Ng