Core Notations (Binary Classification)

each training example is (x, y) where:

$x \in \mathbb{R}^{n_x}$

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

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

$m = m_{train}$

Matrix Representation

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

rows = num features

cols = num examples

$\Rightarrow$ OUR COLUMNS ARE training example.

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

This is a different notation from 10-315.

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

$Y \in \mathbb{R}^{1 \times m}$ $\quad Y.shape = (1, m)$

Logistic Regression

when output is (0 or 1). This is a classification algorithm.

given x, want $\hat{y} = P(y = 1 | x)$

$x \in \mathbb{R}^{n_x}$

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

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

if z is large, $\sigma(z) \approx 1$

if z is small, $\sigma(z) \approx 0$

(large -ve num)

$\Rightarrow$ sigmoid function :

continuous function $\mathbb{R} \to (0, 1)$

Alternative Convention

$\Rightarrow$ feed bias term in

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

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

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

inferior, so in NN, we don't use this notation.

Logistic Regression Cost Function

$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$ we want $\hat{y}^{(i)} \approx y^{(i)}$

Loss (error) function: (for one example)

Square error:

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

$\Rightarrow$ non-convex optimization problem
$\Rightarrow$ multiple local minima
$\Rightarrow$ gradient descent doesn't work well.

Cross-entropy loss: (a.k.a. logarithmic loss)

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

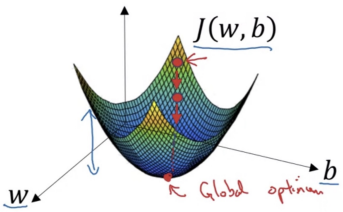
if $y=1$, then we want $\log \hat{y}$ to be large, so $\hat{y}$ as close to 1 as possible

if $y=0$, then we want $\log(1-\hat{y})$ to be large, so $\hat{y}$ as close to 0 as possible

Cost Function: (the cost of parameters — entire training set)

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

Gradient Descent



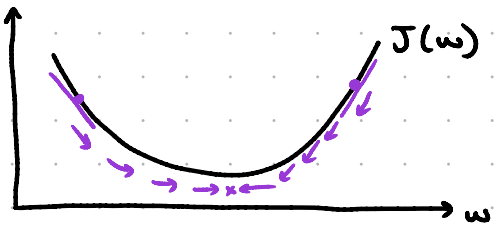
→ This is a good way to illustrate gradient descent.

REPEAT {

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

}

→ slope of function
⇒ as (−) in front, we move down
⇓
move in opposite way



When generalized: $J(w,b)$

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

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

∂ : partial derivative symbol.

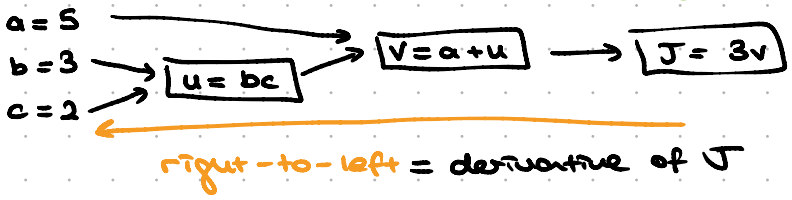
∂ : when 2 or more variables

d : when only 1 variable in function we are taking derivative of

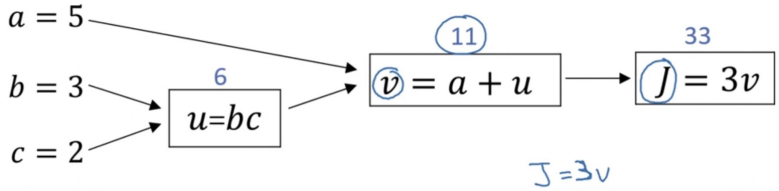
Computation Graph

$J = 3(a + bc)$

$u = bc$
$v = a + u$
$J = 3v$



Computation Graph: Computing Derivatives



If we want to calculate derivative of final output

w.r.t. v ⇒ 1 step back prop
w.r.t. a ⇒ 2 step back prop

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

$J = 3v$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

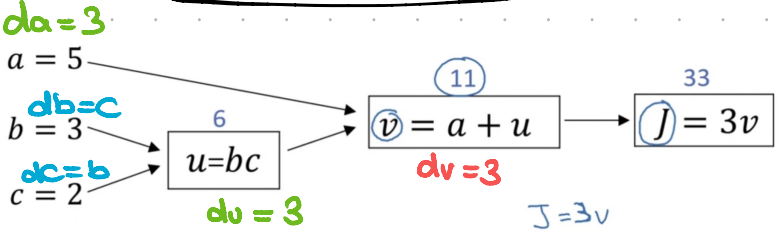
Chain rule

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

$a = 5 \rightarrow 5.001$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

Note: because we are always taking derivative of the final output we can name

$\frac{d\,\text{Final Output Var}}{d\,\text{var}} \rightarrow \text{as} \rightarrow d\,\text{var}$ (in code)



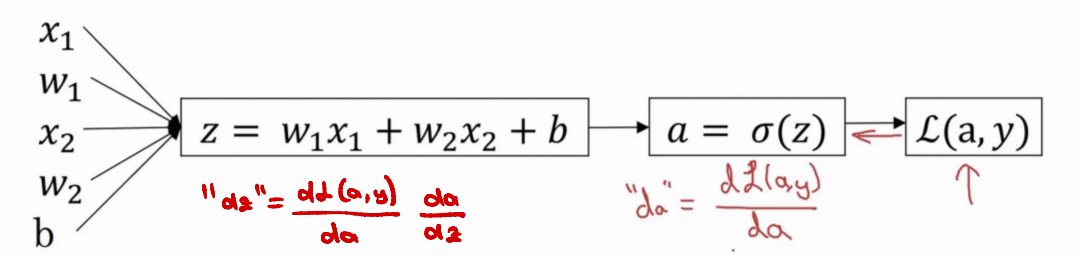
$\frac{dJ}{dv} = ? = 3$

$J = 3v$
$v = 11 \rightarrow 11.001$
$J = 33 \rightarrow 33.003$

Logistic Regression Gradient Descent

$$z = w^T x + b$$
$$\hat{y} = a = \sigma(z) = e^{-z}$$
$$\mathcal{L}(a, y) = -\left(y \log(a) + (1-y)\log(1-a)\right)$$

Consider that we have two features.



$$dw_1 = x_1 \cdot dz \qquad = \hat{y} - y = a - y \qquad = -\frac{y}{a} + \frac{1-y}{1-a}$$
$$dw_2 = x_2 \cdot dz$$
$$db = dz$$

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

Extra: Derivation of dz

$$\frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{da}\frac{da}{dz}$$

Step 1: $\frac{d\mathcal{L}}{da} \Rightarrow \mathcal{L} = -(y \log a + (1-y)\log(1-a))$

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

Lets give both terms a same denominator

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

Step 2: $\frac{da}{dz}$

$$\frac{da}{dz} = \frac{d}{dz}\sigma(z) = \sigma(z) \times (1 - \sigma(z))$$
$$= a(1-a) \quad [\text{as } a = \sigma(z)]$$

Step 3: $\frac{d\mathcal{L}}{dz}$

you can just multiply outcomes

$$\frac{d\mathcal{L}}{dz} = \frac{d\mathcal{L}}{da} \times \frac{da}{dz} = \frac{a-y}{a(1-a)} \cdot a \cdot (1-a) = a - y$$

Gradient Descent for m - Examples

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

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

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

For $i = 1$ to m: → For loop over training examples

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

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

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

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

For loop over features:

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

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

} assuming we have 2 features $n=2$

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

$J \mathrel{/}= m$

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

The left side is ONE step of gradient descent

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

To get rid of For loops → Vectorization