Binary Classification

Logistic Regression

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

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

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

Cost function

Given $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots (x^{(m)}, y^{(m)})\}$

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

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

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

Gradient Descent

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

Derivatives

Computation Graph

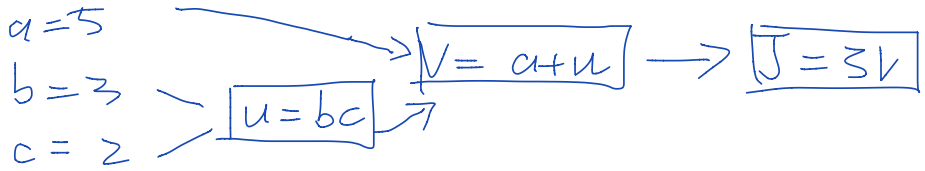
example:

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

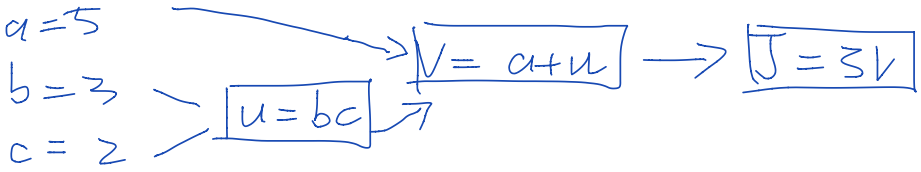
$u = bc$

$v = a + u$

$J = 3v$



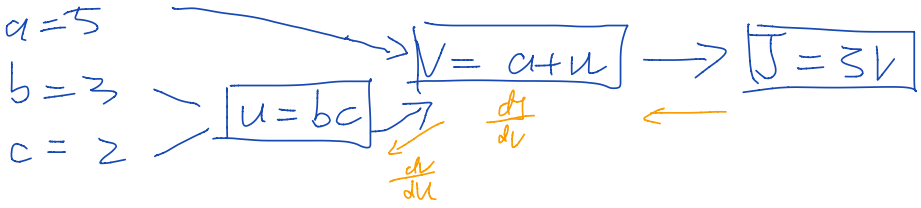
Derivatives with a computation Graph.



Chain rule:

change a → change V → change J

$$\frac{\partial J}{\partial a} = \frac{\partial J}{\partial V}\frac{\partial V}{\partial a} = 3$$



Logistic regression gradient descent

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

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

$$J = \frac{1}{m}\sum -(y \log(\hat{y}_i) + (1-y)\log(1-\hat{y}^{(i)}))$$

$$\frac{\partial J}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z} = (-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}})\frac{-1}{(1+e^{-z})^2}e^{-z}\cdot(-1)$$

$$= \left(-\frac{y}{\hat{y}} + \frac{1-y}{1-\hat{y}}\right)\hat{y}(1-\hat{y})$$
$$= -y(1-\hat{y}) + (1-y)\hat{y}$$
$$= \hat{y} - y$$

$$dz = \frac{\partial z}{\partial w}dw + \frac{\partial z}{\partial b}db = x\,dw + db$$

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial z}\frac{\partial z}{\partial w} = (\hat{y} - y)x$$

$$w := w - \alpha\frac{\partial \mathcal{L}}{\partial w}$$
$$= w - \alpha(\hat{y} - y)x$$
$$b := b - \alpha\frac{\partial \mathcal{L}}{\partial b}$$

Gradient descent examples

m examples

for $i = 1$ to m:

$$z^{(i)} = w^T x^{(i)} + b$$
$$\hat{y}^{(i)} = \sigma(z^{(i)}) = \frac{1}{1 + e^{-(w^T x^{(i)} + b)}}$$
$$J\mathrel{+}= -(y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)}))$$
$$"dz^{(i)}" = \hat{y}^{(i)} - y^{(i)}$$
$$"dw_1" \mathrel{+}= x_1^{(i)} \cdot "dz^{(i)}" = x_1^{(i)}(\hat{y}^{(i)} - y^{(i)})$$
$$"dw_2" \mathrel{+}= x_2^{(i)} \cdot "dz^{(i)}" = x_2^{(i)}(\hat{y}^{(i)} - y)$$
$$"db" \mathrel{+}= 1 \cdot "dz^{(i)}" = \hat{y}^{(i)} - y$$

$J = J/m$ $\quad "dw_1" = "dw_1"/m$ $\quad "dw_2" = "dw_2"/m$ $\quad db = \frac{"db"}{m}$

$$w_1 := w_1 - \alpha \text{"}dw_1\text{"} = w_1 - \frac{\alpha}{m}\sum_i x_1^{(i)}(\hat{y}^{(i)} - y^{(i)})$$
$$w_2 := w_2 - \alpha \text{"}dw_2\text{"} = w_2 - \frac{\alpha}{m}\sum_i x_2^{(i)}(\hat{y}^{(i)} - y^{(i)})$$
$$b := b - \alpha \text{"}db\text{"} = b - \frac{\alpha}{m}\sum_i (\hat{y}^{(i)} - y^{(i)})$$

Vectorization

GPU } SIMD ⇒ "Single instruction multiple data"
CPU }

Neural network programming guideline

Whenever possible, avoid explicit for loops

$u = Av \Rightarrow u = np.dot(A, v)$

$u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \end{bmatrix} \Rightarrow u = np.exp(v)$

Vectorize logistic regression

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

m: training examples
n: features

$$w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}_{n \times 1} \qquad b = [b, b, \cdots b]$$

$$[z^{(1)}, z^{(2)}, \cdots z^{(m)}] = w^T X + b = Z$$

$Z = np.(w.T, X) + b$ ← broad casting

Logistic regression prediction:

$$\hat{Y} = [\hat{y}^{(1)} \dots \hat{y}^{(m)}] = \frac{1}{1+e^{-w^T x + b}}$$

Vectorize Gradient Descent

"dz" $= \hat{Y} - Y$

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

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

① $z = w^T x + b$ = np.dot(w.T, x) + b

② $\hat{Y} = \sigma(z)$

③ "dz" $= \hat{Y} - Y$

④ "dw" $= \frac{1}{m} X("dz")^T$

⑤ "db" $= \frac{1}{m}$ np.sum("dz")

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

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

Broadcasting In Python

example

	Apple	Beef	Eggs	Potatos
Carb.				
Protein				
Fat				

question: calculate % of carbons from carb, Protein, Fat.
Can you do ~~that~~ w/o for loop ?

Broadcasting Example

$$\begin{bmatrix}1\\2\\3\\4\end{bmatrix} + 100 \rightarrow \begin{bmatrix}100\\100\\100\\100\end{bmatrix} = \begin{bmatrix}101\\103\\103\\104\end{bmatrix}$$

$$\underset{m\times n}{\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}} + \underset{1\times n}{\begin{bmatrix}100&200&300\end{bmatrix}} \rightarrow (m\times n) = \begin{bmatrix}101&102&103\\104&105&106\end{bmatrix}$$

$$\underset{m\times n}{\begin{bmatrix}1&2&3\\4&5&6\end{bmatrix}} + \underset{m\times 1}{\begin{bmatrix}100\\200\end{bmatrix}} \rightarrow m\times n = \begin{bmatrix}101&102&103\\201&202&203\end{bmatrix}$$

General Principle

$$\begin{matrix}(m,n) & + \\ & - \\ & * \\ & / \end{matrix} \begin{matrix}(1,n) \rightarrow (m,n)\\ (m,1) \rightarrow (m,n)\end{matrix}$$

Python vector sidenotes

a = np.random.randn(5) → shape (5,) → rank 1 array

a = - - - - - (5,1) → shape (5,1) → col. vector

a = - - - - - (1,5) → shape (1,5) → row vector

Explaination cost function of logistic regression

$$\hat{y} = \frac{1}{1+e^{-w^T x}}$$

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

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

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

$$\log P(y|x) = y \log \hat{y} + (1-y)\log(1-\hat{y}) = -\mathcal{L}$$

then minimize the cost function accordingly maximize the $\log(P(y|x))$ func

Logistic regression is maximum likelihood estimation

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

$$\log P(\dots) = \sum_{i=1}^{m} \log(P(y^{(i)}|x^{(i)})) = -\sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

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

minimize $J \rightarrow$ maximize P

$\rightarrow$ maximum likelihood optimization