Warming Up
- Logistic Regression

Most of this material is from Prof. Andrew Ng'and Chang's slides.

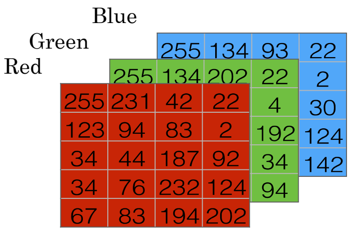
Binary Classification

→ 1 (cat) vs 0 (non cat)

Binary Classification

x

$\longrightarrow$ y 1 (cat) vs 0 (non cat)



input feature vector:

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

if 64 x 64 x 3 = 12288
dimension of x = n_x = 12288

Notation

- single training example

  $(x, y) \quad x \in \mathbb{R}^{n_x}, \quad y \in \{0, 1\}$
- m training examples

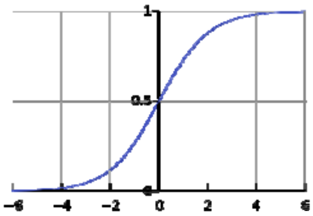
  $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
- more compact notation using matrix

$$\mathrm{X} = \underbrace{\begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \cdots & x^{(m)} \\ | & | & & | \end{bmatrix}}_{m} \Big\updownarrow n_x \qquad \mathrm{X} \in \mathbb{R}^{n_x \times m}$$

$$\mathrm{Y} = [y^{(1)} \quad y^{(2)} \quad \cdots \quad y^{(m)}] \qquad \mathrm{Y} \in \mathbb{R}^{1 \times m}$$

Logistic Regression

- Given $x \in \mathbb{R}^{n_x}$, we want to estimate $\hat{y} = \mathrm{P}(y = 1|x)$
  - We want $\hat{y}$ to be probability: $0 \leq \hat{y} \leq 1$
- (Model) Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$
- (Model) Output: $\hat{y} = \sigma(w^T x + b)$
  - Sigmoid function $\sigma(z)$



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

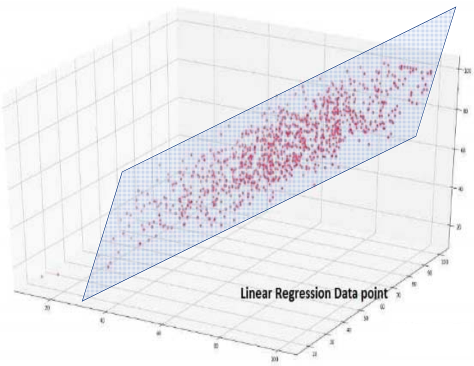
If z is large positive number: $\sigma(z) \approx \frac{1}{1+0} = 1$

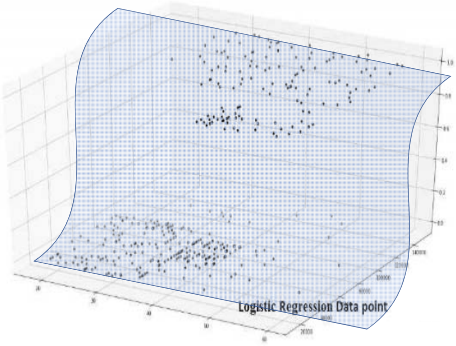
If z is large negative number: $\sigma(z) \approx \frac{1}{1+\infty} \approx 0$

Linear Regression vs Logistic Regression

$$y = w_1 x_1 + w_2 x_2 + b$$

$$y = \frac{1}{1+e^{-(w_1 x_1 + w_2 x_2 + b)}}$$





Logistic Regression

- Given $x \in \mathbb{R}^{n_x}$, we want to estimate $\hat{y} = \mathrm{P}(y|x)$
  - We want $\hat{y}$ to be probability: $0 \leq \hat{y} \leq 1$
- (Model) Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$
- (Model) Output: $\hat{y} = \sigma(w^T x + b)$
- The goal of logistic regression
  - try to learn the parameters w and b so that $\hat{y}$ becomes a good estimate of the probability of y

Logistic Regression

- Given $x \in \mathbb{R}^{n_x}$, we want to estimate $\hat{y} = \mathrm{P}(y|x)$
  - We want $\hat{y}$ to be probability: $0 \leq \hat{y} \leq 1$
- (Model) Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$
- (Model) Output: $\hat{y} = \sigma(w^T x + b)$
- The goal of logistic regression
- Notational convention
  - $x_0 = 1, x \in \mathbb{R}^{n_x+1}$
  - $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_{n_x} \end{bmatrix} \begin{array}{l} \rightarrow b \\ \left.\begin{array}{l} \\ \\ \\ \end{array}\right\} w \end{array}$ $\qquad \hat{y} = \sigma(\theta^T x)$

Logistic Regression Cost Function

- Model: $\hat{y} = \sigma(w^T x + b)$, where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Given training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$

Logistic Regression Cost Function

- Model:

  $\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$ and $z^{(i)} = w^T x^{(i)} + b$
- Given training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$

Logistic Regression Cost Function

- Model:

  $\hat{y}^{(i)} = \sigma\left(w^T x^{(i)} + b\right)$, where $\sigma\left(z^{(i)}\right) = \frac{1}{1+e^{-z^{(i)}}}$ and $z^{(i)} = w^T x^{(i)} + b$
- Given training set $\left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$
- Loss (error) function:
  - $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ $\rightarrow$ non-convex and multiple local optima

Logistic Regression Cost Function

- Model:

  $\hat{y}^{(i)} = \sigma\left(w^T x^{(i)} + b\right)$, where $\sigma\left(z^{(i)}\right) = \frac{1}{1+e^{-z^{(i)}}}$ and $z^{(i)} = w^T x^{(i)} + b$
- Given training set $\left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$
- Loss (error) function:
  - $\cancel{\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2}$ $\rightarrow$ non-convex and multiple local optima
  - $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y)\log(1-\hat{y}))$: cross-entropy loss

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

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

Logistic Regression Cost Function

- Model:

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

- Given training set $\left\{\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(m)}, y^{(m)}\right)\right\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$

- Loss (error) function:
  - $\cancel{\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2}$ $\rightarrow$ non-convex and multiple local optima
  - $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y}))$: cross-entropy loss

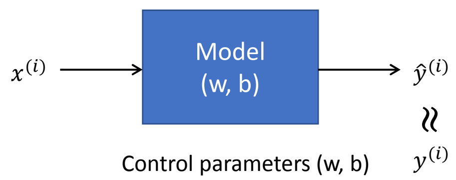
    If $y = 1 : \mathcal{L}(\hat{y}, y) = -y \log \hat{y}$ $\rightarrow$ want $\log \hat{y}$ large $\rightarrow$ want $\hat{y}$ large $\rightarrow$ want $\hat{y} \approx 1$

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

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

Recap

- Logistic regression model
- Loss function (cross-entropy loss)
  - measures how well your parameters w and b are doing on a single training example
- Cost function
  - measures how well your parameters w and b are doing on your entire training set
- Now let's investigate how to use the gradient descent algorithm to train the parameters w and b on your training set



Gradient Descent

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

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

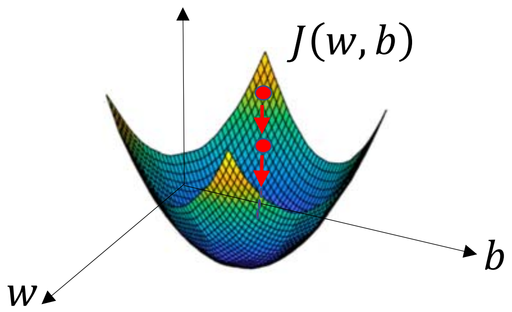
- We want to find w, b that minimize J(w,b)

Gradient Descent

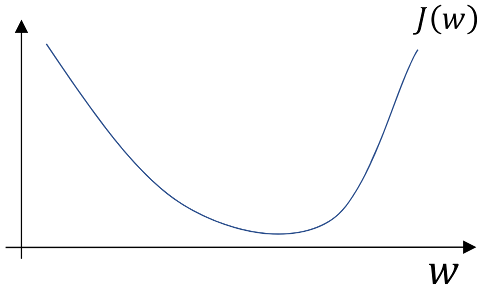
- Model: $\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b)$, where $\sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$ and $z^{(i)} = w^T x^{(i)} + b$
- Cost function:

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

- We want to find w, b that minimize J(w,b)



Gradient Descent

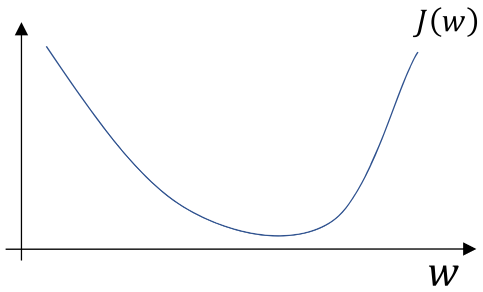


Repeat {

$$w \leftarrow w - \overset{\text{learning rate}}{\alpha} \underbrace{\frac{dJ(w)}{dw}}_{\text{derivative}}$$

}

Gradient Descent



Repeat {

$$w \leftarrow w - \overset{\text{learning rate}}{\alpha} \underbrace{\frac{dJ(w)}{dw}}_{\text{derivative}}$$

}

Repeat {

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

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

}

Computation Graph

- Computation of neural network
  - Forward propagation: we compute the output of the neural network
  - Backward propagation: we compute gradients (derivatives)

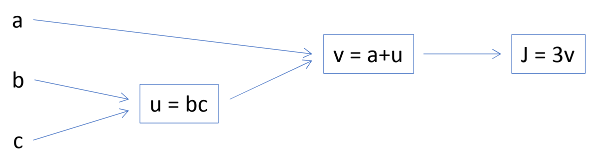
- Computation graph explains why it is organized in this way

Computation Graph

- We're trying to compute a function J of three variables
- J(a,b,c) = 3(a+bc)

  - Computing this function has three distinct steps
    u = bc → v = a+u → J = 3v

  - Computation graph:

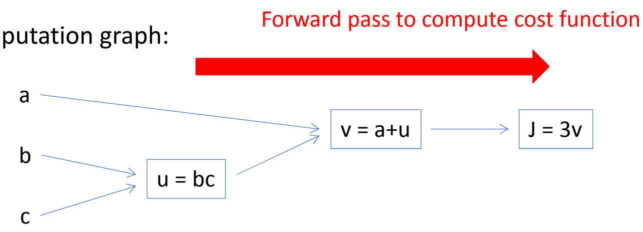


Computation Graph

- We're trying to compute a function J of three variables
- J(a,b,c) = 3(a+bc)

  – Computing this function has three distinct steps
  u = bc → v = a+u → J = 3v

  – Computation graph:

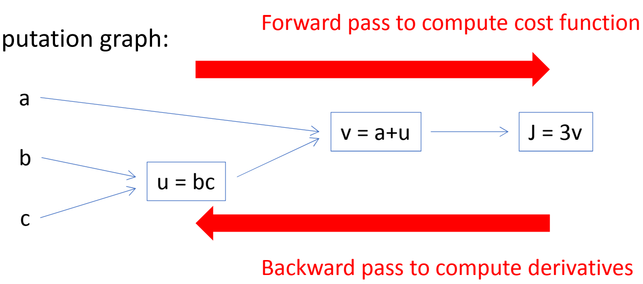


Computation Graph

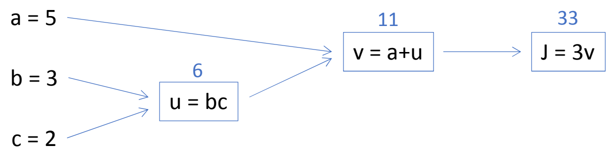
- We're trying to compute a function J of three variables
- J(a,b,c) = 3(a+bc)

  – Computing this function has three distinct steps
  u = bc → v = a+u → J = 3v

  – Computation graph:

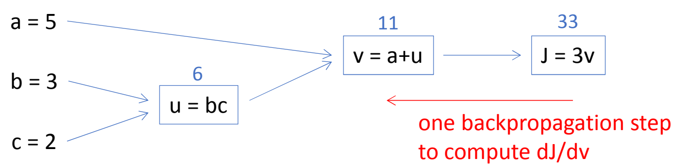


Derivatives with a Computation Graph



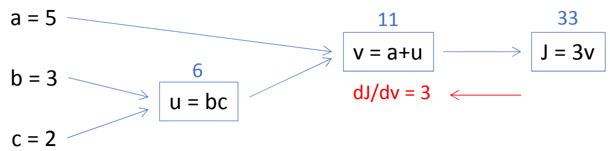
- dJ/dv = ?

Derivatives with a Computation Graph



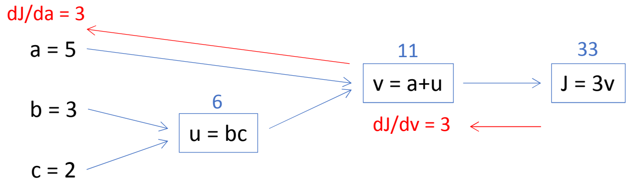
- dJ/dv = 3

Derivatives with a Computation Graph



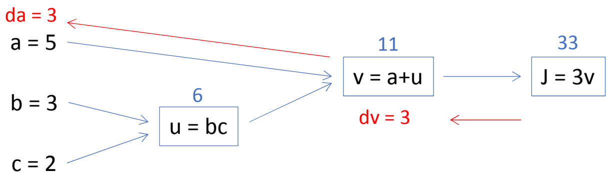
- dJ/dv = 3
- dJ/da = ?

Derivatives with a Computation Graph



- dJ/dv = 3
- dJ/da = dJ/dv · dv/da = 3 · 1 = 3 ("chain rule" in calculus)

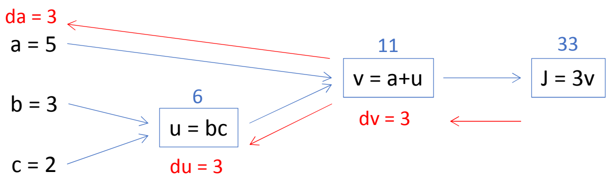
Derivatives with a Computation Graph



- dJ/dv = 3
- dJ/da = dJ/dv · dv/da = 3 · 1 = 3 ("chain rule" in calculus)

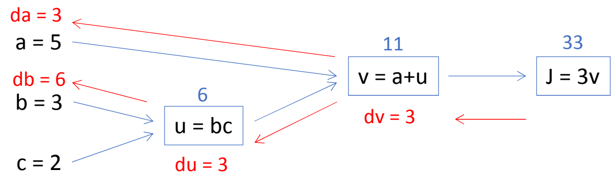
NOTE: dJ/d(var) is usually simplified into d(var) in implementing code

Derivatives with a Computation Graph



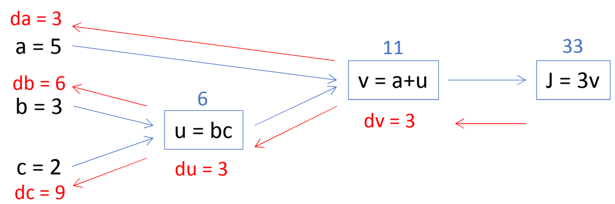
- dJ/dv = 3
- dJ/da = dJ/dv · dv/da = 3 · 1 = 3
- dJ/du = dJ/dv · dv/du = 3 · 1 = 3

Derivatives with a Computation Graph



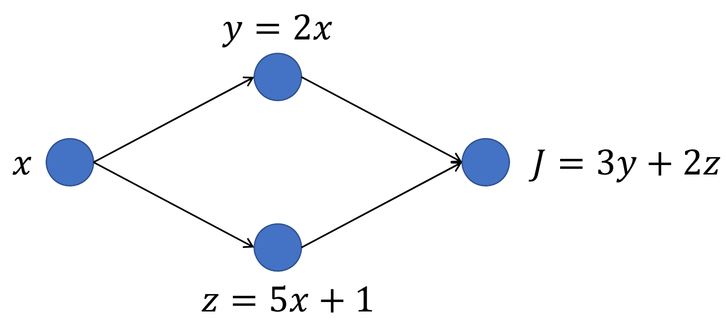
- dJ/dv = 3
- dJ/da = dJ/dv · dv/da = 3 · 1 = 3
- dJ/du = dJ/dv · dv/du = 3 · 1 = 3
- dJ/db = dJ/dv · dv/du · du/db = 3 · 1 · c = 3 · 1 · 2 = 6

Derivatives with a Computation Graph



- dJ/dv = 3
- dJ/da = dJ/dv · dv/da = 3 · 1 = 3
- dJ/du = dJ/dv · dv/du = 3 · 1 = 3
- dJ/db = dJ/dv · dv/du · du/db = 3 · 1 · c = 3 · 1 · 2 = 6
- dJ/dc = dJ/dv · dv/du · du/dc = 3 · 1 · b = 3 · 1 · 3 = 9

Exercise



$$\frac{\partial J}{\partial x} \ ?$$

Recap

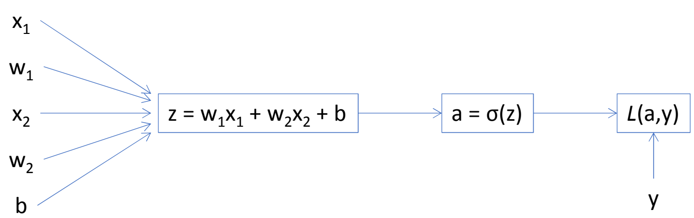
- Logistic regression model
- Loss function (cross-entropy loss)
  - measures how well your parameters w and b are doing on a single training example
- Cost function
  - measures how well your parameters w and b are doing on your entire training set
- Gradient descent
  - how to minimize the cost function by using gradient (derivatives)
  - gradient: how does a variable affect the value of the cost?
- Computing gradient
  - backpropagation with respect to computation graph

Logistic Regression Derivatives

- We will investigate how to compute derivatives for you to implement gradient descent for logistic regression
- Logistic regression:
  - $z = w^T x + b$ → linear function
  - $\hat{y} = a = \sigma(z)$ → nonlinear function
  - $\mathcal{L}(a, y) = -(y \log a + (1 - y) \log(1 - a))$ → loss for one example

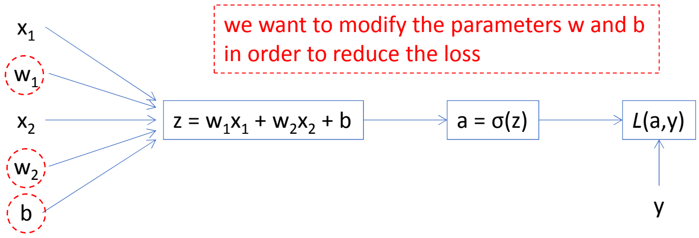
Logistic Regression Derivatives

- We will investigate how to compute derivatives for you to implement gradient descent for logistic regression
- Logistic regression:
  - $z = w^T x + b$ → linear function
  - $\hat{y} = a = \sigma(z)$ → nonlinear function
  - $\mathcal{L}(a, y) = -(y \log a + (1 - y) \log(1 - a))$ → loss for one example

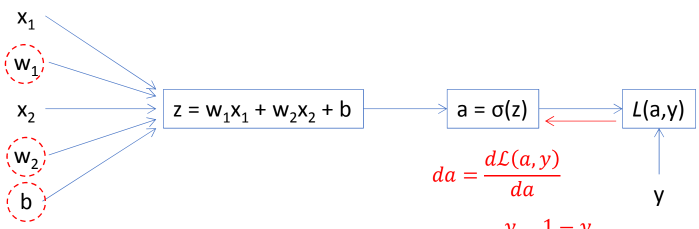


Logistic Regression Derivatives

- We will investigate how to compute derivatives for you to implement gradient descent for logistic regression
- Logistic regression:
  - $z = w^T x + b$ → linear function
  - $\hat{y} = a = \sigma(z)$ → nonlinear function
  - $\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$ → loss for one example



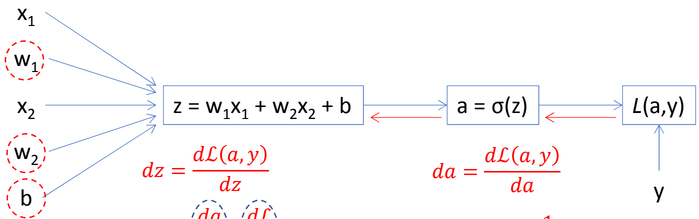
Logistic Regression Derivatives



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

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

Logistic Regression Derivatives



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

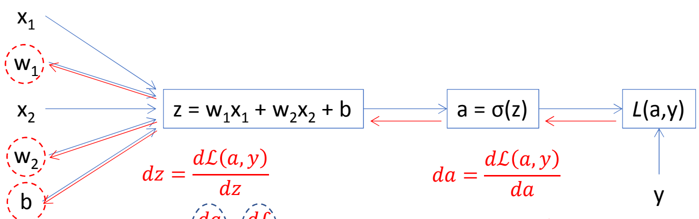
$$= \frac{da}{dz} \cdot \frac{d\mathcal{L}}{da} = a - y$$

$$a(1-a)$$

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

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

Logistic Regression Derivatives



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

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

$$a(1-a)$$

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

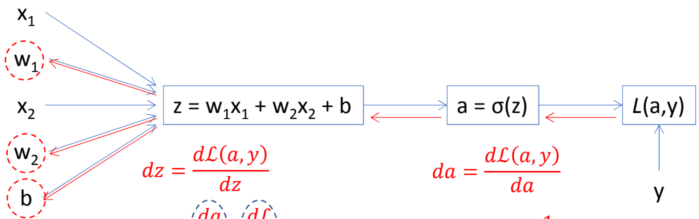
$$= -\frac{y}{a} + \frac{1-y}{1-a}$$

$$dw_1 = \frac{d\mathcal{L}(a,y)}{dw_1} = \frac{dz}{dw_1} \cdot \frac{d\mathcal{L}}{dz} = x_1 \cdot dz = x_1 \cdot (a-y)$$

$$dw_2 = \frac{d\mathcal{L}(a,y)}{dw_2} = \frac{dz}{dw_2} \cdot \frac{d\mathcal{L}}{dz} = x_2 \cdot dz = x_2 \cdot (a-y)$$

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

Logistic Regression Derivatives



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

$$\frac{da}{dz} = a(1-a)$$

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

$$dw_1 = \frac{d\mathcal{L}(a,y)}{dw_1} = \frac{dz}{dw_1} \cdot \frac{d\mathcal{L}}{dz} = x_1 \cdot dz = x_1 \cdot (a-y)$$

$$dw_2 = \frac{d\mathcal{L}(a,y)}{dw_2} = \frac{dz}{dw_2} \cdot \frac{d\mathcal{L}}{dz} = x_2 \cdot dz = x_2 \cdot (a-y)$$

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

Gradient Descent Algorithm

$$w_1 \longleftarrow w_1 - \alpha \cdot dw_1$$

$$w_2 \longleftarrow w_2 - \alpha \cdot dw_2$$

$$b \longleftarrow b - \alpha \cdot db$$

Logistic Regression on *m* Examples

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

- Derivatives:

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

$$\vdots$$

Logistic Regression on m Examples

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

For $i = 1$ to m

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

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

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

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

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

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

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

$J \mathrel{/}= m$

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

Logistic Regression on m Examples

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

For $i = 1$ to m

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

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

forward propagation

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

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

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

backward propagation

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

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

$J \mathrel{/}= m$

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

Logistic Regression on m Examples

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

For $i = 1$ to m

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

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

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

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

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

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

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

$J \ /= \ m$

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

gradient computation

$$w_1 \longleftarrow w_1 - \alpha \cdot dw_1$$

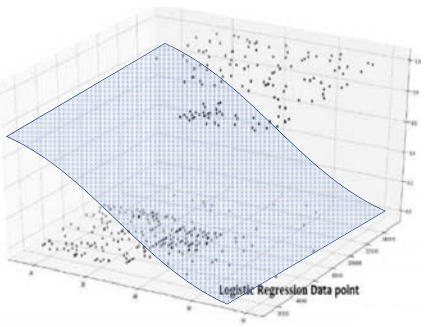
$$w_2 \longleftarrow w_2 - \alpha \cdot dw_2$$

$$b \longleftarrow b - \alpha \cdot db$$

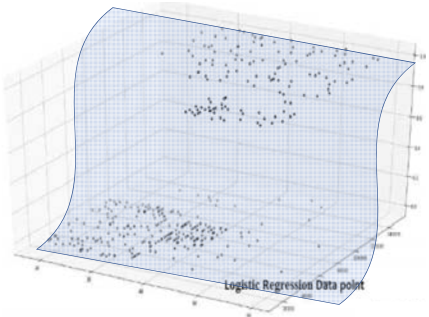
parameter update

Logistic Regression by Gradient Descent Algorithm

$$y = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + b)}}$$



when w_1, w_2, and b are (randomly) initialized



when w_1, w_2, and b are converged by their sufficient updates