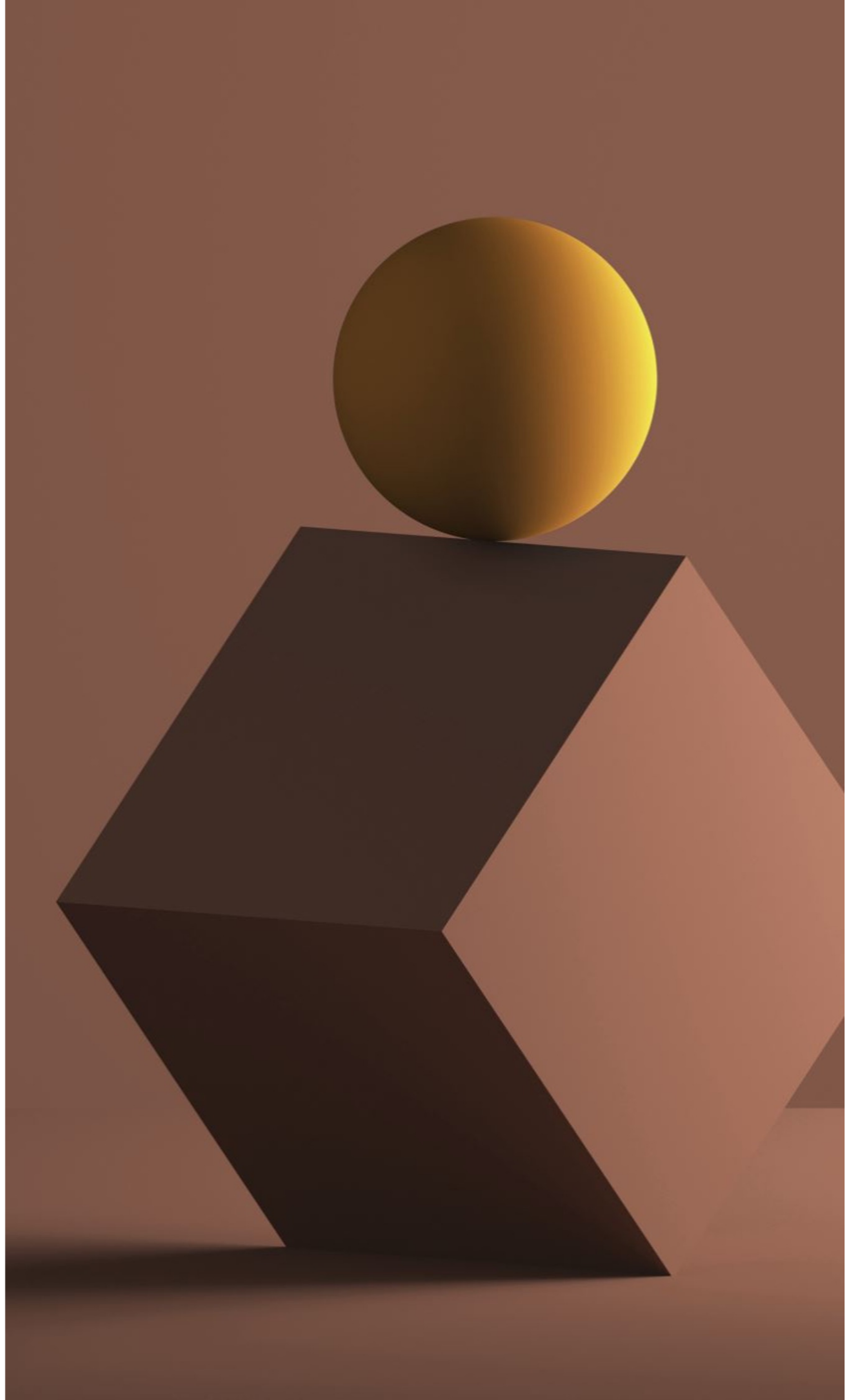
Logistic Regression

Deep Learning

Woohwan Jung



오픈카톡방

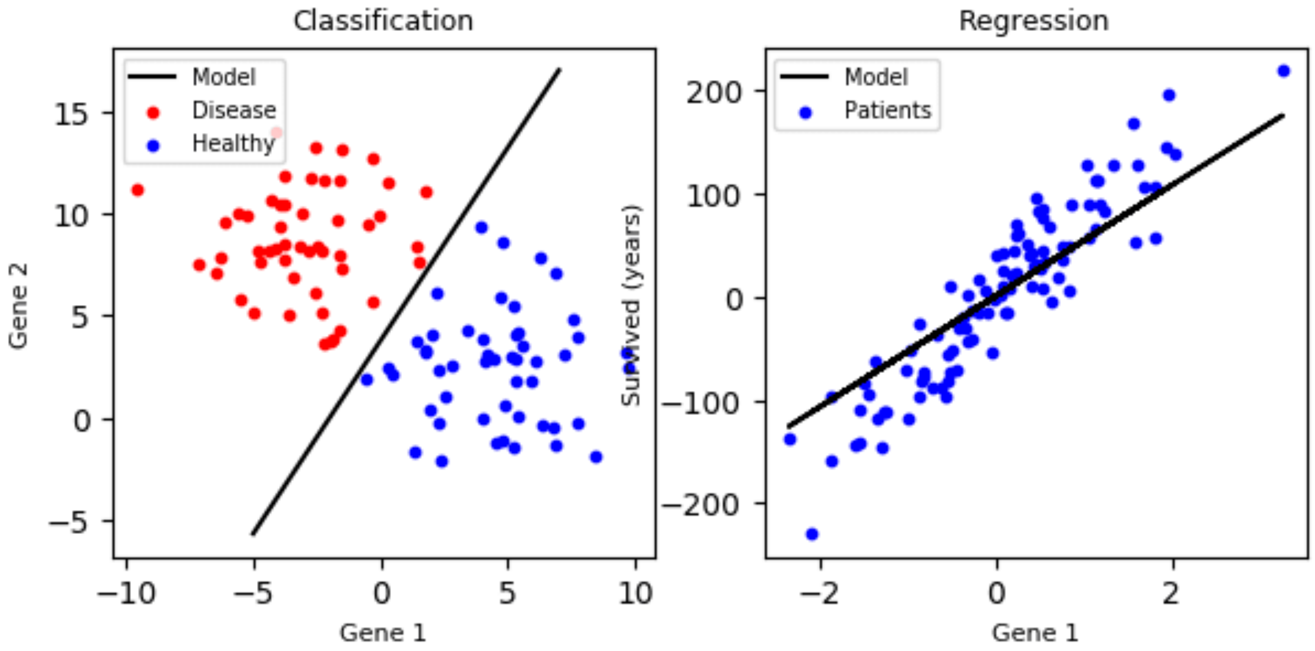
- https://open.kakao.com/o/gCLPx6Ce

- 수업중
  - 말로하기 어려운 복잡한 질문은 카톡으로 (말로하는걸 권장)
- 수업시간외
  - 학생들간 의견 및 자료교환
  - 학생들간 질문 (오픈카톡방에서 공유된 내용은 Copy X)

Topics

- Binary Classification
- Logistic Regression
  - Model
  - Cost/Loss function
  - Optimization

Classification vs Regression

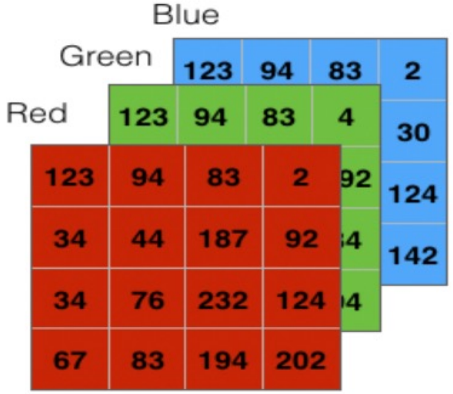


Logistic regression algorithm is used for **(binary) classification**!

Binary Classification

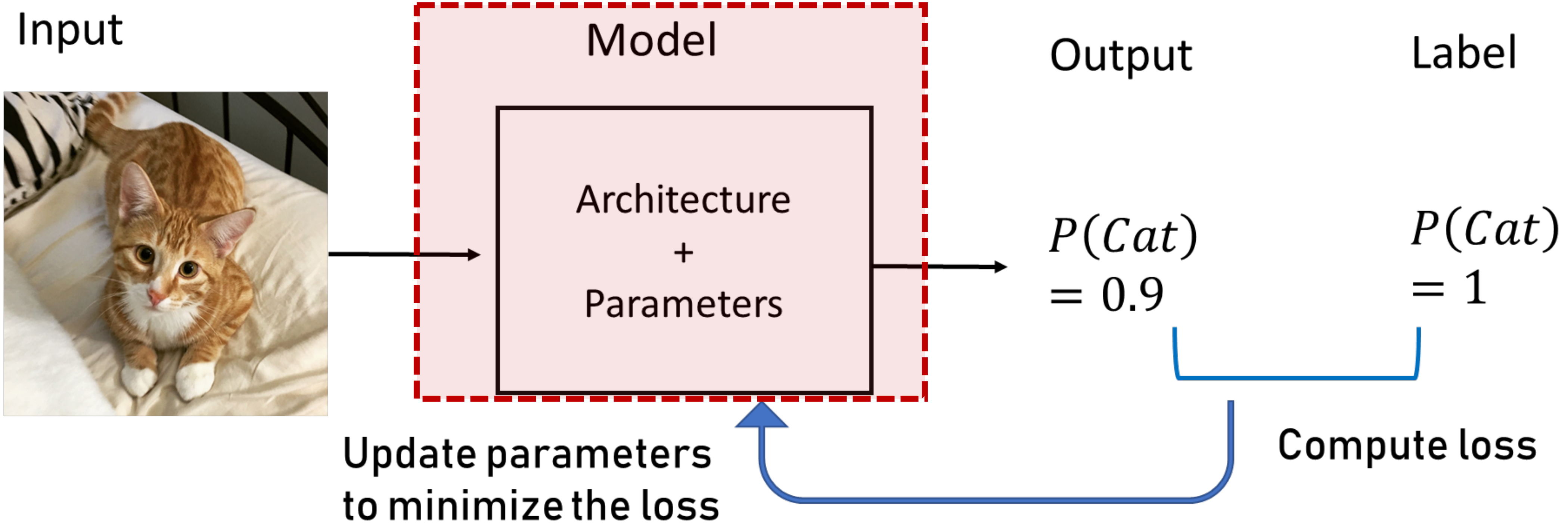
Model
=
Architecture
+
Parameters

1(cat) vs 0 (non cat)



$$x = \begin{bmatrix} 123 \\ 94 \\ \dots \\ 202 \\ 123 \\ 94 \\ \dots \\ 142 \end{bmatrix}$$

Model



Logistic Regression

Logistic Regression

- A simple model for binary classification
- Maybe one of the simplest neural network
- A training example $(\boldsymbol{x}, y)$
  - Input: $\boldsymbol{x} \in \mathbb{R}^n$
  - Output: $y \in \{0,1\}$
- m training examples
  - $\{(\boldsymbol{x}^{(\mathbf{1})}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \dots (\boldsymbol{x}^{(m)}, y^{(m)})\}$

Model = Architecture
+ Parameters

Review: Linear Regression

- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} \approx y$

$$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

Logistic Regression

Model = Architecture
+ Parameters

- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} = P(y = 1|\boldsymbol{x})$

$$\hat{y} = \sigma(\boldsymbol{w}^{\top}\boldsymbol{x} + b)$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

$$\Theta = \{\boldsymbol{w}, b\}$$

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

$$\sigma(-\infty) = 0$$

$$\sigma(+\infty) = 1$$

Logistic Regression

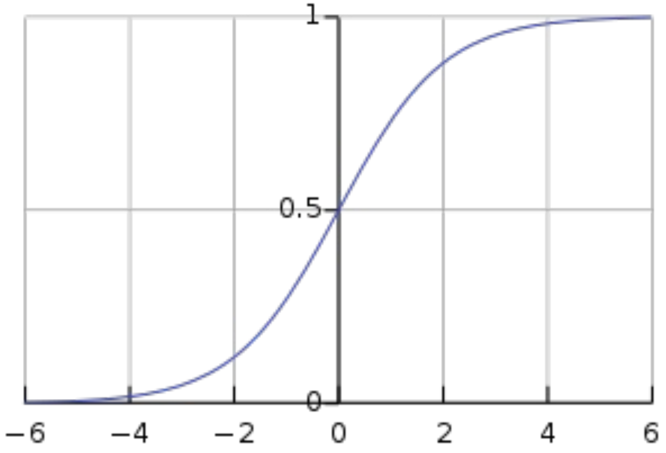
Model = Architecture
+ Parameters

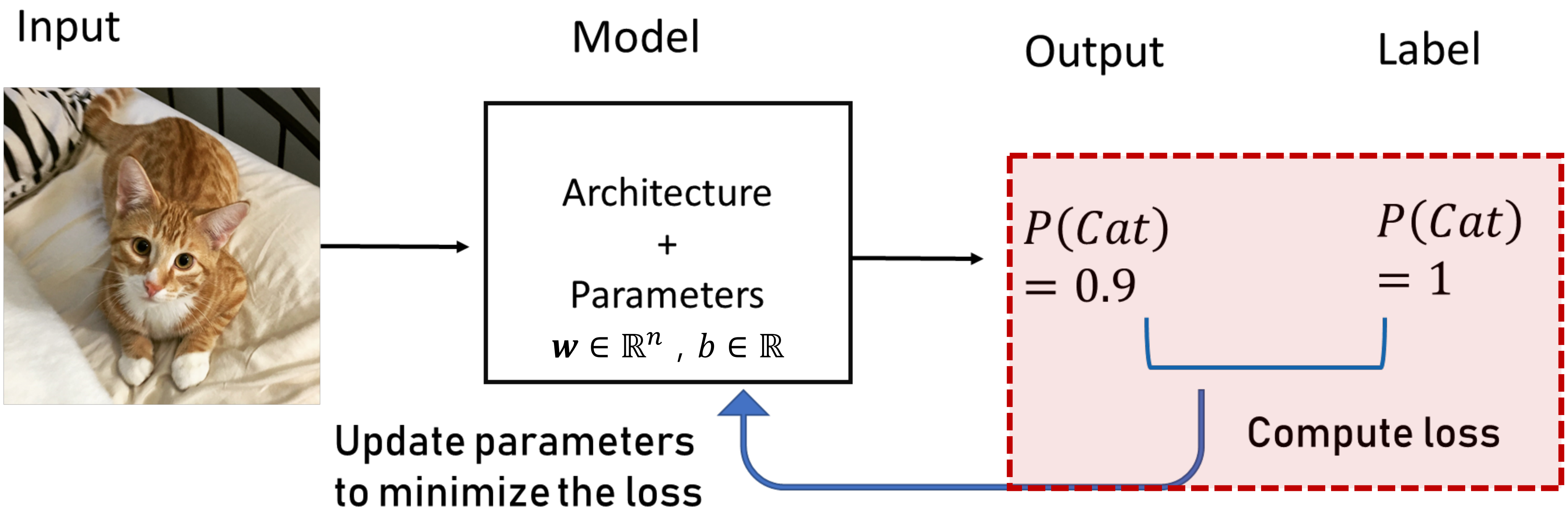
- Given $\boldsymbol{x} \in \mathbb{R}^n$
- Want $\hat{y} = P(y = 1|\boldsymbol{x})$

$$\hat{y} = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$$

Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$

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





Cost function & Loss

Logistic Regression

Logistic Regression: Cost function

Roughly, $\hat{y}^{(i)} \approx y^{(i)}$

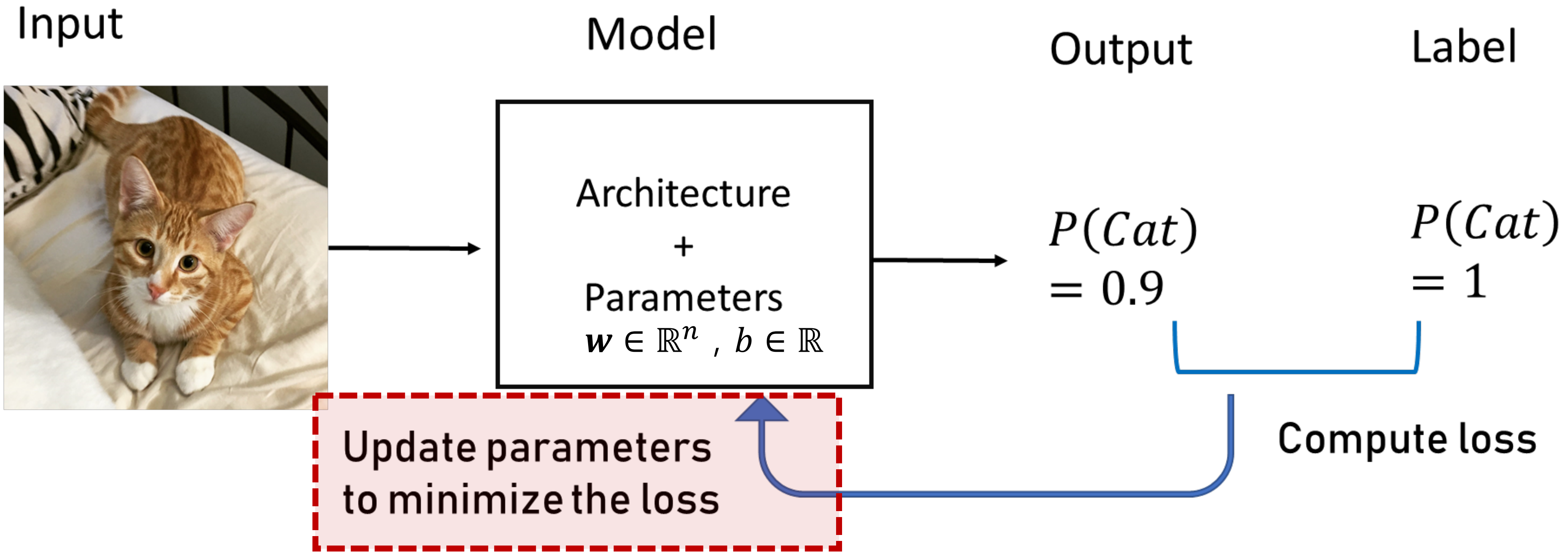
- $\hat{y} = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Given $\{(\boldsymbol{x}^{(1)}, y^{(1)}), \ldots (\boldsymbol{x}^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} = P(y^{(i)} = 1)$
- Loss function: Binary Cross Entropy

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

If $y = 1$: $L(\hat{y}, y) = -\log \hat{y}$
If $y = 0$: $L(\hat{y}, y) = -\log(1-\hat{y})$

- Cost function:

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



Optimization

Logistic Regression

Optimization

- Logistic regression model
  - $\hat{y} = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- Cost function

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

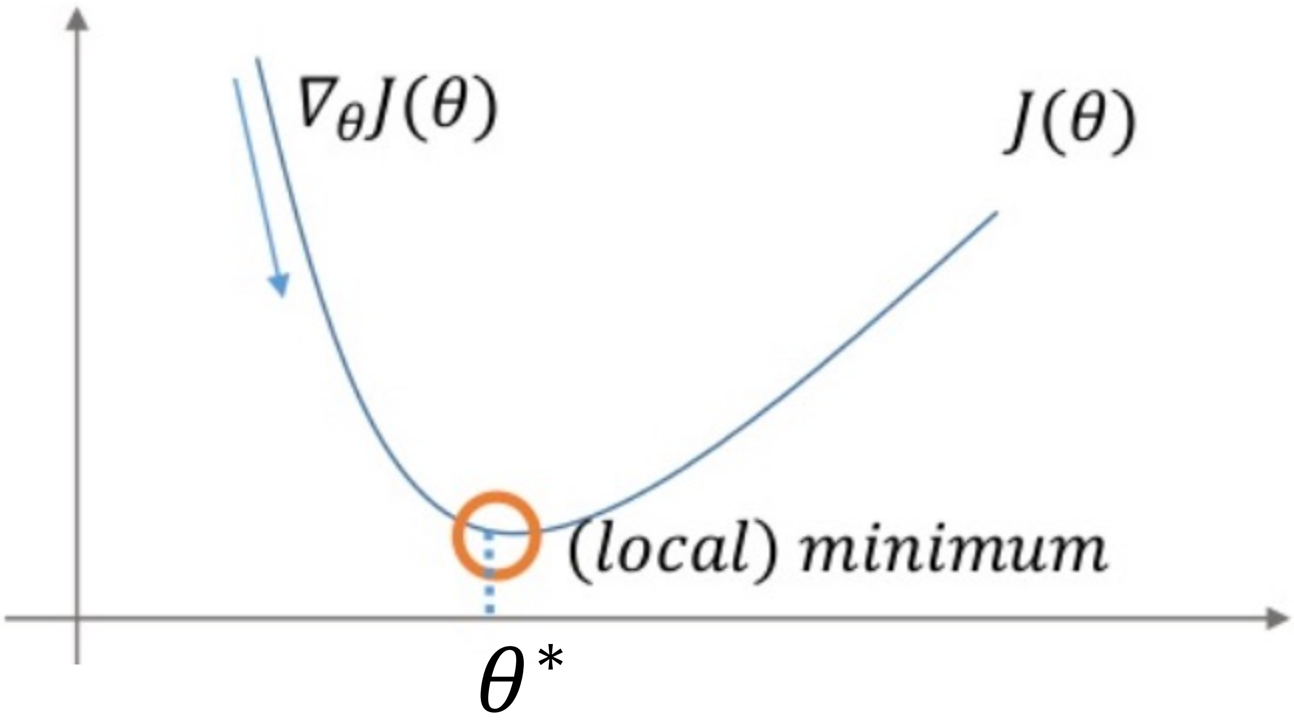
- Our goal
  - Find parameters $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$ that minimize $J(\boldsymbol{w}, b)$
- Gradient Descent!

Gradient Descent

- Algorithm to minimize a cost function $J(\theta)$
- $J(\theta)$: cost function
- θ: model parameters
- η: Learning rate

Repeatedly update

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

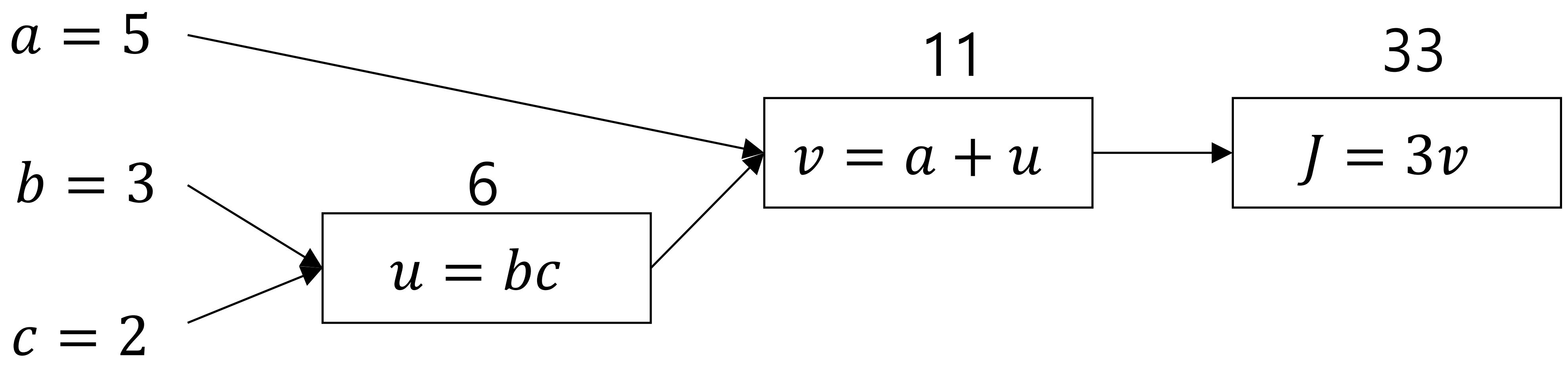


By Hakky St

Derivative with a Computation Graph

Computation Graph

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

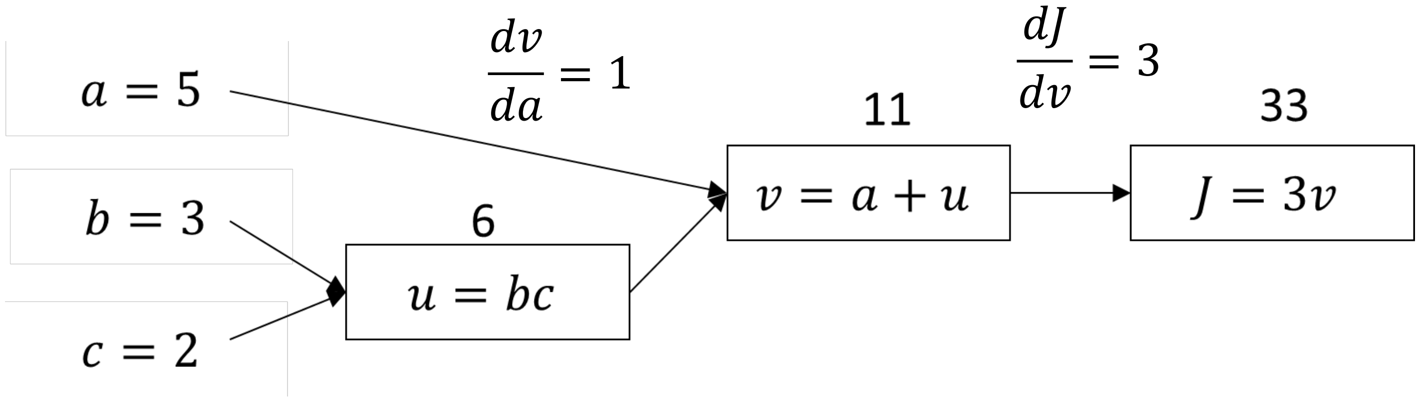


Chain Rule (calculus)

$$(f \circ g)'(c) = f'(g(c)) \cdot g'(c)$$

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Derivatives with a Computation Graph

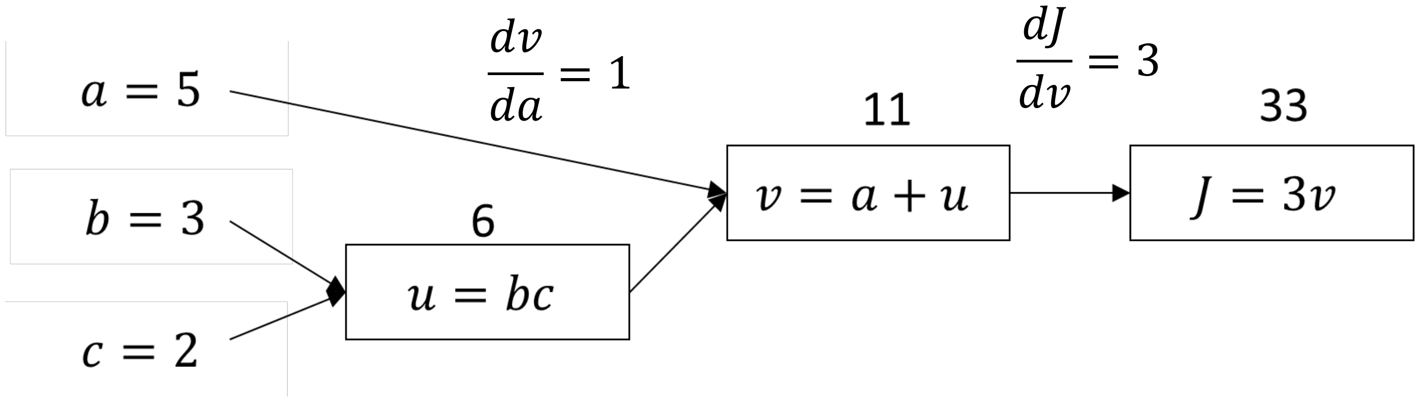


$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Chain rule

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

Derivatives with a Computation Graph



$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

Chain rule

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

$$\frac{dJ}{db} = \frac{dJ}{du}\frac{du}{db} = 2\ \frac{dJ}{du} = 2\ \frac{dv}{du}\frac{dJ}{dv} = 6$$

$$\frac{dJ}{dc} = \frac{dJ}{du}\frac{du}{dc} = 3\ \frac{dJ}{du} = 3\ \frac{dv}{du}\frac{dJ}{dv} = 9$$

Gradient Descent
:Logistic Regression

Background: Derivatives

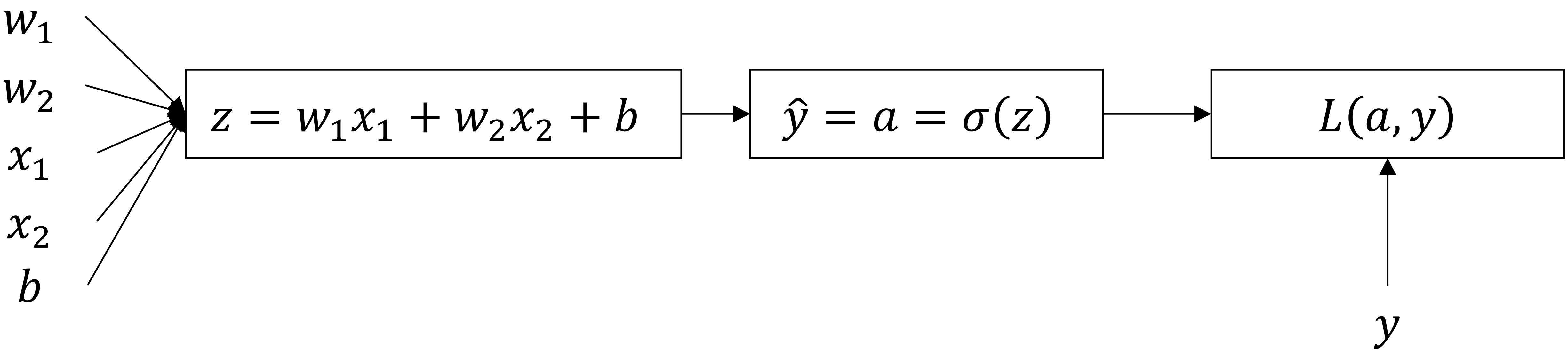
1. $\frac{d}{dx}\log_{\mathrm{e}} x = \frac{1}{x}$

2. $\frac{d}{dx}\sigma(x) = \sigma(x)(1 - \sigma(x))$

Logistic Regression Recap

- $\hat{y} = \sigma(\boldsymbol{w}^{\top}\boldsymbol{x} + b)$ where $\sigma(z) = \frac{1}{1+e^{-z}}$
- $L(\hat{y}, y) = -y \log \hat{y} - (1-y)\log(1-\hat{y})$

For the simplicity
$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$



$$L(a,y) = -y\log a - (1-y)\log(1-a)$$

Logistic Regression Derivative

$$\begin{tikzcd} & & & y \arrow[d] \\ w_1,\ w_2,\ x_1,\ x_2,\ b \arrow[r, "\frac{dz}{dw_k}"'] & z = w_1x_1 + w_2x_2 + b \arrow[r, "\frac{da}{dz}"'] & \hat{y} = a = \sigma(z) \arrow[r, "\frac{dL}{da}"'] & L(a,y) \end{tikzcd}$$

$$\frac{dL}{dw_k} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_k} = \left(-\frac{y}{a} + \frac{1-y}{1-a}\right)\sigma(z)(1-\sigma(z))x_k$$

$$L(a, y) = -y \log a - (1 - y) \log(1 - a)$$

Logistic Regression Derivative

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

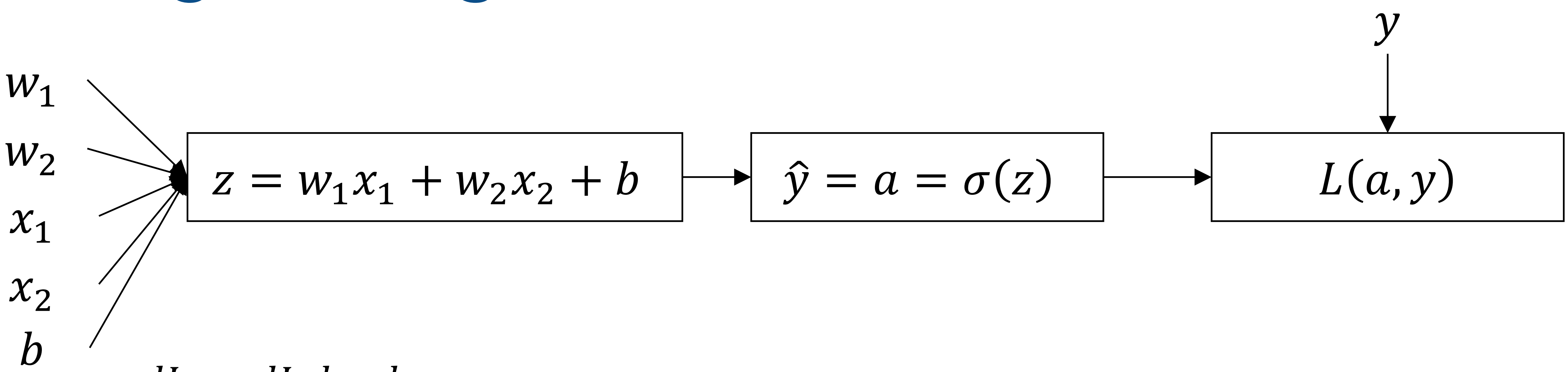
$$\frac{dL}{dw_k} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_k} = \left(-\frac{y}{a} + \frac{1-y}{1-a}\right)\sigma(z)(1 - \sigma(z))x_k$$

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

$$= (a - y)x_k$$

$$L(a,y) = -y\log a - (1-y)\log(1-a)$$

Logistic Regression Derivative



$$\frac{dL}{dw_1} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_1} = (a-y)x_1 \qquad w_1 := w_1 - \eta\frac{dL}{dw_1}$$

$$\frac{dL}{dw_2} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{dw_2} = (a-y)x_2 \qquad w_2 := w_2 - \eta\frac{dL}{dw_2}$$

$$\frac{dL}{db} = \frac{dL}{da}\frac{da}{dz}\frac{dz}{db} = a-y \qquad b := b - \eta\frac{dL}{db}$$

Gradient descent on m examples

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

$$\frac{d}{dw_k} J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{d}{dw_k} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y\right) x_k$$

$$\frac{d}{db} J(\boldsymbol{w}, b) = \frac{1}{m}\sum_{i=1}^{m} \frac{d}{db} L\left(\hat{y}^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m} \left(\hat{y}^{(i)} - y\right)$$

Logistic Regression on m example

- Initialize $\boldsymbol{w}, b$
- lr = 0.1
- For e = 1 to n_{epoch}:
    - J = 0; d_w1 = 0; d_w2 = 0; d_b=0
    - For i= 1 to m:
        - $z = w_1 x_1^{(i)} + w_2 x_2^{(i)} + b$
        - $a = \sigma(z)$
        - d_w1 += $(a - y)x_1^{(i)}$
        - d_w2 += $(a - y)x_2^{(i)}$
        - d_b += $a - y$
    - w_1 −= lr ∗ d_w1/m
    - w_2 −= lr ∗ d_w2/m
    - b −= lr ∗ d_b/m

$$\boldsymbol{w}, \boldsymbol{x}, b \rightarrow \boxed{\boldsymbol{w}^{\top}\boldsymbol{x} + b} \rightarrow \boxed{a = \sigma(z)} \rightarrow \boxed{L(a, y)}$$

$$\frac{d}{dw_2} J(\boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{d}{dw_2} L\left(a^{(i)}, y^{(i)}\right) = \frac{1}{m} \sum_{i=1}^{m} \left(a^{(i)} - y^{(i)}\right) x_2$$

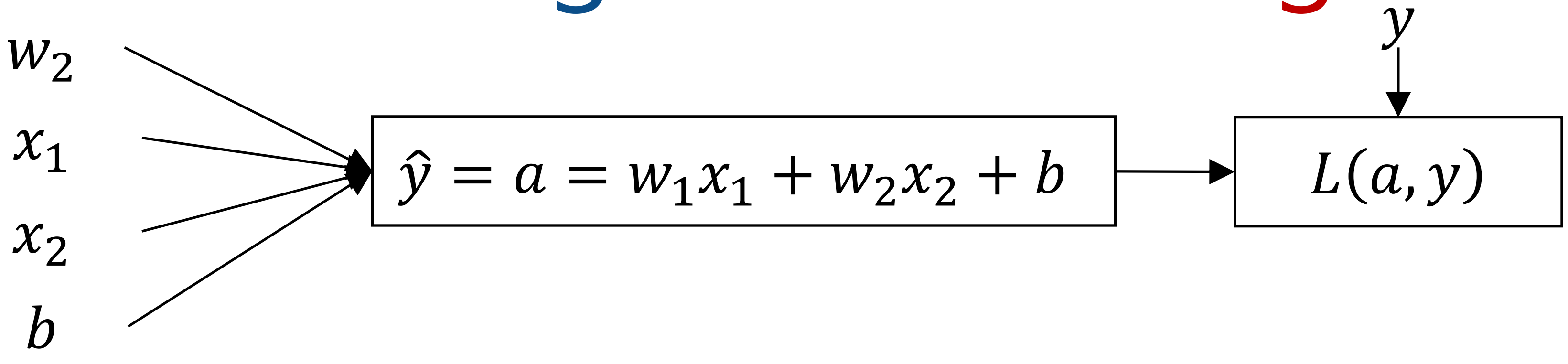
$$\frac{d}{dw_2} J(\boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{d}{dw_2} L\left(a^{(i)}, y^{(i)}\right) = \frac{1}{m} \sum_{i=1}^{m} \left(a^{(i)} - y^{(i)}\right) x_2$$

$$\frac{d}{db} J(\boldsymbol{w}, b) = \frac{1}{m} \sum_{i=1}^{m} \frac{d}{db} L\left(a^{(i)}, y^{(i)}\right) = \frac{1}{m} \sum_{i=1}^{m} \left(a^{(i)} - y^{(i)}\right)$$

	Linear Regression	Logistic Regression
Problem	Regression	Classification
Model	$\hat{y} = \boldsymbol{w}^\top \boldsymbol{x} + b$ Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$	$\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{x} + b\right)$ Parameters: $\boldsymbol{w} \in \mathbb{R}^n$, $b \in \mathbb{R}$
Loss	Squared Error $L(y, \hat{y}) = (y - \hat{y})^2$	Binary Cross Entropy (BCE) $L(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$

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

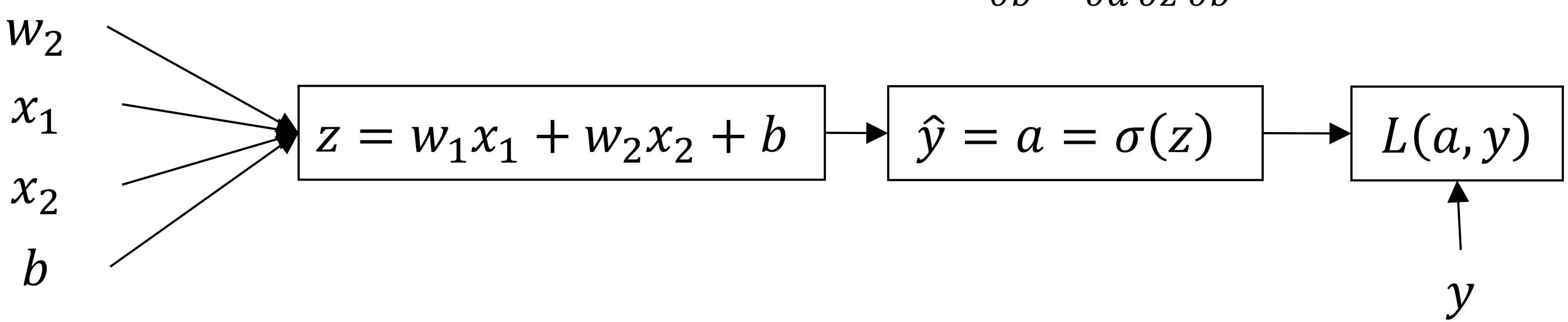
Linear regression vs Logistic regression



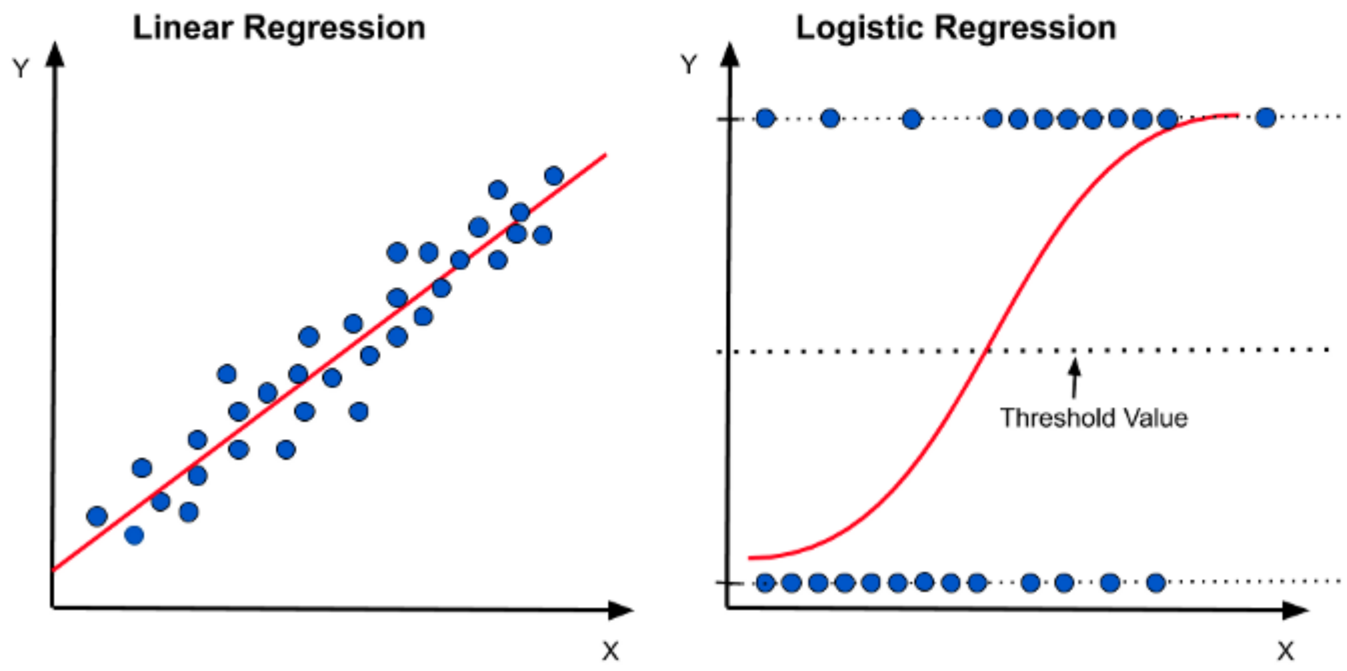
For the simplicity
$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $\boldsymbol{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial w_i} = 2(a - y)x_i \qquad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial b} = 2(a - y)$$

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial w_i} = (a - y)x_i \qquad \frac{\partial L}{\partial b} = \frac{\partial L}{\partial a}\frac{\partial a}{\partial z}\frac{\partial z}{\partial b} = (a - y)$$



Why is Logistic Regression Called Logistic Regression?



Logistic Regression

Programming in Python

Logistic regression: logical AND

x1	x2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$y = x_1 \; AND \; x_2$$

Data preparation

Logistic regression (AND)

In [ ]:

```
import random
from math import exp,log
```

Data prepration

In [12]:

```
X = [(0,0),(1,0),(0,1),(1,1)]
Y = [0,0,0,1]
```

Model

Model

```
In [14]: class logistic_regression_model():
             def __init__(self):
                 self.w = [random.random(), random.random()]
                 self.b = random.random()

             def sigmoid(self,z):
                 return 1/(1 + exp(-z))

             def predict(self,x):
                 z = self.w[0] * x[0] + self.w[1] * x[1] + self.b
                 a = self.sigmoid(z)
                 return a
```

```
In [15]: model = logistic_regression_model()
```

Training

Training

In [16]:
```python
def train(X, Y, model, lr = 0.1):
    dw0 = 0.0
    dw1 = 0.0
    db = 0.0
    m = len(X)
    cost = 0.0
    for x,y in zip(X,Y):
        a = model.predict(x)
        if y == 1:
            cost -= log(a)
        else:
            cost -= log(1-a)

        dw0 += (a-y)*x[0]
        dw1 += (a-y)*x[1]
        db += (a-y)

    cost /= m
    model.w[0] -= lr * dw0/m
    model.w[1] -= lr * dw1/m
    model.b -= lr*db/m

    return cost
```

In [17]:
```python
for epoch in range(10000):
    cost = train(X,Y, model, 0.1)
    if epoch %100==0:
        print(epoch, cost)
```

```
0 0.9799277803394626
100 0.4455359447221918
200 0.3527852128241023‍6
300 0.2946984536660345‍3
400 0.25432071172280113
500 0.22425431605184998
600 0.20079558352997323

...

9000 0.019352012397599427
9100 0.019139595049452222
9200 0.018931735521070026
9300 0.018728289777080104
9400 0.018529119755095403
9500 0.01833409306055272
9600 0.018143082680009734
9700 0.01795596671161366
9800 0.017772628111558907
9900 0.017592954455438015
```

Testing

Testing

In [22]:
```
model.predict((0,0))
```
Out[22]: 1.2451625968657186e-05

In [23]:
```
model.predict((0,1))
```
Out[23]: 0.020240526677753723

In [24]:
```
model.predict((1,0))
```
Out[24]: 0.02024051938891944

In [25]:
```
model.predict((1,1))
```
Out[25]: 0.9716510306648906