

Introduction to deep learning

STAT 601.28 Week 2

Qingrun Zhang

Table of content

- What is neural network?
- Supervised Learning with Neural Network
- Logistic regression in NN view

This startup wants to help small businesses survive expensive lawsuits by funding them

Sound familiar, Peter Thiel?

Not every person or business can brush aside a multimillion dollar lawsuit — but for those that can afford to fight, a winning case can mean a profitable outcome.

That's what Legalist, a startup in Y Combinator's Summer 2016 class, is banking on by offering funding to companies in litigation battles. For example, the startup has invested \$75,000 for one lawsuit and said it expects to receive \$1 million in the outcome.

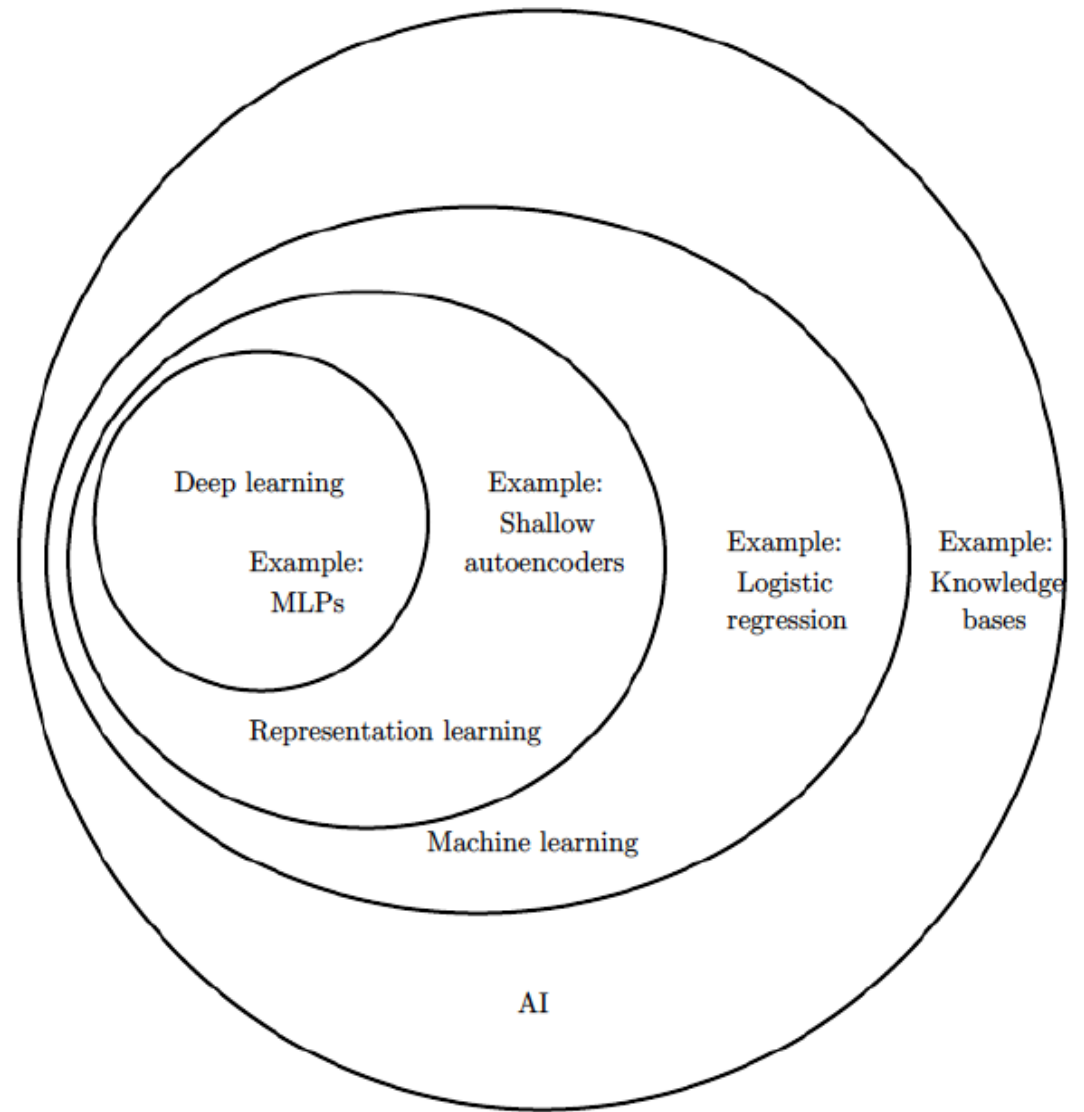
<https://mashable.com/article/legalist-startup-lawsuit>

What is Neural Network



© dreamstime.com

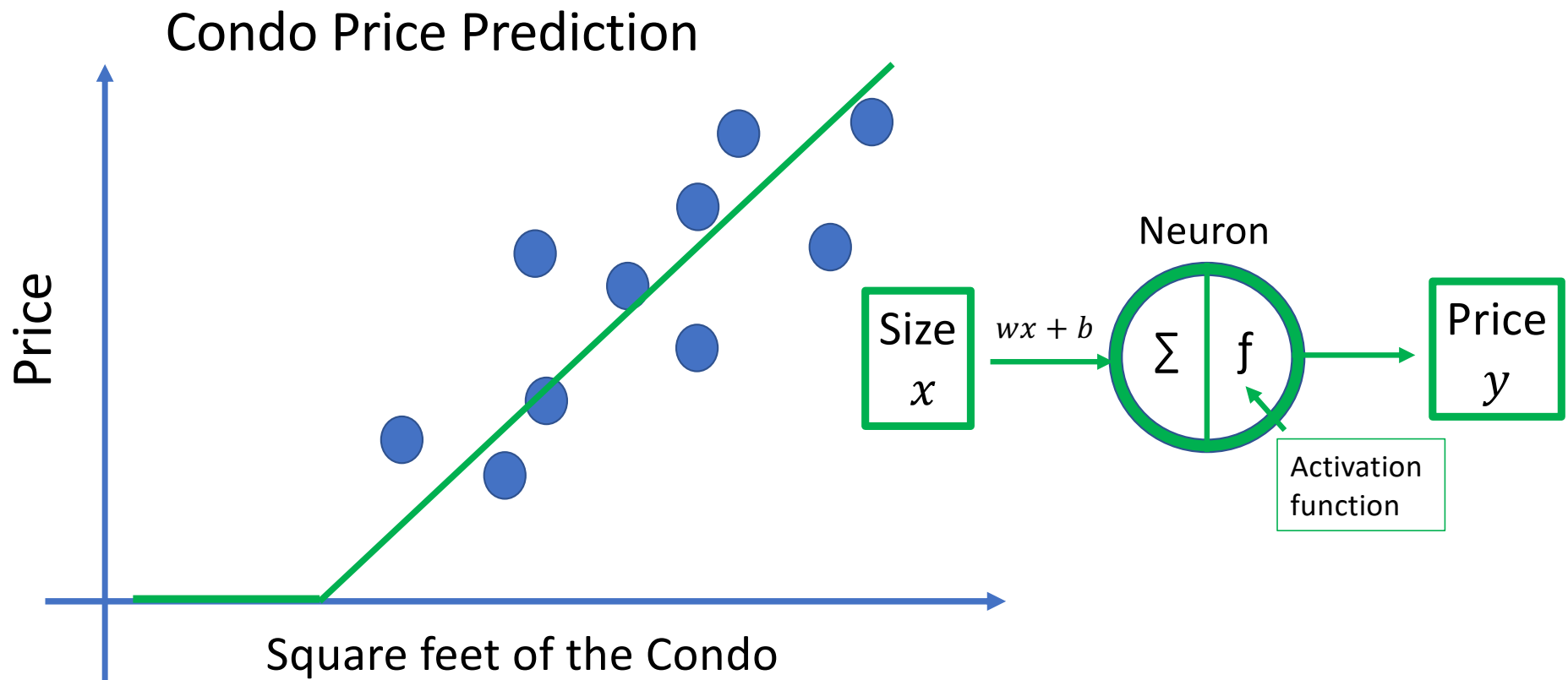
ID 186922137 © Nickylarson974



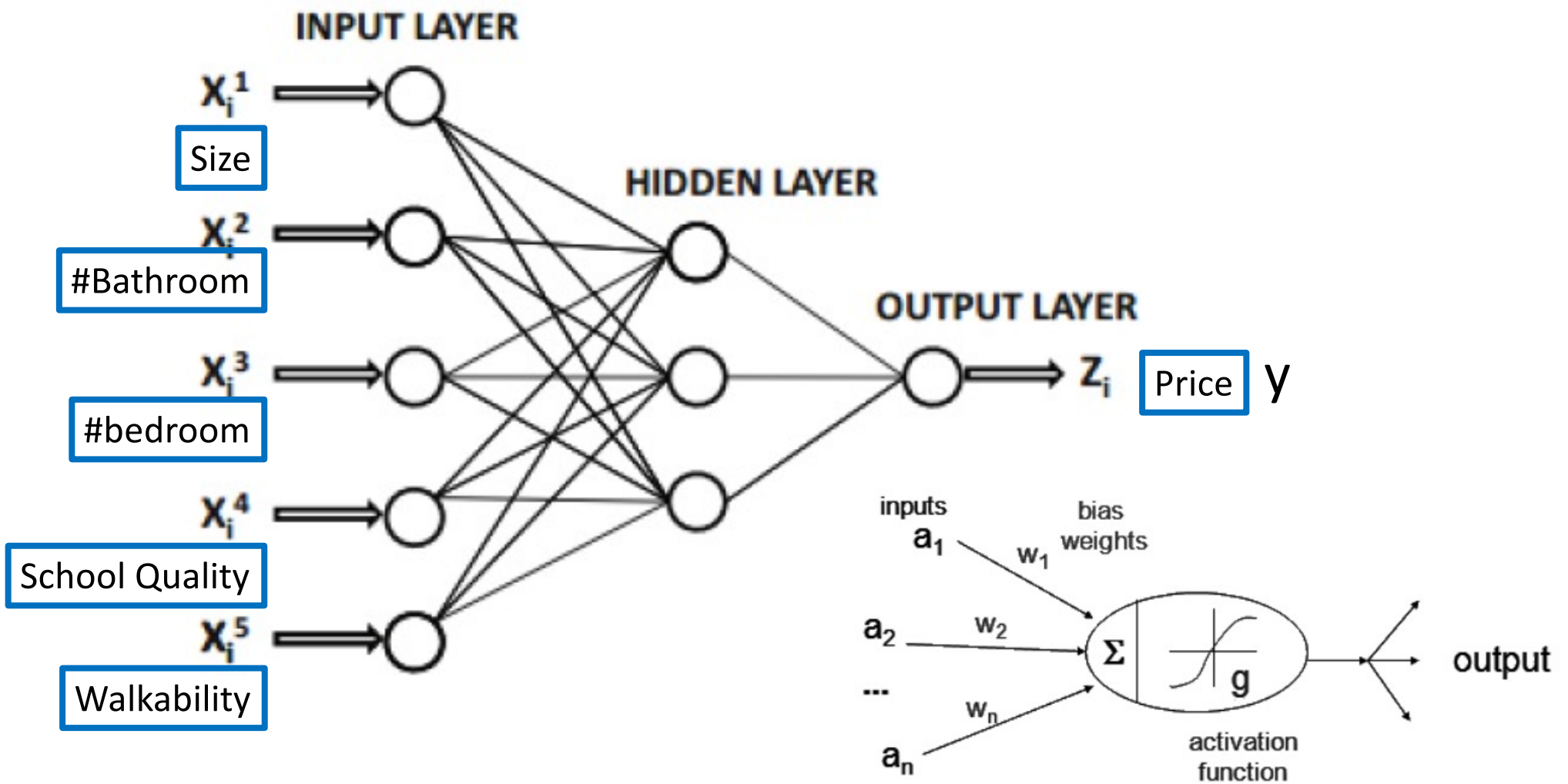
Types of neural networks

- **The perceptron**, the simplest NN, created by Frank Rosenblatt in 1958, has a single neuron
- **Feedforward neural networks, or multi-layer perceptrons (MLPs)** are comprised of an input layer, a hidden layer or layers, and an output layer. they are actually comprised of sigmoid neurons.
- **Convolutional neural networks (CNNs)** are utilized for image recognition, pattern recognition, and/or computer vision
- **Recurrent neural networks (RNNs)** are identified by their feedback loops. These learning algorithms are primarily leveraged when using time-series data to make predictions about future outcomes,

<https://www.ibm.com/cloud/learn/neural-networks>



Neural networks are good at figuring out functions relating an input x to an output y given enough examples.



Supervised Learning with Neural Network

Input (X)	Output (Y)	Application	
Housing features	Price	Real Estate	Standard NN
Ad, user info	Probability of click an ad	Online advertising	Standard NN
Image	Object	Photo Tagging	CNN
Audio	Text Transcript	Speech Recognition	RNN
Lawsuit features	Probability of success	Lawsuit	Customize Hybrid
Image, Radar info	Position of other objects	Autonomous driving	Customize Hybrid

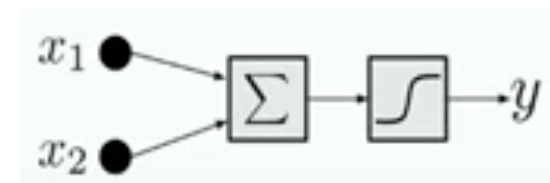
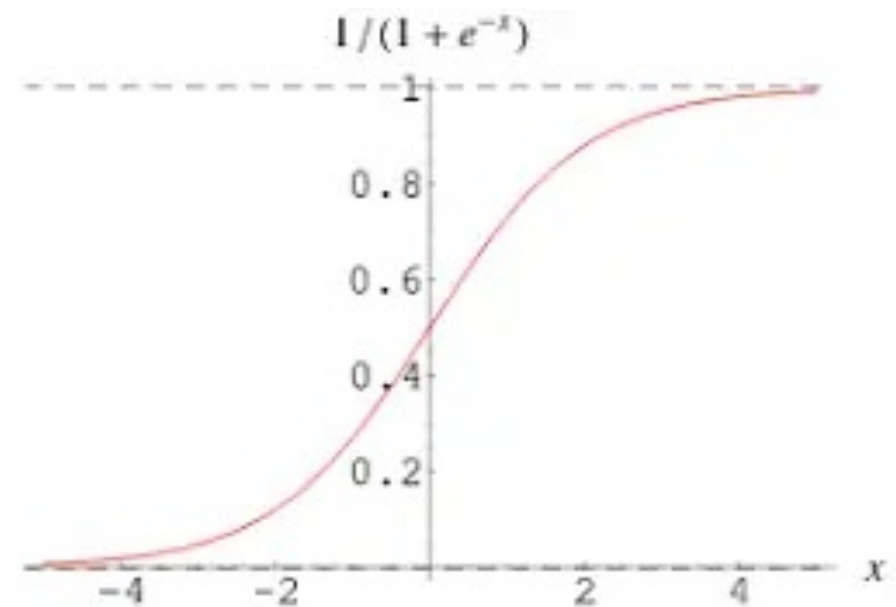
Structured Data	Unstructured Data
Housing price	Audio
Advertising	image
	text

Which of the following are examples of structured data?

- A dataset with short poems
- A dataset of weight, height, age, blood sugar level, and arterial pressure
- A set of audio recordings of a person reading a sentence
- A dataset with zip code, house price, house square feet, # of bathrooms, # of bedrooms.

Binary classification – Logistic Regression

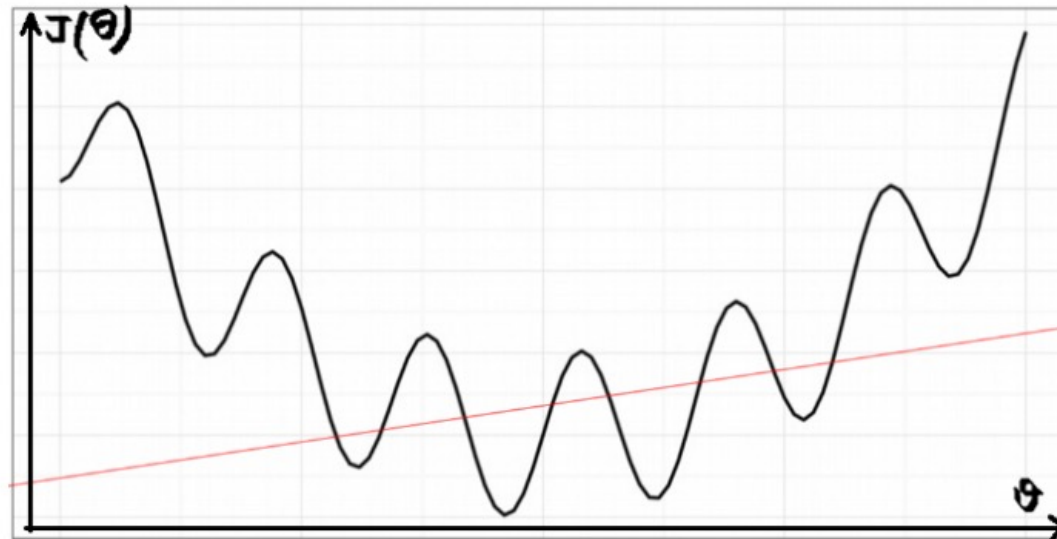
- Give input x , we want to predict the output $\hat{y} = P(y = 1|x)$
- Input: $x \in \mathbb{R}^{n_x}$
- Parameters: $w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$
- Output: $\hat{y} = \sigma(w^T X + b)$
- $z = w^T X + b$
- $\hat{y} = \frac{1}{(1+e^{-z})}$
- When z is large (∞), $\hat{y} \approx \frac{1}{1+0} = 1$
- When z is small ($-\infty$), $\hat{y} \approx \frac{1}{1+\infty} = 0$



What are the parameters of logistic regression?

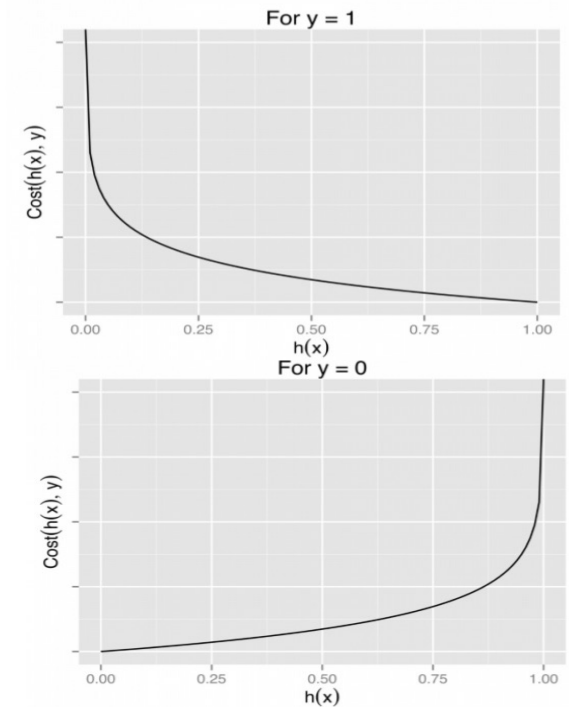
Cost function

In linear regression: cost function defined as $J(\theta) = \frac{1}{2} \sum_{i=1}^m (y - \hat{y})^2$ If we try to use the cost function of the linear regression in 'Logistic Regression' then it would be of no use as it would end up being a **non-convex** function with many local minimums, in which it would be very **difficult** to **minimize the cost value** and find the global minimum.



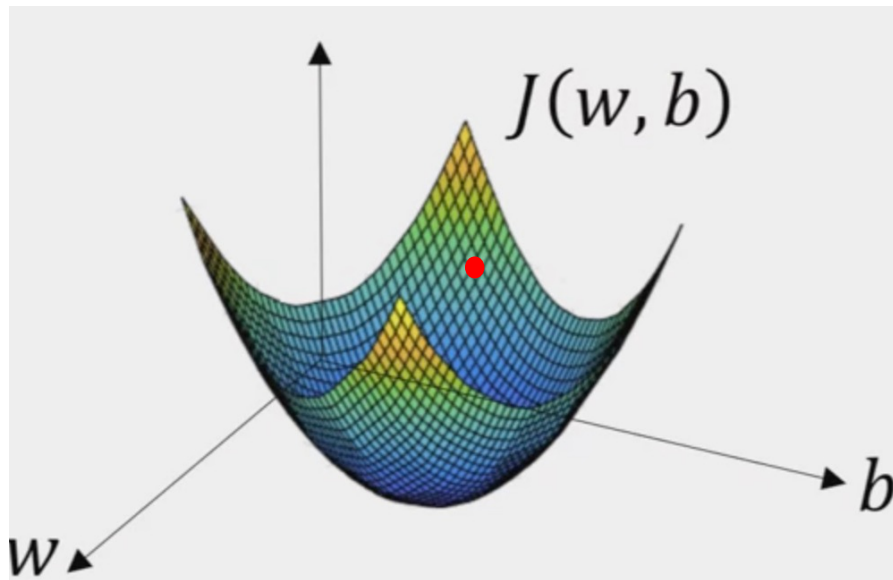
Logistic Regression cost function

- $\hat{y} = \sigma(z)$ where $z = w^T X + b$
- Given $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$
- Loss (error) function:
- $\mathcal{L}(\hat{y}, y) = -(y \log \hat{y} + (1 - y) \log(1 - \hat{y}))$
- If $y = 1$: $\mathcal{L}(\hat{y}, y) = -\log \hat{y}$. We want $\log \hat{y}$ large, \hat{y} large
- If $y = 0$: $\mathcal{L}(\hat{y}, y) = -\log(1 - \hat{y})$. *we want* $(1 - \hat{y})$ large, \hat{y} small
- Cost function:
- $J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$



Gradient Descent

- $\hat{y} = \sigma(z)$ where $z = w^T X + b$
- $J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$
- We want to find w, b that minimize cost function $J(w, b)$



Repeat{

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

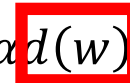
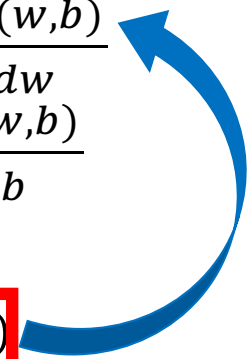
$$b := b - \alpha \frac{dJ(w, b)}{db}$$

}

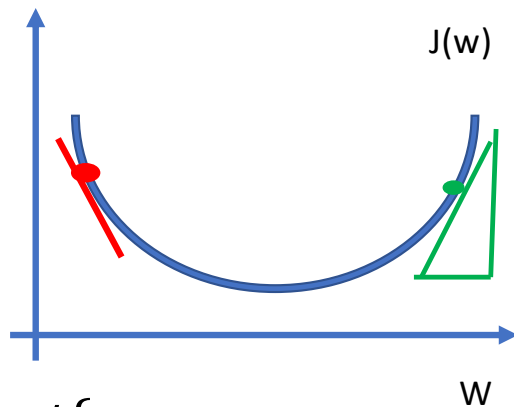
$$w := w - \alpha d(w)$$

$$b := b - \alpha d(b)$$

Learning Rate



Gradient descent



Repeat{

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

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

}

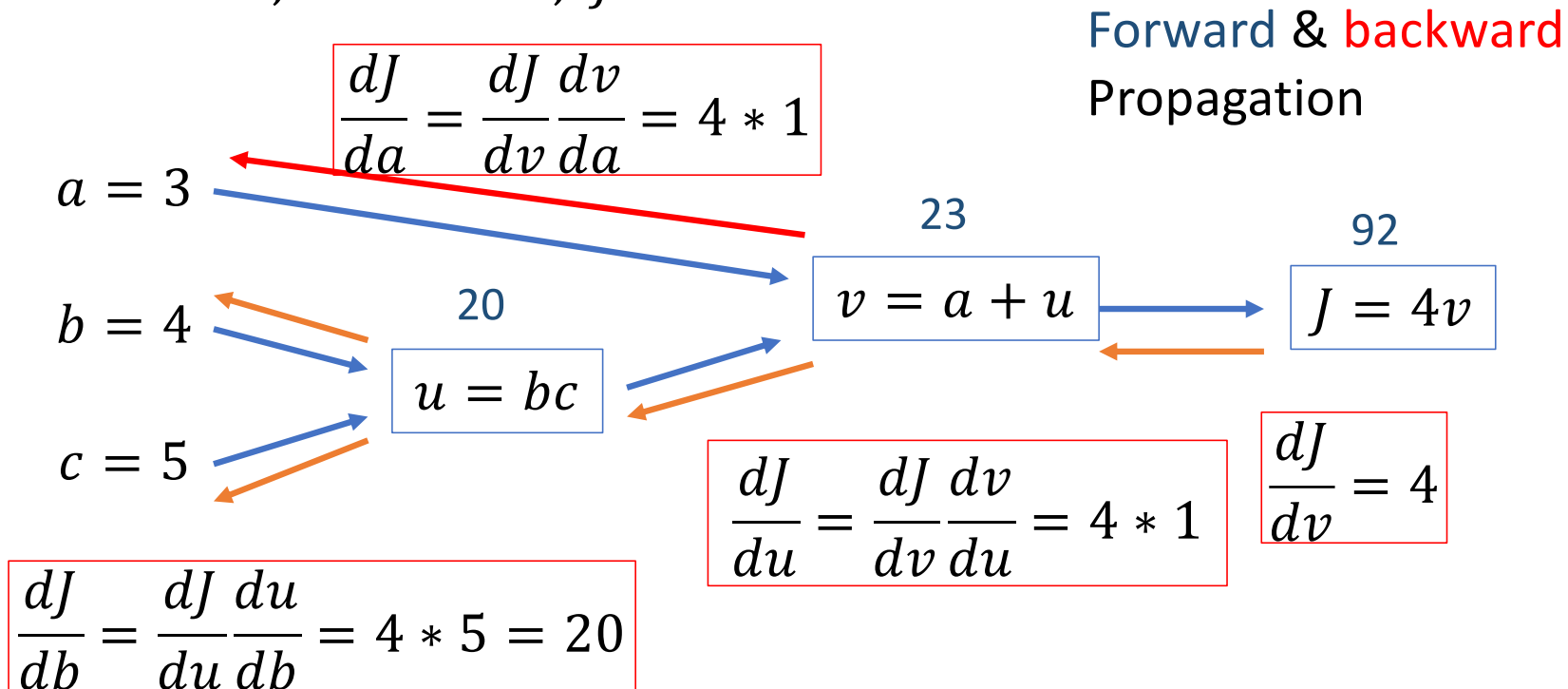
- If you start with large w at the green point, the derivative is positive,

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

- w will update as w minus a positive number, so gradient descent will make your algorithm slowly decrease the parameter w .
- If you start with small w , the derivative at red point is negative, w will update as w minus a negative number, gradient descent will make your algorithm slowly increase the parameter w .

Computation graph

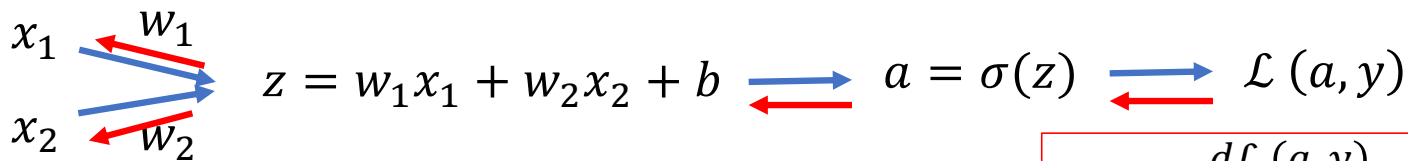
- $J(a, b, c) = 4(a + bc)$
- $u = bc, v = a + u, J = 4v$



Logistic Regression Gradient Descent: 1 example

Activation

- $\hat{y} = a = \sigma(z)$ where $z = w^T X + b$
- $\mathcal{L}(a, y) = -(y \log a + (1 - y) \log(1 - a))$



Forward &
backward
Propagation

Learning Rate

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

$$\begin{aligned} dz &= \frac{d\mathcal{L}(a, y)}{dz} = \frac{d\mathcal{L}(a, y)}{da} \frac{da}{dz} \\ &= \left(-\frac{y}{a} + \frac{1-y}{1-a} \right) (a(1-a)) \\ &= a - y \end{aligned}$$

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

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

$$\begin{aligned} w_1 &:= w_1 - \alpha dw_1 \\ w_2 &:= w_2 - \alpha dw_2 \\ b &:= b - \alpha db \end{aligned}$$

Logistic Regression on m examples

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

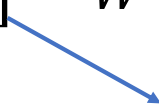
- $dw_1^{(i)}, dw_1^{(i)}, db^{(i)},$

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

Logistic Regression on m examples

- $J = 0; dw_1 = 0; dw_2 = 0; db = 0$
- *for* $i = 1$ *to* m
 - $z^{(i)} = w^T x^{(i)} + b$
 - $a^{(i)} = \sigma(z^{(i)})$
 - $J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$
 - $dz^{(i)} = a^{(i)} - y^{(i)}$
 - $dw_1 += x_1^{(i)} dz^{(i)}$
 - $dw_2 += x_2^{(i)} dz^{(i)}$
 - $db += dz^{(i)}$
- $J = J/m; dw_1 = dw_1/m; dw_2 = dw_2/m; db = db/m$
- $w_1 := w_1 - \alpha dw_1; w_2 := w_2 - \alpha dw_2; b := b - \alpha db$

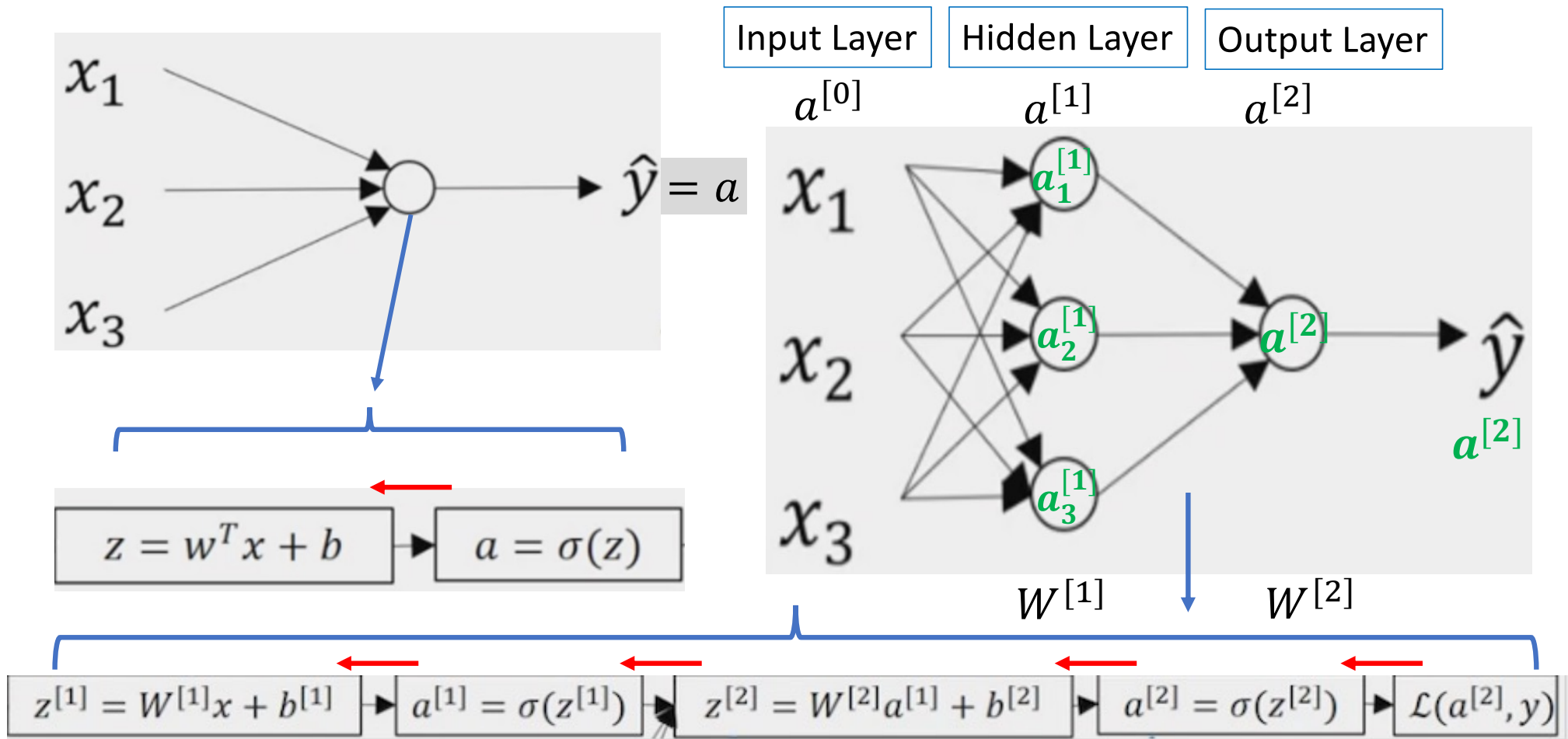
Vectorization

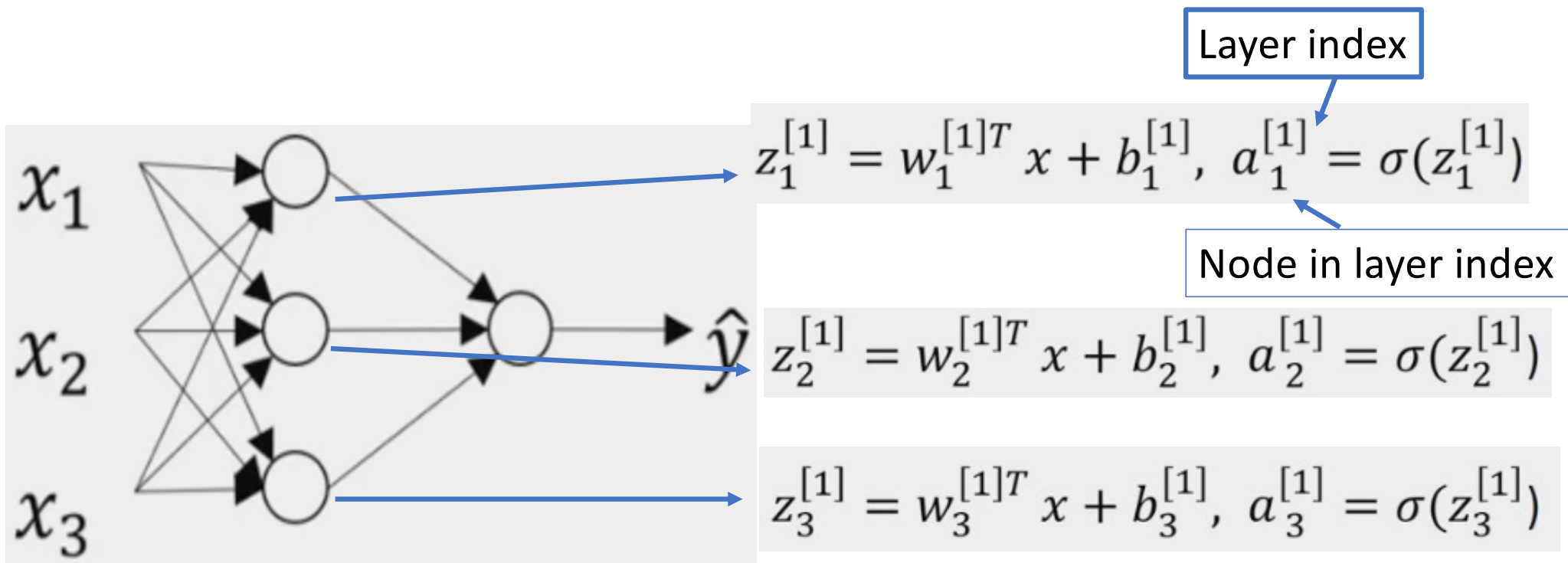
- $z = np.dot(w, x) + b$ which is equivalent to $z = w^T X + b$
- Avoid explicit for-loops can speed up the program
- $X = \begin{bmatrix} | & & | \\ x^{(1)} & \dots & x^{(m)} \\ | & & | \end{bmatrix}, X \in \mathbb{R}^{(n_x, m)}$
- $Z = [z^{(1)}, \dots, z^{(m)}] = W^T X + [b, \dots, b] = [W^T x^{(1)} + b, \dots, W^T x^{(m)} + b]$
- $Z \in \mathbb{R}^{(1, m)}$ $b \in \mathbb{R}^{(1, 1)}$ 
- $Z = np.dot(W.T, X) + b$ **Broadcasting**
- $A = [a^{(1)}, \dots, a^{(m)}] = \sigma(Z)$

Colab example: <https://colab.research.google.com/drive/1WfhHRoseY8wxhy2hx5J3u60hU-VEI28-#scrollTo=8xbJdrbhK5Lf>

- $dZ^{(1)} = a^{(1)} - y^{(1)}, \dots, dZ^{(m)} = a^{(m)} - y^{(m)}$
- $dZ = [dz^{(1)}, \dots, dz^{(m)}]$
- $A = [a^{(1)}, \dots, a^{(m)}]$
- $Y = [y^{(1)}, \dots, y^{(m)}]$
- $dZ = A - Y = [a^{(1)} - y^{(1)}, \dots, a^{(m)} - y^{(m)}]$
- $dW = \frac{1}{m} X dZ^T = \frac{1}{m} \begin{bmatrix} | & & | \\ x^{(1)} & \dots & x^{(m)} \\ | & & | \end{bmatrix} \begin{bmatrix} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{bmatrix} = \frac{1}{m} [x^{(1)} dz^{(1)}, \dots, x^{(m)} dz^{(m)}]$
- For iter in range(1000):
 - $Z = np.dot(W.T, X) + b, A = \sigma(Z),$
 - $dZ = A - Y, dW = \frac{1}{m} X dZ^T, db = \frac{1}{m} np.sum(dZ)$
 - $W := W - \alpha dW, b := b - \alpha db$

Review what is Neural Network?



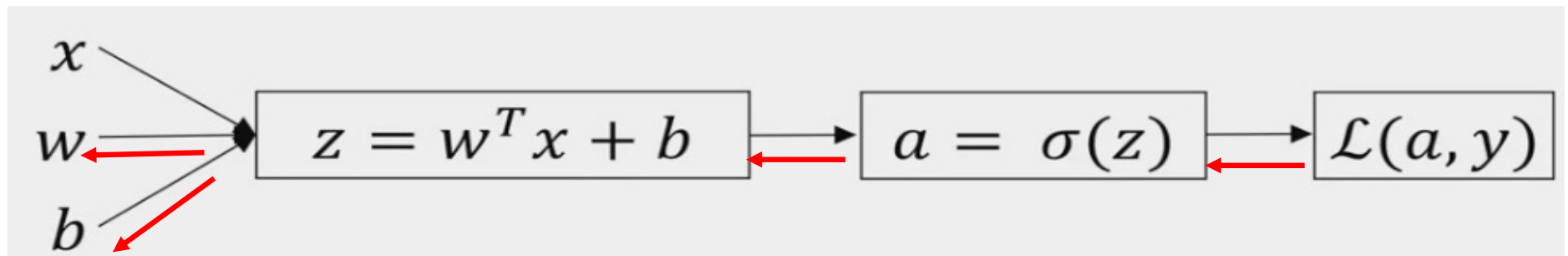


Layer l , sample m , node i

$$a_{i}^{l(m)}$$

Logistic Regression

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



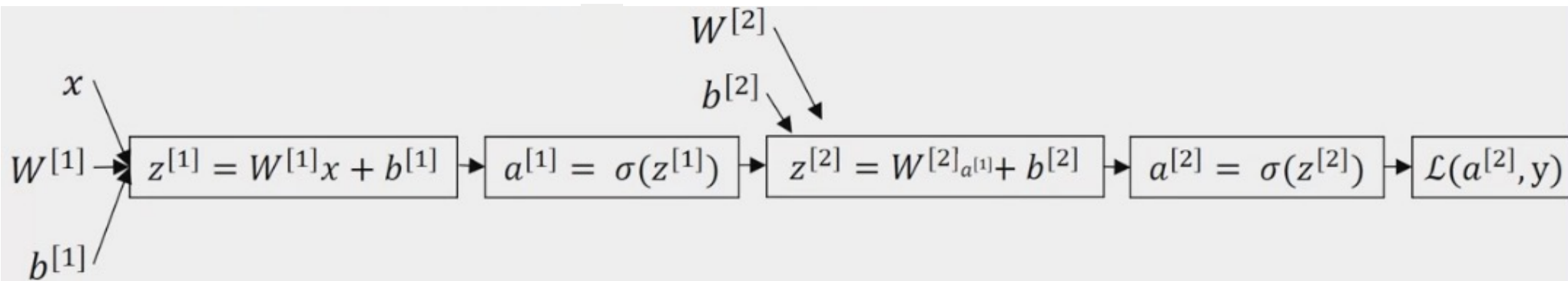
$$dw = dz * x$$

$$db = dz$$

$$\begin{aligned} dz &= \frac{d\mathcal{L}(a, y)}{dz} = \frac{d\mathcal{L}(a, y)}{da} \frac{da}{dz} \\ &= \left(-\frac{y}{a} + \frac{1-y}{1-a} \right) (a(1-a)) \\ &= a - y \end{aligned}$$

$$\begin{aligned} "da" &= \frac{d\mathcal{L}(a, y)}{da} \\ &= -\frac{y}{a} + \frac{1-y}{1-a} \end{aligned}$$

Neural Network gradients



$$dz^{[1]} = w^{[2]T} dz^{[2]} * g^{[1]'}(z^{[1]})$$

$$(n^{[1]}, 1) = n^{[1]}, n^{[2]}(n^{[2]}, n^{[1]}) * (n^{[1]}, 1)$$

$$w^{[2]}(n^{[2]}, n^{[1]})$$

$$z^{[2]}(n^{[2]}, 1)$$

$$dz^{[2]}(1, 1)$$

$$dz^{[2]} = a^{[2]} - y$$

$$dw^{[2]} = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

$$\begin{aligned} "da" &= \frac{d\mathcal{L}(a, y)}{da} \\ &= -\frac{y}{a} + \frac{1-y}{1-a} \end{aligned}$$

Forward and Backward propagation for layer l

- Forward propagation:
- Input $A^{[l-1]}$; output: $A^{[l]}, \text{cache}(Z^{[l]})$
- $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$ $A^{[l]} = g^{[l]}(Z^{[l]})$
- Backward propagation:
- Input: $dA^{[l]}$; output: $dZ^{[l]}, dW^{[l]}, db^{[l]}$
- $dZ^{[l]} = dA^{[l]} * g^{[l]}(Z^{[l]})$
- $dW^{[l]} = \frac{1}{m} dZ^{[l]} A^{[l-1]T}$
- $db^{[l]} = \frac{1}{m} \text{np.sum}(dZ^{[l]}, \text{axis} = 1, \text{keepdims} = \text{True})$
- $dA^{[l-1]} = W^{[l]T} dZ^{[l]}$

Logistic Regression with a Neural Network mindset

- `W2_BuildUp_NN_mindset.ipynb`

Readings for Feedforward Neural Network

- <https://jonaslalin.com/2021/12/10/feedforward-neural-networks-part-1/>
- <https://jonaslalin.com/2021/12/21/feedforward-neural-networks-part-2/>
- <https://jonaslalin.com/2021/12/22/feedforward-neural-networks-part-3/>