université de BORDEAUX

Deep Neural Network

LE Van Linh

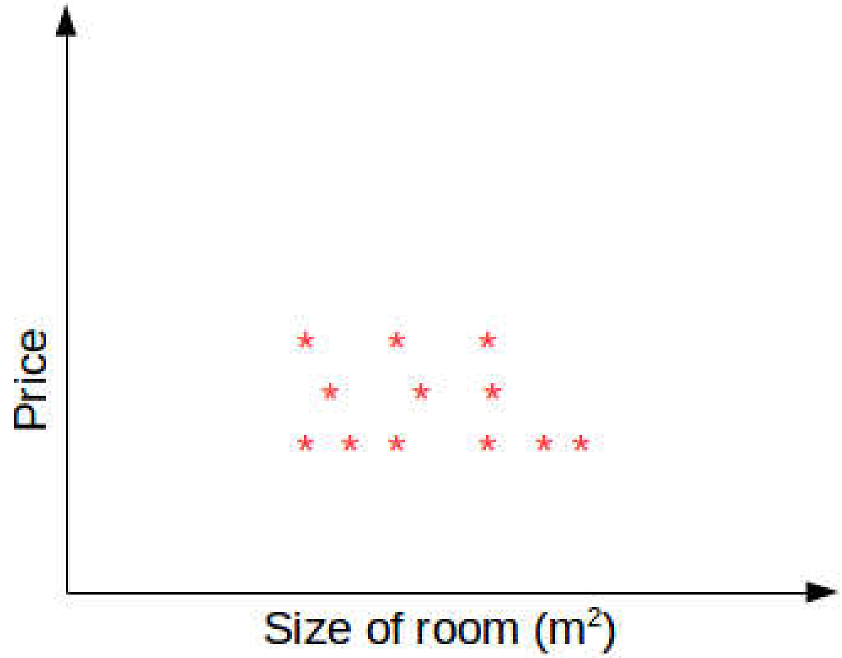
van-linh.le@u-bordeaux.fr

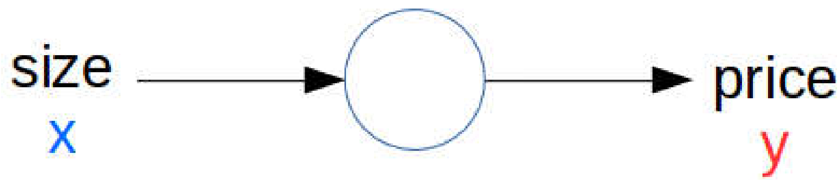
http://www.labri.fr/perso/vle

Contents

What is a neural network?

Example

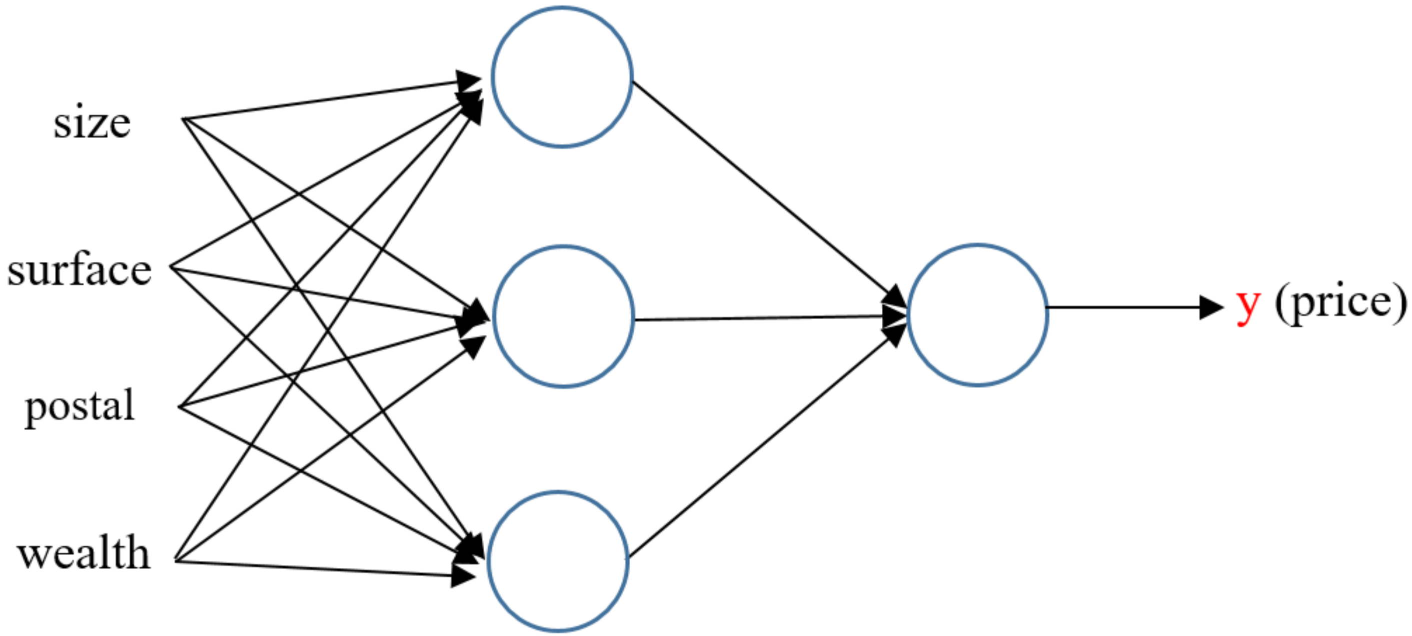




van-linh.le@u-bordeaux.fr

What is a neural network?

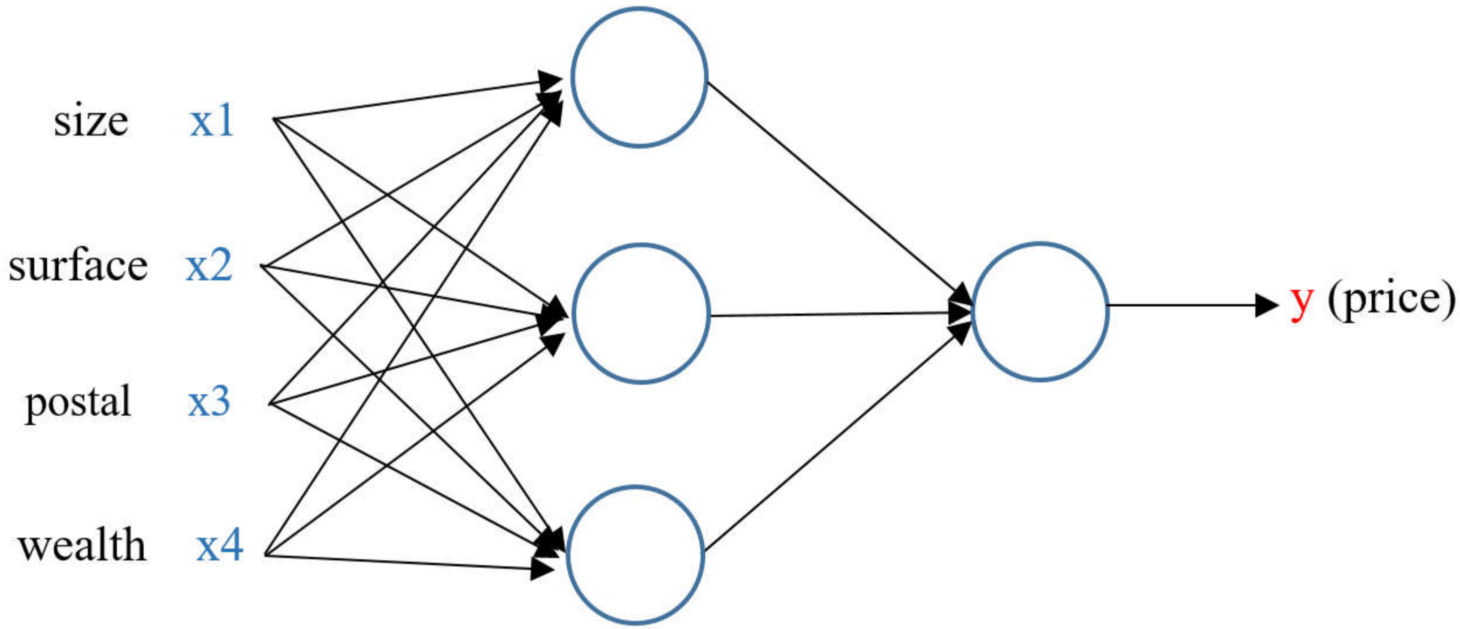
Example



van-linh.le@u-bordeaux.fr

What is a neural network?

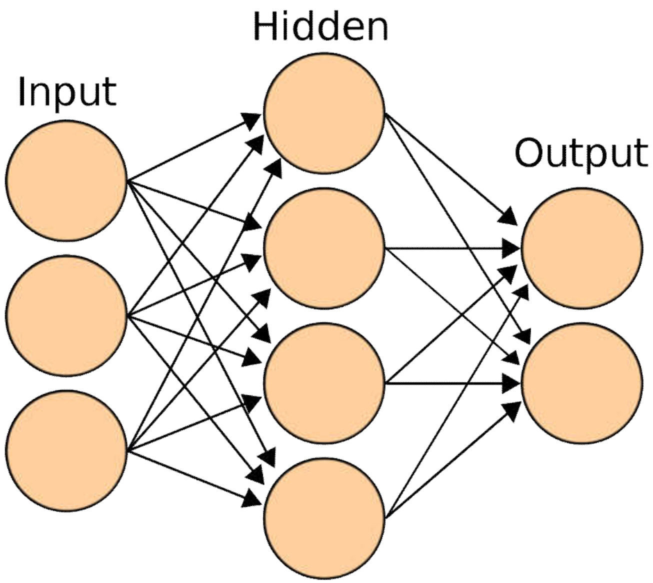
Example



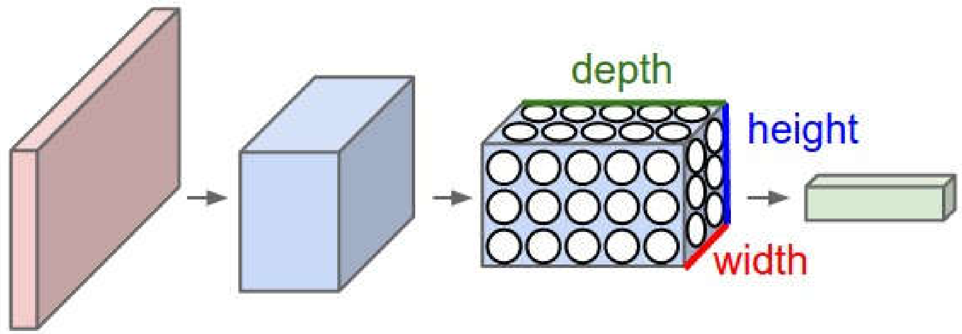
Given enough training data (***X***,***y***), neural network remarkable figuring out **a function** to map from **X** to **y**

What is a neural network?

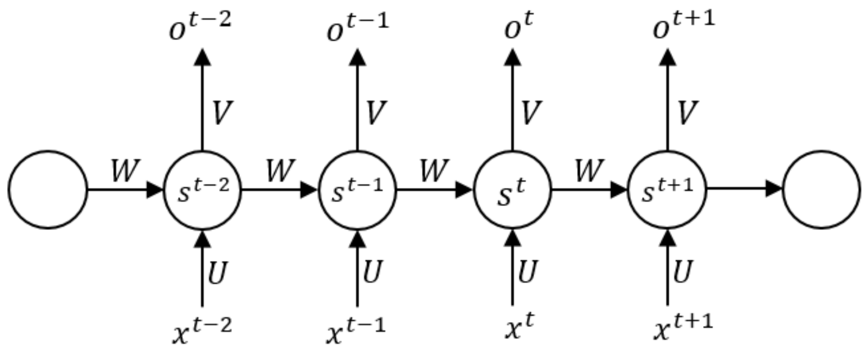
Neural Network examples



Standard NN



Convolutional NN



Recurrent NN

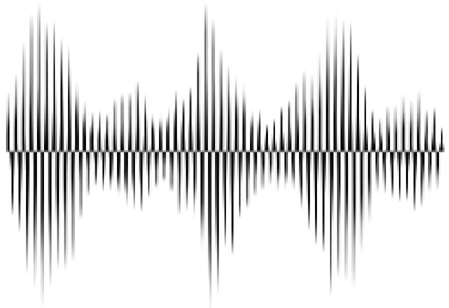
van-linh.le@u-bordeaux.fr

What is a neural network?

Structured data

Size(m^2)	Acreage	....	Price (x 100 euros)
10	1		3
14	1		3.5
....	...		...
50	2		7
100	3		12

Unstructured data



Bonjour,
Comment ca va?

What is a neural network?

Application domains

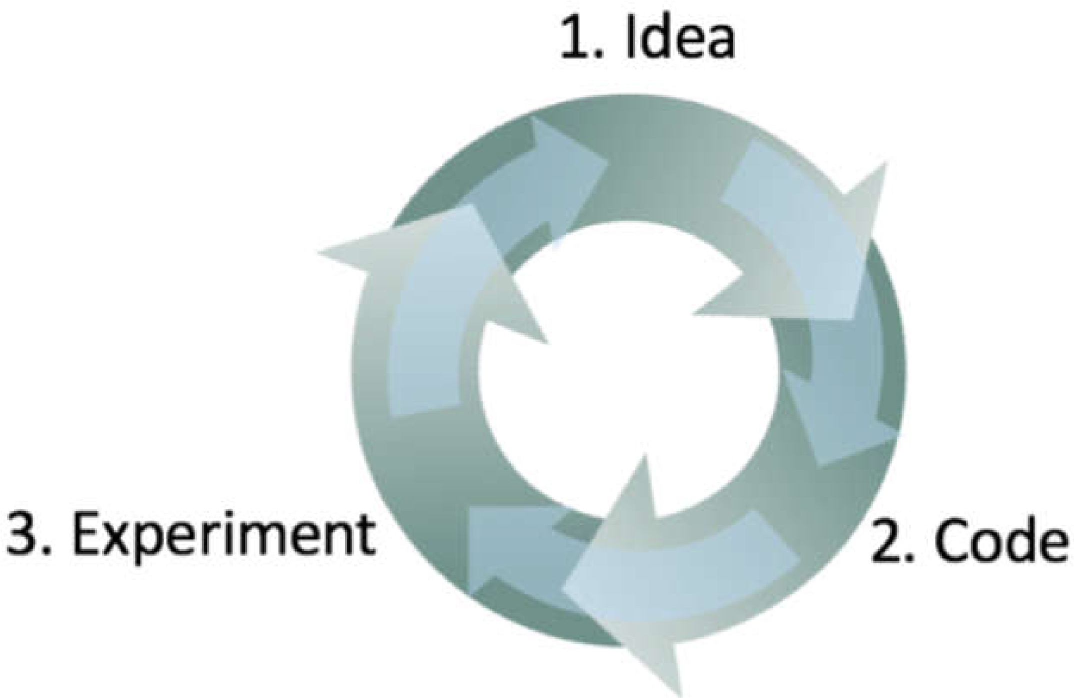
Input (X)	Output (y)	Application
Room features	Prices	Real estate
Advertisement, user info	Click on ad? (0,1)	Online advertising
Image	Object	Photo tagging
Audio	Text transcript	Speech recognition
French	Vietnamese	Machine translation
Image, radar information	Position of other cars	Automatically driving

van-linh.le@u-bordeaux.fr

Driving of Deep Learning

Three components

1. Data
2. Computation
3. Algorithms

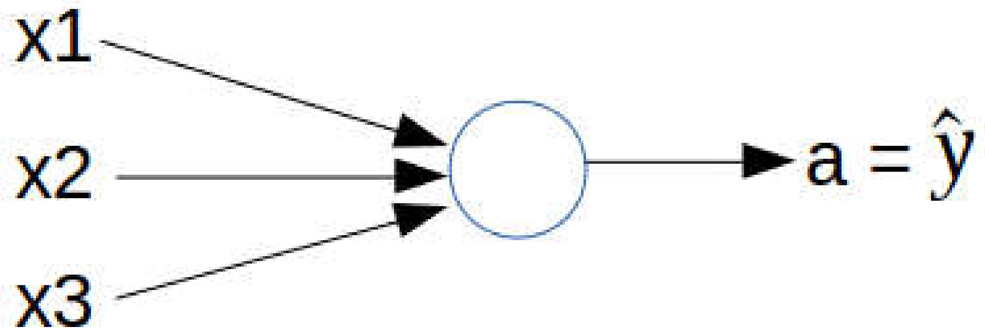


van-linh.le@u-bordeaux.fr

Contents

van-linh.le@u-bordeaux.fr

Logistic Regression

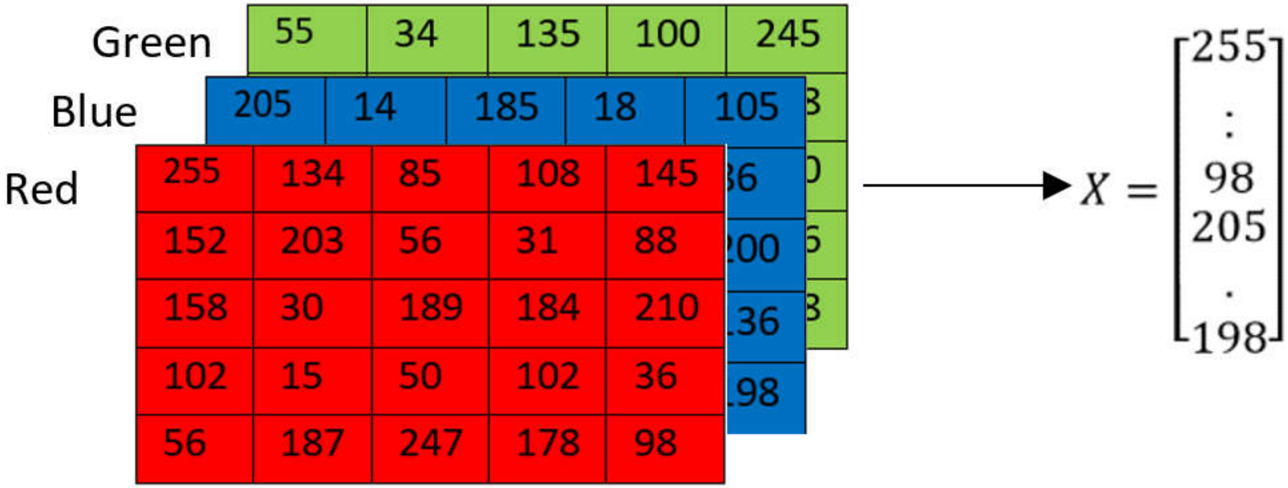


One node network

Logistic Regression

Binary classification problem

CAT (1) or NOT CAT (0)



van-linh.le@u-bordeaux.fr

Logistic Regression

Notations:

- A training data (x, y): $x \in R^{n_x}$, $y \in \{0,1\}$
- With ***m*** training examples: $(x^1, y^1), (x^2, y^2), \ldots, (x^m, y^m)$

$$X = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ x^1 & x^2 & \vdots & \vdots & x^m \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$X \in R^{n_x \times m}$$

$$X.shape = (n_x, m)$$

$$Y = [y^1, y^2, \ldots, y^m]$$

$$Y \in R^{1 \times m}$$

$$Y.shape = (1, m)$$

van-linh.le@u-bordeaux.fr

Logistic Regression

Given $\boldsymbol{x}$, want:

$\hat{\mathbf{y}} = \mathbf{P}(\mathbf{y} = \mathbf{1}|\mathbf{x})$ where $(x \in R^{n_x})$

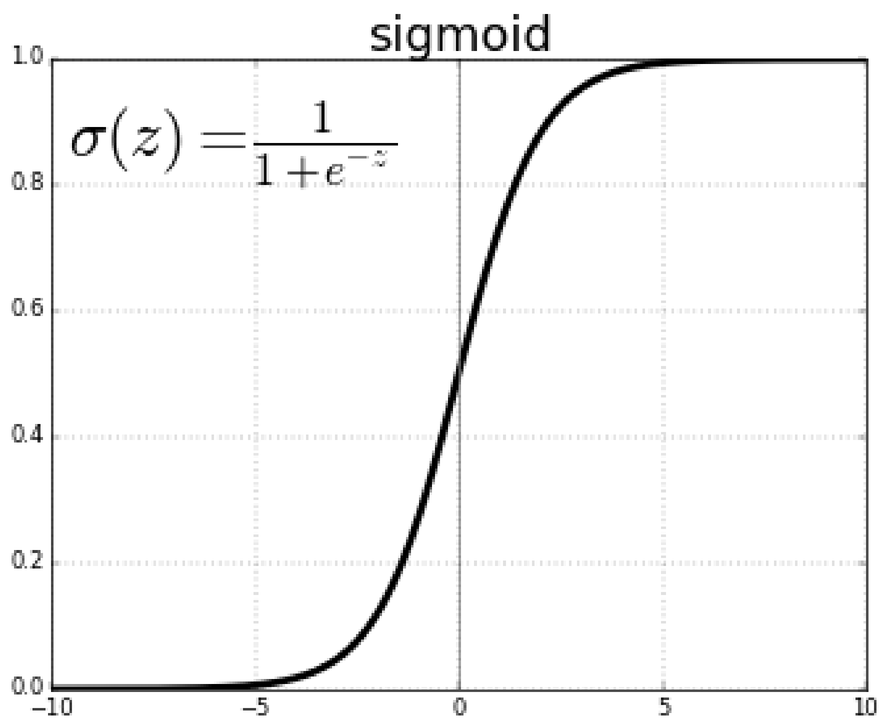
Parameters:

$$\boldsymbol{w} \in \boldsymbol{R}^{n_x}, \boldsymbol{b} \in \boldsymbol{R}$$

Output:

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

where $\boldsymbol{\sigma}$ is an activation function.



van-linh.le@u-bordeaux.fr

Logistic Regression

Loss (error) function:

- L1 loss: $\mathcal{L}(\hat{y}, y) = |\hat{y} - y|$

- L2 loss: $\mathcal{L}(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$

- Cross-entropy loss:

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

van-linh.le@u-bordeaux.fr

Logistic Regression

Cost function: $\{(x^1, y^1), (x^2, y^2), \dots, (x^m, y^m)\}$, want: $\hat{y}^{(i)} \approx y^{(i)}$

- L1 cost: $\mathbf{J(w, b)} = \frac{1}{m}\sum_1^m \mathcal{L}(\hat{y}, y) = \frac{1}{m}\sum_1^m |\hat{y} - y|$

- L2 cost: $\mathbf{J(w, b)} = \frac{1}{m}\sum_1^m \mathcal{L}(\hat{y}, y) = \frac{1}{m}\sum_1^m (\hat{y} - y)^2$

- Cross-entropy cost:

$$\mathbf{J(w, b)} = \frac{1}{m}\sum_1^m \mathcal{L}(\hat{y}, y) = -\frac{1}{m}\sum_1^m \left(y^{(i)}.\log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right).\log\left(1 - \hat{y}^{(i)}\right)\right)$$

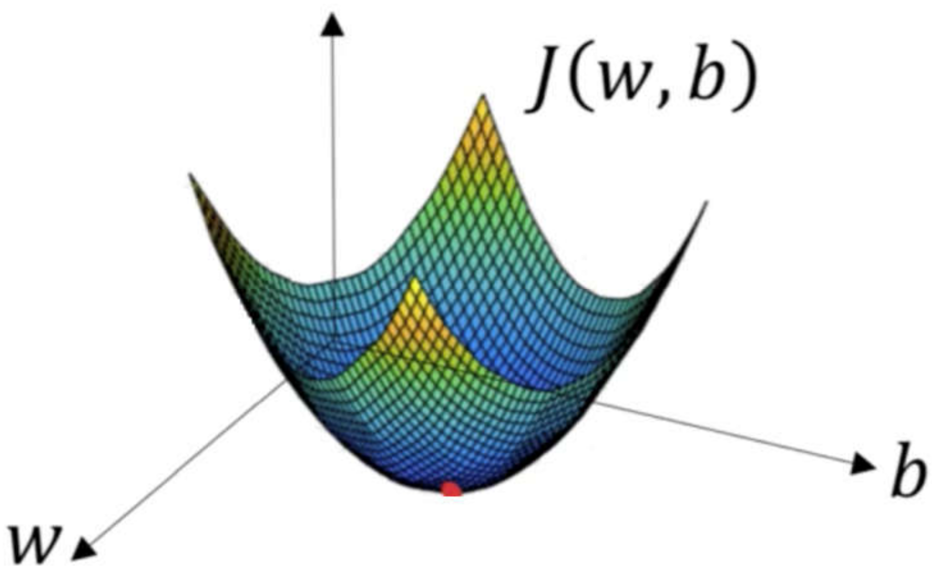
van-linh.le@u-bordeaux.fr

Contents

van-linh.le@u-bordeaux.fr

Derivatives and Gradient Descent

Objective: To find ***w, b*** that minimize ***J(w,b)***



van-linh.le@u-bordeaux.fr

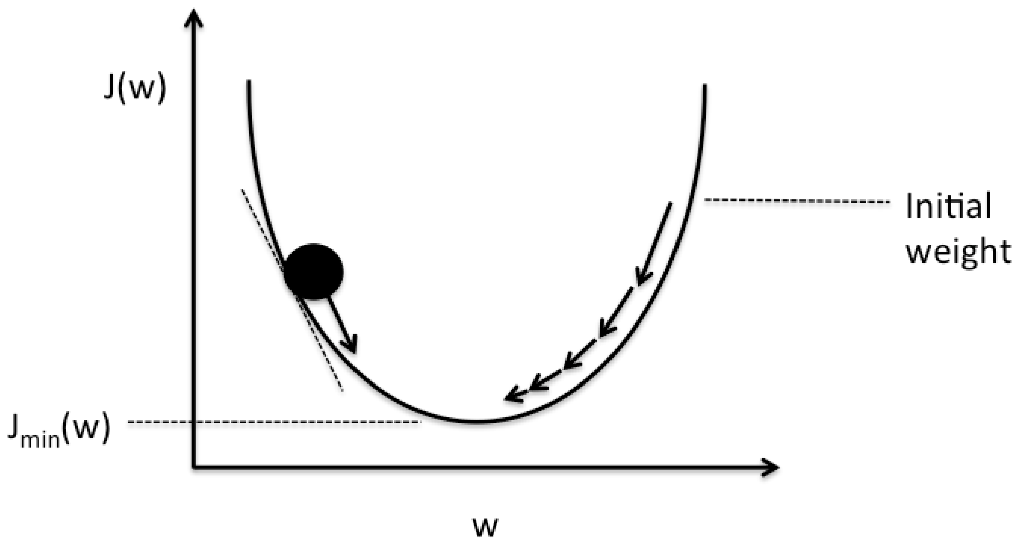
Derivatives and Gradient Descent

Repeat to update:

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

$$b := b - \alpha \frac{dJ}{db}$$

Where: α is learning rate



Schematic of gradient descent.

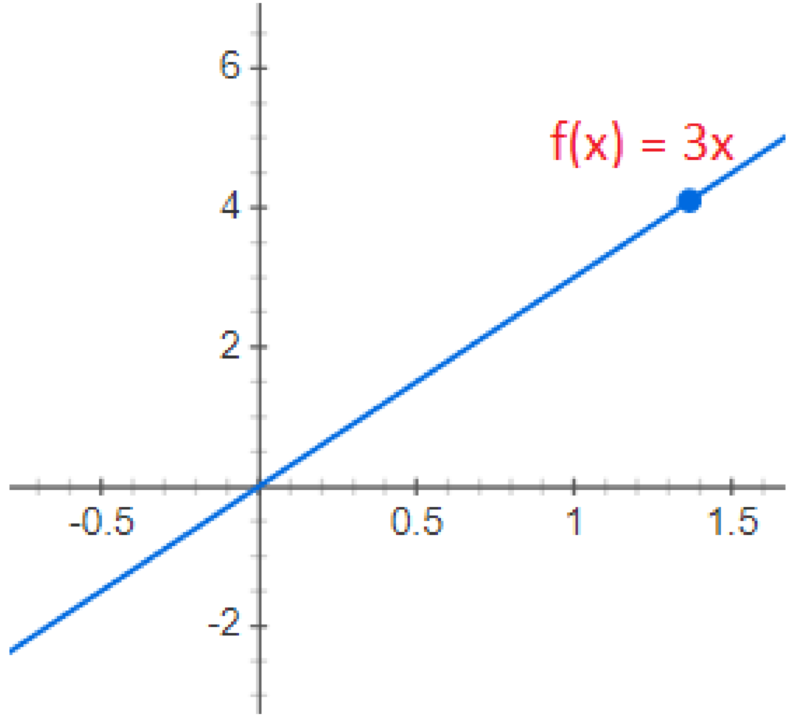
Derivatives and Gradient Descent

$x = 1 \;\rightarrow f(x) = 3$
$x = 1.001 \;\rightarrow f(x) = 3.003$

Slope of $f(x)$ at $x = 1$:

$$\boldsymbol{\frac{df(x)}{dx} = \frac{0{,}003}{0{,}001} = 3}$$

Which is slope of $f(x)$ at 5?



van-linh.le@u-bordeaux.fr

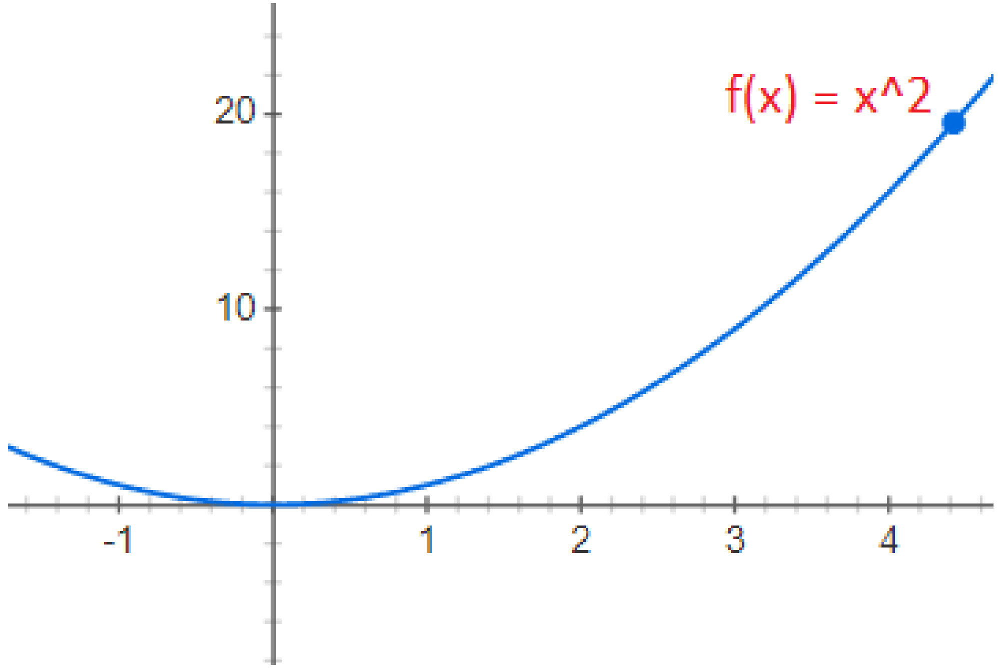
Derivatives and Gradient Descent

$x = 3 \ \rightarrow f(x) = 9$
$x = 3.001 \ \rightarrow f(x) = 9.006$

Slope of $f(x)$ at $x = 1$:

$$\frac{df(x)}{dx} = \frac{0{,}006}{0{,}001} = 6$$

Which is slope of $f(x)$ at 5?



With $\boldsymbol{f(x) = x^3 \rightarrow \frac{df(x)}{dx} = 3x^2}$

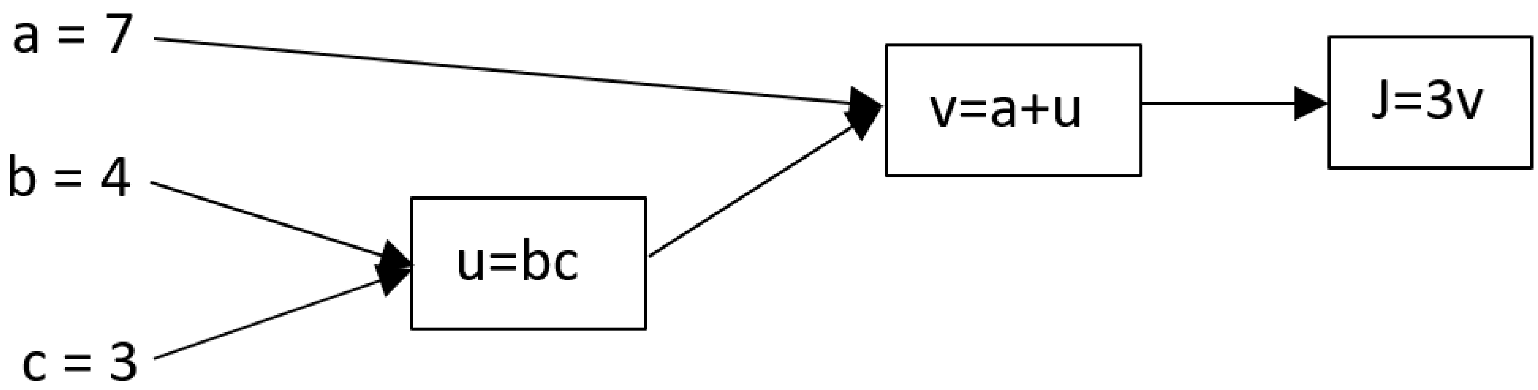
With $\boldsymbol{f(x) = \log(x) \rightarrow \frac{df(x)}{dx} = \frac{1}{x}}$

van-linh.le@u-bordeaux.fr

Derivatives with Computation Graph

Consider a function: $\boldsymbol{J(a, b, c) = 3(a + bc)}$

Assign: $u = bc, v = (a + bc) = (a + u) \Rightarrow J = 3v$

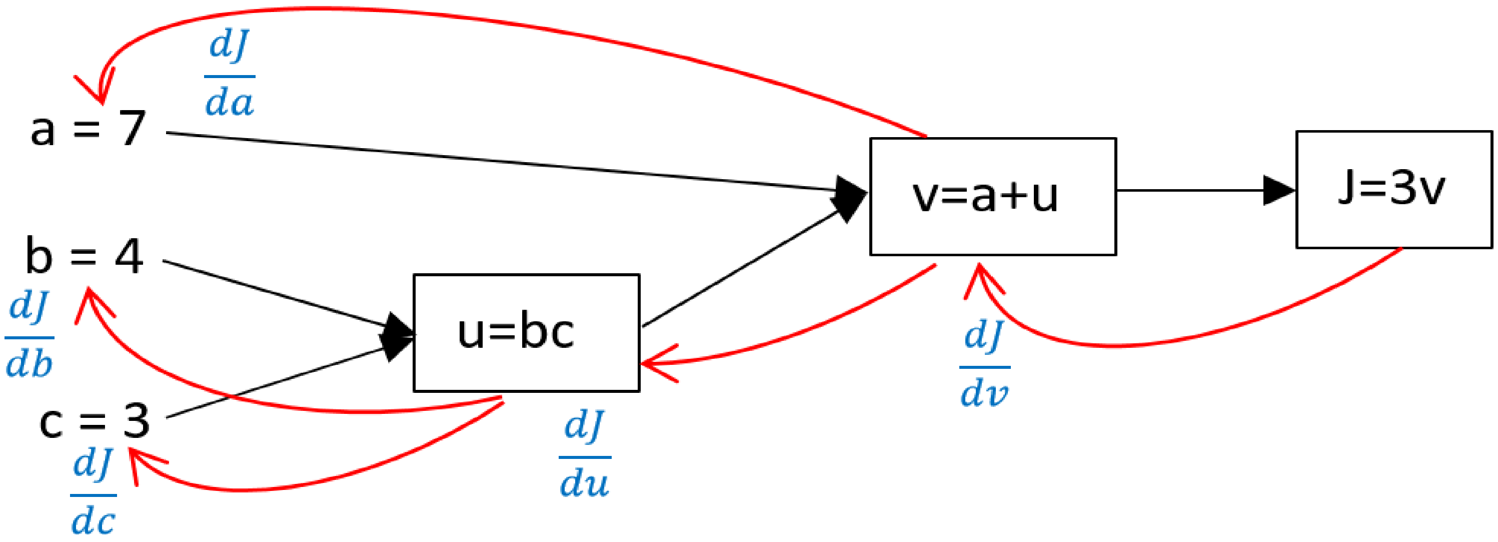


van-linh.le@u-bordeaux.fr

Derivatives with Computation Graph

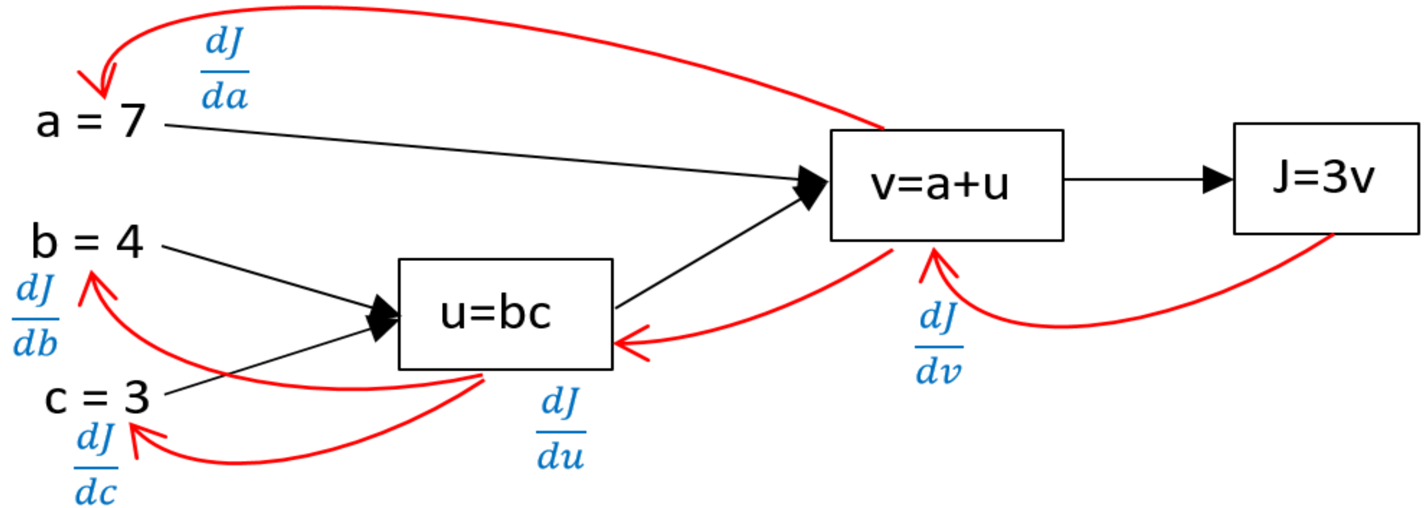
Consider a function: $\boldsymbol{J(a,b,c) = 3(a+bc)}$

Assign: $u = bc, v = (a+bc) = (a+u) \Rightarrow J = 3v$



van-linh.le@u-bordeaux.fr

Derivatives with Computation Graph



$J = 3v$: $v = 19 \;\rightarrow 19{,}001$

$$J = 57 \rightarrow 57{,}003 \Rightarrow \frac{dJ}{dv} = \frac{0{,}003}{0{,}001} = 3$$

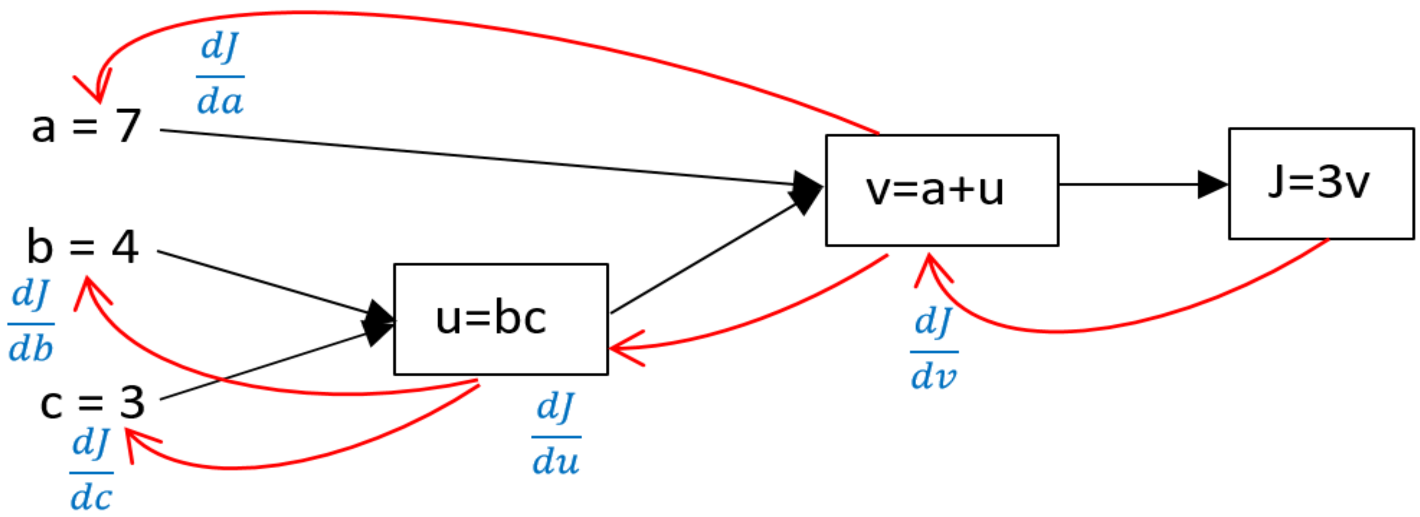
$v = a + u$: $a = 7 \;\rightarrow 7{,}001$

$$v = 19 \rightarrow 19{,}001$$

$$J = 57 \rightarrow 57{,}003 \Rightarrow \frac{dJ}{da} = \frac{0{,}003}{0{,}001} = 3, \frac{dv}{da} = \frac{0{,}001}{0{,}001} = 1$$

van-linh.le@u-bordeaux.fr

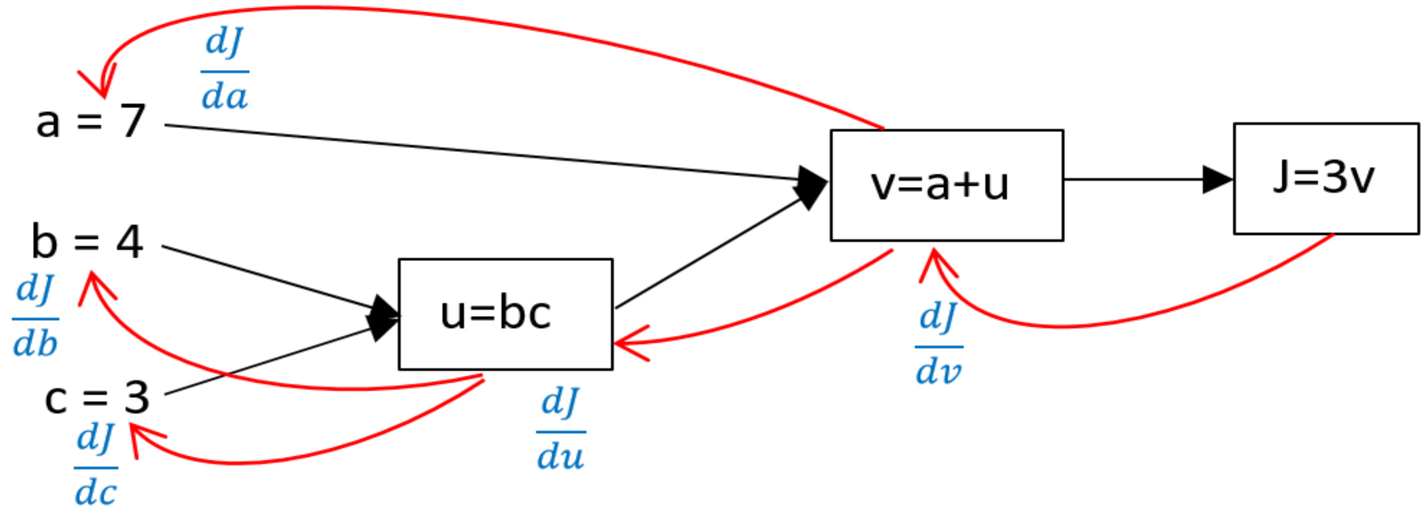
Derivatives with Computation Graph



$$v = a + u: u = 12 \ \rightarrow 12{,}001$$
$$v = 19 \rightarrow 19{,}001$$
$$J = 57 \rightarrow 57{,}003 \Rightarrow \frac{dJ}{du} = \frac{0{,}003}{0{,}001} = 3, \frac{dv}{du} = \frac{0{,}001}{0{,}001} = 1$$
$$u = bc: b = 4 \ \rightarrow 4{,}001$$
$$u = 12 \rightarrow 12{,}003 \Rightarrow \frac{du}{db} = \frac{0{,}003}{0{,}001} = 3 \Rightarrow \frac{dJ}{db} = \frac{dJ}{du}\frac{du}{db} = 3 \times 3 = 9$$

van-linh.le@u-bordeaux.fr

Derivatives with Computation Graph



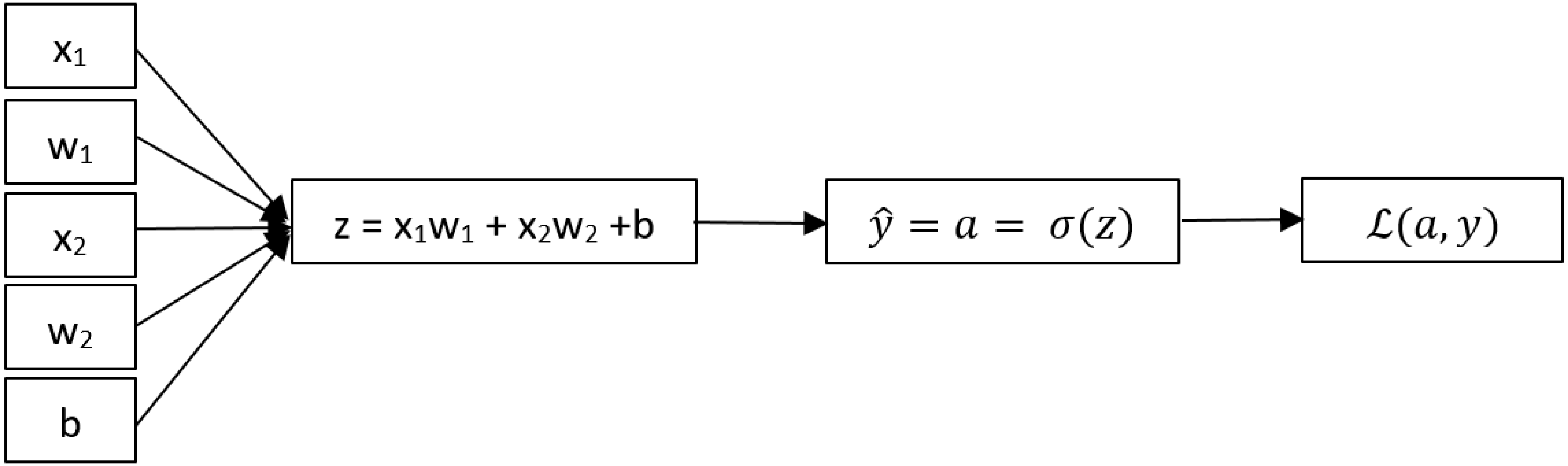
$u = bc$: $c = 3 \rightarrow 3{,}001$

$$u = 12 \rightarrow 12{,}004 \Rightarrow \frac{du}{db} = \frac{0{,}004}{0{,}001} = 4 \Rightarrow \frac{dJ}{dc} = \frac{dJ}{du}\frac{du}{dc} = 3 \times 4 = 12$$

Derivatives in Logistic Regression

$$z = w^T x + b \text{ and } a = \hat{y} = \sigma(z)$$

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



van-linh.le@u-bordeaux.fr

Derivatives in Logistic Regression

- $\frac{d\mathcal{L}}{da} = (\frac{-y}{a} + \frac{1-y}{1-a})$
- $\frac{d\mathcal{L}}{dz} = (a - y), \frac{da}{dz} = a(1-a)$
- $dw_1 = \frac{d\mathcal{L}}{dw_1} = x_1 \frac{d\mathcal{L}}{dz}$
- $dw_2 = \frac{d\mathcal{L}}{dw_2} = x_2 \frac{d\mathcal{L}}{dz}$
- $db = \frac{d\mathcal{L}}{db} = dz$

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

van-linh.le@u-bordeaux.fr

Gradient descent on **m** training data

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

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

- $\frac{dJ(w,b)}{dw_1} = \frac{1}{m}\sum_{i=1}^{m} \frac{d\mathcal{L}(a^{(i)},y^{(i)})}{dw_1}$
- $\frac{dJ(w,b)}{dw_2} = \frac{1}{m}\sum_{i=1}^{m} \frac{d\mathcal{L}(a^{(i)},y^{(i)})}{dw_2}$
- $\frac{dJ(w,b)}{db} = \frac{1}{m}\sum_{i=1}^{m} \frac{d\mathcal{L}(a^{(i)},y^{(i)})}{db}$

van-linh.le@u-bordeaux.fr

Gradient descent on **m** training data

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

For $i = 1$ to m:

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

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

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

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

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

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

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

$$J = \frac{J}{m}, dw_1 = \frac{dw_1}{m}, dw_2 = \frac{dw_2}{m}, db = \frac{db}{m}$$

$$\boldsymbol{w_1 := w_1 - \alpha.dw_1\ ;\ \ w_2 := w_2 - \alpha.dw_2;\ b := b - \alpha.db}$$

van-linh.le@u-bordeaux.fr

Contents

van-linh.le@u-bordeaux.fr

Vectorization

Consider the computing: $z = w^T x + b$

Where w and x are large vectors features. $(w, x \in \mathcal{R}^{n_x})$

Non-vectorize

$z = 0$

for i in $\mathrm{range}(n_x)$:

$\quad z += w[i] * x[i]$

$z += b$

Vectorize

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

- **Avoid the loops**
- **Make program more faster**

van-linh.le@u-bordeaux.fr

Vectorization

Consider the computing: $u = A.v$, where A is a matrix and v is a vector and $u_i = \sum_j A_{ij} v_j$

$u = np.zeros((n, 1))$

for i in $\mathrm{rows(A)}$:

for j in $\mathrm{cols(A)}$:

$u[i] += A[i][j] \; * v[j]$

The for loops are equal to $\mathrm{u} = \mathrm{np.dost}(A, v)$

van-linh.le@u-bordeaux.fr

Vectorization in logistic regression

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

For $i = 1$ to m:

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

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

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

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

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

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

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

$$J = \frac{J}{m}, dw_1 = \frac{dw_1}{m}, dw_2 = \frac{dw_2}{m}, db = \frac{db}{m}$$

How to change the derivatives into the form of vectorization?

van-linh.le@u-bordeaux.fr

Vectorization

Consider 3 training examples:

$$z^{(1)} = w^T x^{(1)} + b \qquad z^{(2)} = w^T x^{(2)} + b \qquad z^{(3)} = w^T x^{(3)} + b$$
$$a^{(1)} = \sigma\left(z^{(1)}\right) \qquad a^{(2)} = \sigma\left(z^{(2)}\right) \qquad a^{(3)} = \sigma\left(z^{(3)}\right)$$

Vectorizing:

$$\boldsymbol{X} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$\boldsymbol{Z} = [z^{(1)} \quad z^{(2)} \quad \dots \quad z^{(m)}] = w^T X + [b \quad b \quad \dots \quad b]$$
$$= [w^T x^{(1)} + b \quad w^T x^{(2)} + b \quad \dots \quad w^T x^m + b]$$

$$\boldsymbol{A} = [a^{(1)} \quad a^{(2)} \quad \dots \quad a^{(m)}] = \sigma(Z)$$

van-linh.le@u-bordeaux.fr

Vectorization in gradient descent

$$dz^{(1)} = a^{(1)} - y^{(1)} \qquad dz^{(2)} = a^{(2)} - y^{(2)} \qquad dz^{(3)} = a^{(3)} - y^{(3)}$$

$$dZ = [dz^{(1)} \quad dz^{(2)} \quad \dots \quad dz^{(m)}] \; (dZ.shape = (1 \times m))$$

$$A = [a^{(1)} \quad a^{(2)} \quad \dots \quad a^{(m)}] \quad Y = [y^{(1)} \quad y^{(2)} \quad \dots \quad y^{(m)}]$$

$$\rightarrow dZ = A - Y = [a^{(1)} - y^{(1)} \quad a^{(2)} - y^{(2)} \quad \dots \quad a^{(m)} - y^{(m)}]$$

In numpy: $\mathbf{db} = \boldsymbol{\frac{1}{m} np.sum(dZ)}$, $\mathbf{dw} = \boldsymbol{\frac{1}{m} X dZ^T}$

van-linh.le@u-bordeaux.fr

Vectorization in logistic regression

The for loop in slide 31 is equal to:

$$\boldsymbol{Z = w^T X + b = np.dot(w.T, X) + b}$$

$$\boldsymbol{A = \sigma(Z)}$$

$$\boldsymbol{dZ = A - Y}$$

$$\boldsymbol{dw = \frac{1}{m} X.dZ^T}$$

$$\boldsymbol{db = \frac{1}{m} np.sum(dZ)}$$

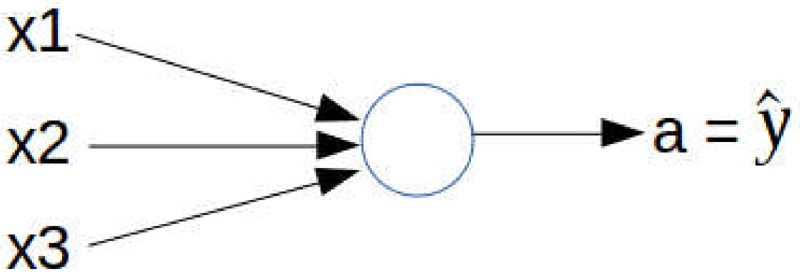
$$\boldsymbol{w \ := w - \alpha.dw}$$

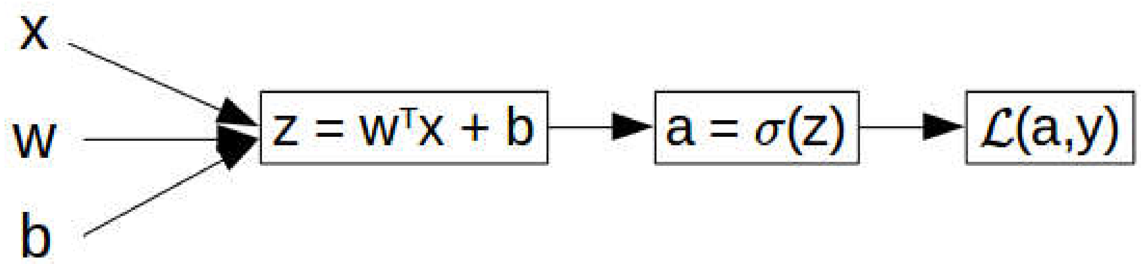
$$\boldsymbol{b = b - \alpha.db}$$

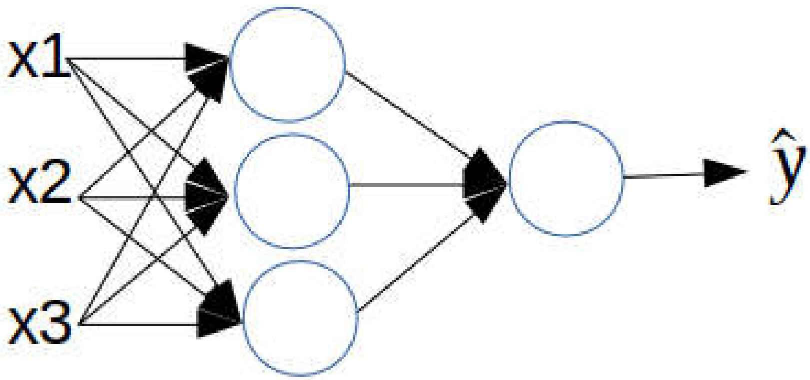
van-linh.le@u-bordeaux.fr

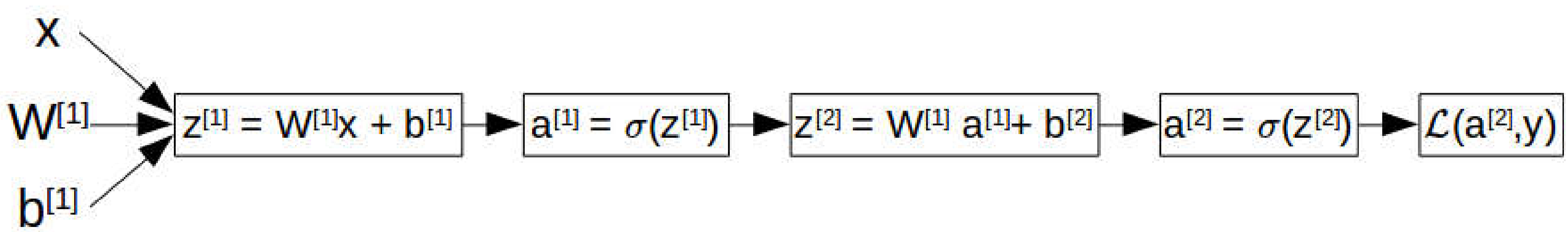
Contents

Neural Network Representation





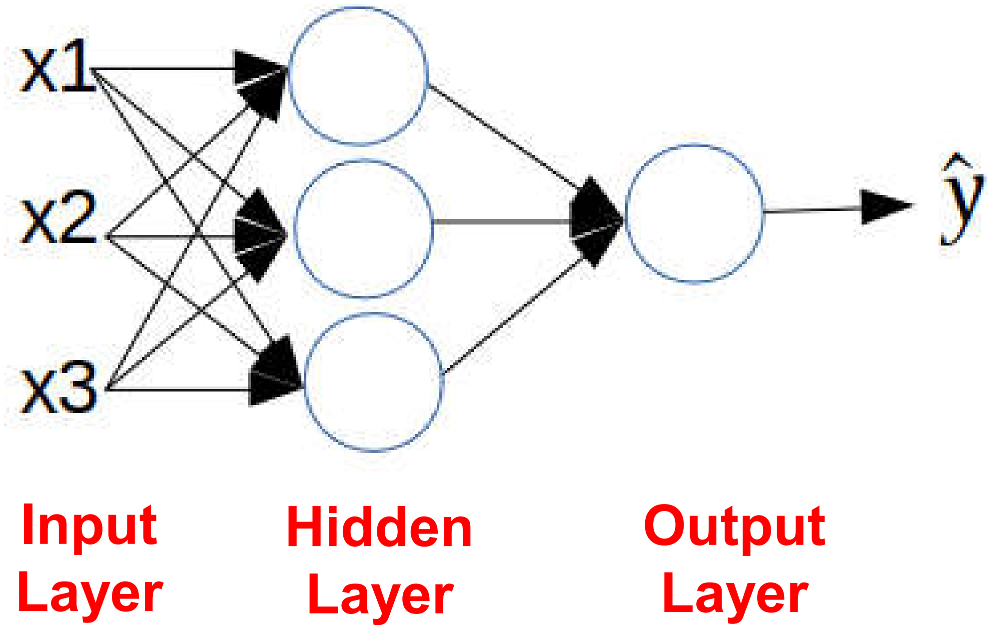




Neural Network Representation

This is a:

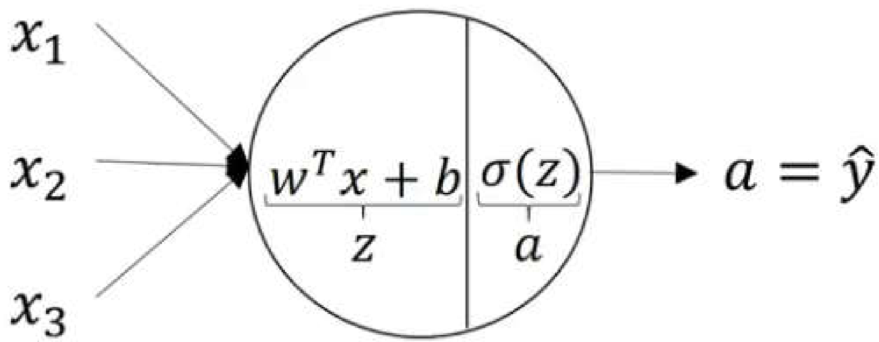
- 2 layers neural network
- The hidden layers and output layer (sometime) will have the parameters (W, b)



van-linh.le@u-bordeaux.fr

Neural Network Representation

The computing at one node of NN.

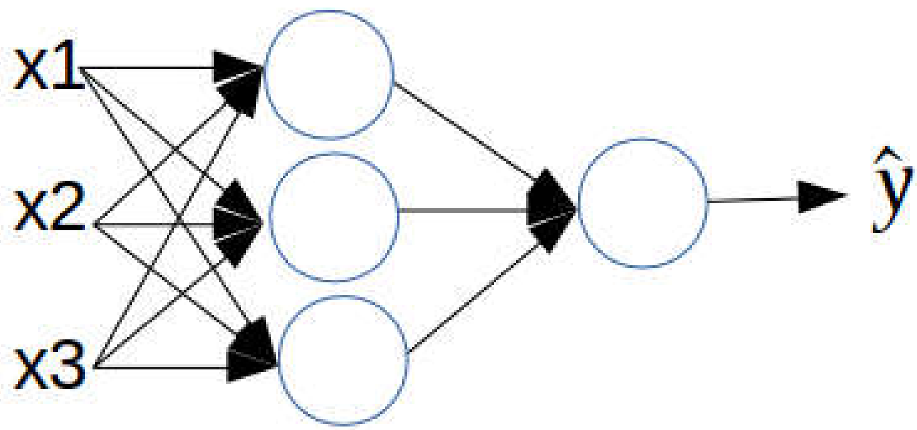


$$z = w^T x + b$$

$$a = \sigma(z)$$

Lets us consider a simple NN.

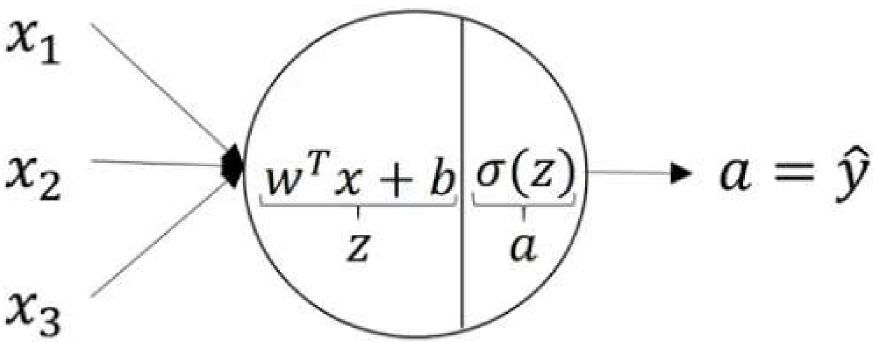
How about computing at each node?



van-linh.le@u-bordeaux.fr

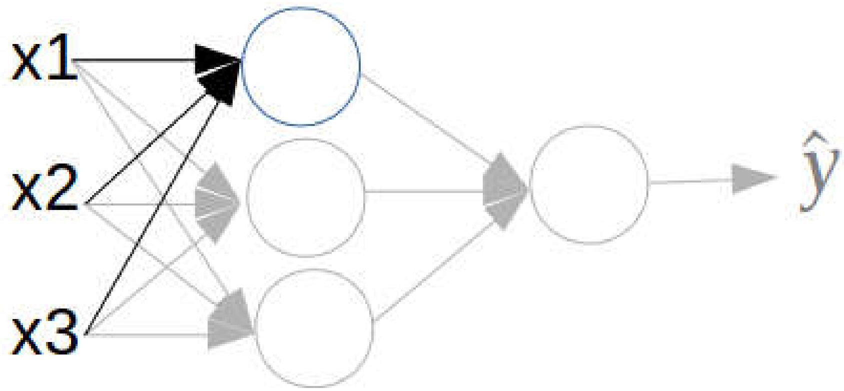
Neural Network Representation

The computing at one node of NN.



$z = w^T x + b$

$a = \sigma(z)$



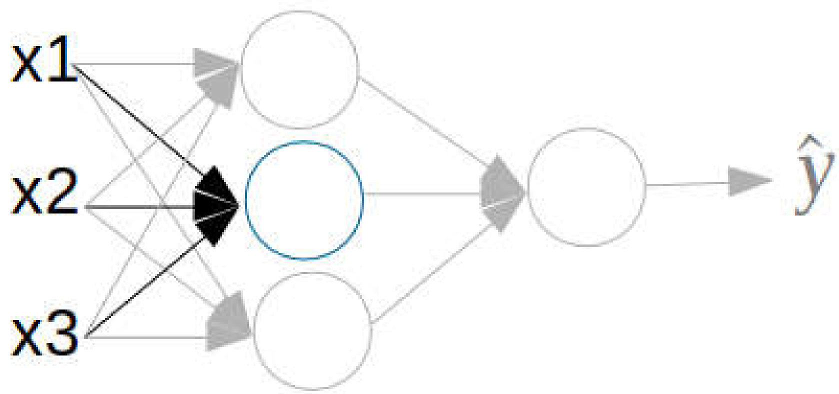
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$$

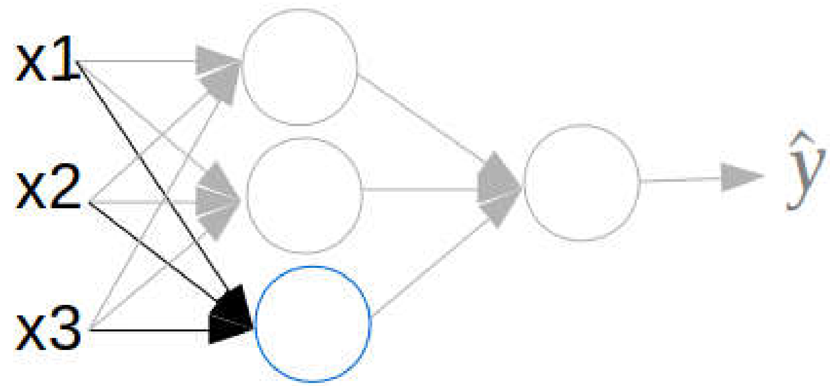
$$a_1^{[1]} = \sigma(z_1^{[1]})$$

Note: $a_i^{[l]}$ denotes node a_i at layer l

van-linh.le@u-bordeaux.fr

Neural Network Representation





$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$$

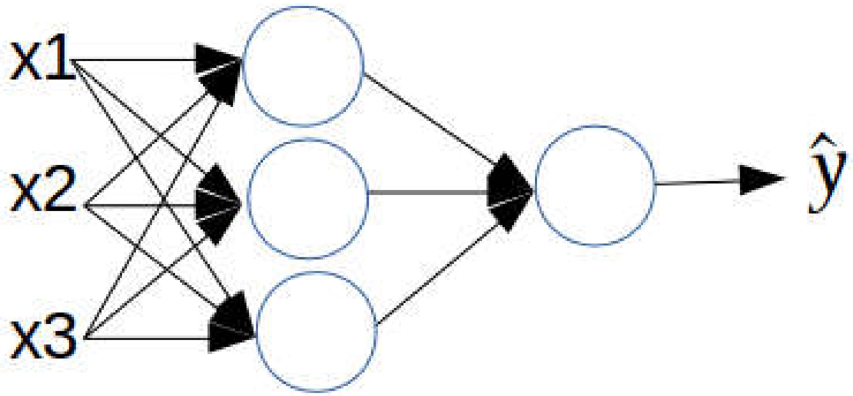
$$a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}$$

$$a_3^{[1]} = \sigma(z_3^{[1]})$$

van-linh.le@u-bordeaux.fr

Neural Network Representation



$$z_1^{[1]} = w_1^{[1]T}x + b_1^{[1]},\ a_1^{[1]} = \sigma(z_1^{[1]})$$

$$z_2^{[1]} = w_2^{[1]T}x + b_2^{[1]},\ a_2^{[1]} = \sigma(z_2^{[1]})$$

$$z_3^{[1]} = w_3^{[1]T}x + b_3^{[1]},\ a_3^{[1]} = \sigma(z_3^{[1]})$$

van-linh.le@u-bordeaux.fr

Neural Network Representation

Vectorize:

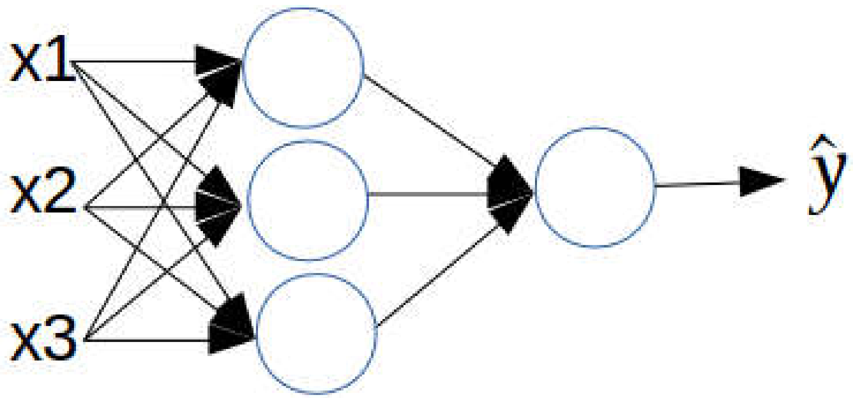
$$W^{[1]} = \begin{bmatrix} \cdots & w_1^{[1]T} & \cdots \\ \cdots & w_2^{[1]T} & \cdots \\ \cdots & w_3^{[1]T} & \cdots \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \text{ and } b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{bmatrix}$$

$$\Rightarrow z^{[1]} = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \end{bmatrix} = \begin{bmatrix} w_1^{[1]T} x + b_1^{[1]} \\ w_2^{[1]T} x + b_2^{[1]} \\ w_3^{[1]T} x + b_3^{[1]} \end{bmatrix} \Longrightarrow a^{[1]} = \sigma(z^1) = \begin{bmatrix} \sigma(z_1^{[1]}) \\ \sigma(z_2^{[1]}) \\ \sigma(z_3^{[1]}) \end{bmatrix}$$

van-linh.le@u-bordeaux.fr

Neural Network Representation

Computing for an input x:



$$z^{[1]} = W^{[1]}x + b^{[1]}$$

$$a^{[1]} = \sigma(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = \sigma(z^{[2]})$$

van-linh.le@u-bordeaux.fr

Neural Network Representation

$X \longrightarrow a^{[2]} = \hat{y}$

$X(1) \longrightarrow a^{[2](1)} = \hat{y}$

$X(2) \longrightarrow a^{[2](2)} = \hat{y}$

$\vdots \qquad\qquad \vdots$

$X(n) \longrightarrow a^{[2](m)} = \hat{y}$

Computing for ***m*** input examples:

For $i = 1$ to m:

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma(z^{[2](i)})$$

van-linh.le@u-bordeaux.fr

Neural Network Representation

Computing for ***m*** input examples: Vectorization

For $i = 1$ to m:

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$

$$a^{[2](i)} = \sigma(z^{[2](i)})$$

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$A^{[1]} = \sigma(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A + b^{[2]}$$

$$A^{[2]} = \sigma(Z^{[2]})$$

van-linh.le@u-bordeaux.fr

Neural Network Representation

Vectorization

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

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

$$Z^{[1]} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ z^{[1](1)} & z^{[1](2)} & \dots & z^{[1](m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

$$A^{[1]} = \sigma(Z^{[1]})$$

$$Z^{[2]} = W^{[2]}A + b^{[2]}$$

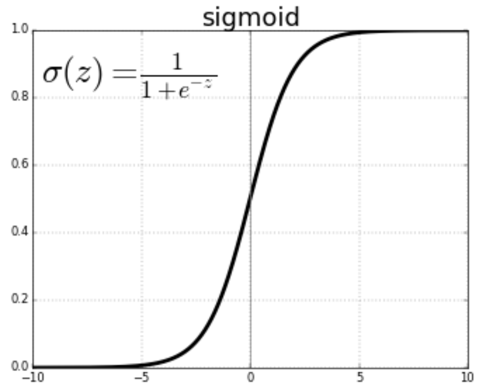
$$A^{[1]} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ a^{[1](1)} & a^{[1](2)} & \dots & a^{[1](m)} \\ \vdots & \vdots & & \vdots \end{bmatrix}$$

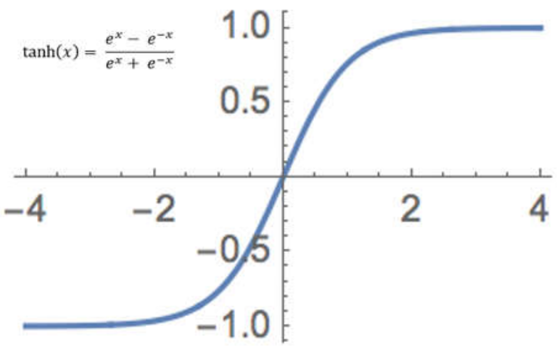
$$A^{[2]} = \sigma(Z^{[2]})$$

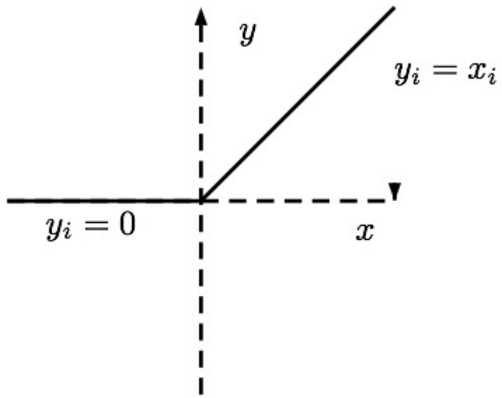
van-linh.le@u-bordeaux.fr

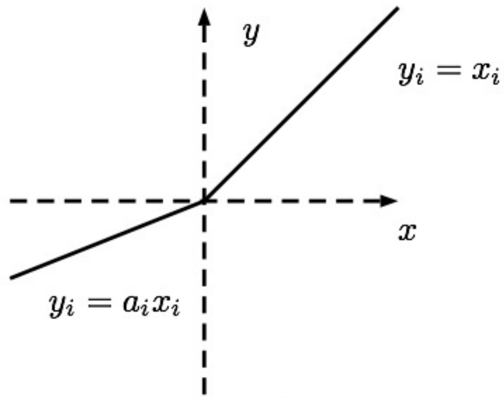
Contents

Activation functions and their derivatives





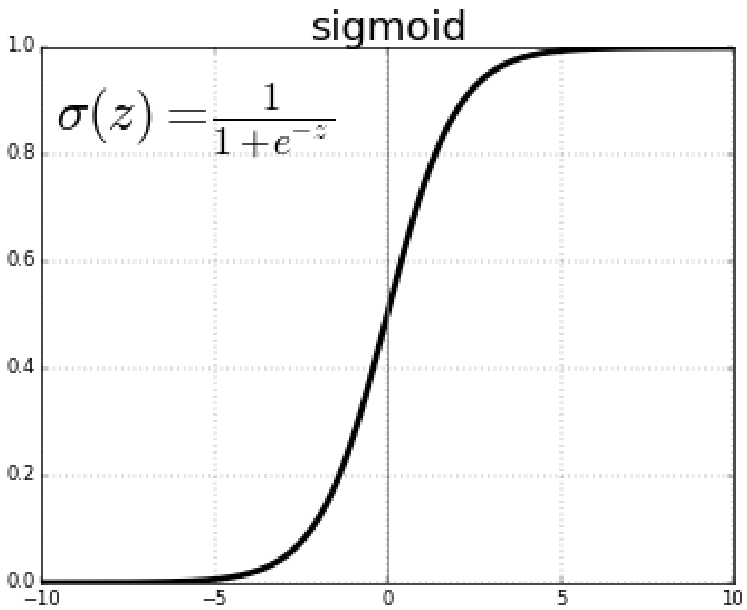




- For hidden units: use ***tanh, ReLU*** should be better than ***sigmoid***
- For output layer ($0 \le \hat{y} \le 1$), we can use ***sigmoid***

van-linh.le@u-bordeaux.fr

Activation functions and their derivatives



***Sigmoid** function*

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

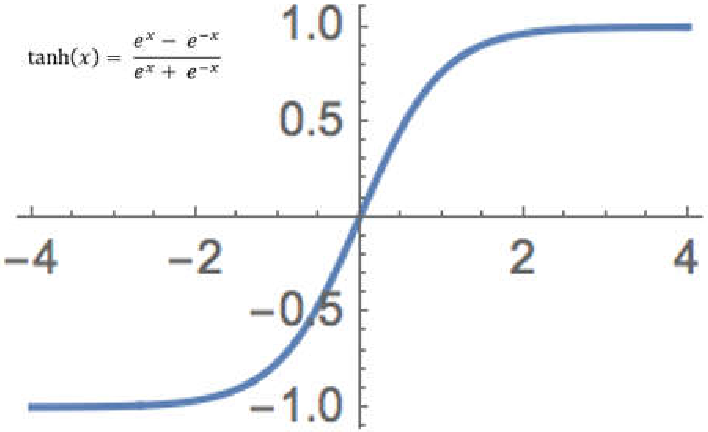
Derivative:

$$\frac{d\sigma(z)}{dz} = slope\ of\ \sigma(z) at\ z$$

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

van-linh.le@u-bordeaux.fr

Activation functions and their derivatives



TanH *function*

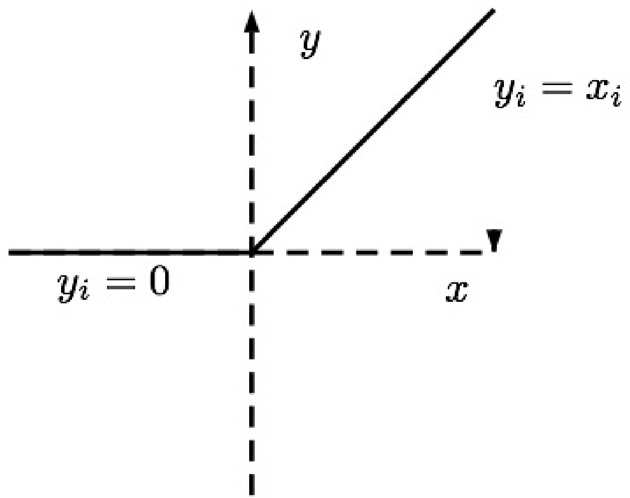
$$\tanh(z) = \sigma(z) = \frac{e^x - e^{-x}}{e^x + e^{-z}}$$

Derivative:

$$\frac{d\sigma(z)}{dz} = \mathbf{1 - (\sigma(z))^2}$$

van-linh.le@u-bordeaux.fr

Activation functions and their derivatives



***ReLU** function*

$$\mathrm{ReLU}(z) = \sigma(z) = \max(0, z)$$

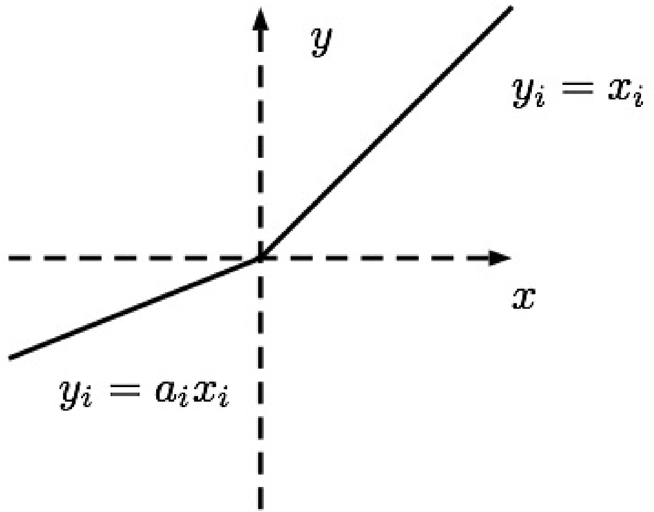
Derivative:

$$\frac{d\sigma(z)}{dz} = \begin{cases} \mathbf{0} & \boldsymbol{if\ z < 0} \\ \mathbf{1} & \boldsymbol{if\ z \geq 0} \end{cases}$$

van-linh.le@u-bordeaux.fr

Activation functions and their derivatives

***LeakyReLU** function*



$$\text{LeakyReLU}(z) = \sigma(z) = \max(0.01z, z)$$

Derivative:

$$\frac{d\sigma(z)}{dz} = \begin{cases} \mathbf{0.01} & \boldsymbol{if\ z < 0} \\ \mathbf{1} & \boldsymbol{if\ z \geq 0} \end{cases}$$

van-linh.le@u-bordeaux.fr

Gradient descent for NN

Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}$

$n_x = n^{[0]}$ is input, $n^{[1]}$ is number of hidden unit, $n^{[2]} = 1$ is output

Size of parameters: $(n^{[1]}, n^{[0]}), (n^{[1]}, 1)$, $(n^{[2]}, n^{[1]})$, $(n^{[2]}, 1)$

Cost function: $\mathcal{J}(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}) = \frac{1}{m}\sum_{i=1}^{m} \mathcal{L}(a, y)$

van-linh.le@u-bordeaux.fr

Gradient descent for NN

Gradient descent:

Repeat {

Compute predict: $a^{(i)} = \hat{y}^{(i)}$ $(i = 1..m)$

$dW^{[1]} = \frac{dJ}{dW^{[1]}}, db^{[1]} = \frac{dJ}{db^{[1]}}, \ldots$

$W^{[1]} = W^{[1]} - \alpha.dW^{[1]}$

$b^{[1]} = b^{[1]} - \alpha.db^{[1]}$

$W^{[2]} = W^{[2]} - \alpha.dW^{[2]}$

$b^{[2]} = b^{[2]} - \alpha.db^{[2]}$

}

van-linh.le@u-bordeaux.fr

Gradient descent for -N

Forward propagation

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$

Where $g^{[1]}, g^{[2]}$ are activation function at layer 1 and 2.

Backward propagation

$$dZ^{[2]} = A^{[2]} - Y$$
$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$
$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$

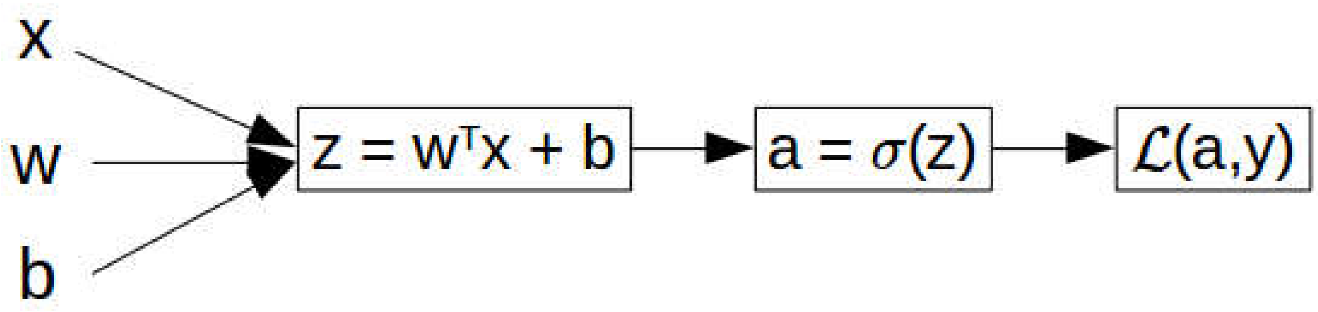
$$dZ^{[1]} = W^{[2]T}.dZ^{[2]} * g^{[1]'}(Z^{[1]})$$
$$dW^{[1]} = \frac{1}{m} dZ^{[1]} X^T$$
$$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$$

Where:

- $g^{[1]'}$ is derivative of activation function that we used in the first layer.
- * : is element-wise product

van-linh.le@u-bordeaux.fr

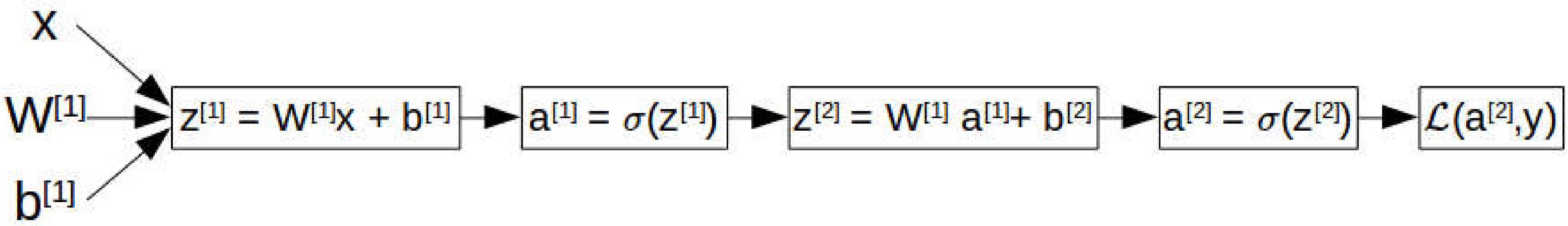
Backward propagation intuition



What are values of da, dz, dw and db?

van-linh.le@u-bordeaux.fr

Backward propagation intuition



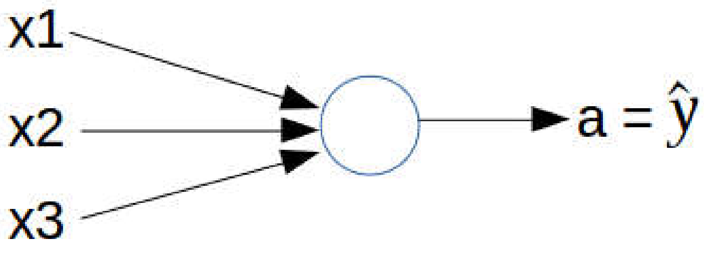
What are values of $da^{[2]}, dz^{[2]}, dw^{[2]}, db^{[2]}, da^{[1]}, dz^{[1]}, dw^{[1]}$ *and* $db^{[1]}$?

van-linh.le@u-bordeaux.fr

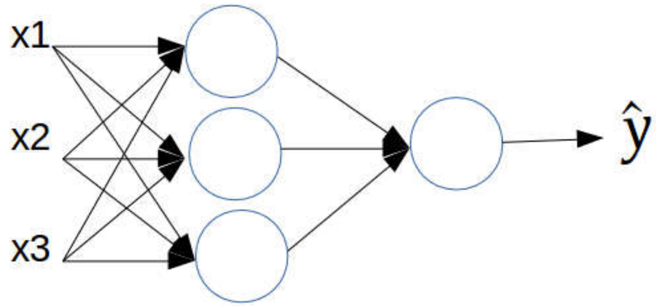
Contents

van-linh.le@u-bordeaux.fr

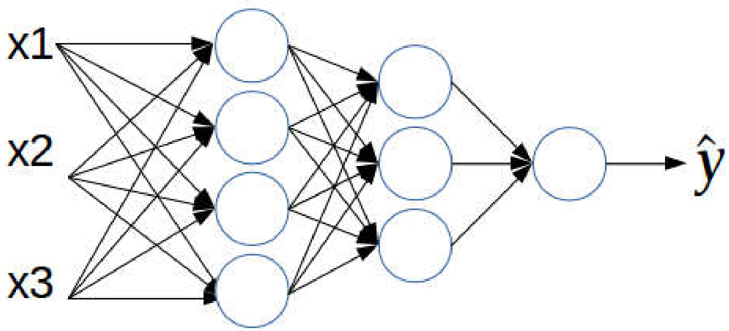
Deep N-layers Neural Network



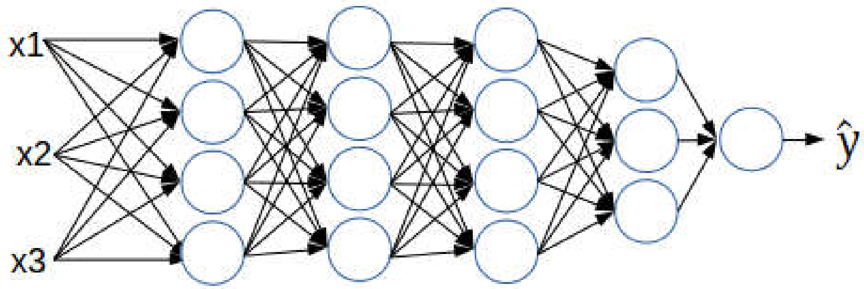
Logistic Regression



1 hidden layer



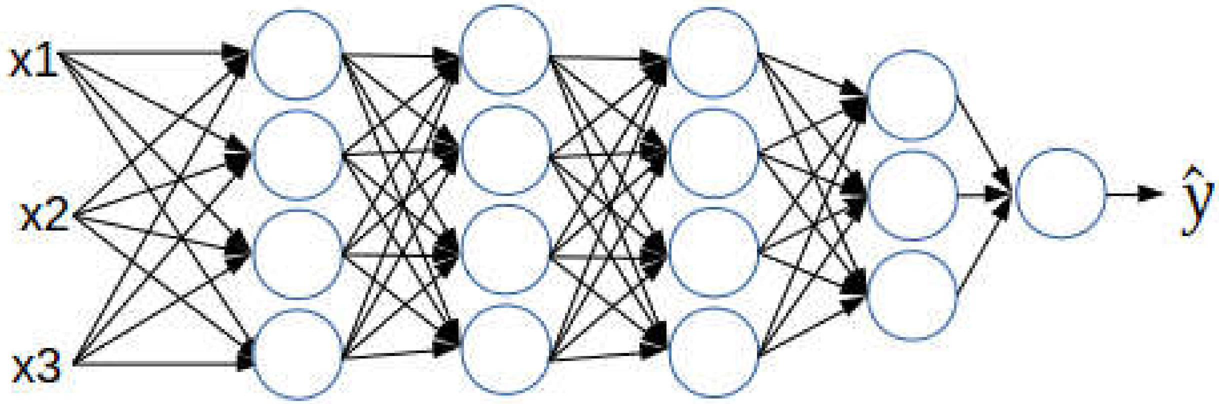
2 hidden layers



4 hidden layers

Deep N-layers Neural Network

Consider a 5-layers Neural Network



- L = 5 (number of layers)
- $n^{[l]}$ = number hidden unit at layer l
- $a^{[l]}$= activation at layer l
- $W^{[l]}$= weights for $Z^{[l]}$
- $b^{[l]}$= bias for $Z^{[l]}$

van-linh.le@u-bordeaux.fr

Deep N-layers Neural Network

Forward propagation at layer $\boldsymbol{l}$

- **Input:** $a^{[l-1]}$
- **Ouput**: $a^{[l]}$
- **Parameters** of $z^{[l]}$: $(W^{[l]}, b^{[l]})$
- Calculating:

$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

- Vectorize:

$$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$$
$$A^{[l]} = g^{[l]}(Z^{[l]})$$

Backward propagation at layer $\boldsymbol{l}$

- **Input:** $da^{[l]}$
- **Ouput**: $da^{[l-1]}, dW^{[l]}, db^{[l]}$
- **Parameters** of $z^{[l]}$: $(W^{[l]}, b^{[l]})$
- Calculating:

$$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]})$$
$$dW^{[l]} = dz^{[l]}.a^{[l-1]}$$
$$db^{[l]} = dz^{[l]}$$
$$da^{[l-1]} = W^{[l]T}.dz^{[l]}$$

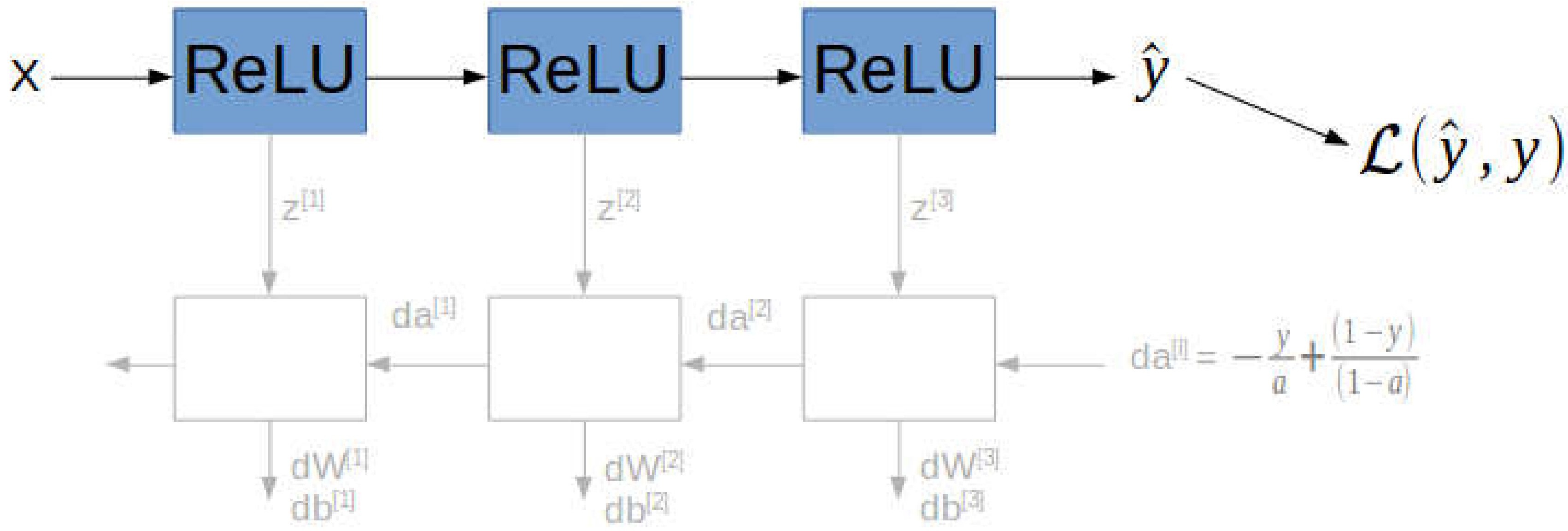
- Vectorize:

$$dZ^{[l]} = da^{[l]} * g^{[l]\prime}(Z^{[l]})$$
$$dW^{[l]} = \frac{1}{m}dZ^{[l]}.A^{[l-1]}$$
$$db^{[l]} = \frac{1}{m}np.sum(dZ^{[l]})$$
$$dA^{[l-1]} = W^{[l]T}.dZ^{[l]}$$

van-linh.le@u-bordeaux.fr

Deep N-layers Neural Network

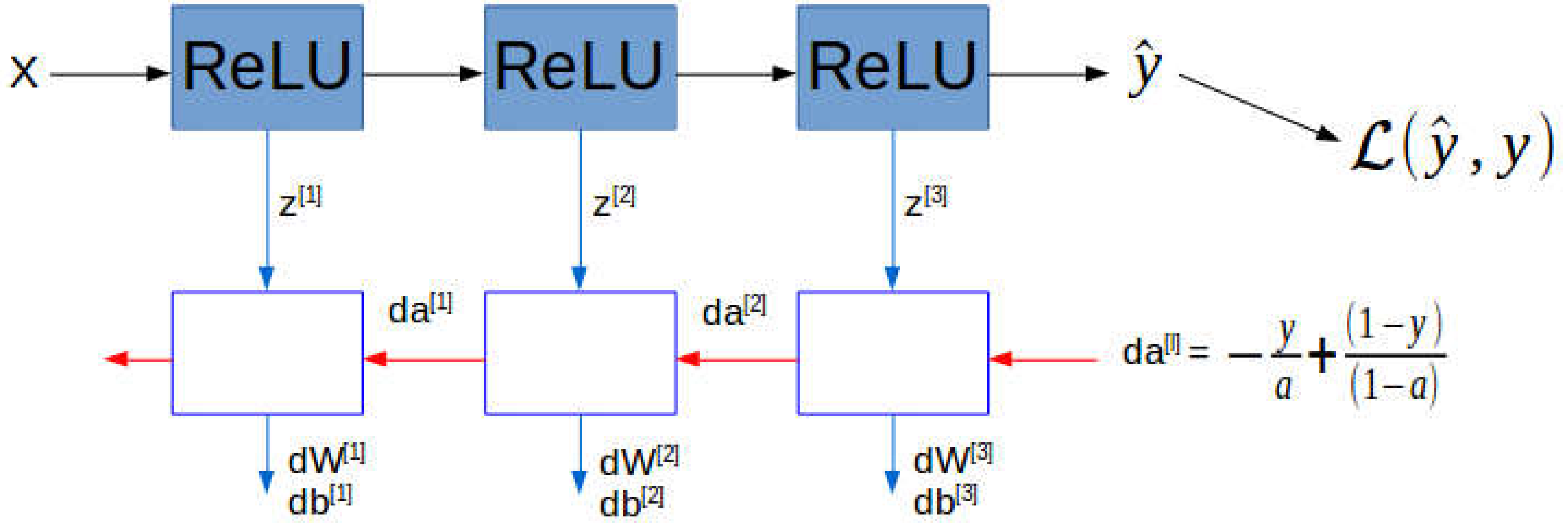
Forward and backward propagation



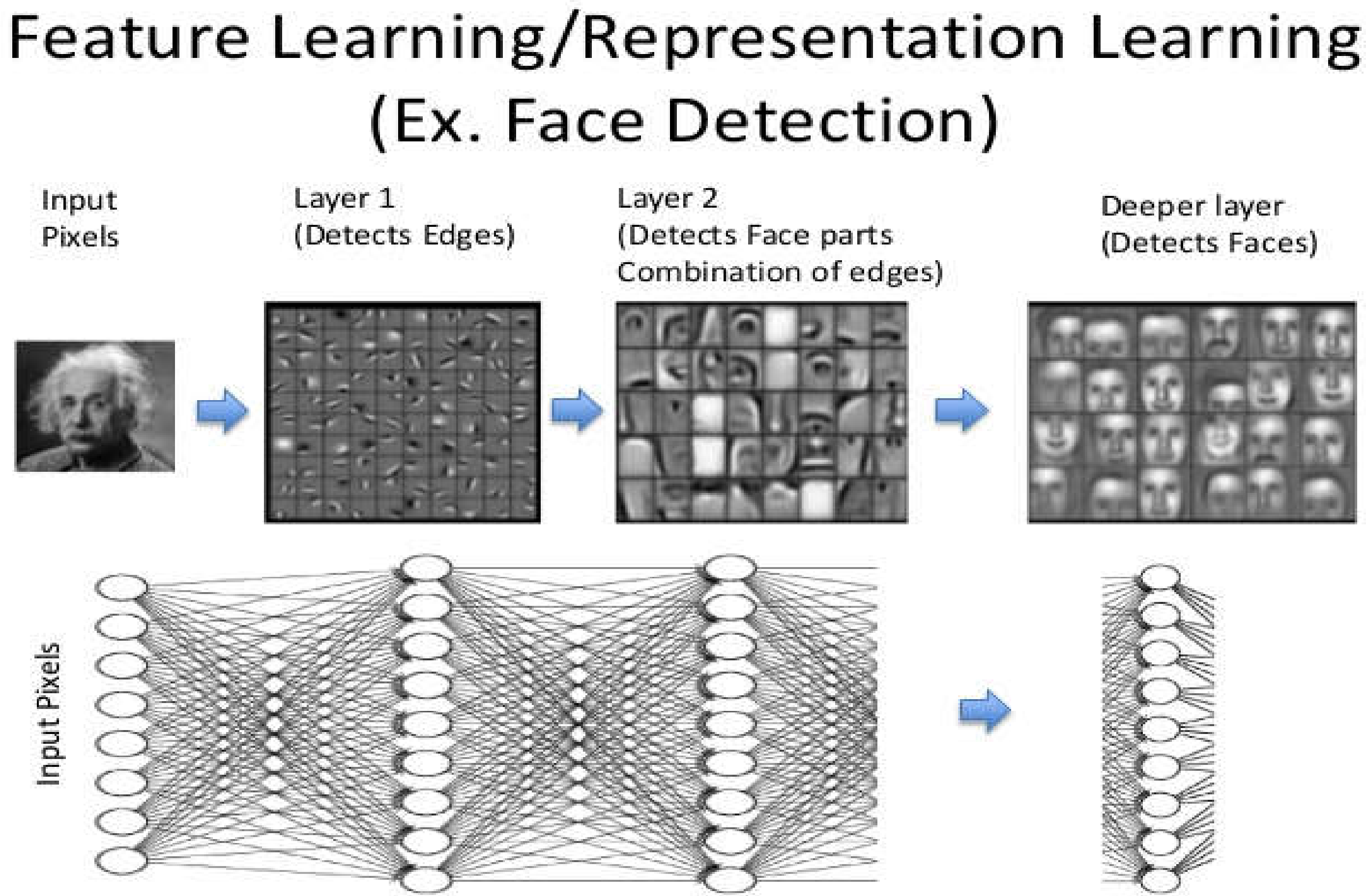
van-linh.le@u-bordeaux.fr

Deep N-layers Neural Network

Forward and backward propagation



Deep N-layers Neural Network



[slideshare.net]

van-linh.le@u-bordeaux.fr

Deep N-layers Neural Network

Parameters:

- W of the layers
- b of the layers

Hyper-parameters:

- Learning rate α
- Number of iteration
- Number of hidden layers and its number units
- Activation functions

van-linh.le@u-bordeaux.fr

Summary

- Neural networks
- Logistic Regression problem
- Derivative and gradient descent
- Vectorization
- Deep N-layers neural network
- Forward and backward propagation

van-linh.le@u-bordeaux.fr

université de BORDEAUX