Lecture slides for this course
have been prepared by Dr. Le Minh Huy,
EEE, Phenikaa University

PHENIKAA UNIVERSITY

Deep Learning

Chapter 2 Building Neural Network from Scratch

Dr. Van-Toi NGUYEN
EEE, Phenikaa University

1

PHENIKAA UNIVERSITY

Chapter 2: Building Neural Network from Scratch

1. Shallow neural network
2. Deep neural network
3. Building neural network: step-by-step (modulation)
4. Regularization
5. Dropout
6. Batch Normalization
7. Optimizers
8. Hyper-parameters
9. Practice

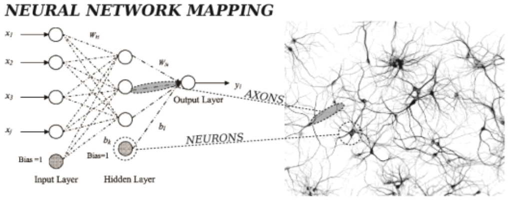
2

Previous Lecture Overview

PHENIKAA UNIVERSITY

Biologically inspired (akin to the neurons in a brain)

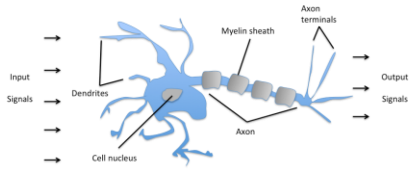
NEURAL NETWORK MAPPING



3

Previous Lecture Overview

Artificial Neurons and the McCulloch-Pitts Model (1943)

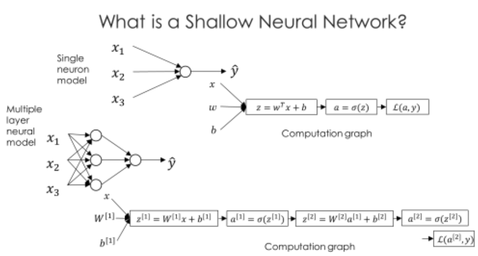


Schematic of a biological neuron.

W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4):115–133, 1943.

4

Previous Lecture Overview

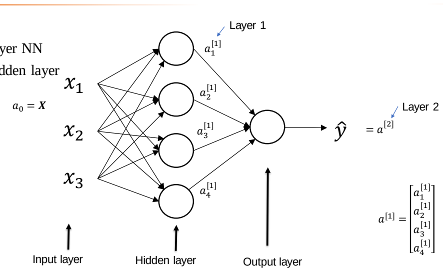


5

Previous Lecture Overview

One hidden layer Neural Network

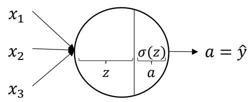
- 2-layer NN
- 1 hidden layer



6

Previous Lecture Overview

Computing NN's Output



$z = w^T x + b$

$a = \sigma(z)$

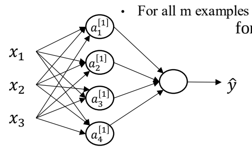
These slides are provided by Minhhuy Le, ICSLab, Phenikaa Uni.

7

7

Previous Lecture Overview

Vectorizing across multiple examples



- For all m examples …

for i = 1 to m:

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

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

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

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

These slides are provided by Minhhuy Le, ICSLab, Phenikaa Uni.

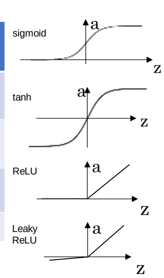
8

8

Previous Lecture Overview

Activation functions

Activation Function	Formula (g(z))	Derivative (g'(z))
sigmoid	$a = \frac{1}{1+e^{-z}}$	$a(1-a)$
tanh	$a = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$1 - a^2$
ReLU	$\max(0, z)$	0 if $z < 0$ 1 if $z \geq 0$
Leaky ReLU	$\max(0.01z, z)$	0.01 if $z < 0$ 1 if $z \geq 0$



These slides are provided by Minhhuy Le, ICSLab, Phenikaa Uni.

9

9

Previous Lecture Overview

Vectorizing across multiple examples

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

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

```
for i = 1 to m:
```

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

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

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

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

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

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

$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$

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

These slides are provided by Minhhuy Le, ICSLab, Phenikaa Uni.

10

10

Previous Lecture Overview

Gradient descent for one hidden layer

x, $W^{[1]}$, $b^{[1]}$ → $z^{[1]} = W^{[1]}x + b^{[1]}$ → $a^{[1]} = \sigma(z^{[1]})$

$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$

$dW^{[1]} = dz^{[1]}x^T$ $\quad db^{[1]} = dz^{[1]}$

$W^{[2]}$, $b^{[2]}$ → $z^{[2]} = W^{[2]}x + b^{[2]}$ → $a^{[2]} = \sigma(z^{[2]})$ → $\mathcal{L}(a^{[2]}, y)$

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

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

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

These slides are provided by Minhhuy Le, ICSLab, Phenikaa Uni.

11

11

Previous Lecture Overview

Vectorizing Gradient Descent

$dz^{[2]} = a^{[2]} - y$	$dZ^{[2]} = A^{[2]} - Y$
$dW^{[2]} = dz^{[2]}a^{[1]^T}$	$dW^{[2]} = \frac{1}{m}dZ^{[2]}A^{[1]^T}$
$db^{[2]} = dz^{[2]}$	$db^{[2]} = \frac{1}{m}np.sum(dZ^{[2]}, axis = 1, keepdims = True)$
$dz^{[1]} = W^{[2]T}dz^{[2]} * g^{[1]\prime}(z^{[1]})$	$dZ^{[1]} = W^{[2]T}dZ^{[2]} * g^{[1]\prime}(Z^{[1]})$
$dW^{[1]} = dz^{[1]}x^T$	$dW^{[1]} = \frac{1}{m}dZ^{[1]}X^T$
$db^{[1]} = dz^{[1]}$	$db^{[1]} = \frac{1}{m}np.sum(dZ^{[1]}, axis = 1, keepdims = True)$

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

These slides are provided by Minhhuy Le, ICSLab, Phenikaa Uni.

12

12

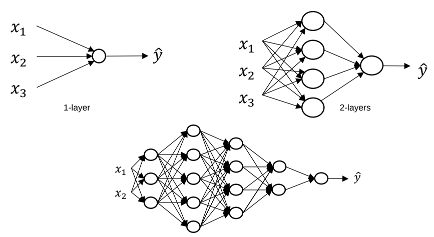
2. Deep neural network

Deep Learning???

13

2. Deep neural network

Multiple layers

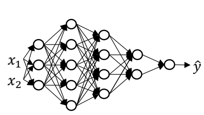


14

2. Deep neural network

Notations in Deep Neural Networks

- #Layers = L = 5
- $n^{[l]}$ = #units in layer l
  - $n^{[1]} = 3, n^{[2]} = 5, n^{[3]} = 4, n^{[4]} = 2, n^{[5]} = n^{[L]} = 1$
  - $n^{[0]} = n^{[x]} = 2$
- $a^{[l]} = g^{[l]}(z^{[l]})$
- $W^{[l]}$ = weights for computing $z^{[l]}$
- $b^{[l]}$ = bias for computing $z^{[l]}$
- $a^{[0]} = X$ (input)
- $a^{[L]} = \hat{y}$ (prediction output)



15

2. Deep neural network

Dimensions of vectorized implementations

- For one single training example:
- $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$
- $(n^{[l]}, 1) = (n^{[l]}, n^{[l-1]}) \times (n^{[l-1]}, 1) + (n^{[l]}, 1)$

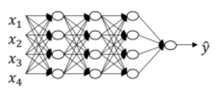
- For a vectorized implementation over m examples
- $Z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$
- $(n^{[l]}, m) = (n^{[l]}, n^{[l-1]}) \times (n^{[l-1]}, m) + (n^{[l]}, m)$

Matrix	Dimensions
$Z^{[l]}, A^{[l]}, b^{[l]}, dZ^{[l]}, dA^{[l]}, db^{[l]}$	$(n^{[l]}, m)$
$W^{[l]}, dW^{[l]}$	$(n^{[l]}, n^{[l-1]})$

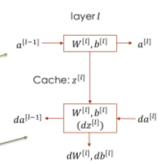
16

2. Deep neural network

Building Blocks of Deep Neural Networks



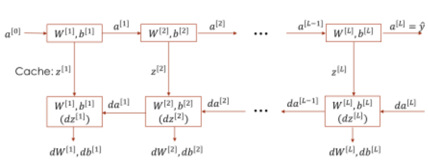
- Focus on some layer l, parameters include $W^{[l]}$, $b^{[l]}$
- Forward Propagation
  - Input: $a^{[l-1]}$
  - Output: $a^{[l]}$
  - Method: $z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]}$, $a^{[l]} = g^{[l]}(z^{[l]})$
  - Cache: $z^{[l]}$
- Backward Propagation
  - Input: $da^{[l]}$, $z^{[l]}$ (from cache)
  - Output: $da^{[l-1]}$, $dW^{[l]}, db^{[l]}$



17

2. Deep neural network

Forward and backward functions



$$W^{[l]} = W^{[l]} - \alpha dW^{[l]}$$
$$b^{[l]} = b^{[l]} - \alpha db^{[l]}$$

18

2. Deep neural network

Forward Propagation for DNN

- Input: $a^{[l-1]}$
- Forward Propagation:
  - For single example:
    $$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}, \quad a^{[l]} = g^{[l]}(z^{[l]})$$
  - Vectorized version:
    $$Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}, \quad A^{[l]} = g^{[l]}(Z^{[l]})$$
- Output: $a^{[l]}$
- Cache: $z^{[l]}, W^{[l]}, b^{[l]}$

19

2. Deep neural network

Backward Propagation for DNN

- Inputs: $da^{[l]}, z^{[l]}, W^{[l]}, b^{[l]}$ (last 3 from cache)
- Outputs: $da^{[l-1]}, dW^{[l]}, db^{[l]}$
- Backward Propagation:
- For **single example**
  $$dz^{[l]} = da^{[l]} * g^{[l]\prime}(z^{[l]})$$
  $$dW^{[l]} = dz^{[l]}.a^{[l-1]}$$
  $$db^{[l]} = dz^{[l]}$$
  $$da^{[l-1]} = W^{[l]T}dz^{[l]}$$
  $$dz^{[l]} = W^{[l+1]T}dz^{[l+1]} * g^{[l]\prime}(z^{[l]})$$

Vectorized implementation

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

20

2. Deep neural network

Parameters vs. Hyperparameters

- Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, \ldots$
- Hyperparameters
  - Learning rate: α
  - Number of iterations
  - Number of hidden layers or L
  - Number of hidden units in each layer: $n^{[1]}, n^{[2]}, \ldots$
  - Choice of activation function: sigmoid, ReLU, tanh, etc.
  - Momentum, mini-batch size, regularization parameters, … (in the next Chapter)

21

2. Deep neural network

Summary of Forward/Backward Computations

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$
$$A^{[1]} = g^{[1]}(Z^{[1]})$$
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$
$$A^{[2]} = g^{[2]}(Z^{[2]})$$
$$\vdots$$
$$A^{[L]} = g^{[L]}\left(Z^{[L]}\right) = \hat{Y}$$

Forward Propagation

$$dZ^{[L]} = A^{[L]} - Y$$
$$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L]^T}$$
$$db^{[L]} = \frac{1}{m} np.\text{sum}(\text{d}Z^{[L]}, axis = 1, keepdims = True)$$
$$dZ^{[L-1]} = dW^{[L]^T} dZ^{[L]} g'^{[L]}(Z^{[L-1]})$$
$$\vdots$$
$$dZ^{[1]} = dW^{[L]^T} dZ^{[2]} g'^{[1]}(Z^{[1]})$$
$$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[1]^T}$$
$$db^{[1]} = \frac{1}{m} np.\text{sum}(\text{d}Z^{[1]}, axis = 1, keepdims = True)$$

Backward Propagation

22

3. Building neural network: step-by-step (modulation)

On Assignments
Coding time !!!

23