K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

# CS540 Introduction to Artificial Intelligence Lecture 3

#### Young Wu

Based on lecture slides by Jerry Zhu, Yingyu Liang, and Charles Dyer

July 1, 2022

<span id="page-1-0"></span>

[Neural Network](#page-1-0)<br>● Neural Network [Backpropagation](#page-9-0) [Multi-Layer Network](#page-22-0)<br>● Network Backpropagation Multi-Layer Network

**KORK ERKER ADAM ADA** 

### Single Layer Perceptron **Motivation**

- Perceptrons can only learn linear decision boundaries.
- Many problems have non-linear boundaries.
- One solution is to connect perceptrons to form a network.

[Neural Network](#page-1-0) and the Care of [Backpropagation](#page-9-0) and the Care of [Multi-Layer Network](#page-22-0) and the Care of Secondo Control of American Secondo Control of American Secondo Control of American Secondo Control of American Secondo Co

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

#### Multi-Layer Perceptron Motivation

• The output of a perceptron can be the input of another.

$$
a = g \left( w^T x + b \right)
$$
  
\n
$$
a' = g \left( w'^T a + b' \right)
$$
  
\n
$$
a'' = g \left( w''^T a' + b'' \right)
$$
  
\n
$$
\hat{y} = \mathbb{I}_{\{a'' > 0\}}
$$

[Neural Network](#page-1-0) and the second state of the [Backpropagation](#page-9-0) and the second of the [Multi-Layer Network](#page-22-0)<br>
Multi-Layer Network and the second of t

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

#### Learning XOR Operator, Part 1 **Motivation**

• XOR cannot be modeled by a single perceptron.



**KORK ERKER ADA DI VOLO** 

# Learning XOR Operator, Part 2 **Motivation**

- OR, AND, NOT AND can be modeled by perceptrons.
- If the outputs of OR and NOT AND is used as inputs for AND, then the output of the network will be XOR.

 $000000000$ 

[Neural Network](#page-1-0) [Backpropagation](#page-9-0) [Multi-Layer Network](#page-22-0)  $0000$ 

**KORK ERKER ADA DI VOLO** 

#### Neural Network Biology **Motivation**

- Human brain: 100, 000, 000, 000 neurons.
- Each neuron receives input from 1,000 others.
- An impulse can either increase or decrease the possibility of nerve pulse firing.
- If sufficiently strong, a nerve pulse is generated.
- The pulse forms the input to other neurons.

[Neural Network](#page-1-0) and the Communication [Backpropagation](#page-9-0) and the Communication [Multi-Layer Network](#page-22-0)<br>
Multi-Layer Network and the Communication Backpropagation and the Communication Communication of Communicatio  $0000$ 

**KORK ERKER ADA DI VOLO** 

# Theory of Neural Network **Motivation**

- In theory:
- **1** Hidden-layer with enough hidden units can represent any continuous function of the inputs with arbitrary accuracy.
- 2 2 Hidden-layer can represent discontinuous functions.
- In practice:
- **4** AlexNet: 8 layers.
- **2** GoogLeNet: 27 layers (or  $22 +$  pooling).
- **3** ResNet: 152 layers.

[Neural Network](#page-1-0) and the Constitution [Backpropagation](#page-9-0) and the Constitution [Multi-Layer Network](#page-22-0)<br>
Multi-Layer Network and the Constitution Backpropagation and the Constitution of Constitution of Constitution

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

### Neural Network Examples **Motivation**

- **Classification tasks.**
- Approximate functions.
- Store functions (after midterm).

**KORK ERKER ADA DI VOLO** 

#### Gradient Descent **Motivation**

- The derivatives are more difficult to compute.
- The problem is no longer convex. A local minimum is no longer guaranteed to be a global minimum.
- Need to use chain rule between layers called backpropagation.

<span id="page-9-0"></span>

[Neural Network](#page-1-0) **[Backpropagation](#page-9-0)** Backpropagation [Multi-Layer Network](#page-22-0) Multi-Layer Network<br>Backpropagation Multi-Layer Network  $0000$ 

**KORK ERKER ADAM ADA** 

# Backpropagation **Description**

- Initialize random weights.
- (Feedforward Step) Evaluate the activation functions.
- (Backpropagation Step) Compute the gradient of the cost function with respect to each weight and bias using the chain rule.
- Update the weights and biases using gradient descent.
- Repeat until convergent.

**KORK ERKER ADAM ADA** 

# Cost Function Definition

For simplicity, assume there are only two layers (one hidden layer), and  $g$  is the sigmoid function for this lecture.

$$
g'(z) = g(z) (1 - g(z))
$$

Let the output in the second layer be  $a_i$  for instance  $x_i$ , then cost function is the squared error,

$$
C = \frac{1}{2} \sum_{i=1}^{n} (y_i - a_i)^2
$$

0000

**KORKARYKERKER POLO** 

#### Interal Activations Definition

Let the output in the first layer be  $a_{ij}^{(1)}, j = 1, 2, ..., m^{(1)}$ .

$$
a_i = g(z_i)
$$
  

$$
z_i = \sum_{j=1}^{m^{(1)}} a_{ij}^{(1)} w_j^{(2)} + b^{(2)}
$$

• Let the input in the zeroth layer be  $x_{ii}$ ,  $j = 1, 2, ..., m$ .

$$
a_{ij}^{(1)} = g\left(z_{ij}^{(1)}\right)
$$
  

$$
z_{ij}^{(1)} = \sum_{j'=1}^{m} x_{ij'} w_{j'j}^{(1)} + b_j^{(1)}
$$

00000000

[Neural Network](#page-1-0) [Backpropagation](#page-9-0) [Multi-Layer Network](#page-22-0)  $0000$ 

# **Notations** Definition

- $a_{ij}^{(l)}$  is the hidden unit activation of instance  $i$  in layer I, unit  $j$
- $z_{ij}^{(l)}$  is the linear part of instance  $i$  in layer  $l$ , unit  $j$
- $w_{i'i}^{(I)}$  $j_{j\,j}^{(l)}$  is the weights between layers  $l-1$  and *I*, from unit  $j^{\prime}$  in layer  $l - 1$  to unit *j* in layer *l*.
- $b_i^{(l)}$  $j_j^{(t)}$  is the bias for layer / unit *j*.
- $m^{(1)}$  is the number of units in layer *l*.
- Superscript *l* is omitted for the last layer.

00000000

[Neural Network](#page-1-0) Network [Backpropagation](#page-9-0) [Multi-Layer Network](#page-22-0) Network Network Backpropagation Multi-Layer Network<br>Network Decision Multi-Layer Network Decision Decision Decision Decision Decision Decision Decision Decision D 0000

**KORKARYKERKER POLO** 

## Required Gradients Definition

• The derivatives that are required for the gradient descents are the following.

> $\partial C$  $\partial w_{i'i}^{(1)}$ j 1 j  $j = 1, 2, ..., m<sup>(1)</sup>, j' = 1, 2, ..., m$  $\partial C$  $\partial b_i^{(1)}$ j  $j = 1, 2, ..., m<sup>(1)</sup>$  $\partial C$  $\partial w_i^{(2)}$ j  $j = 1, 2, \dots, m^{(1)}$  $\partial C$  $\partial b^{(2)}$

**KORK ERKER ADAM ADA** 

### Gradients of Second Layer Definition

Apply chain rule once to get the gradients for the second layer.

$$
\frac{\partial C}{\partial w_j^{(2)}} = \sum_{i=1}^n \frac{\partial C}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial w_j^{(2)}}, j = 1, 2, ..., m^{(1)}
$$

$$
\frac{\partial C}{\partial b^{(2)}} = \sum_{i=1}^n \frac{\partial C}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial b^{(2)}}
$$

[Neural Network](#page-1-0) and the second state of the [Backpropagation](#page-9-0) and the second state of the [Multi-Layer Network](#page-22-0) and the second state of the second

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

#### Gradients of First Layer Definition

• Chain rule twice says,

$$
\frac{\partial C}{\partial w_{j'j}^{(1)}} = \sum_{i=1}^{n} \frac{\partial C}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial a_{ij}^{(1)}} \frac{\partial a_{ij}^{(1)}}{\partial z_{ij}^{(1)}} \frac{\partial z_{ij}^{(1)}}{\partial w_{j'j}^{(1)}}
$$

$$
j = 1, 2, ..., m^{(1)}, j' = 1, 2, ..., m
$$

$$
\frac{\partial C}{\partial b_j^{(1)}} = \sum_{i=1}^{n} \frac{\partial C}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial a_{ij}^{(1)}} \frac{\partial a_{ij}^{(1)}}{\partial z_{ij}^{(1)}} \frac{\partial z_{ij}^{(1)}}{\partial b_j^{(1)}}
$$

$$
j = 1, 2, ..., m^{(1)}
$$

[Neural Network](#page-1-0) and the Communication [Backpropagation](#page-9-0) and the Communication [Multi-Layer Network](#page-22-0)<br>
Multi-Layer Network and the Communication Communication Communication and the Communication Communication Com

K ロ ▶ K @ ▶ K 할 ▶ K 할 ▶ | 할 | © 9 Q @

#### Derivative of Error Definition

• Compute the derivative of the error function.

$$
C = \frac{1}{2} \sum_{i=1}^{n} (y_i - a_i)^2
$$

$$
\Rightarrow \frac{\partial C}{\partial a_i} = a_i - y_i
$$

K ロ ▶ K 個 ▶ K 결 ▶ K 결 ▶ │ 결 │ K 9 Q Q

#### Derivative of Interal Outputs, Part 1 Definition

Compute the derivative of the output in the second layer.

$$
a_i = g(z_i)
$$
  
\n
$$
\Rightarrow \frac{\partial a_i}{\partial z_i} = g(z_i) (1 - g(z_i)) = a_i (1 - a_i)
$$
  
\n
$$
z_i = \sum_{j=1}^{m^{(1)}} a_{ij}^{(1)} w_j^{(2)} + b^{(2)}
$$
  
\n
$$
\Rightarrow \frac{\partial z_i}{\partial w_j^{(2)}} = a_{ij}^{(1)}, \frac{\partial z_i}{\partial b^{(2)}} = 1
$$

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

#### Derivative of Internal Outputs, Part 2 Definition

• Compute the derivative of the output in the first layer.

$$
a_{ij}^{(1)} = g\left(z_{ij}^{(1)}\right)
$$
  
\n
$$
\Rightarrow \frac{\partial a_{ij}^{(1)}}{\partial z_{ij}^{(1)}} = g\left(z_{ij}^{(1)}\right)\left(1 - g\left(z_{ij}^{(1)}\right)\right) = a_{ij}^{(1)}\left(1 - a_{ij}^{(1)}\right)
$$
  
\n
$$
z_{ij}^{(1)} = \sum_{j'=1}^{m} x_{ij}' w_{j'j}^{(1)} + b_{j}^{(1)}
$$
  
\n
$$
\Rightarrow \frac{\partial z_{ij}^{(1)}}{\partial w_{j'j}^{(1)}} = x_{ij'}, \frac{\partial z_{ij}^{(1)}}{\partial b_{j}^{(1)}} = 1
$$

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

### Derivative of Internal Outputs, Part 3 Definition

• Compute the derivative between the outputs.

$$
z_i = \sum_{j=1}^{m^{(1)}} a_{ij}^{(1)} w_j^{(2)} + b^{(2)}
$$

$$
\Rightarrow \frac{\partial z_i}{\partial a_{ij}^{(1)}} = w_j^{(2)}
$$

**KORKARYKERKER POLO** 

# Gradient Step, Combined Definition

• Put everything back into the chain rule formula. (Please check for typos!)

 $\partial C$  $\partial w_{i'i}^{(1)}$ j 1 j  $=\sum_{n=1}^{n}$  $i=1$  $p(a_i - y_i) a_i (1 - a_i) w_i^{(2)}$  $\tilde{a}_{ij}^{(2)}$ a $_{ij}^{(1)}\left(1-a_{ij}^{(1)}\right)$  Xij'  $\partial C$  $\partial b^{(1)}_i$ j  $=\sum_{n=1}^{n}$  $i=1$  $p(a_i - y_i) a_i (1 - a_i) w_i^{(2)}$  $g_j^{(2)}$ a $_{ij}^{(1)}$   $\left(1 - a_{ij}^{(1)}\right)$  $\partial C$  $\partial w_i^{(2)}$ j  $=\sum_{n=1}^{n}$  $i=1$  $p\left(a_i-y_i\right)a_i\left(1-a_i\right)a_{ii}^{(1)}$ ij  $\partial C$  $\frac{\partial C}{\partial b^{(2)}} = \sum_{i=1}^n$  $i=1$  $(a_i - y_i) a_i (1 - a_i)$ 

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

#### Gradient Descent Step Definition

• The gradient descent step is the same as the one for logistic regression.

$$
w_j^{(2)} \leftarrow w_j^{(2)} - \alpha \frac{\partial C}{\partial w_j^{(2)}}, j = 1, 2, ..., m^{(1)}
$$
  
\n
$$
b^{(2)} \leftarrow b^{(2)} - \alpha \frac{\partial C}{\partial b^{(2)}},
$$
  
\n
$$
w_{j'j}^{(1)} \leftarrow w_{j'j}^{(1)} - \alpha \frac{\partial C}{\partial w_{j'j}^{(1)}}, j' = 1, 2, ..., m, j = 1, 2, ..., m^{(1)}
$$
  
\n
$$
b_j^{(1)} \leftarrow b_j^{(1)} - \alpha \frac{\partial C}{\partial b_j^{(1)}}, j = 1, 2, ..., m^{(1)}
$$

**KORKARYKERKER POLO** 

# Backpropogation, Part 1 Algorithm

- <span id="page-22-0"></span>Inputs: instances:  $\{x_i\}_i^n$  $\sum_{i=1}^{n}$  and  $\{y_i\}_{i}^{n}$  $\int_{i=1}^{n}$ , number of hidden layers  $L$  with units  $m^{(1)}, m^{(2)}, ..., m^{(L-1)}$ , with  $m^{(0)} = m$ ,  $m^{(L)} = 1$ , and activation function g is the sigmoid function.
- Outputs: weights and biases:  $w_{i'i}^{(I)}$  $\boldsymbol{b}_j^{(I)},\boldsymbol{b}_j^{(I)}$  $j^{(l)}, j' = 1, 2, ...., m^{(l-1)}, j = 1, 2, ...., m^{(l)}, l = 1, 2, ..., L$
- $\bullet$  Initialize the weights.

 $w_{i'i}^{(I)}$  $j^{\left(\prime\right)}_{j^{\prime}j},b^{\left(\prime\right)}_{j}\sim\,$  Unif  $\left[-1,1\right]$ 

[Neural Network](#page-1-0) and the Constitution of [Backpropagation](#page-9-0) and the Constitution of the [Multi-Layer Network](#page-22-0) and the<br>Network and the Constitution of Constitution of Constitution of the Constitution of Constitution of Constituti

K ロ ▶ K 個 ▶ K 할 ▶ K 할 ▶ 이 할 → 9 Q Q →

### Backpropagation, Part 2 Algorithm

• Evaluate the activation functions.

$$
a_i = g\left(\sum_{j=1}^{m^{(L-1)}} a_{ij}^{(L-1)} w_j^{(L)} + b^{(L)}\right)
$$
  
\n
$$
a_{ij}^{(l)} = g\left(\sum_{j'=1}^{m^{(l-1)}} a_{ij'}^{(l-1)} w_{j'j}^{(l)} + b_j^{(l)}\right), l = 1, 2, ..., L - 1
$$
  
\n
$$
a_{ij}^{(0)} = x_{ij}
$$

[Neural Network](#page-1-0) and the Control of [Backpropagation](#page-9-0) and the Control of Control of [Multi-Layer Network](#page-22-0)<br>
Network and the Control of Contr

**KORK ERKER ADAM ADA** 

### Backpropagation, Part 3 Algorithm

• Compute the  $\delta$  to simplify the expression of the gradient.

$$
\delta_i^{(L)} = (a_i - y_i) a_i (1 - a_i)
$$
  
\n
$$
\delta_{ij}^{(l)} = \sum_{j'=1}^{m^{(l+1)}} \delta_{j'}^{(l+1)} w_{jj'}^{(l+1)} a_{ij}^{(l)} (1 - a_{ij}^{(l)}) , l = 1, 2, ..., L - 1
$$

• Compute the gradient using the chain rule.

$$
\frac{\partial C}{\partial w_{j'j}^{(l)}} = \sum_{i=1}^{n} \delta_{ij}^{(l)} a_{ij'}^{(l-1)}, l = 1, 2, ..., L
$$

$$
\frac{\partial C}{\partial b_j^{(l)}} = \sum_{i=1}^{n} \delta_{ij}^{(l)}, l = 1, 2, ..., L
$$

[Neural Network](#page-1-0) and the Control of [Backpropagation](#page-9-0) and the Control of Control of [Multi-Layer Network](#page-22-0)<br>
Multi-Layer Network and the Control of Contr<br>
Multi-La  $000$ 

**KORKARYKERKER POLO** 

## Backpropagation, Part 4 Algorithm

• Update the weights and biases using gradient descent. For  $l = 1, 2, ..., L$  $w_{i'i}^{(I)}$  $y_j^{(l)} \leftarrow w_{j'j}^{(l)}$  $\hat{d}^{(l)}_{j'j} - \alpha \frac{\partial C}{\partial (l)}$  $\partial w_{i'i}^{(I)}$ j 1 j  $j' = 1, 2, ..., m^{(l-1)}, j = 1, 2, ..., m^{(l)}$  $b_j^{(l)} \leftarrow b_j^{(l)} - \alpha \frac{\partial C}{\partial l_l}$  $\partial b_i^{(I)}$ j  $j = 1, 2, \dots, m^{(l)}$ 

• Repeat the process until convergent.

$$
|C - C^{\text{prev}}| < \varepsilon
$$