CS540 Introduction to Artificial Intelligence
Lecture 3

Young Wu
Based on lecture slides by Jerry Zhu and Yingyu Liang

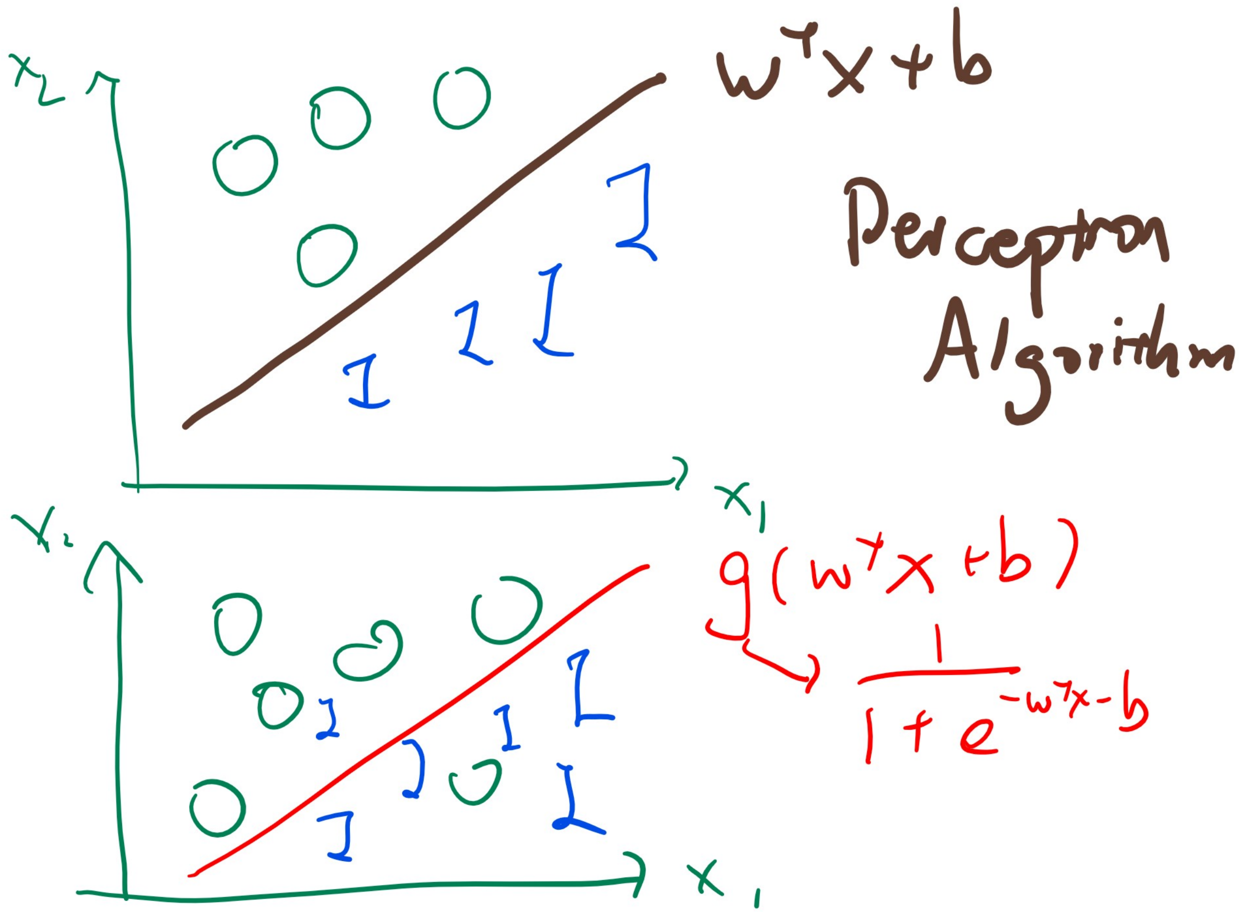
May 30, 2019

Test

Quiz (Graded)

- A:
- B:
- C:
- D: Choose this.
- E:

Gradient Descent.

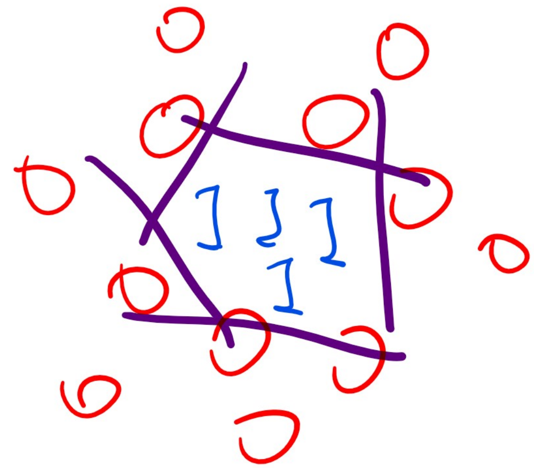


Homework

Quiz (Participation)

prog

- Have you finished homework 1
- A: Waiting for solution.
- B: Will start soon.
- C: Started.
- D: Does not work due to bugs.
- E: Finished: 90+ percent accuracy.



Neural Network.

AND Operator Data

Quiz (Particpation)

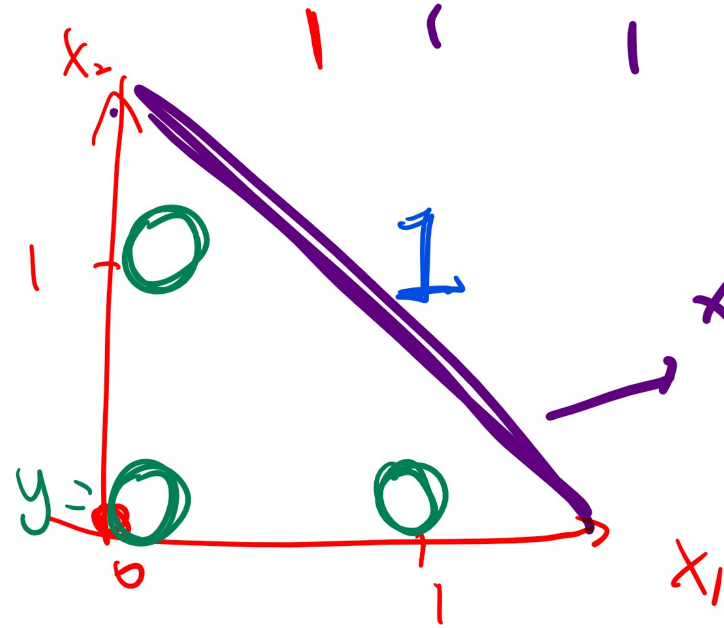
- Sample data for AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Learning AND Operator

Quiz (Particpation)

- Which one of the following is AND?
- A: $\hat{y} = \mathbb{1}_{\{1x_1+1x_2-1.5\geqslant 0\}}$
- B: $\hat{y} = \mathbb{1}_{\{1x_1+1x_2-0.5\geqslant 0\}}$
- C: $\hat{y} = \mathbb{1}_{\{-1x_1+0.5\geqslant 0\}}$
- D: $\hat{y} = \mathbb{1}_{\{-1x_1-1x_2+0.5\geqslant 0\}}$
- E: None of the above



OR Operator Data

Quiz (Graded)

- Sample data for OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

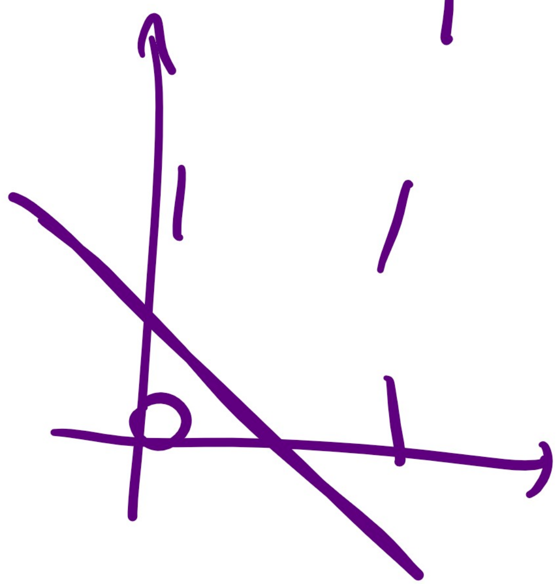
Learning OR Operator

Quiz (Graded)

Q5

- Which one of the following is OR?
- A: $\hat{y} = \mathbb{1}_{\{1x_1 + 1x_2 - 1.5 \geqslant 0\}}$
- B: $\hat{y} = \mathbb{1}_{\{1x_1 + 1x_2 - 0.5 \geqslant 0\}}$
- C: $\hat{y} = \mathbb{1}_{\{-1x_1 + 0.5 \geqslant 0\}}$
- D: $\hat{y} = \mathbb{1}_{\{-1x_1 - 1x_2 + 0.5 \geqslant 0\}}$
- E: None of the above

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



XOR Data

Quiz (Graded)

- Sample data for XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Learning XOR Operator

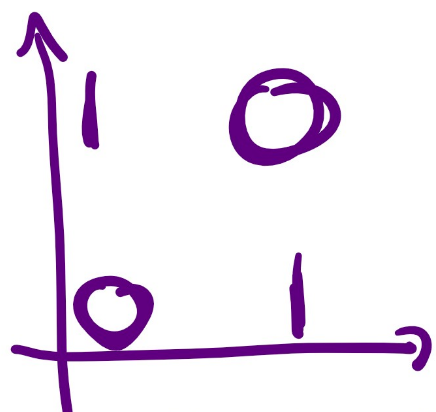
Quiz (Graded)

- Which one of the following is XOR?
- A: $\hat{y} = \mathbb{1}_{\{1x_1+1x_2-1.5 \geqslant 0\}}$
- B: $\hat{y} = \mathbb{1}_{\{1x_1+1x_2-0.5 \geqslant 0\}}$
- C: $\hat{y} = \mathbb{1}_{\{-1x_1+0.5 \geqslant 0\}}$
- D: $\hat{y} = \mathbb{1}_{\{-1x_1-1x_2+0.5 \geqslant 0\}}$
- E: None of the above

NOT

NOT OR

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0



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.

Multi Layer Perceptron

Motivation

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

$$
\begin{aligned}
a &= g\left(w^T x + b\right) \\
a' &= g\left(w'^T a + b'\right) \\
a'' &= g\left(w''^T a' + b''\right) \\
\hat{y} &= \mathbb{1}_{\{a''>0\}}
\end{aligned}
$$

Learning XOR Operator, Part 1

Motivation

- XOR cannot be modelled by a single perceptron.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

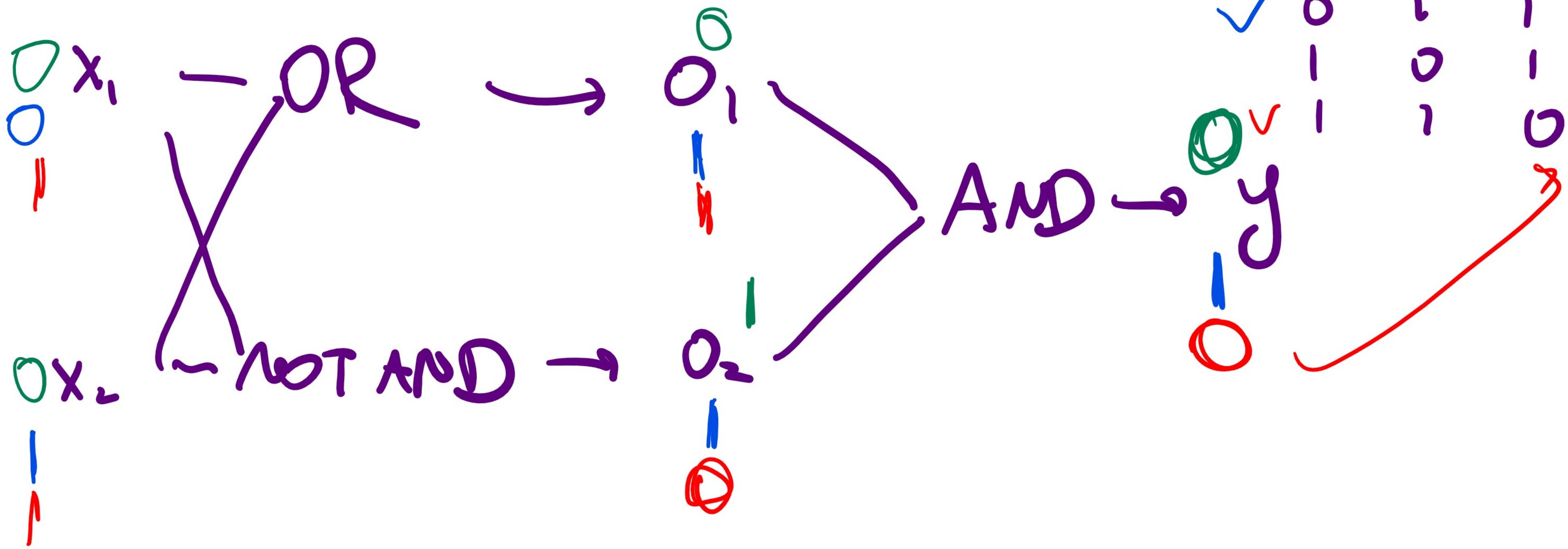
Learning XOR Operator, Part 2

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.

XOR Neural Network Diagram

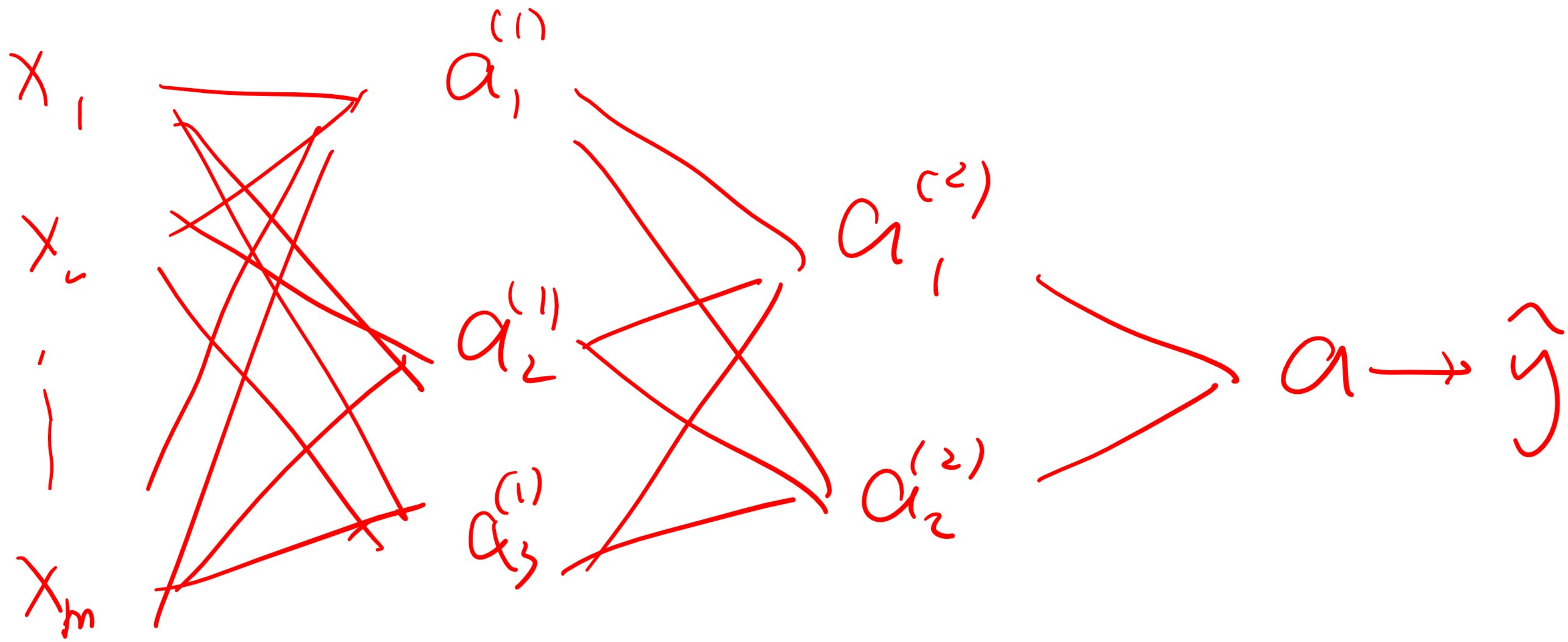
Motivation



x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Multi-Layer Neural Network Diagram

Motivation



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

Theory of Neural Network

Motivation

- In theory:

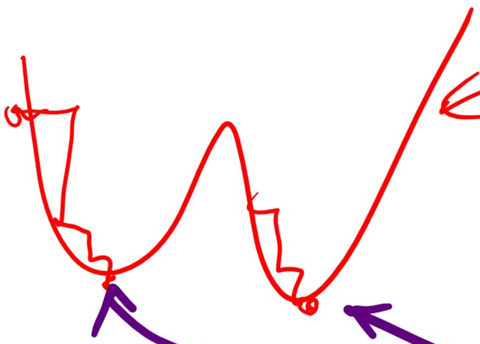
1. 1 Hidden-layer with enough hidden units can represent any continuous function of the inputs with arbitrary accuracy
2. 2 Hidden-layer can represent discontinuous functions

- In practice:

1. AlexNet: 8 layers
2. GoogLeNet: 27 layers
3. ResNet: 152 layers

Gradient Descent

Motivation



logistic

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

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

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

many random initial w, b

Backpropagation

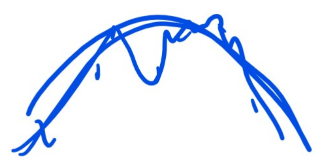
Description

Same as logistic

- 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.

link

$\mathbb{R} \to \mathbb{R}$



$a_i = g(\underbrace{w^T x + b}_{z})$

$a_1 = g(w, x)$

$w^2 x + \sqrt{wx}$

$\nabla_w g(f(w)) = g' \underbrace{\nabla_w f(w)}_{x}$

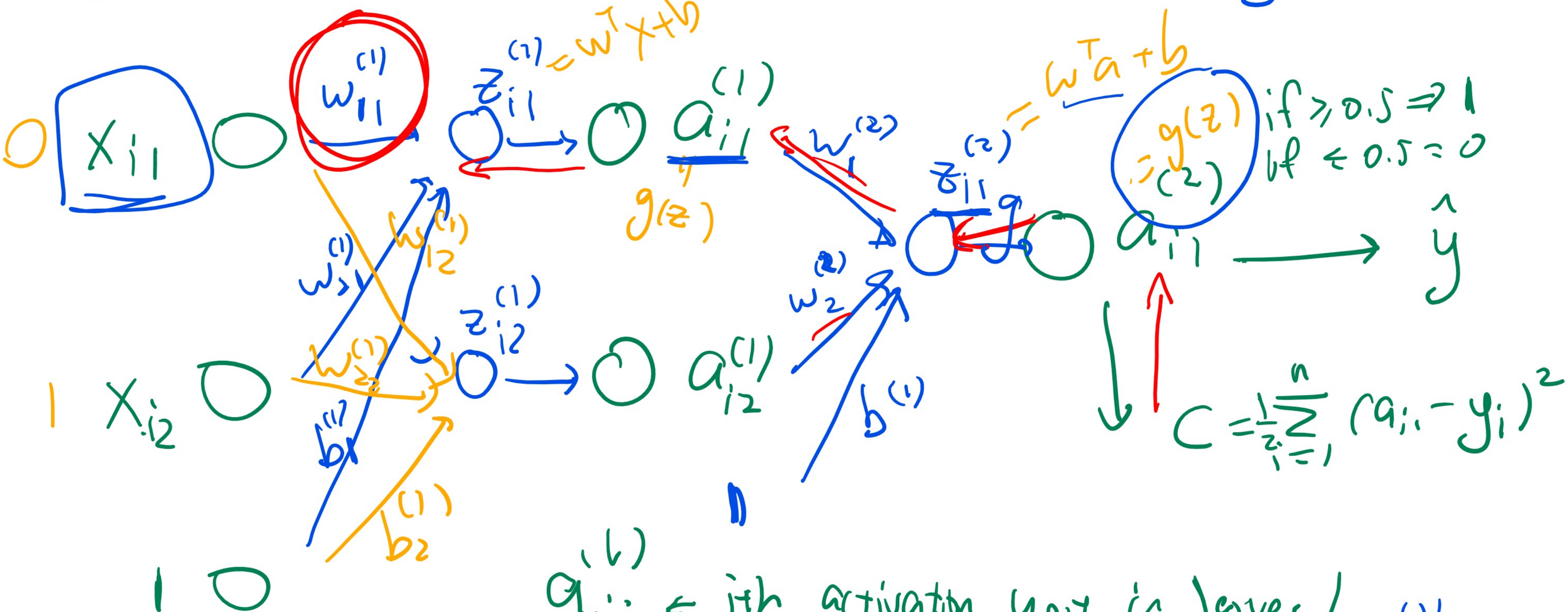
$w_{11}^{(1)} \uparrow$ by 0.001 → how much C changes

Two Layer Neural Network Weights Diagram

Motivation

$w_{j \to j'}^{(l)}$ = from layer $l-1$ unit j to layer l unit j'

logistic activation $g' = g(1-g)$



$a_{ij}^{(l)} \leftarrow$ jth activation unit in layer l for data point i

$a_{i1}^{(1)}(1-a_{i1}^{(1)})$

goal $\dfrac{\partial C}{\partial w_{11}^{(1)}} = \sum_{i=1}^{n} \dfrac{\partial C}{\partial a_{i1}^{(2)}} \cdot \dfrac{\partial a_{i1}^{(2)}}{\partial z_{i1}^{(2)}} \cdot \dfrac{\partial z_{i1}^{(2)}}{\partial a_{i1}^{(1)}} \cdot \dfrac{\partial a_{i1}^{(1)}}{\partial z_{i1}^{(1)}} \cdot \dfrac{\partial z^{(1)}}{\partial w_{11}^{(1)}}$ x_{i1}

$a_{i,1}^{(2)}(1-a_{i,1}^{(2)})$ $w_1^{(2)}$

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 same squared error,

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

Interal Activations

Definition

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

$$a_i = g\left(z_i\right)$$

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

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

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

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

Required Gradients

Definition

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

$$\frac{\partial C}{\partial w_{j'j}^{(1)}}, j = 1, 2, ..., m^{(1)}, j' = 1, 2, ..., m$$

$$\frac{\partial C}{\partial b_{j'}^{(1)}}, j' = 1, 2, ..., m$$

$$\frac{\partial C}{\partial w_{j}^{(2)}}, j = 1, 2, ..., m^{(1)}$$

$$\frac{\partial C}{\partial b^{(2)}}$$

Gradients of Second Layer

Definition

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

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

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

Gradients of First Layer

Definition

- Chain rule twice says,

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

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

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

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

Derivative of Error

Definition

- Compute the derivative of the error function.

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

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

$a_i - y_i$

Derivative of Interal Outputs, Part 1

Definition

- Compute the derivative of the output in the second layer.

$$
\begin{aligned}
a_i &= g\left(z_i\right) \\
\Rightarrow \frac{\partial a_i}{\partial z_i} &= g\left(z_i\right)\left(1 - g\left(z_i\right)\right) = a_i\left(1 - a_i\right) \\
z_i &= \sum_{j=1}^{m^{(1)}} a_{ij}^{(1)} w_j^{(2)} + b^{(2)} \\
\Rightarrow \frac{\partial z_i}{\partial w_j^{(2)}} &= a_{ij}^{(1)}, \frac{\partial z_i}{\partial b^{(2)}} = 1
\end{aligned}
$$

Derivative of Interal Outputs, Part 2

Definition

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

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

$$\Rightarrow \frac{\partial a_{ij}^{(1)}}{\partial z_{ij}^{(1)}} = g\left(z_{ij}^{(1)}\right)\left(1 - g\left(z_{ij}^{(1)}\right)\right) = a_{ij}^{(1)}\left(1 - a_{ij}^{(1)}\right)$$

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

$$\Rightarrow \frac{\partial z_{ij}^{(1)}}{\partial w_{j'j}^{(1)}} = x_{ij'}, \frac{\partial z_{ij}^{(1)}}{\partial b_j^{(1)}} = 1$$

Derivative of Interal Outputs, Part 3

Definition

- Compute the derivative between the outputs.

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

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

Gradient Step, Combined

Definition

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

$$\frac{\partial C}{\partial w_{j'j}^{(1)}} = \sum_{i=1}^{n} \left(y_i - a_i\right) a_i \left(1 - a_i\right) w_j^{(2)} a_{ij}^{(1)} \left(1 - a_{ij}^{(1)}\right) x_{ij'}$$

$$\frac{\partial C}{\partial b_{j'}^{(1)}} = \sum_{i=1}^{n} \left(y_i - a_i\right) a_i \left(1 - a_i\right) w_j^{(2)} a_{ij}^{(1)} \left(1 - a_{ij}^{(1)}\right)$$

$$\frac{\partial C}{\partial w_{j}^{(2)}} = \sum_{i=1}^{n} \left(y_i - a_i\right) a_i \left(1 - a_i\right) a_{ij}^{(1)}$$

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

$a_i - y_i$

Gradient Descent Step

Definition

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

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

$$b^{(2)} \leftarrow b^{(2)} - \alpha \frac{\partial C}{\partial b^{(2)}},$$

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

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

Back Propagation

Quiz (Graded)

8

- 2018 May Final Exam Q4
- Which one best describes backpropagation?
- A: Activation values are propagated from input nodes to output nodes. → feed-forward
- B: Activation values are propagated from output nodes to input nodes.
- C: Do not choose this.
- D: Weights are modified based on values propagated from input nodes to output nodes.
- → E: Weights are modified based on values propagated from output nodes to input nodes. back prop.

Backpropogation, Part 1

Algorithm

- Inputs: instances: $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^n$, 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:

$$w_{j'j}^{(l)}, b_j^{(l)}, j' = 1, 2, ..., m^{(l-1)}, j = 1, 2, ..., m^{(l)}, l = 1, 2, ..., L$$

- Initialize the weights.

$$w_{j'j}^{(l)}, b_j^{(l)} \sim \text{ Unif } [0, 1]$$

Backpropogation, Part 2

Algorithm

- Evaluate the activation functions.

$$a_i = g\left(\sum_{j=1}^{m^{(L-1)}} a_{ij}^{(L-1)} w_j^{(L)} + b^{(L)}\right)$$

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

$$a_{ij}^{(0)} = x_{ij}$$

Backpropogation, Part 3

Algorithm

- Compute the δ to simplify the expression of the gradient.

$a_i - y_i$

$$\delta_i^{(L)} = (y_i - a_i)\, a_i\, (1 - a_i)$$

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

- Compute the gradient using the chain rule.

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

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

Backpropogation, Part 4

Algorithm

- Update the weights and biases using gradient descent.

For $l = 1, 2, ..., L$

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

$$b_{j}^{(l)} \leftarrow b_{j}^{(l)} - \alpha \frac{\partial C}{\partial b_{j}^{(l)}}, j = 1, 2, ..., m^{(l)}$$

- Repeat the process until convergent.

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