CS540 Introduction to Artificial Intelligence
Lecture 3

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

May 17, 2020

Piazza, Socrative

Admin

- Make sure you sign up on Piazza
- Make sure you can log into Socrative with your ID

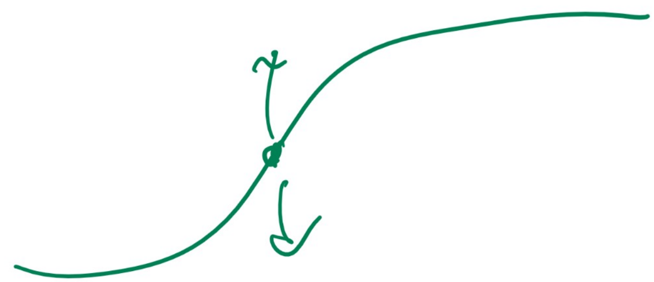
1. Room CS540 for anonymous feedback
2. Room CS540E for section 002 quizzes
3. Room CS540C for section 001 quizzes

- Please report any hacking etc.

Single Layer Perceptron

Motivation

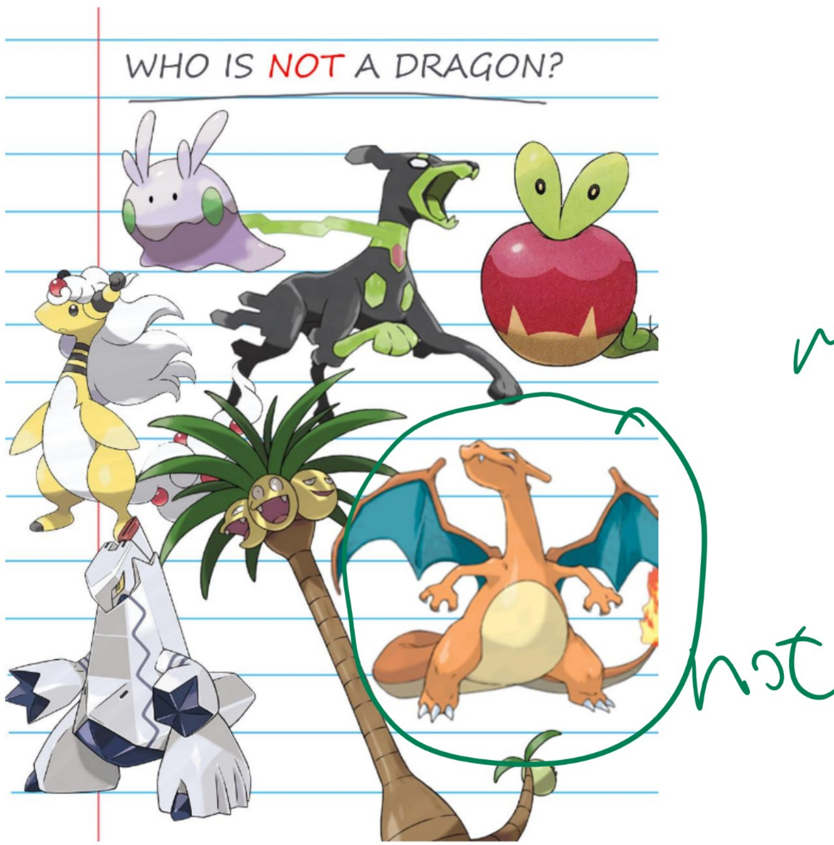
LTU Logistic

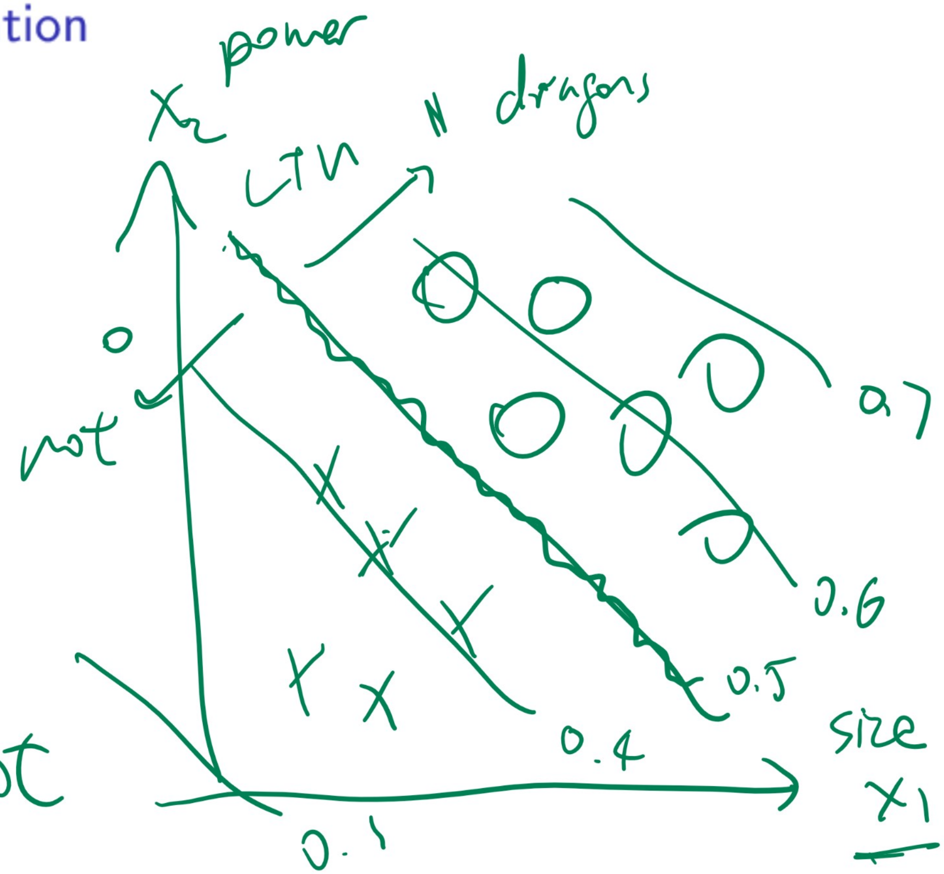


- Perceptrons can only learn linear decision boundaries.
- Many problems have non-linear boundaries.
- One solution is to connect perceptrons to form a network.

Decision Boundary Diagram

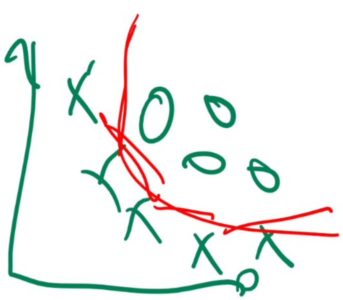
Motivation

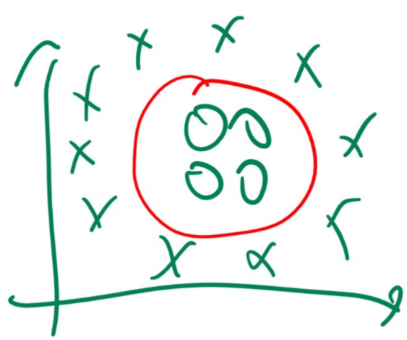




$w^T x + b \geq 0$?

$g(w^T x + b) \geq 0.5$





Multi Layer Perceptron

Motivation

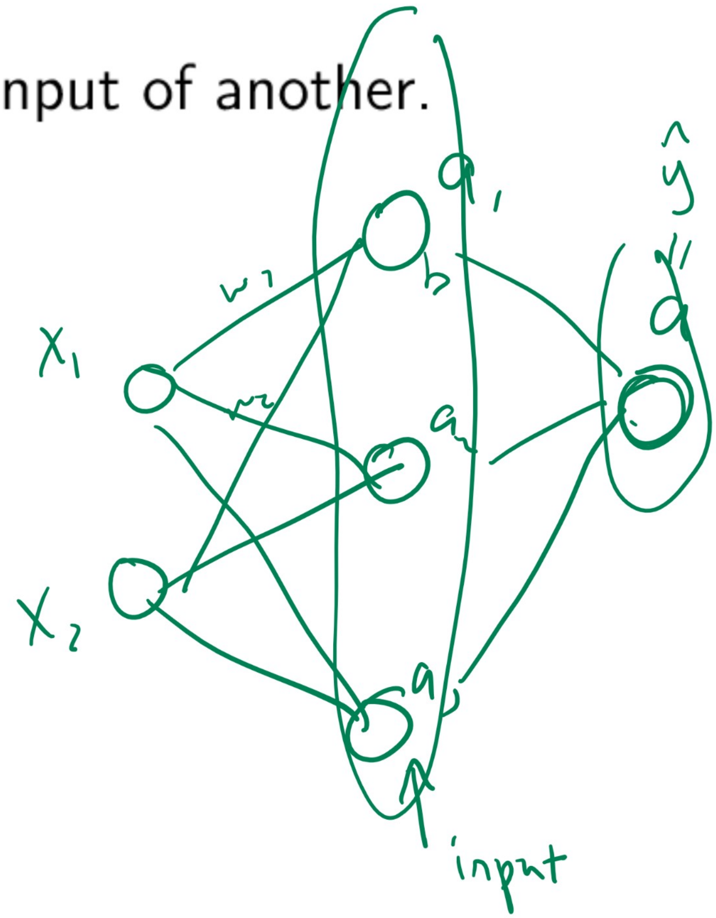
- The output of a perceptron can be the input of another.

$$a = g\left(w^T x + b\right)$$

$$a' = g\left(w'^T a + b'\right)$$

$$a'' = g\left(w''^T a' + b''\right)$$

$$\hat{y} = \mathbb{1}_{\{a''>0\}}$$

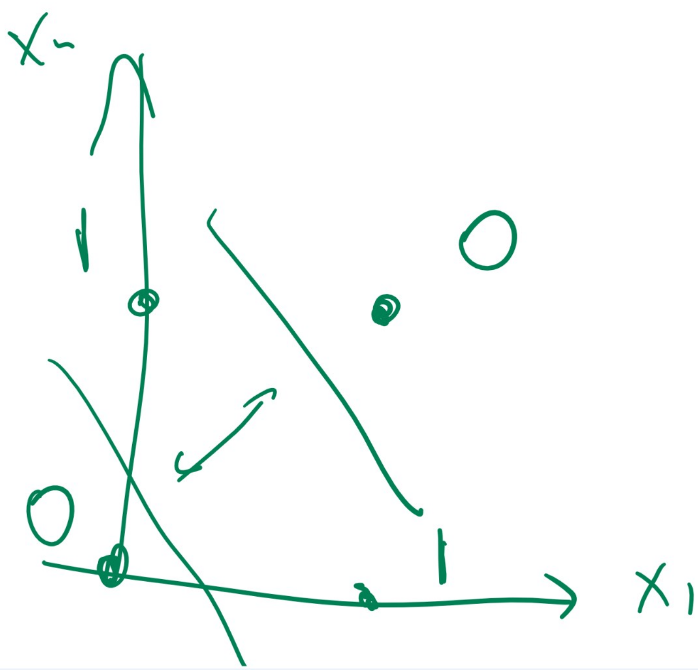


Learning XOR Operator, Part 1

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

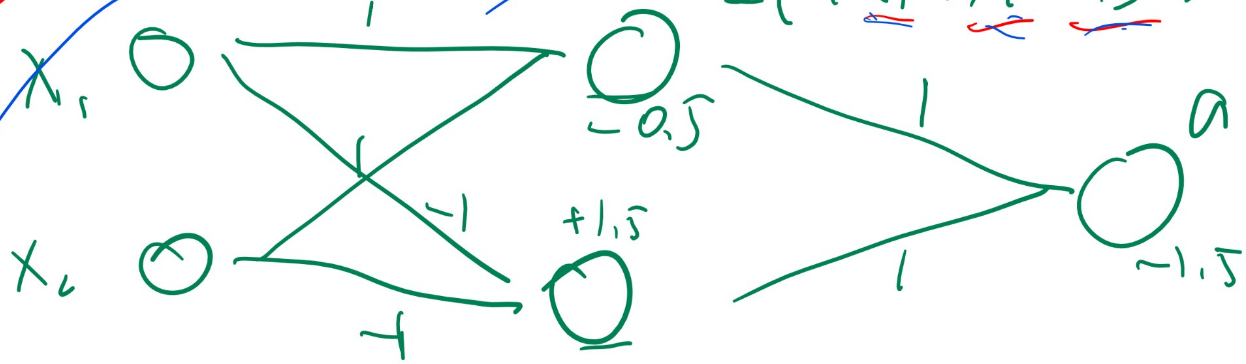
- OR, AND, NOT AND can be modeled by perceptrons.
- If the outputs of OR and NOT AND is used as inputs for AND, then the output of the network will be XOR.

XOR Neural Network Diagram

LTU Motivation

$a_1 = \mathbb{1}\{(x_1 + x_2 - 0.5 \geq 0\}$

$\hat{y}$



$a = \mathbb{1}\{a_1 + a_2 - 1.5 \geq 0\}$

$a_2 = \mathbb{1}\{-x_1 - x_2 + 1.5 \geq 0\}$

x_1	x_2	a_1	a_2	$a = \hat{y}$		y	
0	0	0	1	0	=	0	✓
1	0	1	1	1	=	1	✓
0	1	1	1	1	=	1	✓
1	1	1	0	0	=	0	✓

Neural Network Biology

Motivation

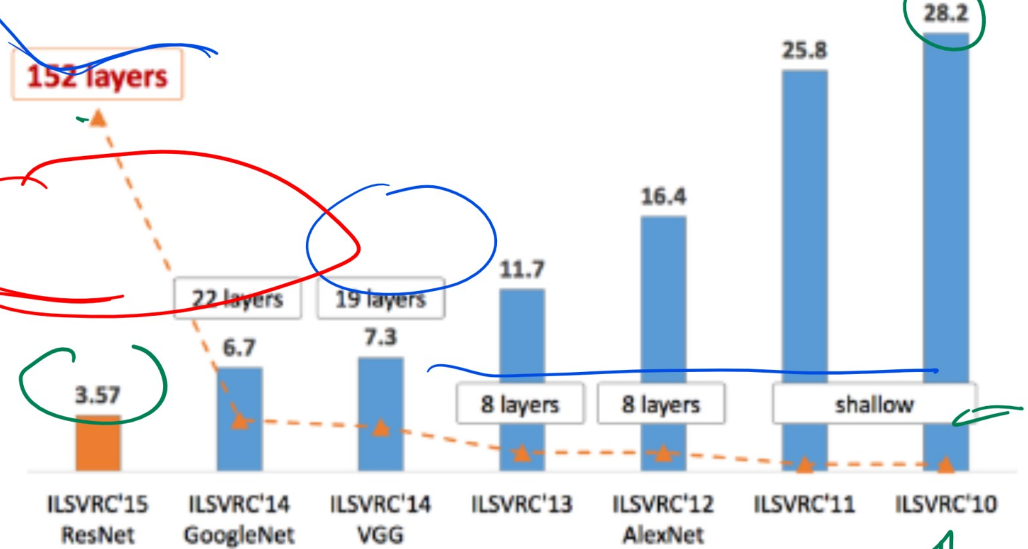
- Human brain: $100,000,000,000$ neurons.
- Each neuron receives input from $1,000$ others.
- An impulse can either increase or decrease the possibility of nerve pulse firing.
- If sufficiently strong, a nerve pulse is generated.
- The pulse forms the input to other neurons.

Theory of Neural Network

Motivation

- In theory:

1. 1 Hidden-layer with continuous function
2. 2 Hidden-layer can r



object recognition

- In practice:

1. AlexNet: 8 layers.
2. GoogLeNet: 27 layers (or 22 + pooling).
3. ResNet: 152 layers.

Neural Network Examples

Motivation

- Classification tasks.
- Aprroximate functions.
- Store functions (after midterm).

medical test

$f(x_1, \ldots, x_m) \to \{0,1\}$

$[0,1]$

NN

Gradient Descent

Motivation

- 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.

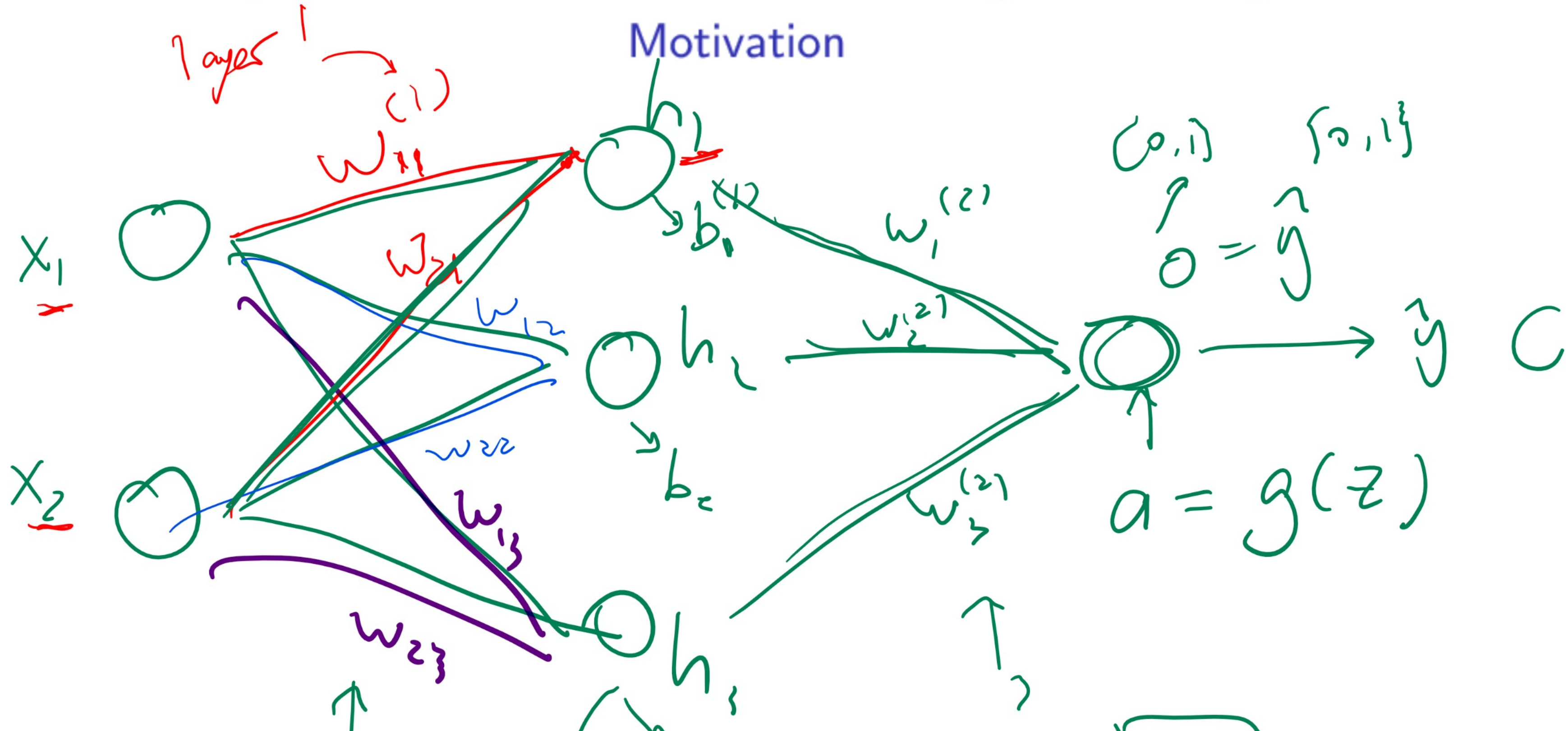
Backpropagation

Description

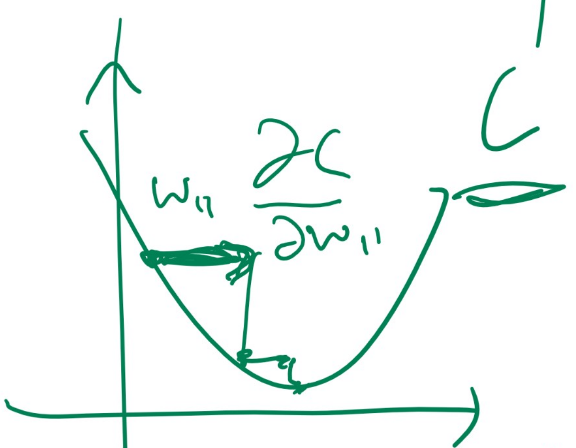
- Initialize random weights.
- (Feedforward Step) Evaluate the activation functions.
- (Backpropagation Step) Compute the gradient of the cost function with respect to each weight and bias using the chain rule.
- Update the weights and biases using gradient descent.
- Repeat until convergent.

Two Layer Neural Network Weights Diagram 1

Motivation



$$\boxed{a_3} = g\Big(\underbrace{w_{13}x_1 + w_{23}x_2 + b_3}_{z_3}\Big)$$

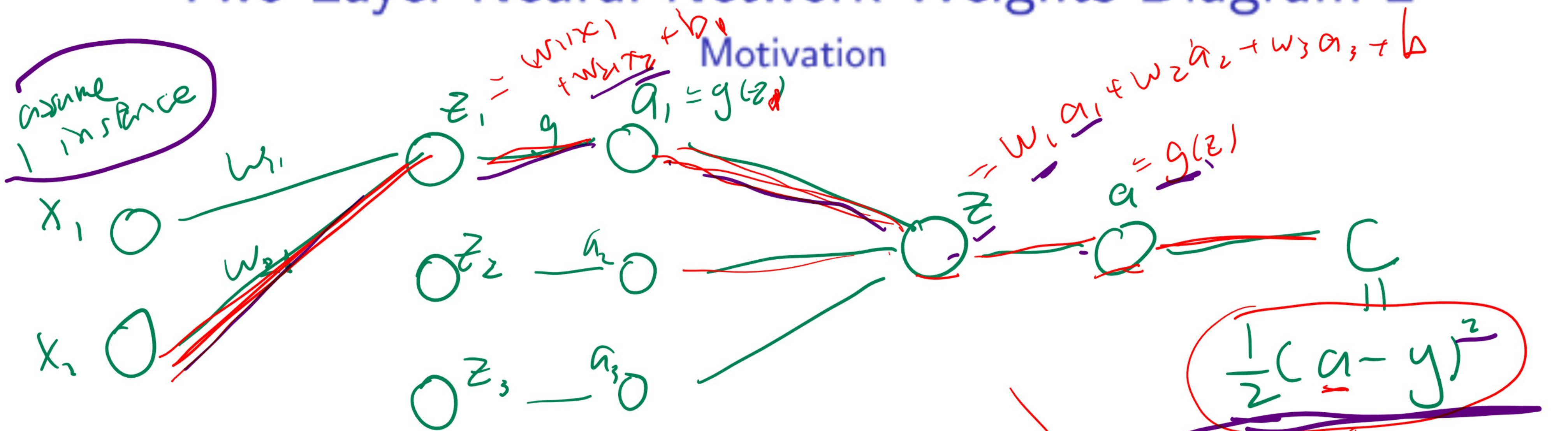


$$C = \frac{1}{2}\sum(\hat{y} - y)^2$$

$$w_{11} = w_{11} - \alpha \boxed{\frac{\partial C}{\partial w_{11}}}$$

Two Layer Neural Network Weights Diagram 2

Motivation



$f(g(x))' = f'(g) \cdot g'(x)$

$$\boxed{\frac{\partial C}{\partial w_{21}}} = \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial w_{21}}$$

$$= (a-y) \cdot \overbrace{a(1-a)}\, w_1\, a_1(1-a_1)\, x_2$$

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

$\to$ logistic

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

$$\boxed{a_i = g\left(z_i\right)}$$

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

\# hidden units in layer 1

instance i

hidden unit j

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

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

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

layer 0 = input x

weight from j' in prev layer j in next layer

Notations

Definition

- $a_{ij}^{(l)}$ is the hidden unit activation of instance i in layer l, unit j
- $z_{ij}^{(l)}$ is the linear part of instance i in layer l, unit j
- $w_{j'j}^{(l)}$ is the weights between layers $l-1$ and l, from unit j' in layer $l-1$ to unit j in layer l.
- $b_{j}^{(l)}$ is the bias for layer l unit j.
- $m^{(l)}$ is the number of units in layer l.
- Superscript l is omitted for the last layer.

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.

$$
\begin{cases}
\dfrac{\partial C}{\partial w_j^{(2)}} = \displaystyle\sum_{i=1}^{n} \dfrac{\partial C}{\partial a_i} \dfrac{\partial a_i}{\partial z_i} \dfrac{\partial z_i}{\partial w_j^{(2)}}, j = 1, 2, ..., m^{(1)} \\
\dfrac{\partial C}{\partial b^{(2)}} = \displaystyle\sum_{i=1}^{n} \dfrac{\partial C}{\partial a_i} \dfrac{\partial a_i}{\partial z_i} \dfrac{\partial z_i}{\partial b^{(2)}}
\end{cases}
$$

Gradients of First Layer

Definition

- Chain rule twice says,

$$\frac{\partial C}{\partial w^{(1)}_{j'j}} = \sum_{i=1}^{n} \frac{\partial C}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial a^{(1)}_{ij}} \frac{\partial a^{(1)}_{ij}}{\partial z^{(1)}_{ij}} \frac{\partial z^{(1)}_{ij}}{\partial w^{(1)}_{j'j}}$$

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

$$\frac{\partial C}{\partial b^{(1)}_{j}} = \sum_{i=1}^{n} \frac{\partial C}{\partial a_i} \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial a^{(1)}_{ij}} \frac{\partial a^{(1)}_{ij}}{\partial z^{(1)}_{ij}} \frac{\partial z^{(1)}_{ij}}{\partial b^{(1)}_{j}}$$

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

Derivative of Error

Definition

- Compute the derivative of the error function.

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

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

Derivative of Interal Outputs, Part 1

Definition

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

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

$$\Rightarrow \dfrac{\partial a_i}{\partial z_i} = g\left(z_i\right)\left(1 - g\left(z_i\right)\right) = a_i\left(1 - a_i\right)$$

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

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

Derivative of Interal Outputs, Part 2

Definition

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

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

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

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

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

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!)

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

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

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

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

Gradient Descent Step

Definition

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

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

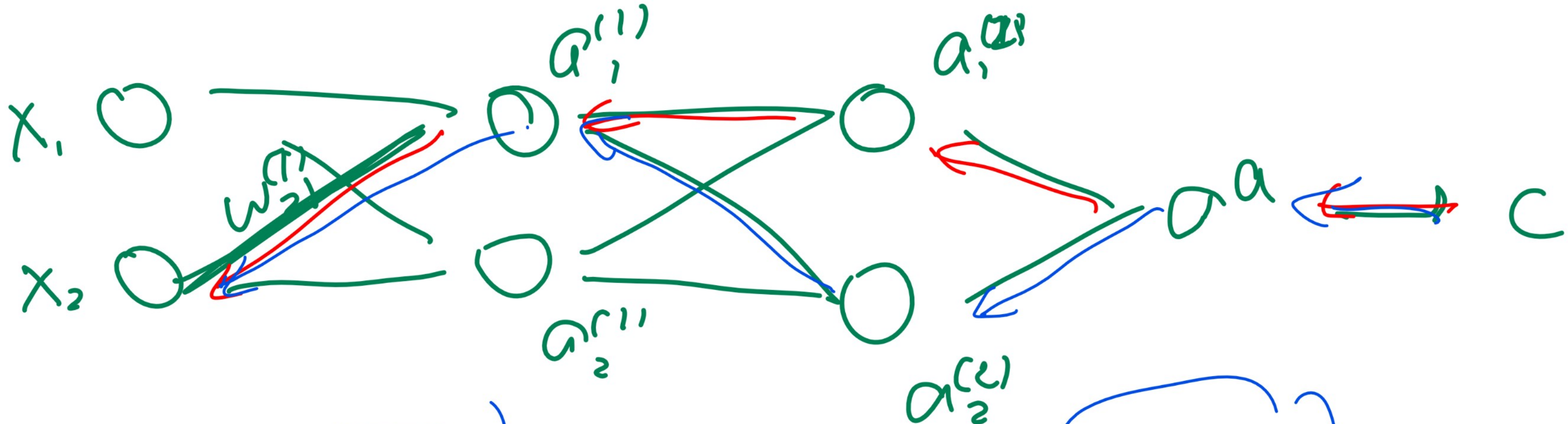
$$b^{(2)} \leftarrow b^{(2)} - \alpha \frac{\partial C}{\partial b^{(2)}},$$

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

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

Back Propagation

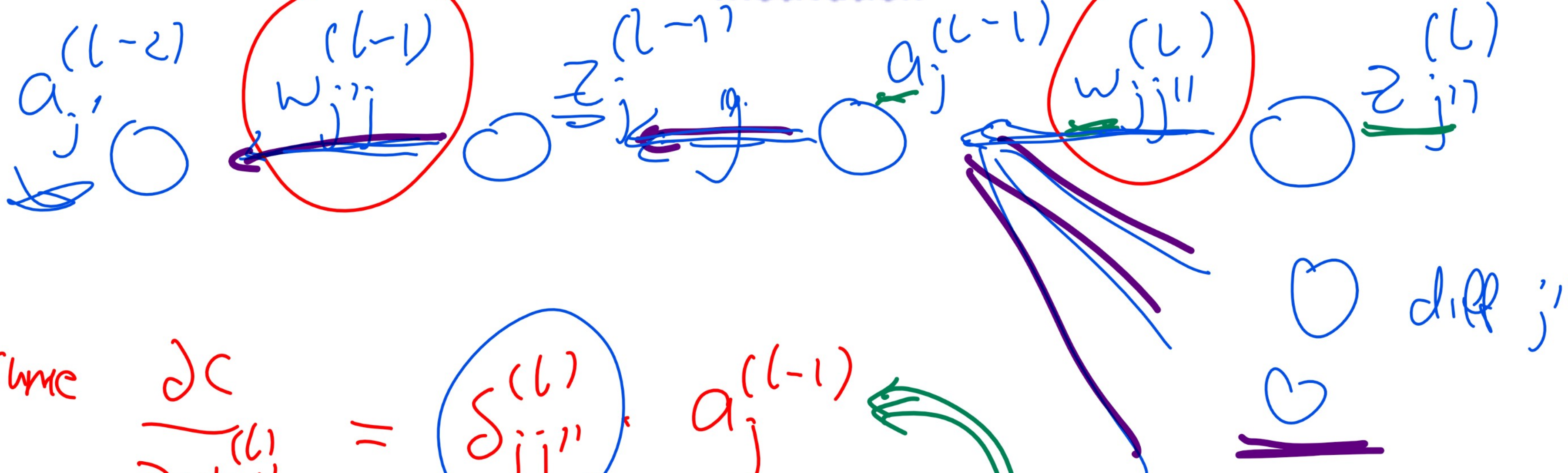
Quiz



$$\frac{\partial C}{\partial w_{21}^{(1)}} = \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial a_1^{(2)}} \cdot \frac{\partial a_1^{(2)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial w_{21}^{(1)}} + \frac{\partial C}{\partial a} \cdot \frac{\partial a}{\partial a_2^{(2)}} \cdot \frac{\partial a_2^{(2)}}{\partial a_1^{(1)}} \cdot \frac{\partial a_1^{(1)}}{\partial w_{21}^{(1)}}$$

Three Layer Neural Network Weights Diagram 1

Motivation



assume $\dfrac{\partial C}{\partial w^{(l)}_{jj''}} = \delta^{(l)}_{jj''} \cdot a^{(l-1)}_{j}$

$$= \boxed{\frac{\partial C}{\partial z^{(l)}_{j''}}} \cdot \frac{\partial z^{(l)}_{j''}}{\partial w^{(l)}_{jj''}}$$

want $\dfrac{\partial C}{\partial w^{(l-1)}_{j'j}} = \displaystyle\sum_{j''=1}^{m^{(l)}} \frac{\partial C}{\partial z^{(l)}_{j''}} \cdot \frac{\partial z^{(l)}_{j''}}{\partial a^{(l-1)}_{j}} \cdot \frac{\partial a^{(l-1)}_{j}}{\partial z^{(l-1)}_{j}} \cdot \frac{\partial z^{(l-1)}_{j}}{\partial w^{(l-1)}_{j'j}}$

$$= \boxed{\sum_{j''=1}^{m^{(l)}} \delta^{(l)}_{jj''} \cdot w^{(l)}_{jj''} \cdot a^{(l-1)}_{j} (1 - a^{(l-1)}_{j})} \; \boxed{a^{(l-2)}_{j'}}$$

$$= \delta^{(l-1)}_{j'j} \; \boxed{a^{(l-2)}_{j'}}$$

Three Layer Neural Network Weights Diagram 2

Motivation

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.

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

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

- 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.

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