

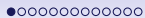
# CS540 Introduction to Artificial Intelligence

## Lecture 3

Young Wu

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

July 1, 2022



# Two-thirds of the Average Game

## Quiz

# AND Operator Data

## Quiz

# Learning AND Operator

## Quiz

# OR Operator Data

## Quiz

# Learning OR Operator

## Quiz

# Learning XOR Operator

## Quiz

# Learning XOR Operator Network

## Quiz



# 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.

# Decision Boundary Diagram

## Motivation

# Multi-Layer Perceptron

## Motivation

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

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

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

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

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

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

# 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.

# 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

# Two-Layer Neural Network Weights Diagram 2

## Motivation

# Two-Layer Neural Network Weights Diagram 3

## Motivation

# Gradient Step, Combined

## Definition

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

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

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

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

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

# Gradient Descent Step

## Definition

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

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

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

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

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

# Learning Logical Operators, XOR

## Quiz

# Learning Logical Operators, XOR, Diagram

## Quiz





# Three-Layer Neural Network Weights Diagram

## Motivation