Artificial Intelligence

Taro Sekiyama

National Institute of Informatics (NII)
sekiyama@nii.ac.jp

Artificial neural networks (ANNs)

- A machine learning model inspired by “brains”
  - Comprising a bunch of “neurons” connected so that they interact with each other
- Main application is supervised learning
  - For both classification and regression
- Recently used for unsupervised and reinforcement learning as well

Pros and Cons

- Advantages
  - **Great success in many application areas!** Computer vision, natural language/audio processing, game AI (Go, Shogi, video game), ...
  - Able to learn a **nonlinear** pattern of data points
- Disadvantages
  - Needs huge datasets
  - Computationally costly
  - No (established) guides to improve the performance yet

Types of NNs

- Feedforward neural networks (FNNs)
  - Simple but powerful architectures
- Convolutional neural networks (CNNs)
  - Specialized for computer vision
- Recurrent neural networks (RNNs)
  - Able to handle sequential data with variable length
  - Used for, e.g., natural language/audio
- … and others!

Types of NNs

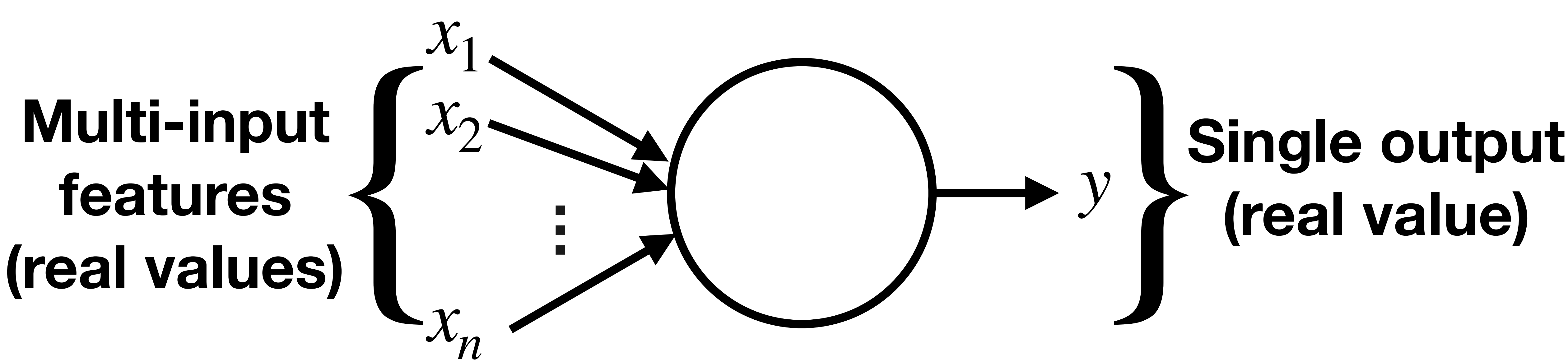
- ***Feedforward neural networks (FNNs)***
  - Simple but powerful architectures
- Convolutional neural networks (CNNs)
  - Specialized for computer vision
- Recurrent neural networks (RNNs)
  - Able to handle sequential data with variable length
  - Used for, e.g., natural language/audio
- … and others!

Outline

- Quick overview of neural networks
- Calculation by neurons
- Activation function with examples
- Cost functions
- Optimization by gradient descent

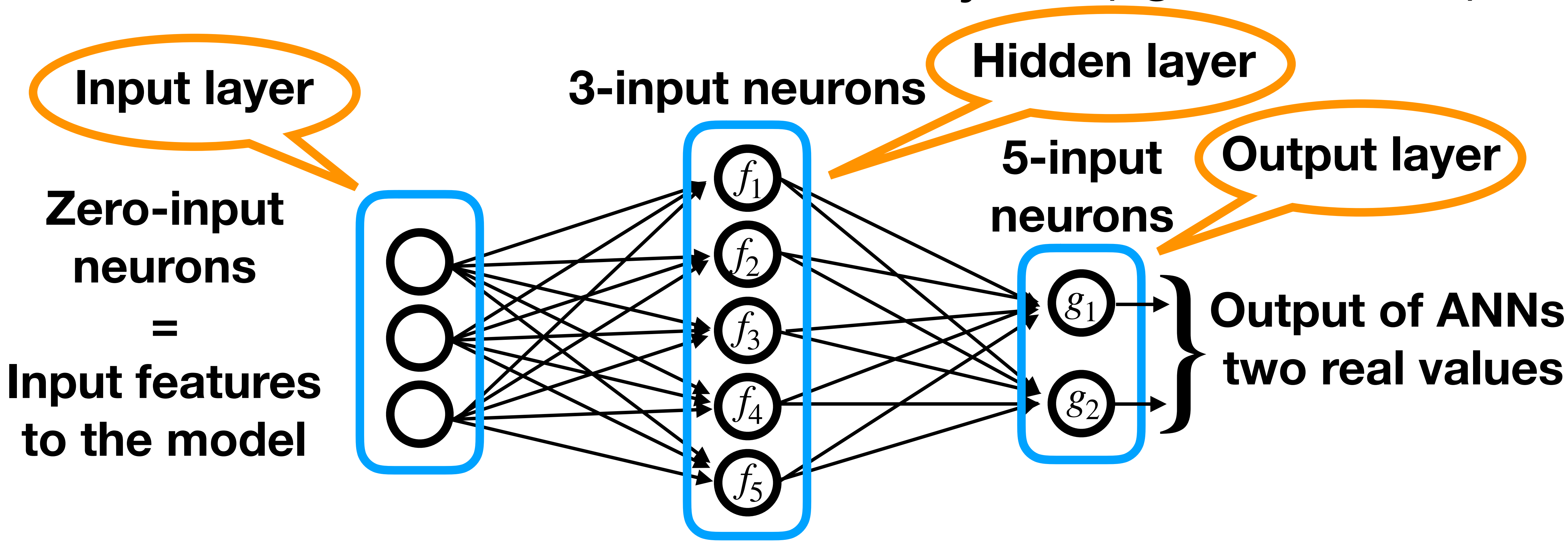
Neuron

- Basic component of neural networks
- Computational unit (a.k.a. perceptron) that
  - Takes multiple real-valued inputs (features)
  - Outputs a single real value



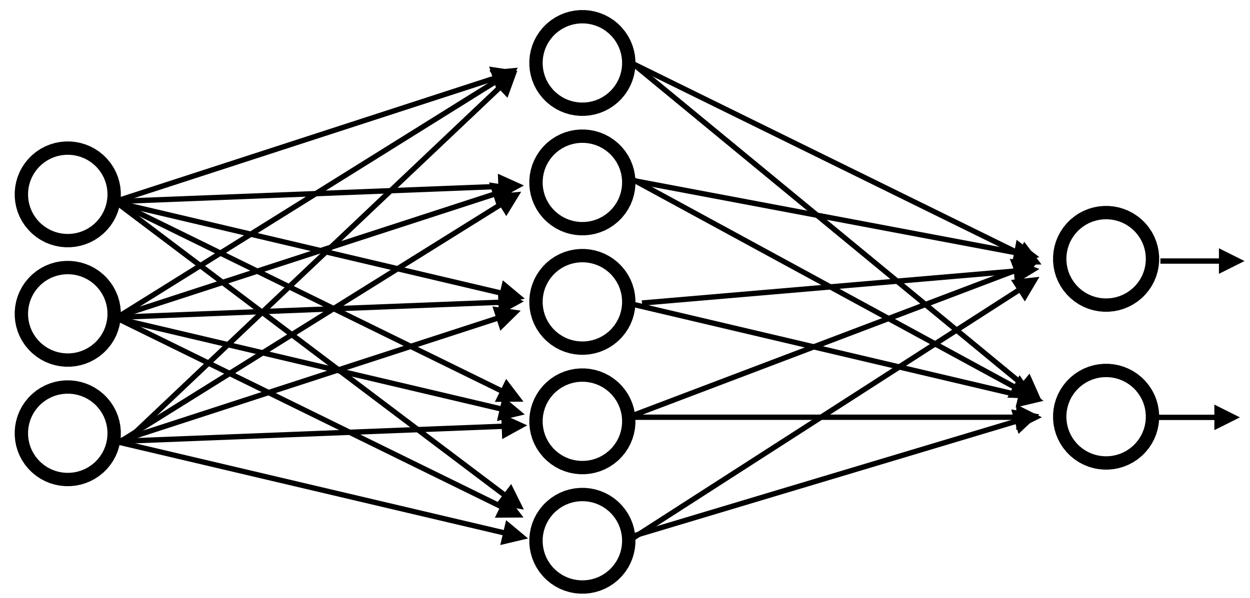
Feedforward neural networks (FNNs)

- Neural networks involve no cycle ("go-forward")

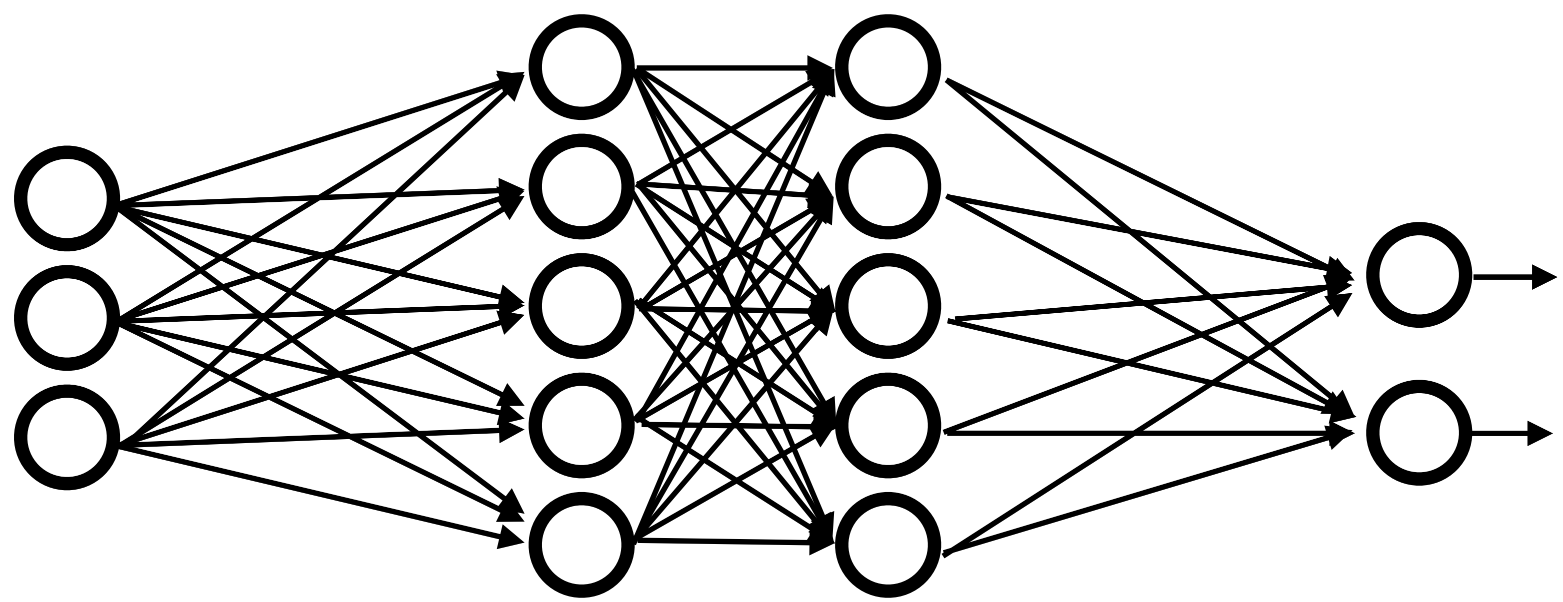


$$X_1 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad X_2 = \begin{bmatrix} f_1(X) \\ f_2(X) \\ f_3(X) \\ f_4(X) \\ f_5(X) \end{bmatrix} \qquad X_3 = \begin{bmatrix} g_1(X_2) \\ g_2(X_2) \end{bmatrix}$$

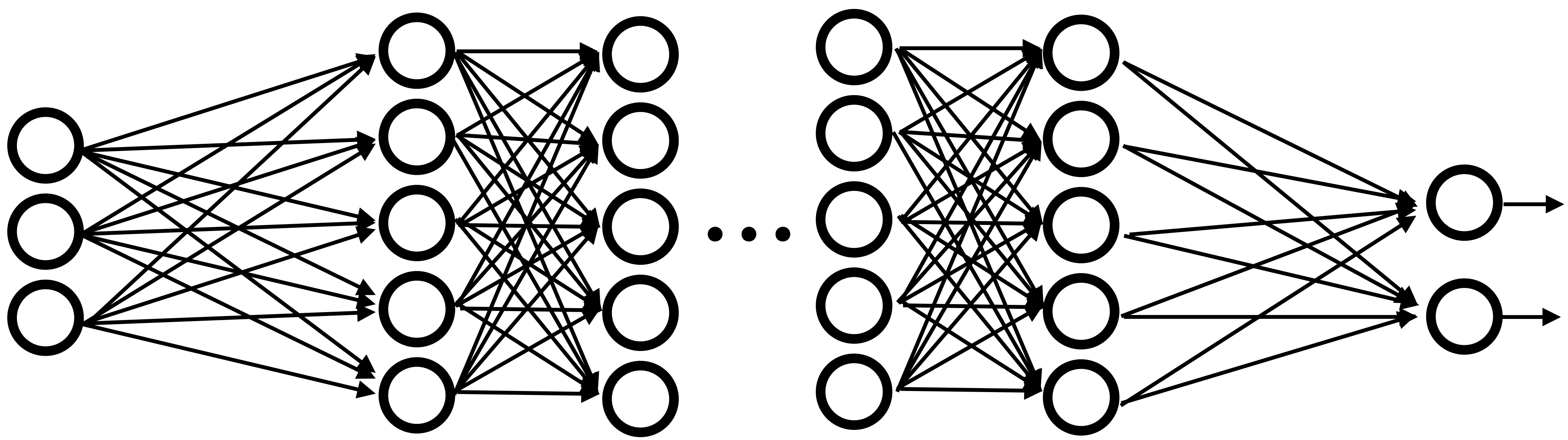
From shallow to deep



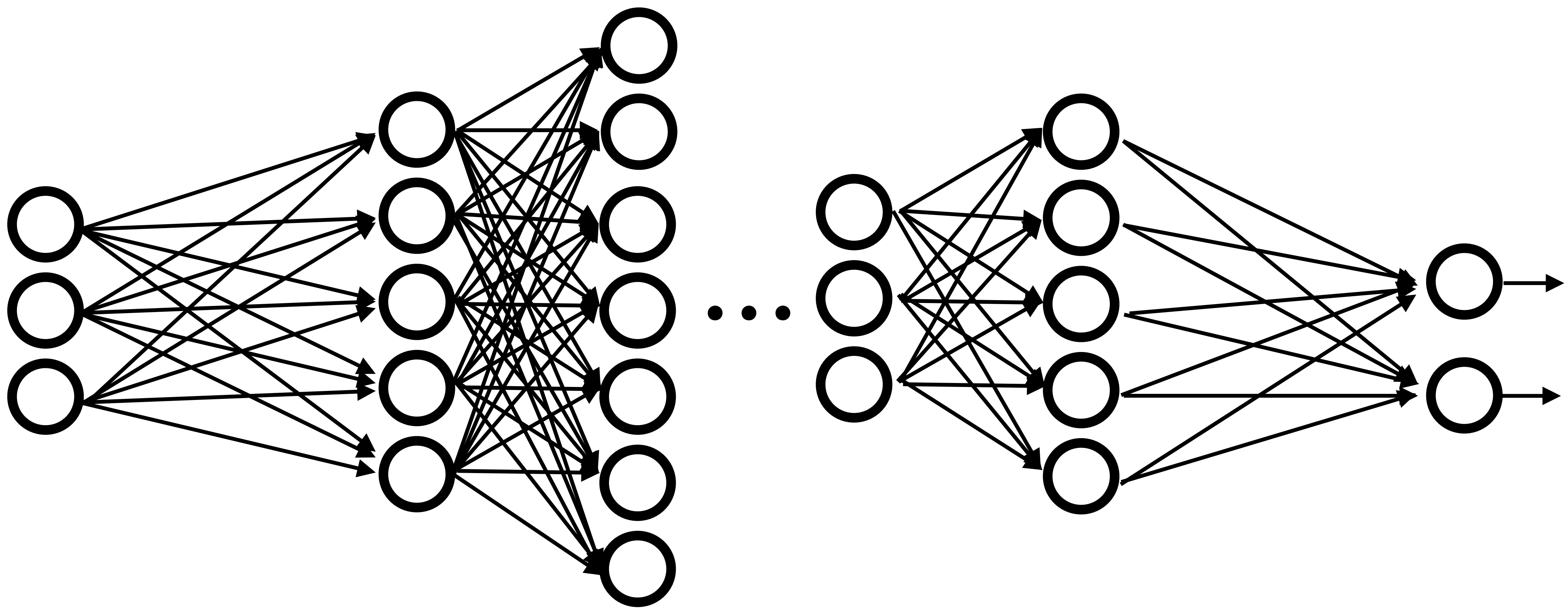
From shallow to deep



From shallow to deep



From shallow to deep



- ***Deep*** neural networks are deep especially when they have multiple hidden layers

Shallow versus deep

- Shallow NNs are surprisingly expressive
  - ***The Universal Approximator Theorem:***
    For many continuous functions f, there is an FNN with a single hidden layer s.t. it can approximate f to a given precision

Shallow versus deep

- Shallow NNs are surprisingly expressive
  - ***The Universal Approximator Theorem:***
    For many continuous functions f, there is an FNN with a single hidden layer s.t. it can approximate f to a given precision
- However, deep NNs work more well in practice

Shallow versus deep

- Shallow NNs are surprisingly expressive
  - ***The Universal Approximator Theorem:***
    For many continuous functions f, there is
    an FNN with a single hidden layer s.t.
    it can approximate f to a given precision
- However, deep NNs work more well in practice
- But very deep NNs (~10^2 layers) have
  the so-called ***vanishing gradient problem***
  - Fitting models to datasets becomes harder

Outline

- Quick overview of neural networks
- **Calculation by neurons**
- Activation function with examples
- Cost functions
- Optimization by gradient descent

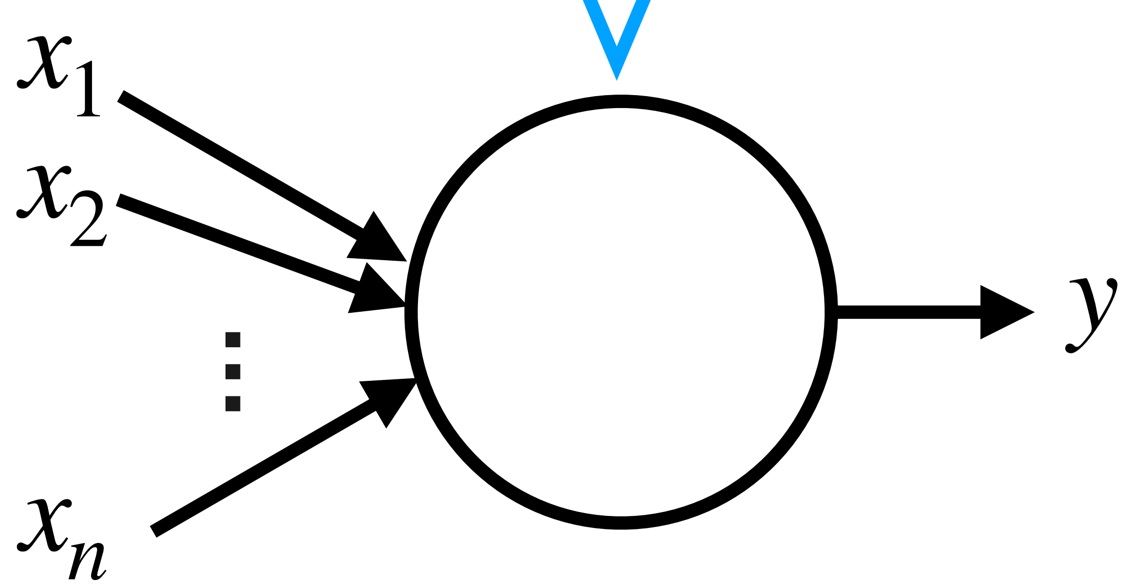
Neuron, detailed

Neuron = linear regression + activation function

Activation functions
$\in \mathbb{R} \rightarrow \mathbb{R}$

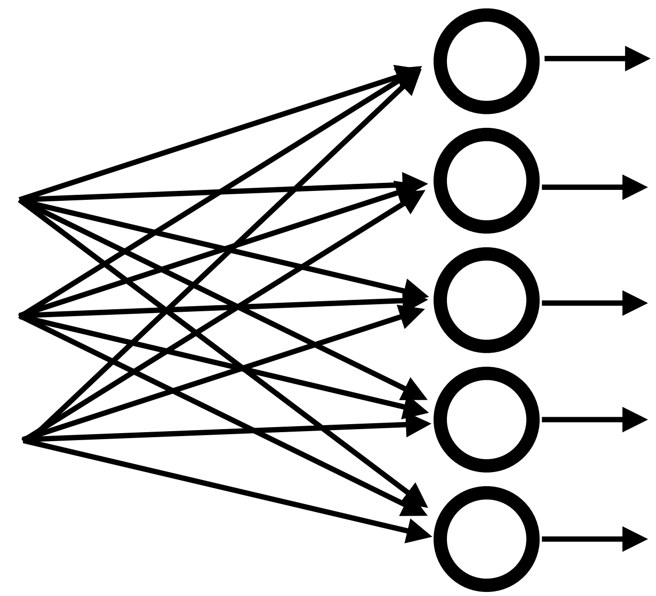
$$y = \sigma\left(\sum_{i}^{n} x_i w_i + b\right)$$

Parameters tuned by training



Calculation on layers

$$X_1 = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



$$\begin{bmatrix} \sigma(\sum_i x_i w_{1i} + b_1) \\ \sigma(\sum_i x_i w_{2i} + b_2) \\ \sigma(\sum_i x_i w_{3i} + b_3) \\ \sigma(\sum_i x_i w_{4i} + b_4) \\ \sigma(\sum_i x_i w_{5i} + b_5) \end{bmatrix} = X_2$$

Calculation on layers

$$\sigma(X_1^T W + B) = X_2$$

$$W = \begin{bmatrix} w_{11} & \cdots & w_{51} \\ w_{12} & \cdots & w_{52} \\ w_{13} & \cdots & w_{53} \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 \\ \vdots \\ b_5 \end{bmatrix}$$

$$\sigma(\begin{bmatrix} x_1 \\ \vdots \\ x_5 \end{bmatrix}) = \begin{bmatrix} \sigma(x_1) \\ \vdots \\ \sigma(x_5) \end{bmatrix}$$

for $\sigma \in \mathbb{R} \to \mathbb{R}$

Outline

- Quick overview of neural networks
- Calculation by neurons
- **Activation function with examples**
- Cost functions
- Optimization by gradient descent

Activation function σ

- ***Nonlinear*** activation functions introduce nonlinearity to neural networks
- Also useful to normalize outputs of NNs

Example: identity

Identity function $I \in \mathbb{R} \to \mathbb{R}$

$$I(x) = x$$

- Neurons with I just performs linear regression
- Used in both hidden and output layers

Example: sigmoid

$\{x \in \mathbb{R} \mid 0 < x < 1\}$

Sigmoid function $S \in \mathbb{R} \rightarrow (0,1)$

$$S(x) = \frac{e^x}{e^x + 1}$$

- Normalizing function
- Used in both hidden and output layers

Sigmoid for binary classification

- Setting: Input $X \in \mathbb{R}^n$, output $y \in \{0,1\}$
- NN $F \in \mathbb{R}^n \rightarrow [0,1]$ if its output layer uses sigmoid function $S \in \mathbb{R} \rightarrow (0,1)$
- Then, y can be predicted by:

$$y = \begin{cases} 0 & (\text{if } f(X) < 0.5) \\ 1 & (\text{if } f(X) \geq 0.5) \end{cases}$$

Example: TanH

TanH function $T \in \mathbb{R} \to (-1,1)$

$$T(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Used in both hidden and output layers

TanH for regression

- Setting: input $X \in \mathbb{R}^n$, output $y \in (-N, N)$
- NN $F \in \mathbb{R}^n \rightarrow (-N, N)$ if its output layer uses the activation function $f \in \mathbb{R} \rightarrow (-N, N)$ as:

$$f(x) = N \times T(x)$$

Example: Rectifier

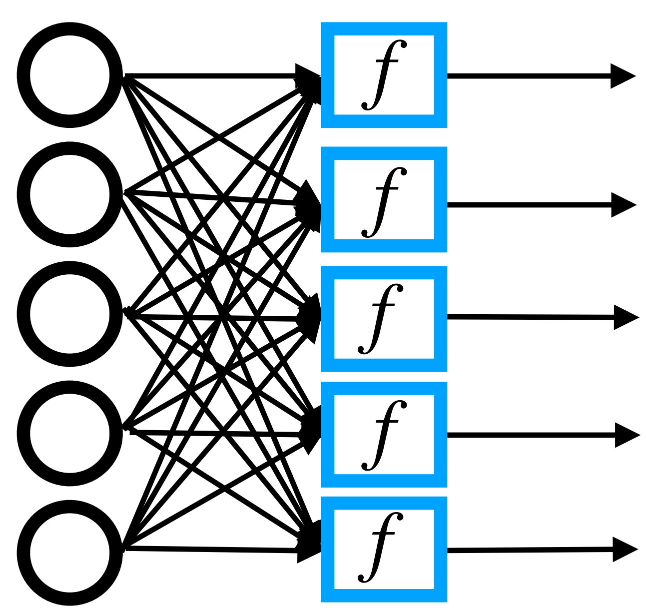
Rectifier (a.k.a. ***ramp function***) $R \in \mathbb{R} \rightarrow \mathbb{R}^{\geq 0}$

$$R(x) = \max(0,x)$$

- Used mainly in hidden layers
- Advantage
  - Relaxing the **vanishing gradient problem** in deep NNs
- Disadvantage
  - Not differentiable at zero
    - In theory, training of NNs requires neurons to be differentiable
    - In practice, setting the derivative at zero to zero works

Multi-input activation function

- Activation functions can be extended to take multiple inputs
- They are functions $f \in \mathbb{R}^n \rightarrow \mathbb{R}$ where m is the # of neurons in the previous layer



Example: softmax

Softmax function $S^i \in \mathbb{R}^n \to (0,1)$ $(i \in [1,n])$

$$S^i(X) = \frac{e^{X[i]}}{\sum_j^n e^{X[j]}}$$

($X[k]$ is the scalar value indexed by k in vector $X \in \mathbb{R}^n$)

- The probability of each value in X

$$\sum^{|X|} S^i(X) = 1$$

- Used in output layers

Softmax for multi-class classification

- Setting: input $X \in \mathbb{R}^n$,
  prediction output $Y \in \{1, \cdots, m\}$
- NN $F \in \mathbb{R}^n \to (0,1)^m$ if its output layer uses **softmax function** $S^i \in \mathbb{R}^m \to (0,1)$
- Then, y can be predicted by:

$$y = \arg\max_i F(X)[i]$$

Outline

- Quick overview of neural networks
- Calculation by neurons
- Activation function with examples
- **Cost functions**
- Optimization by gradient descent

Cost functions

- Mathematical means to describe an amount of differences between a model and expectations

$$C(\bar{\theta}) = \frac{1}{|\mathbf{T}|} \sum_{(X,Y)\in\mathbf{T}} L(F_{\bar{\theta}}(X), Y)$$

  - $\mathbf{T}$: training dataset
  - $F_{\bar{\theta}} \in \mathbb{R}^n \to \mathbb{R}^m$: neural networks with parameters $\bar{\theta}$ (weights and biases)
  - $L \in \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$: ***loss*** function

Loss functions

- Mean squared error: mainly for regression

$$L(Z, Y) = \|Z - Y\|^2$$

😁 Applicable to any NNs

- Cross entropy: mainly for classification

$$L(Z, Y) = Z \log Y + Y \log Z$$

😁 Working more well

😩 Restricting the codomain of applicable NNs

$$F_{\bar{\theta}} \in \mathbb{R}^n \to (0, \infty)^m$$

Outline

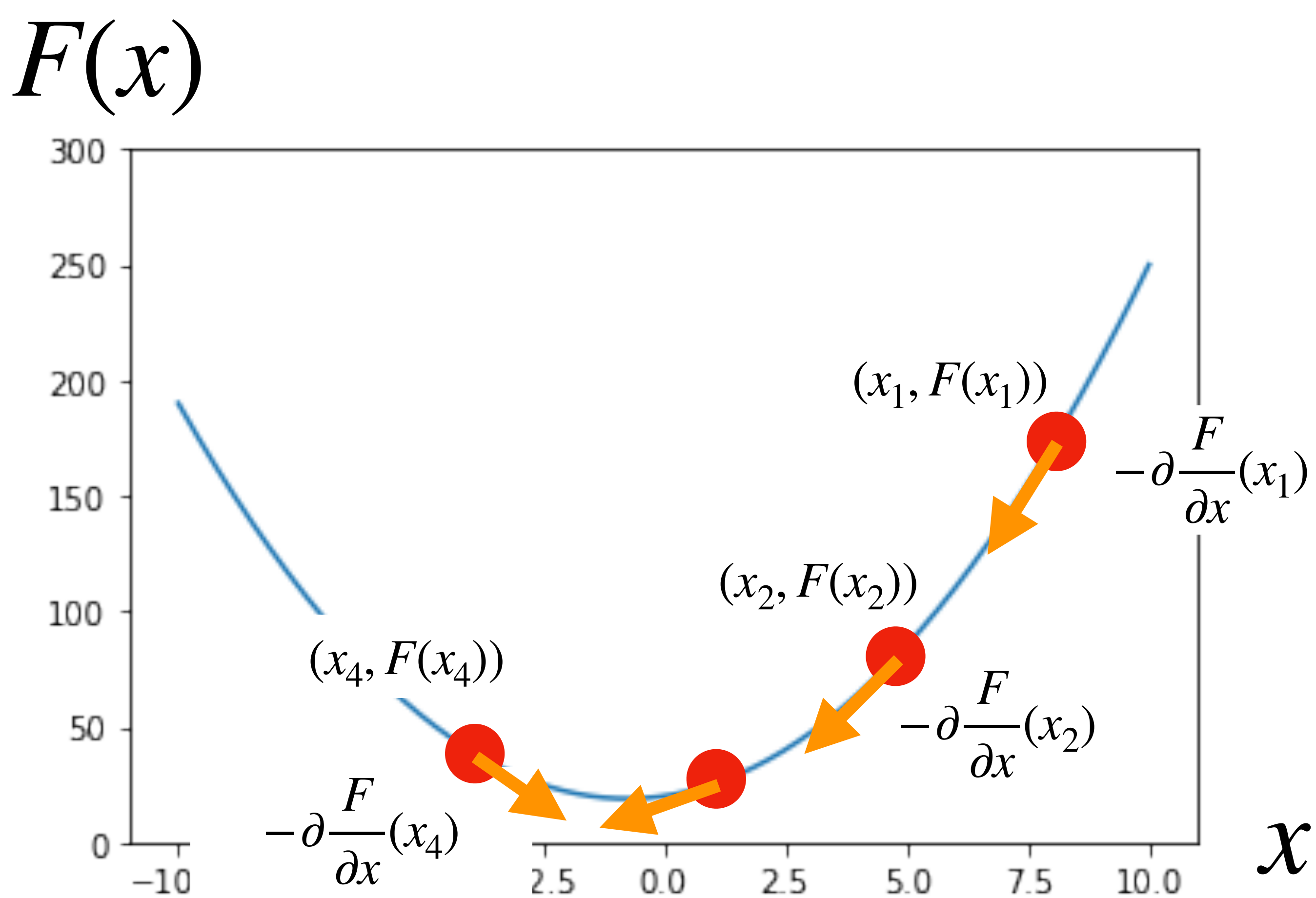
- Quick overview of neural networks
- Calculation by neurons
- Activation function with examples
- Cost functions
- **Optimization by gradient descent**

Optimization

- **Goal:** finding parameters θ such that $C(\theta)$ is minimized
- **Problem:** NNs are nonlinear functions, for which it is hard to find optimal parameters
- **Solution:** Approximation by ***gradient descent***

Gradient descent

- Approaching to optimal parameters by using partial derivatives



Cain rule

$$\partial \frac{f(g(x))}{\partial x} = \partial \frac{f(g(x))}{g(x)} \cdot \partial \frac{g(x)}{\partial x}$$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \partial \frac{C(\bar{\theta})}{\partial \bar{\theta}[i]}$$

$$\partial \frac{C(\bar{\theta})}{\partial \bar{\theta}[i]} = \frac{1}{|T|} \sum_{(X,Y) \in \mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial \bar{\theta}[i]}$$

$$\partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial \bar{\theta}[i]} = \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot \partial \frac{\partial F_{\bar{\theta}}(X)}{\bar{\theta}[i]}$$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot \partial \frac{\partial F_{\bar{\theta}}(X)}{\bar{\theta}[i]}$$

- Suppose
  - σ^l, W^l, b^l are components at the l-the last layer
  - Z^l is the input to σ^l (i.e., $W^l \cdot Z^{l-1} + b^l$ if l is not the first layer)

$$F_{\bar{\theta}}(X) = \sigma^1(Z^1)$$

$$\partial \frac{\partial F_{\bar{\theta}}(X)}{\bar{\theta}[i]} = \partial \frac{\sigma^1(Z^1)}{Z^1} \cdot \partial \frac{Z^1}{\partial \bar{\theta}[i]}$$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot \partial \frac{\sigma^1(Z^1)}{Z^1} \cdot \partial \frac{Z^1}{\partial \bar{\theta}[i]}$$

- Suppose
  - σ^l, W^l, b^l are components at the l-the last layer
  - Z^l is the input to σ^l (i.e., $W^l \cdot Z^{l-1} + b^l$ if l is not the first layer)

$$\partial \frac{Z^1}{\partial \bar{\theta}[i]} = \partial \frac{W^1 \cdot \sigma^2(Z^2) + b^1}{\partial \bar{\theta}[i]} = W^1 \cdot \partial \frac{\sigma^2(Z^2)}{\partial \bar{\theta}[i]}$$

$$= W^1 \cdot \partial \frac{\sigma^2(Z^2)}{\partial Z^2} \cdot \partial \frac{Z^2}{\partial \bar{\theta}[i]}$$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y) \in \mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot W^1 \cdot \partial \frac{\sigma^2(Z^2)}{\partial Z^2} \cdot \partial \frac{Z^2}{\partial \bar{\theta}[i]}$$

- Suppose
  - σ^l, W^l, b^l are components at the l-the last layer
  - Z^l is the input to σ^l (i.e., $W^l \cdot Z^{l-1} + b^l$ if l is not the first layer)

$$\partial \frac{Z^1}{\partial \overrightarrow{\theta}[i]} = \partial \frac{W^1 \cdot \sigma^2(Z^2) + b^1}{\partial \overrightarrow{\theta}[i]} = W^1 \cdot \partial \frac{\sigma^2(Z^2)}{\partial \bar{\theta}[i]}$$

$$= W^1 \cdot \partial \frac{\sigma^2(Z^2)}{\partial Z^2} \cdot \partial \frac{Z^2}{\partial \bar{\theta}[i]}$$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot (\prod_{i} W^i \cdot \partial \frac{\sigma^{i+1}(Z^{i+1})}{\partial Z^{i+1}}) \cdot \partial \frac{Z^k}{\partial \bar{\theta}[i]}$$

- Suppose
  - σ^l, W^l, b^l are components at the l-the last layer
  - Z^l is the input to σ^l (i.e., $W^l \cdot Z^{l-1} + b^l$ if l is not the first layer)
  - The k-th last layer involves the parameter $\bar{\theta}[i]$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot (\prod_{i} W^i \cdot \partial \frac{\sigma^{i+1}(Z^{i+1})}{\partial Z^{i+1}}) \cdot Z^{k-1}[j]$$

- Suppose
  - σ^l, W^l, b^l are components at the l-the last layer
  - Z^l is the input to σ^l (i.e., $W^l \cdot Z^{l-1} + b^l$ if l is not the first layer)
  - The k-th last layer involves the parameter $\bar{\theta}[i]$
  - $\bar{\theta}[i]$ is the weight by $W^k[j]$

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y) \in \mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot (\prod_i W^i \cdot \partial \frac{\sigma^{i+1}(Z^{i+1})}{\partial Z^{i+1}}) \cdot 1$$

- Suppose
  - σ^l, W^l, b^l are components at the l-the last layer
  - Z^l is the input to σ^l (i.e., $W^l \cdot Z^{l-1} + b^l$ if l is not the first layer)
  - The k-th last layer involves the parameter $\bar{\theta}[i]$
  - $\bar{\theta}[i]$ is a bias

Parameter update

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot (\prod_{i} W^{i} \cdot \partial \frac{\sigma^{i+1}(Z^{i+1})}{\partial Z^{i+1}}) \cdot \partial \frac{Z^{k}}{\partial \bar{\theta}[i]}$$

- Suppose
  - Z^{l} be the output vector of l-th last layer
  - σ^{l}, W^{l}, b^{l} are also for the l-the last layer
  - The k-th last layer involves the parameter $\bar{\theta}[i]$

Vanishing gradient problem

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot (\prod_i W^i \cdot \partial \frac{\sigma^{i+1}(Z^{i+1})}{\partial Z^{i+1}}) \cdot \partial \frac{Z^k}{\partial \bar{\theta}[i]}$$

If the derivative $\partial \frac{\sigma(Z)}{\partial Z}$ is small,
the gradient approaches zero

- Derivatives of sigmoid are small in general
- Derivatives of ramp (rectifier) are relative large (1 or 0)

Vanishing gradient problem

$$\bar{\theta}[i] := \bar{\theta}[i] - \eta \cdot \frac{1}{|T|} \sum_{(X,Y)\in\mathbf{T}} \partial \frac{L(F_{\bar{\theta}}(X), Y)}{\partial F_{\bar{\theta}}(X)} \cdot (\prod_i W^i \cdot \partial \frac{\sigma^{i+1}(Z^{i+1})}{\partial Z^{i+1}}) \cdot \partial \frac{Z^k}{\partial \bar{\theta}[i]}$$

If the derivative $\partial \frac{\sigma(Z)}{\partial Z}$ is small,
the gradient approaches zero

- Derivatives of sigmoid are small in general
- Derivatives of ramp (rectifier) are relative large (1 or 0)