PAT 498/598 (Fall 2024)

Special Topics: Generative AI for Music and Audio Creation

Lecture 5: Deep Learning Fundamentals II

Instructor: Hao-Wen Dong

SCHOOL OF MUSIC, THEATRE & DANCE
PERFORMING ARTS TECHNOLOGY
UNIVERSITY OF MICHIGAN

Assignment 1: AI Song Contest

- Please listen to the **ten finalists of AI Song Contest 2024** and **read the about pages** by clicking the cover arts
- **Vote for your favorites**
- **Answer the following questions** (in 10-20 sentences each)
  - Which is your favorite song? What did they do well? What can be improved?
  - What is one dimension that most finalists didn't look into or didn't do well on?
  - What tasks are easy for current AI? What are difficult?

aisongcontest.com/the-2024-finalists

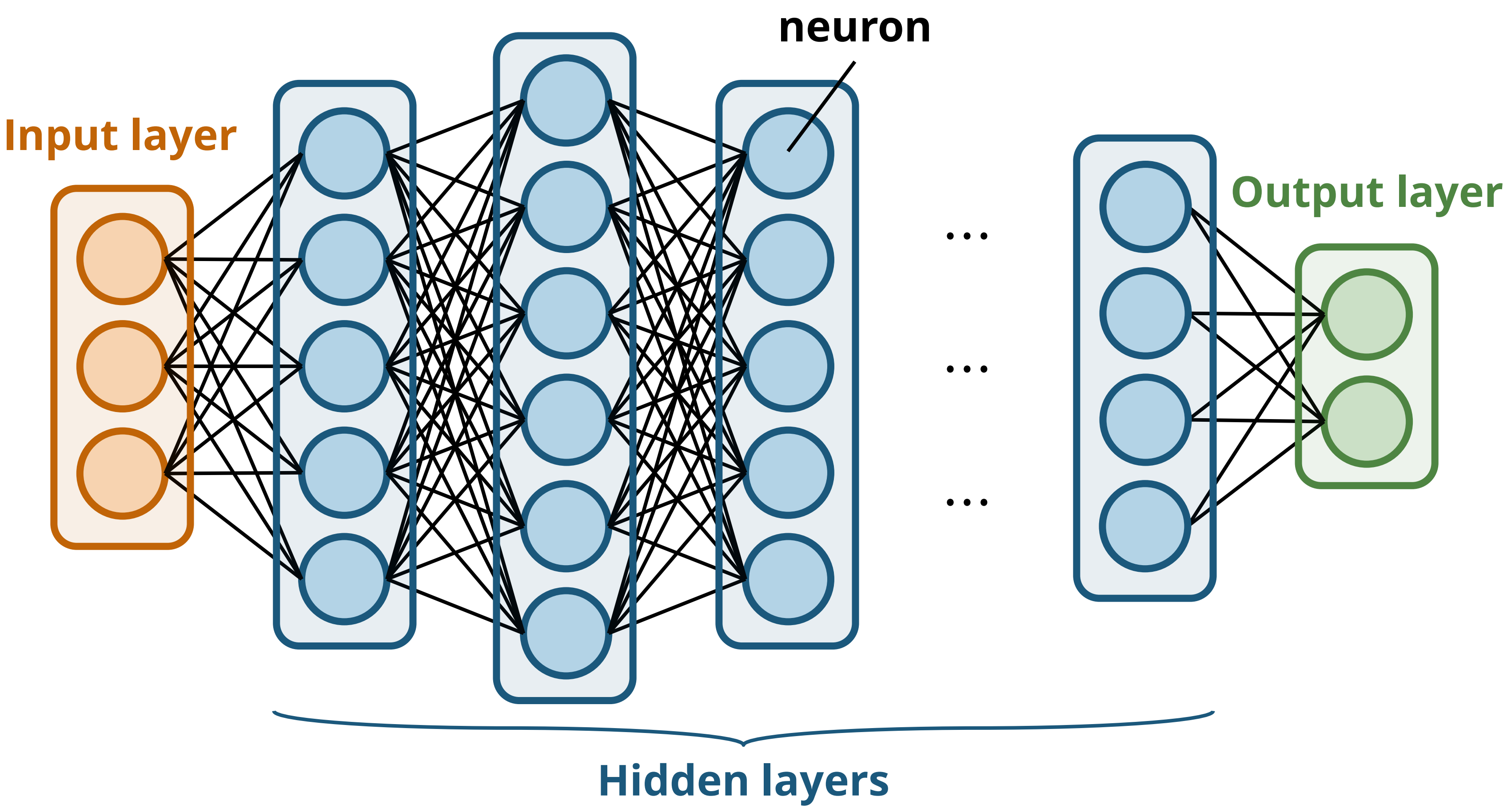
Assignment 1: AI Song Contest

- Instructions will be released on Gradescope
- Due at **11:59pm ET** on **September 20**
- Late submissions: **3 point deducted per day**

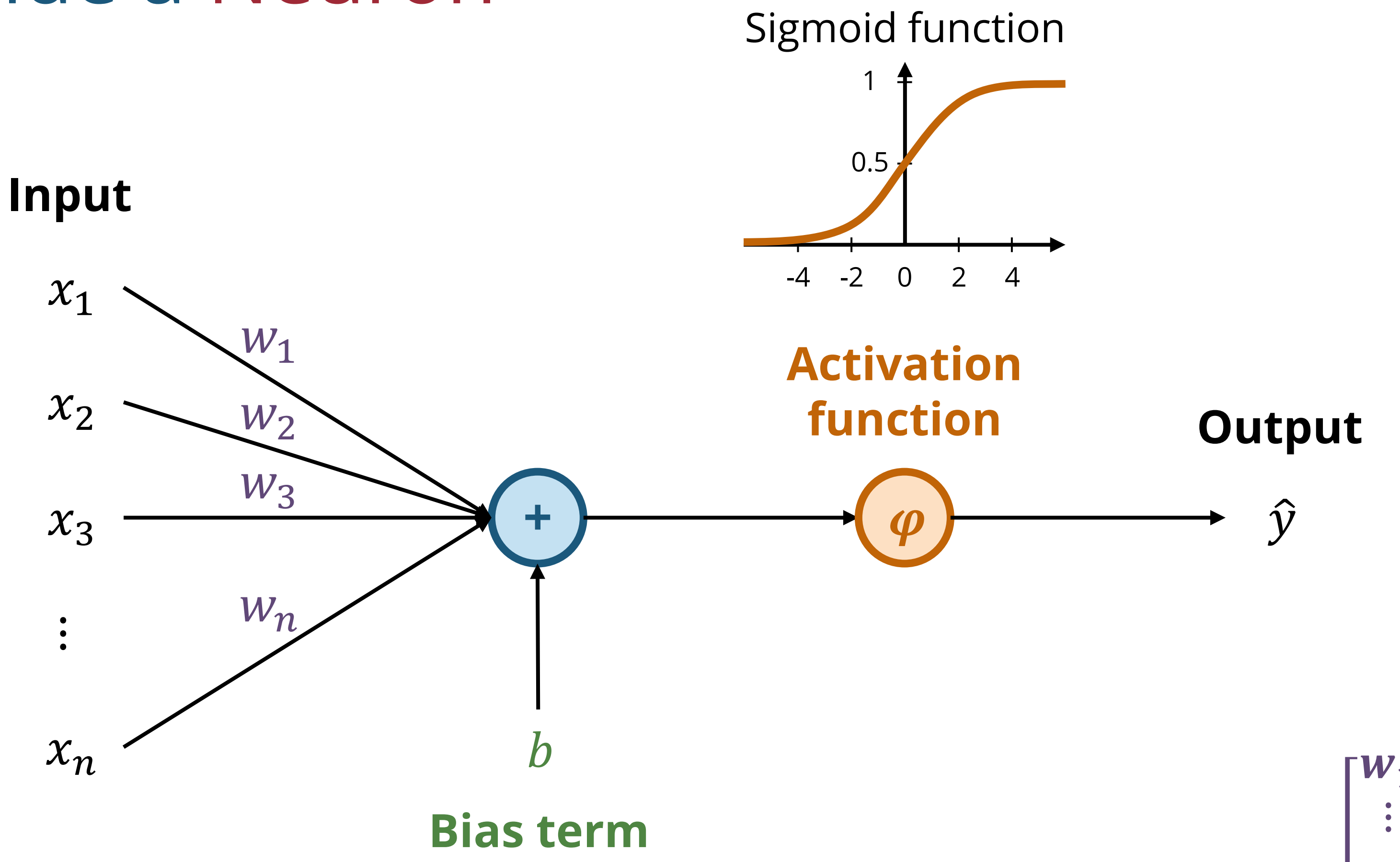
aisongcontest.com/
the-2024-finalists

(Recap) What is Deep Learning?

- A type of machine learning that uses **deep neural networks**



(Recap) Inside a Neuron

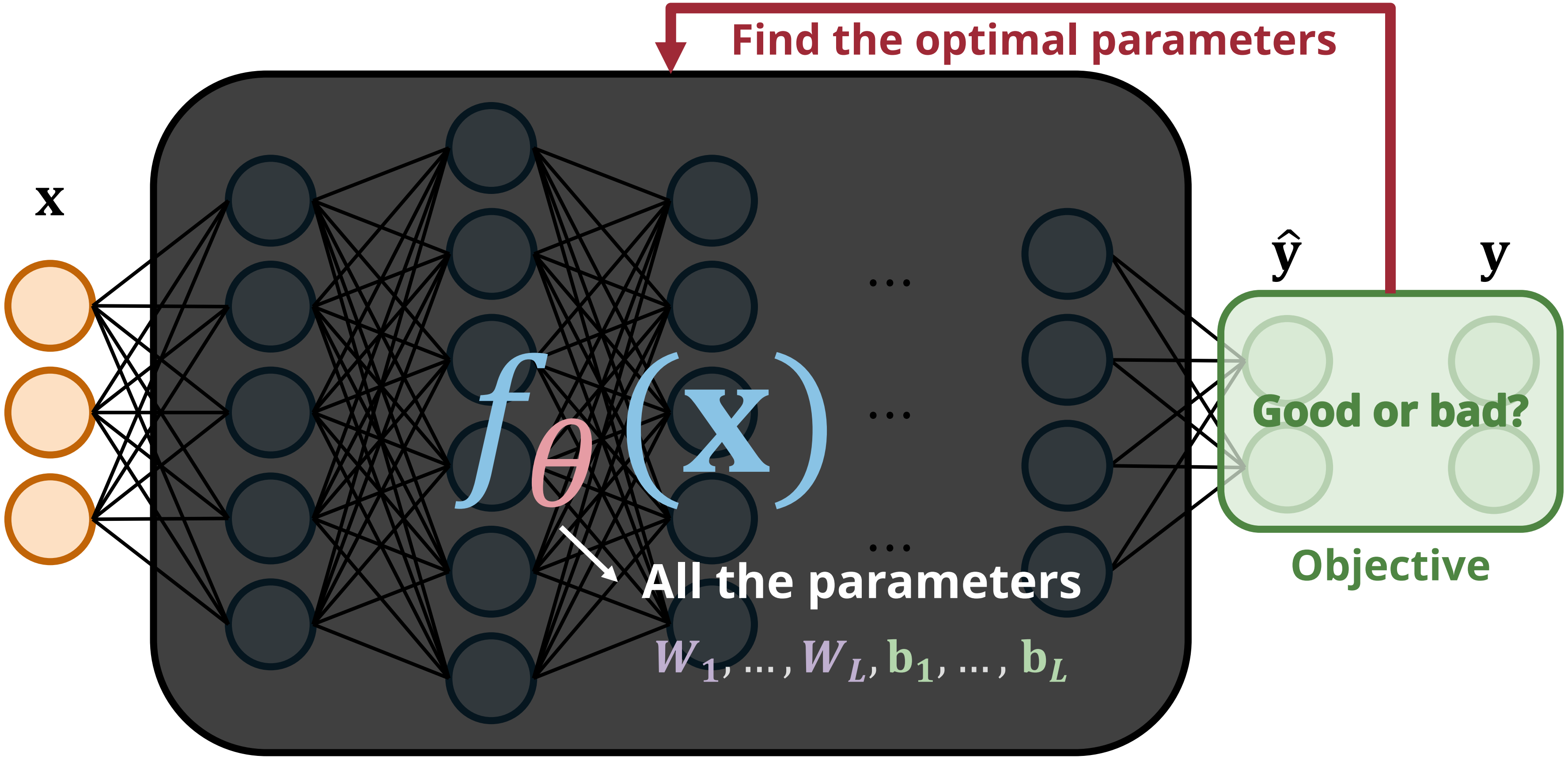


$$\hat{\boldsymbol{y}} = \boldsymbol{\varphi}(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b) = \boldsymbol{\varphi}\left(\sum_{i=1}^{n} w_ix_i + b\right) = \boldsymbol{\varphi}(\mathbf{w}\cdot\mathbf{x} + b)$$

$$\mathbf{w} = \begin{bmatrix} \boldsymbol{w_1} \\ \vdots \\ \boldsymbol{w_n} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \boldsymbol{x_1} \\ \vdots \\ \boldsymbol{x_n} \end{bmatrix}$$

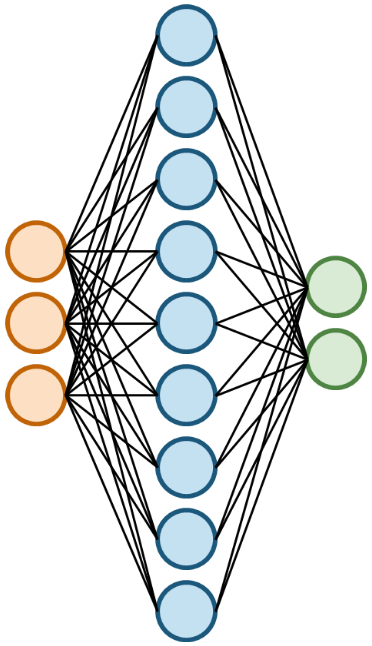
(Recap) Neural Networks are Parameterized Functions

- A neural network represents **a set of functions**



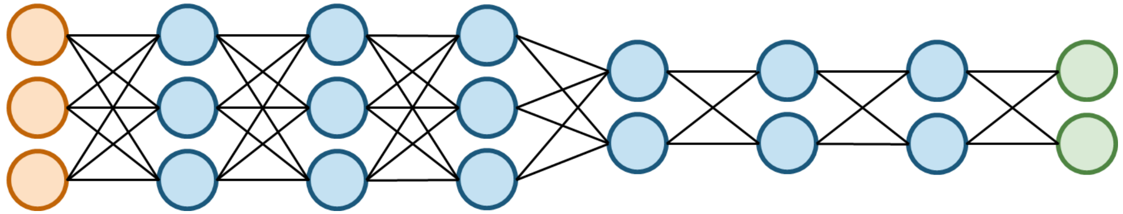
(Recap) Shallow vs Deep Neural Networks – In Practice

Shallow neural nets



Less expressive
(less parameter efficient)

Deep neural nets

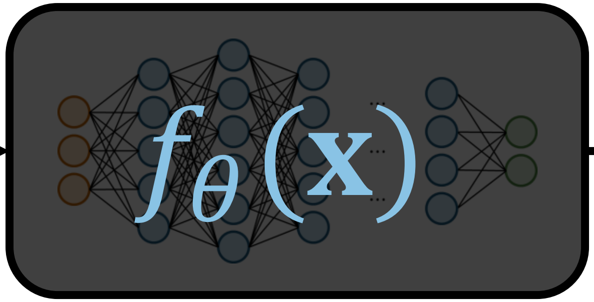


More expressive
(more parameter efficient)

Regression vs Classification

Regression vs Classification

Regression

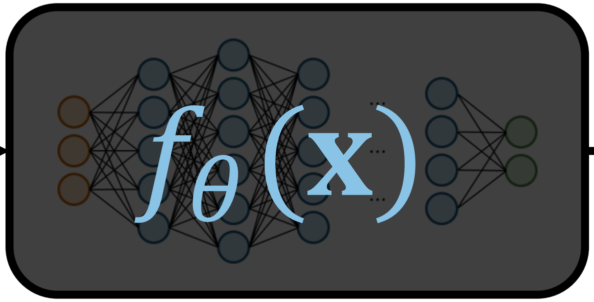


Age

5

Output a number

Classification



Is human?

Yes / No

Output a label

Regression Example: Stock Price Prediction

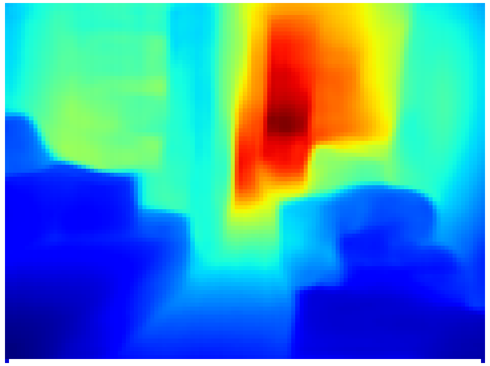
$$y \in [0, \infty)$$

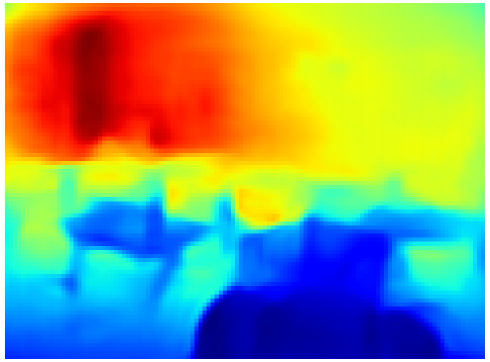
$f\Big($  $\Big) = 108.15$

$f\Big($  $\Big) = 18.95$

Regression Example: Depth Estimation

$$\mathbf{y} \in [0, \infty)^{W \times H}$$

$f\Big($  $\Big) =$ 

$f\Big($  $\Big) =$ 

medium.com/@omarbarakat1995/depth-estimation-with-deep-neural-networks-part-1-5fa6d2237d0d

Classification Example: Image Recognition

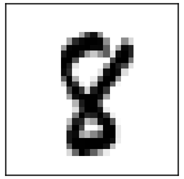
$y \in \{\text{cat}, \text{dog}, \text{bear}, \text{bird}\}$

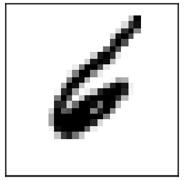
$f($  $) = \text{cat}$

$f($  $) = \text{dog}$

$f($  $) = \text{bear}$

$y \in \{0, 1, 2, \ldots, 9\}$

$f($  $) = 8$

$f($  $) = 6$

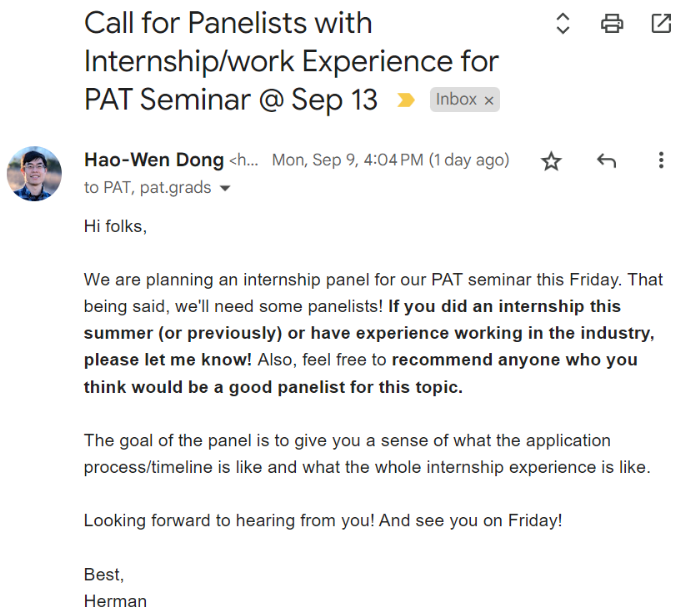
Classification Example: Spam Filter

$f\,($

$) = \text{spam}$

$$y \in \{\text{spam}, \text{not spam}\}$$

$f\,($



$) = \text{not spam}$

How to Train a Neural Network?

Training a Neural Network

Build a neural network
(which defines a set of functions)

↓

Define the objective
(i.e., what is good for a function)

↓

Find the optimal parameters
(which leads to the best function)

Training a Neural Network

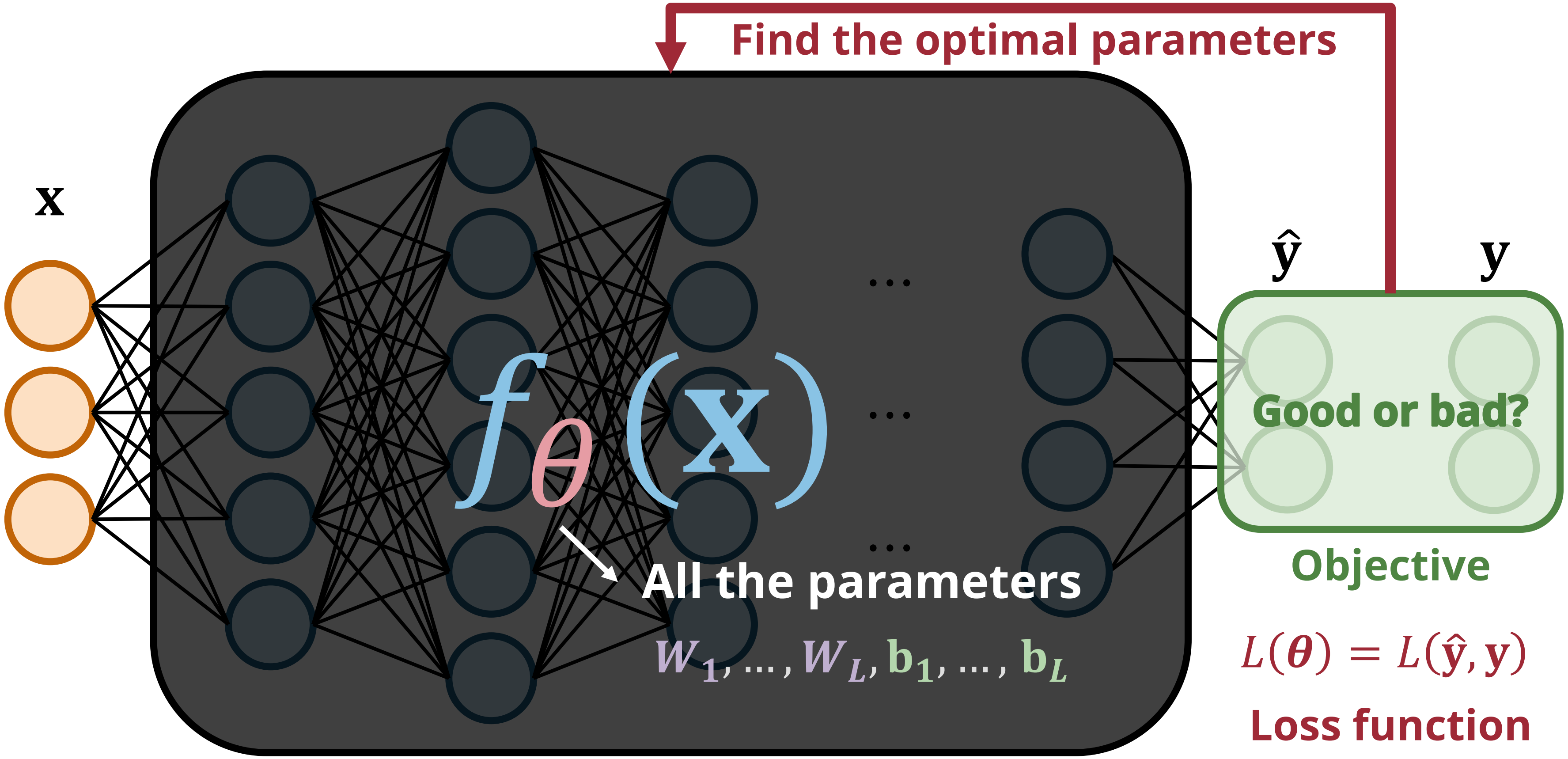
Build a neural network
(which defines a set of functions)

Define the objective
(i.e., what is good for a function)

Find the optimal parameters
(which leads to the best function)

(Recap) Neural Networks are Parameterized Functions

- A neural network represents **a set of functions**



Loss Function

- Measure **how well the model perform** (in the opposite way)
- The choice of loss function depends on the task and the goals

$$L(\boldsymbol{\theta}) = L(\hat{\mathbf{y}}, \mathbf{y})$$

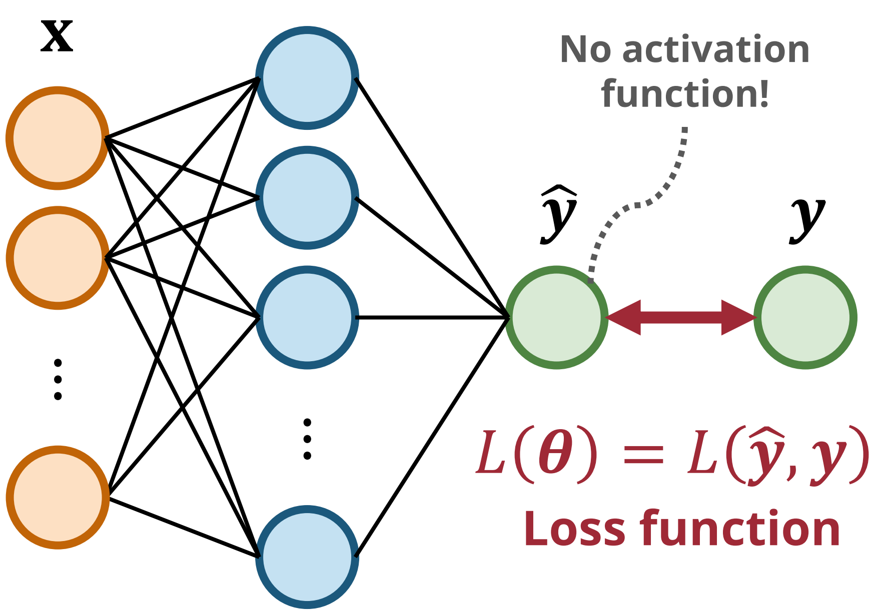
Loss Function – The Many Names

- Sometimes called
  - **Cost** function
  - **Error** function
- The opposite is known as
  - **Objective** function
  - **Reward** function (reinforcement learning)
  - **Fitness** function (evolutionary algorithms & genetic algorithms)
  - **Utility** function (economics)
  - **Profit** function (economics)

Example: Audio Codec

- What would be **a good objective to train a neural audio codec**?
- What do we care about for a codec?
  - Reconstruction quality **Trainable**
  - Bit rate (compression rate) **Likely not trainable but searchable**
  - Encoding/decoding speed **Likely not trainable but searchable**
- How do we measure reconstruction quality?
  - Difference in raw waveforms?
  - Difference in spectrograms?
  - Perceptual quality (psychoacoustics)?

Common Loss Functions for Regression



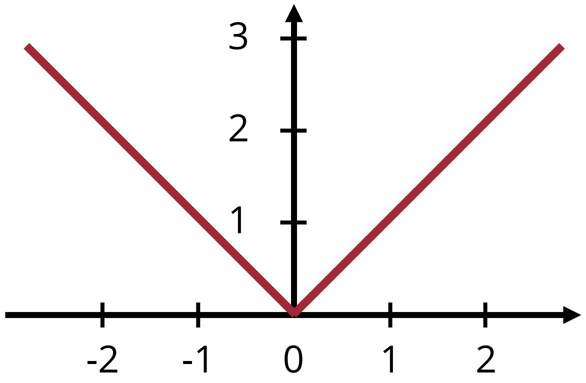
$L(\boldsymbol{\theta}) = L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

Loss function

Why not $L(\hat{y}, y) = \hat{y} - y$?

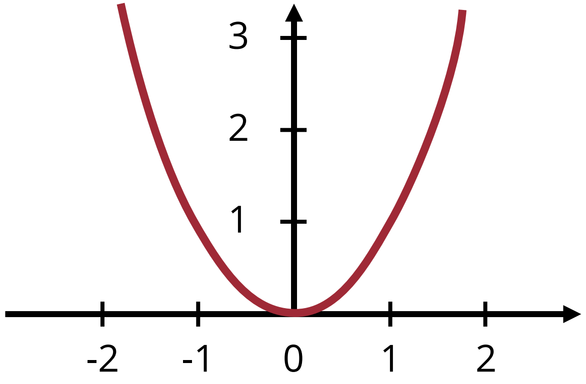
L1 loss

$$L(\hat{y}, y) = |\hat{y} - y|$$



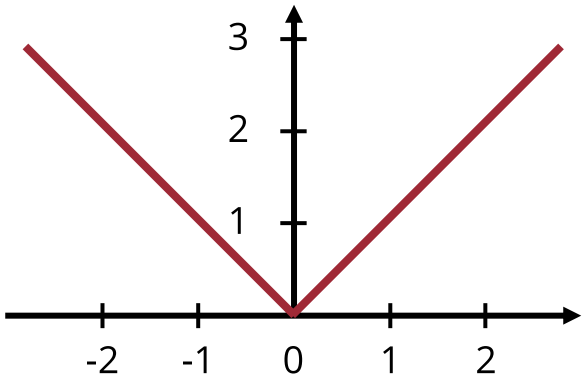
L2 loss

$$L(\hat{y}, y) = (\hat{y} - y)^2$$



L1 vs L2 Losses

L1 loss

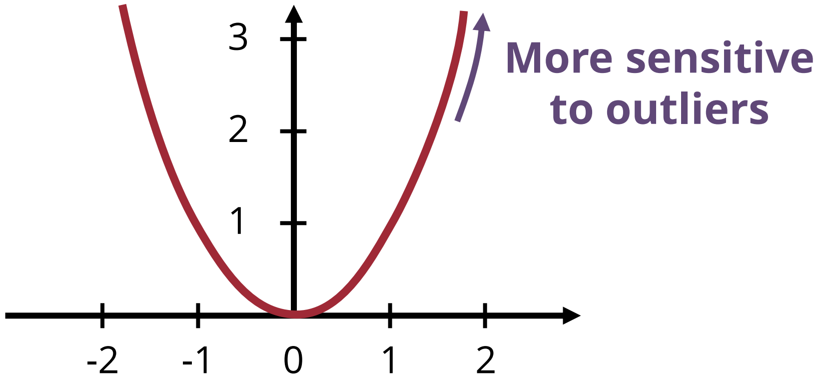


$$L(\hat{y}, y) = |\hat{y} - y|$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \mathbf{MAE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i|$$

Mean Absolute Error (MAE)

L2 loss



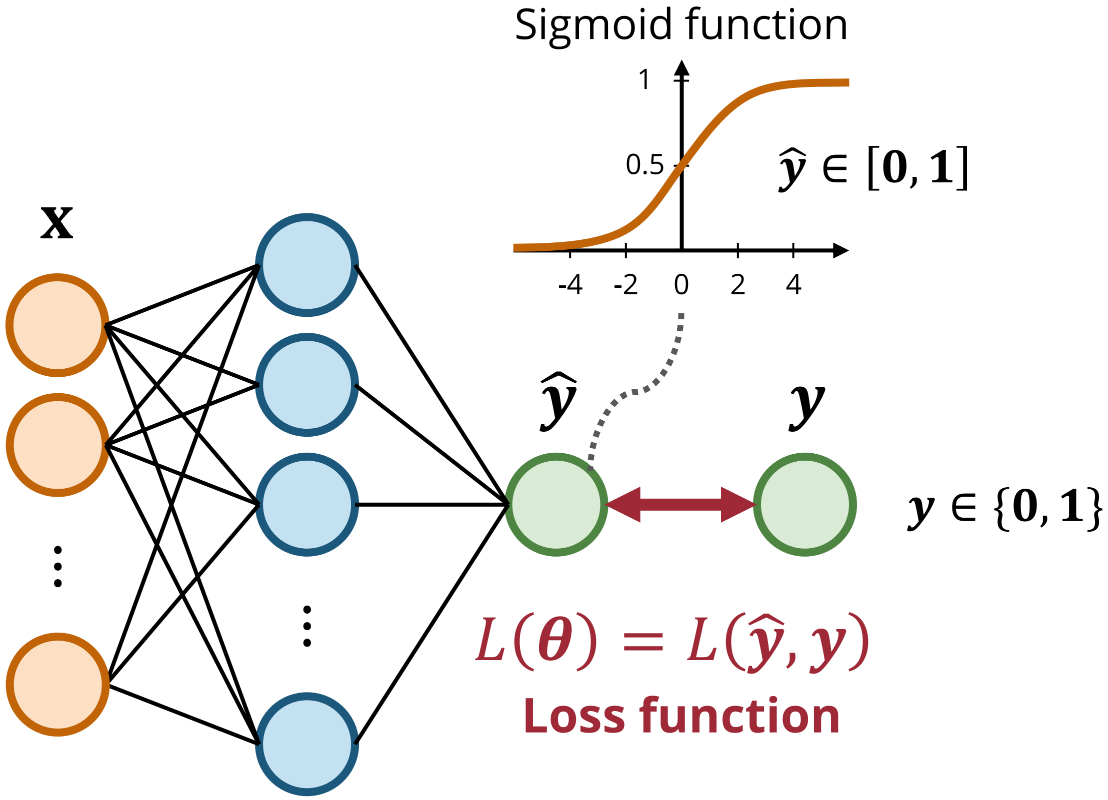
$$L(\hat{y}, y) = (\hat{y} - y)^2$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \mathbf{MSE}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{n}\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

Mean Squared Error (MSE)

Binary Cross Entropy for Binary Classification

- **Logistic regression** approaches classification like regression

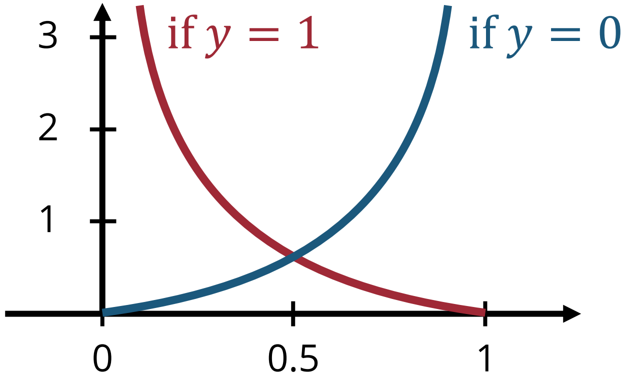


Binary cross entropy

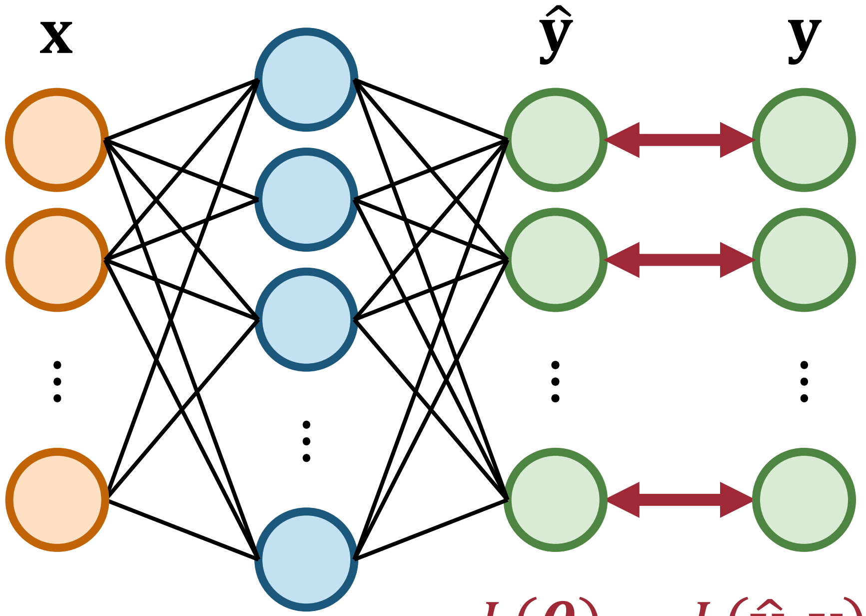
(Also called log loss)

$$L(\hat{y}, y) = \begin{cases} -\log \hat{y}\,, & \text{if } y = 1 \\ -\log(1-\hat{y})\,, & \text{if } y = 0 \end{cases}$$

$$= -y \log \hat{y} + (1-y) \log(1-\hat{y})$$



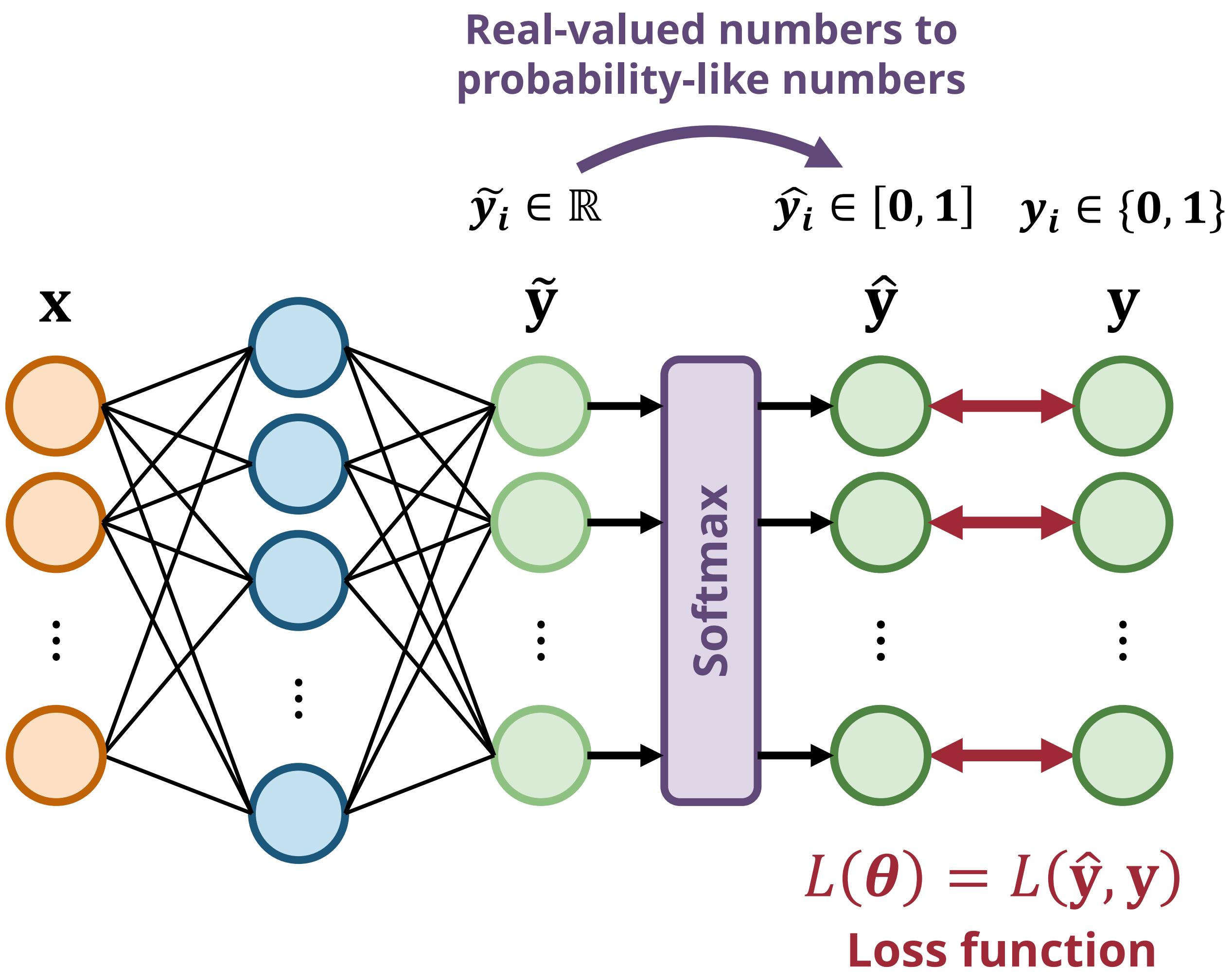
Cross Entropy for Multiclass Classification



$L(\boldsymbol{\theta}) = L(\hat{\mathbf{y}}, \mathbf{y})$

Loss function

Cross Entropy for Multiclass Classification

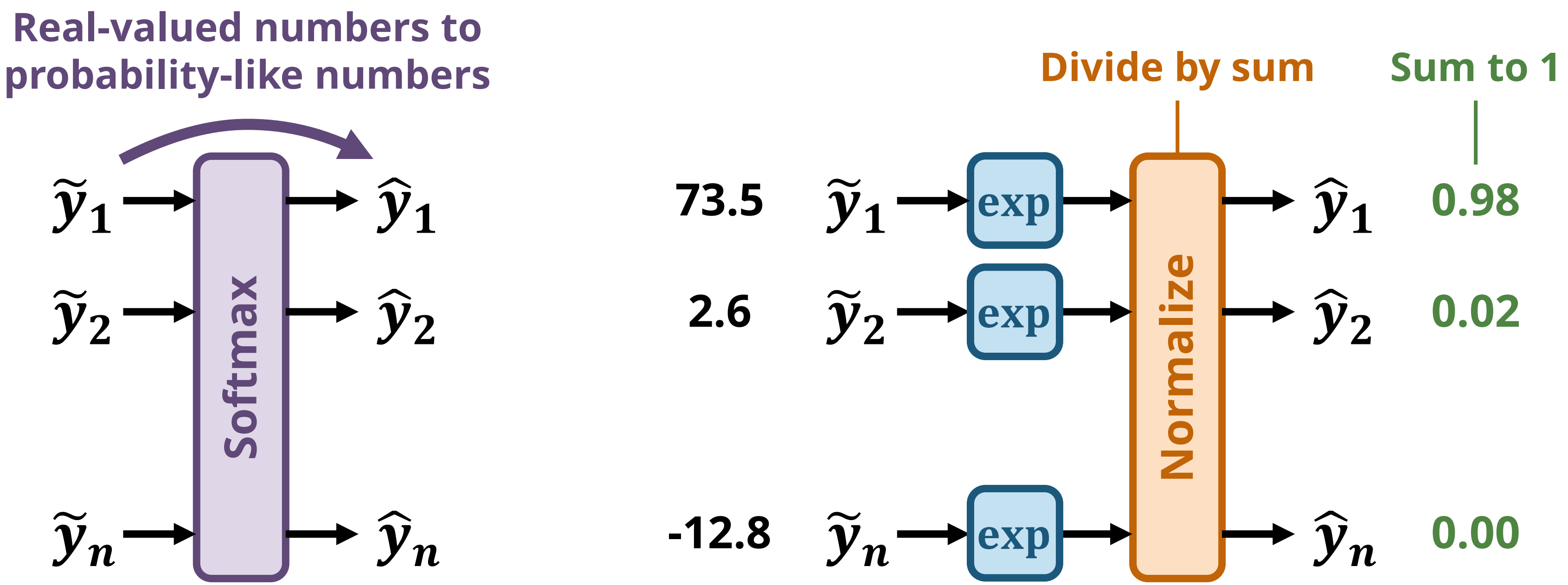


Softmax

$$\hat{y}_i = \frac{e^{\tilde{y}_i}}{\sum_{j=1}^{n} e^{\tilde{y}_j}}$$

Softmax

- **Intuition**: Map several numbers to $[0, 1]$ while **keeping their relative magnitude**
  - Softmax is like the **multivariate version of sigmoid**



Cross Entropy for Multiclass Classification

Binary Cross Entropy

Only one of them will be one!

$$L(\hat{y}, y) = -\boxed{y}\log \hat{y} + \boxed{(1-y)}\log(1-\hat{y})$$

Cross Entropy

Only one of them will be one!

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\boxed{y_1}\log \hat{y}_1 - \boxed{y_2}\log \hat{y}_2 - \cdots - \boxed{y_i}\log \hat{y}_n$$

$$= -\underbrace{\boxed{\sum_i^n y_i \log \hat{y}_i}}_{\text{Log likelihood}}$$

Why *not* MSE Loss for Classification?

- **Minimizing cross-entropy** is equivalent to **maximizing likelihood**!
- However, no one prohibits you from using an MSE loss on Softmax output
  - In fact, it will still train the model

- While loss functions can have the same global minima, they might have led to different **training dynamics** and **weights for different types of errors**
  - For example, MSE is more sensitive to MAE due to the quadratic term even though they have the same global minima

Loss Functions vs Output Space

- Oftentimes, we change the **output space** instead of the loss function
- For example,
  - MSE on spectrograms → MSE on mel spectrograms
  - MSE of magnitude in raw values → MSE of magnitude in dB

$\mathcal{F}$: spectrogram → mel spectrogram

$\mathcal{F}$: raw value → db

- What's the difference?

	Model	Loss
Setup A	$f: x \rightarrow y$	$L(\mathcal{F}(y), \mathcal{F}(\hat{y}))$
Setup B	$f: x \rightarrow \mathcal{F}(y)$	$L(y, \hat{y})$

Optimization

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

Define the objective
(i.e., what is good for a function)

$$L(\boldsymbol{\theta})$$

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Training a Neural Network

Build a neural network
(which defines a set of functions)

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x})$$

↓

Define the objective
(i.e., what is good for a function)

$$L(\boldsymbol{\theta})$$

↓

Find the optimal parameters
(which leads to the best function)

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Optimizing the Parameters of a Neural Network

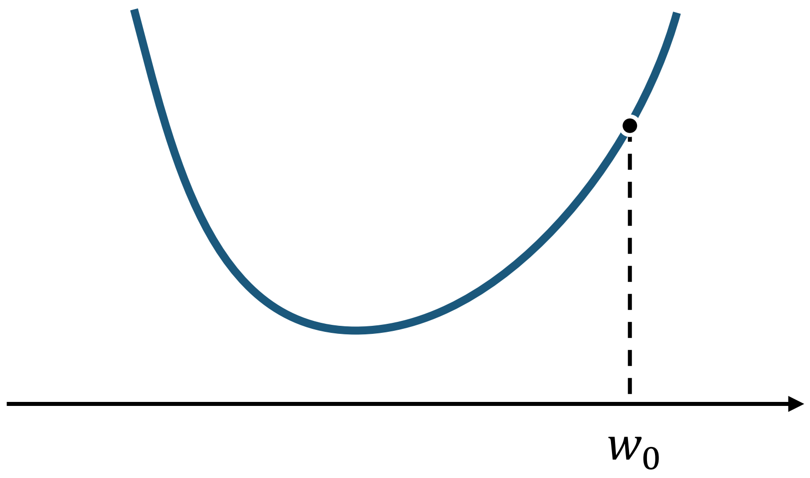
- Many, many ways…
- Most commonly through **gradient descent** in deep learning
- Alternatively, we can use search or genetic algorithm

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Gradient Descent

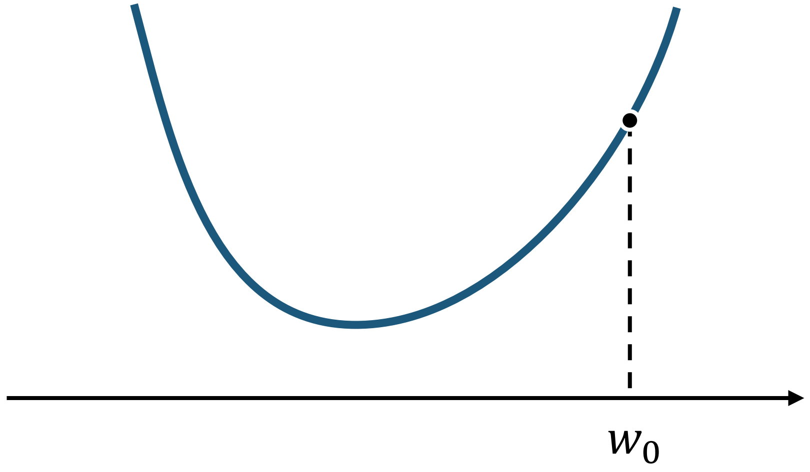
- **Intuition**: Gradient can suggest a good direction to tune the parameters

Derivative for a vector, matrix or tensor



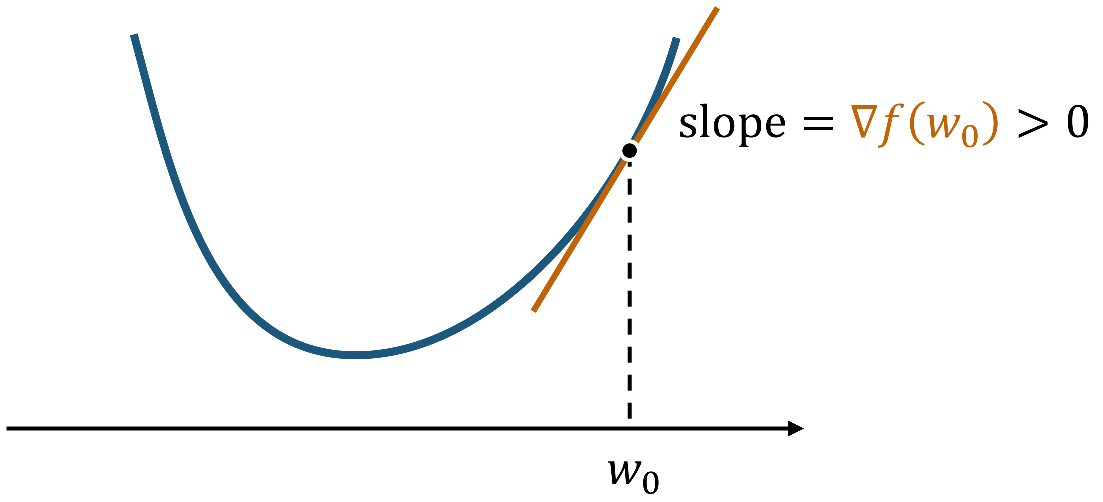
Gradient Descent – Pseudocode

- Pick an **initial weight vector** w_0 and **learning rate** η
- Repeat until convergence: $w_{t+1} = w_t - \eta \boxed{\nabla f(w_t)}$ → **Gradient of function f with respect to weight w**



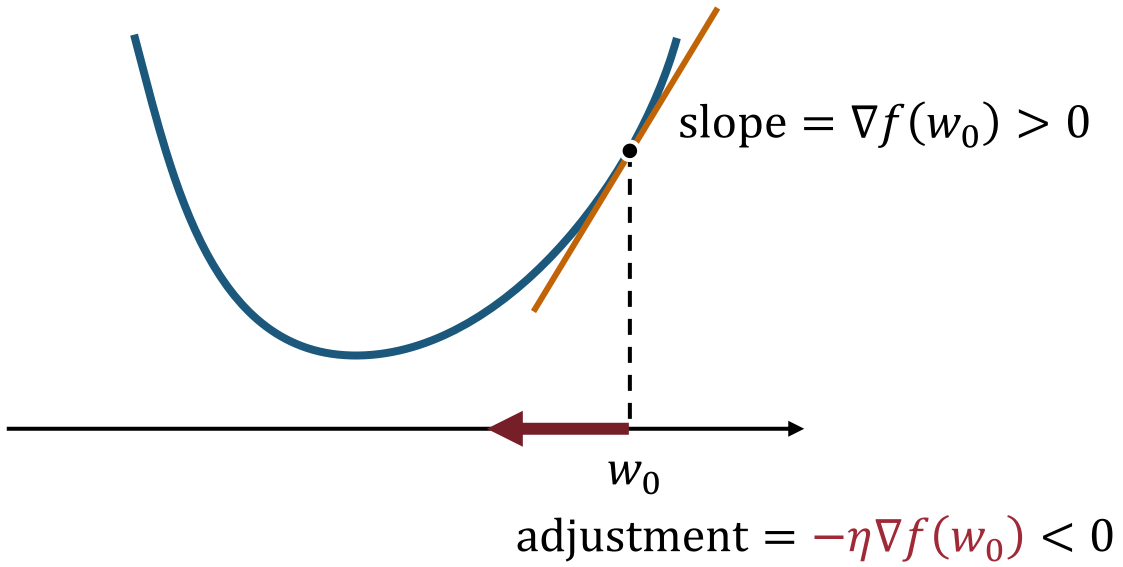
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



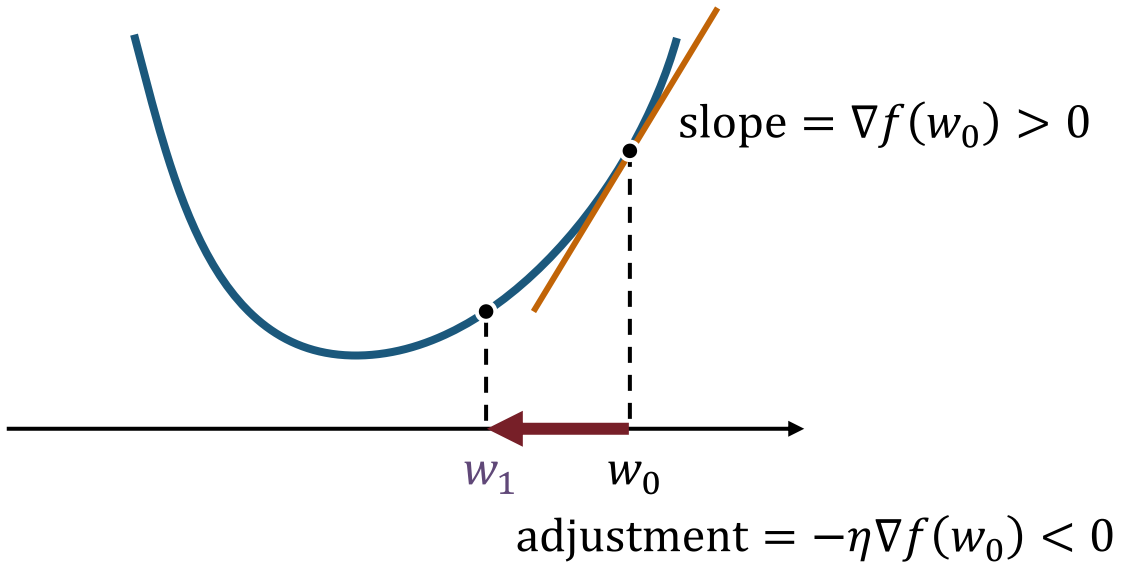
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



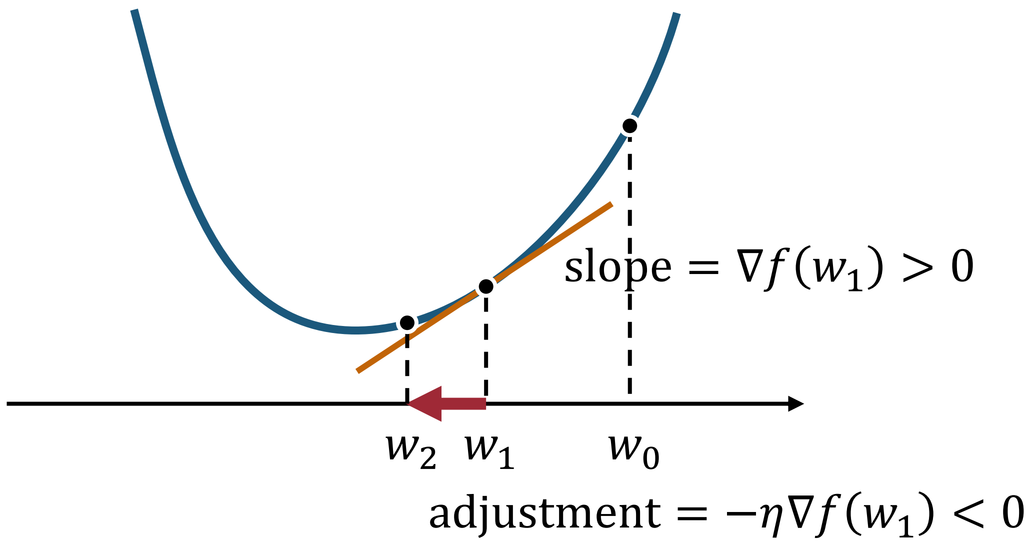
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



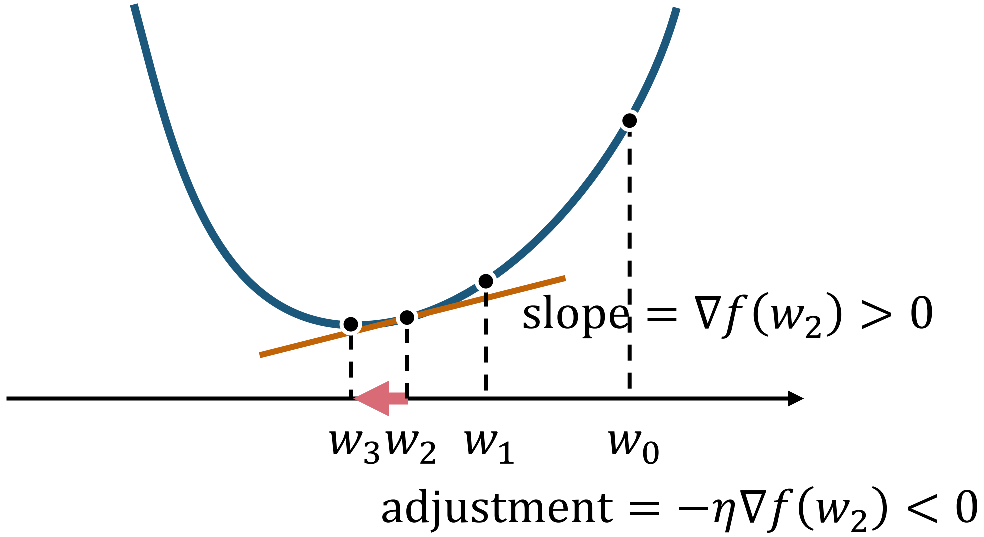
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



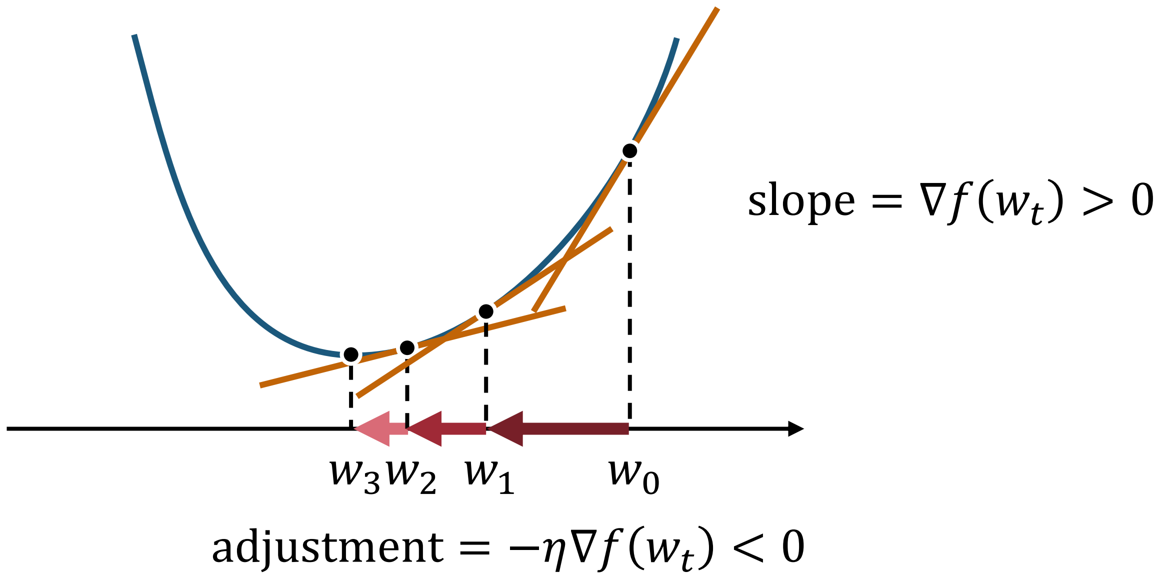
Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$

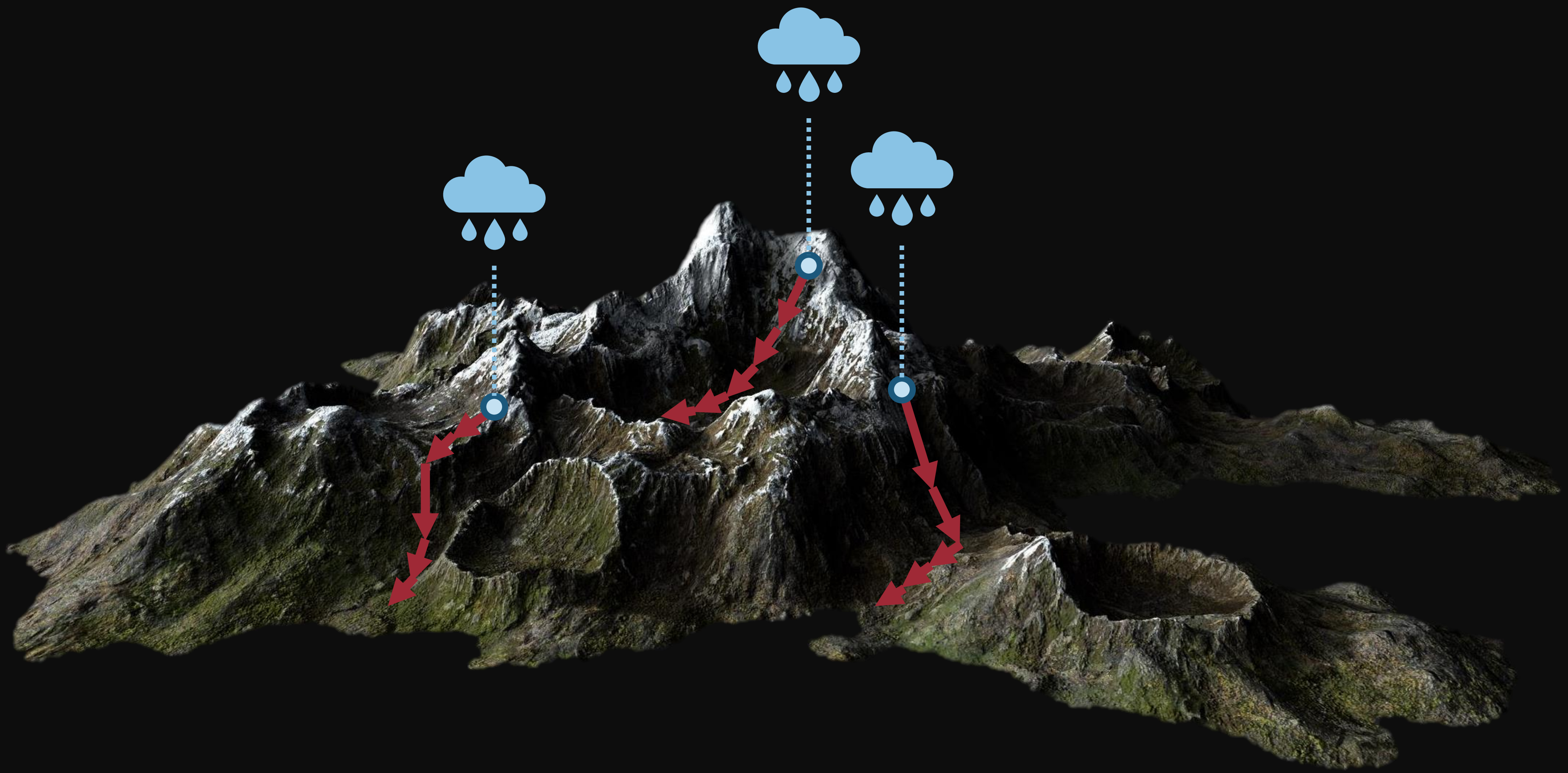


Gradient Descent – Pseudocode

- Pick an initial weight vector w_0 and learning rate η
- Repeat until convergence: $w_{t+1} = w_t - \eta \nabla f(w_t)$



Gradient Descent – 3D Case



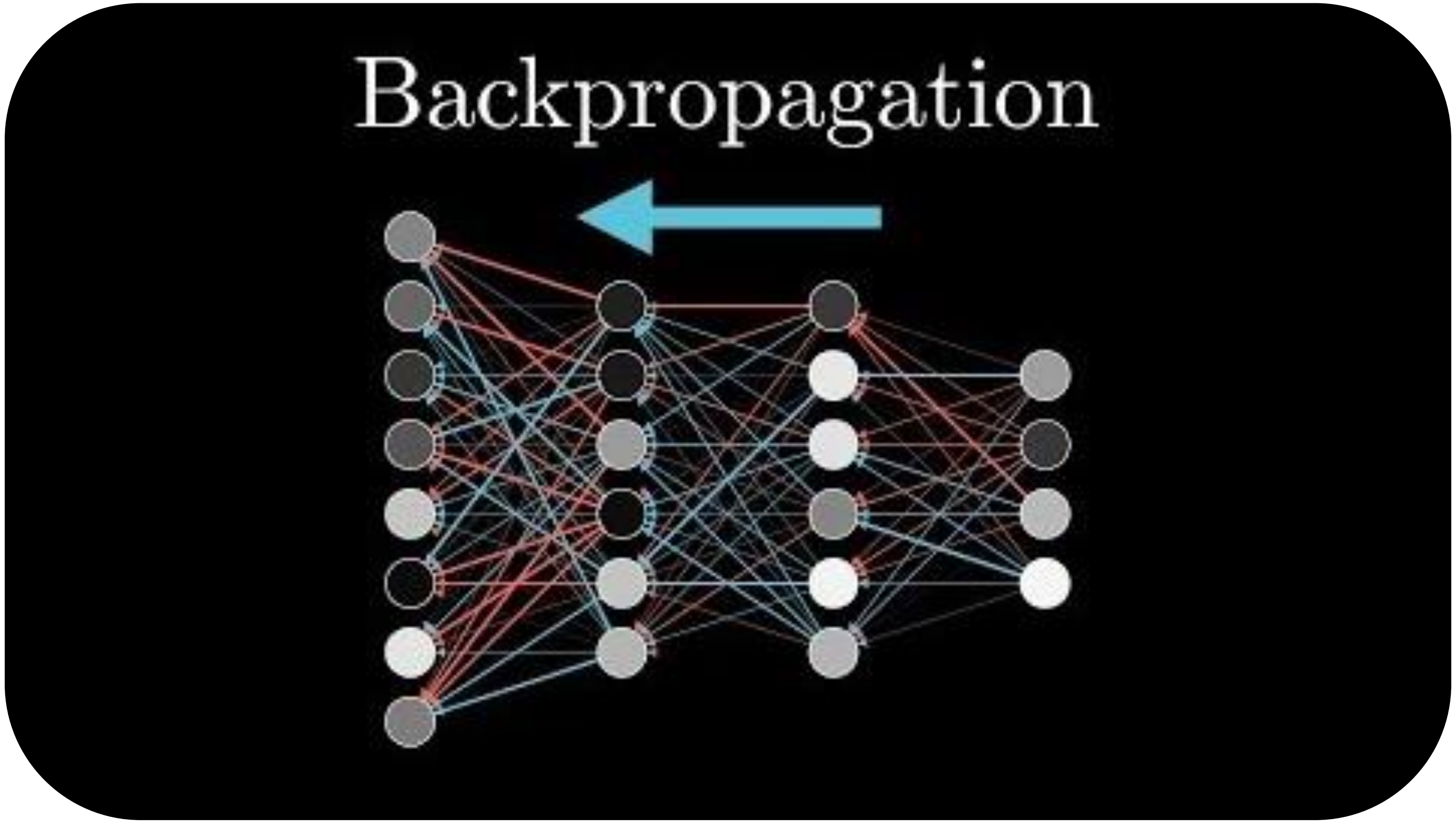
substance3d.adobe.com/community-assets/assets/9fe77d4279e12fe2fbd90b8f7ca18146a565f7ee

Backpropagation: Efficiently Computing the Gradients

- An efficient way of **computing gradients** using chain rule
- The reason why we want **everything to be differentiable** in deep learning

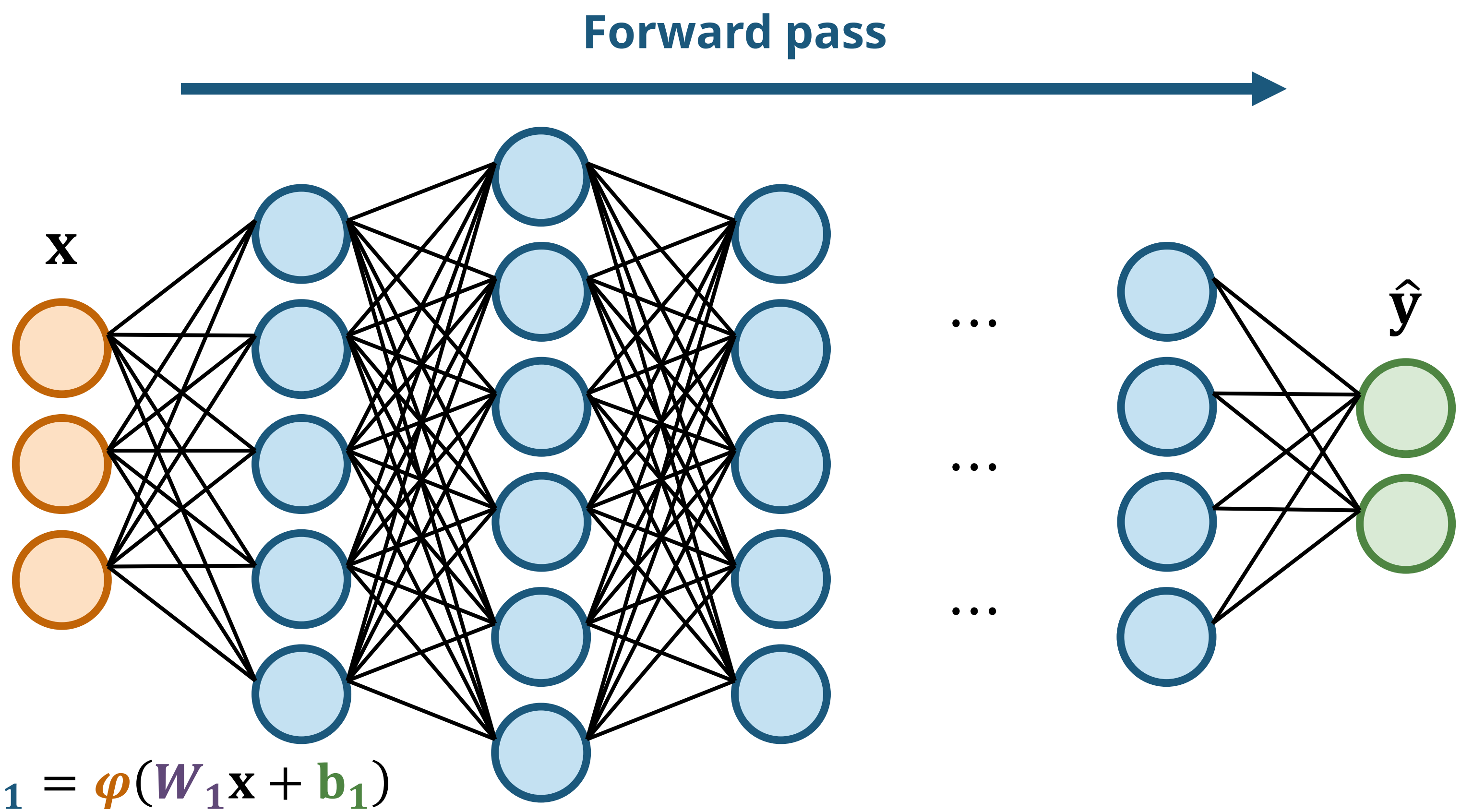
$$w_{t+1} = w_t - \eta \nabla f(w_t)$$

Backpropagation: Efficiently Computing the Gradients



youtu.be/Ilg3gGewQ5U?t=196

Forward Pass & Backward Pass



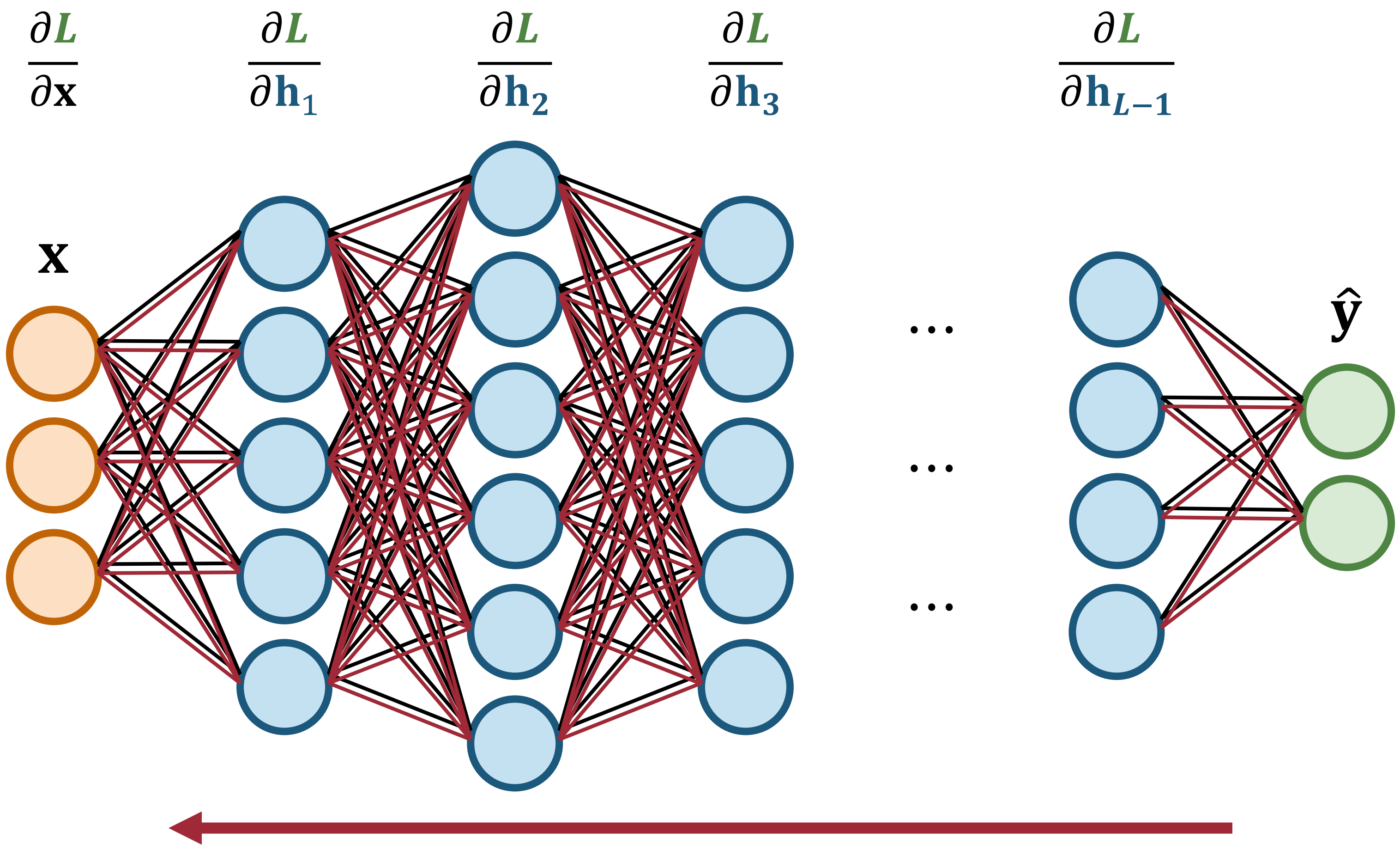
$$\mathbf{h}_1 = \varphi(W_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{h}_2 = \varphi(W_2\mathbf{h}_1 + \mathbf{b}_2)$$

$$\mathbf{h}_3 = \varphi(W_2\mathbf{h}_2 + \mathbf{b}_2)$$

$$\hat{\mathbf{y}} = \varphi(W_L\mathbf{h}_{L-1} + \mathbf{b}_L)$$

Forward Pass & Backward Pass



Backward pass

loss.backward()