CEE 616: Probabilistic Machine Learning
M3 Deep Neural Networks:
Neural Networks for Structured Data I

Jimi Oke

UMass Amherst
College of Engineering

Thu, Oct 16, 2025

Outline

Neural networks

Neural networks

Consider the linear model:

Neural networks

Consider the linear model:

$$f(\boldsymbol{x}; \theta) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

Neural networks

Consider the linear model:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We can increase the flexibility of the model via a basis function expansion (feature extractor) $\phi(\boldsymbol{x})$:

Neural networks

Consider the linear model:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We can increase the flexibility of the model via a basis function expansion (feature extractor) $\phi(\boldsymbol{x})$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x}) + \boldsymbol{b} \tag{2}$$

Neural networks

Consider the linear model:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We can increase the flexibility of the model via a basis function expansion (feature extractor) $\phi(\boldsymbol{x})$:

$$f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x}) + \boldsymbol{b} \tag{2}$$

If further parameterize $\phi(\boldsymbol{x})$ by $\boldsymbol{\theta}_2$ for better fitting, we have:

Neural networks

Consider the linear model:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We can increase the flexibility of the model via a basis function expansion (feature extractor) $\phi(\boldsymbol{x})$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x}) + \boldsymbol{b} \tag{2}$$

If further parameterize $\phi(\boldsymbol{x})$ by $\boldsymbol{\theta}_2$ for better fitting, we have:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x};\boldsymbol{\theta}_2) + \boldsymbol{b} \tag{3}$$

To even further increase complexity, we can recursively fit more feature extractors $f_\ell(\boldsymbol{x};\boldsymbol{\theta}_\ell)$:

Neural networks

Consider the linear model:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We can increase the flexibility of the model via a basis function expansion (feature extractor) $\phi(\boldsymbol{x})$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x}) + \boldsymbol{b} \tag{2}$$

If further parameterize $\phi(\boldsymbol{x})$ by $\boldsymbol{\theta}_2$ for better fitting, we have:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x};\boldsymbol{\theta}_2) + \boldsymbol{b} \tag{3}$$

To even further increase complexity, we can recursively fit more feature extractors $f_\ell(\boldsymbol{x};\boldsymbol{\theta}_\ell)$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f_L(f_{L-1}(\cdots f_1(\boldsymbol{x};\boldsymbol{\theta}_1))\cdots)) \tag{4}$$

Each ℓ can be considered a layer in a **feedforward neural network** (FFNN) of L layers.

Neural networks

Consider the linear model:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b} \tag{1}$$

We can increase the flexibility of the model via a basis function expansion (feature extractor) $\phi(\boldsymbol{x})$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x}) + \boldsymbol{b} \tag{2}$$

If further parameterize $\phi(\boldsymbol{x})$ by $\boldsymbol{\theta}_2$ for better fitting, we have:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}\phi(\boldsymbol{x};\boldsymbol{\theta}_2) + \boldsymbol{b} \tag{3}$$

To even further increase complexity, we can recursively fit more feature extractors $f_\ell(\boldsymbol{x};\boldsymbol{\theta}_\ell)$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = f_L(f_{L-1}(\cdots f_1(\boldsymbol{x};\boldsymbol{\theta}_1))\cdots)) \tag{4}$$

Each ℓ can be considered a layer in a **feedforward neural network** (FFNN) of L layers.

- Also known as a **multilayer perception** (MLP)
- When L is large, this is termed a **deep neural network** (DNN)

Biological neuron

Biological neuron

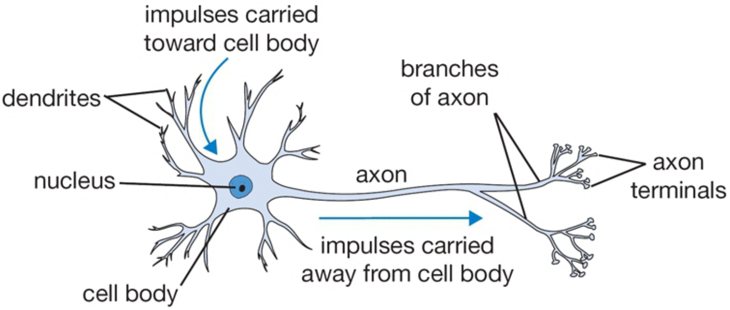


Figure: Biological neuron (Source: `https://cs231n.github.io/neural-networks-1/`)

- $\sim$86 billion neurons are found in the human nervous system

Biological neuron

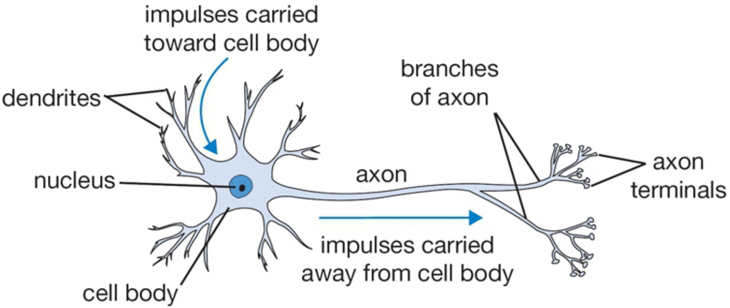


Figure: Biological neuron (Source: `https://cs231n.github.io/neural-networks-1/`)

- ∼86 billion neurons are found in the human nervous system
- These neurons are connected by 10^{14} to 10^{15} synapses

Biological neuron

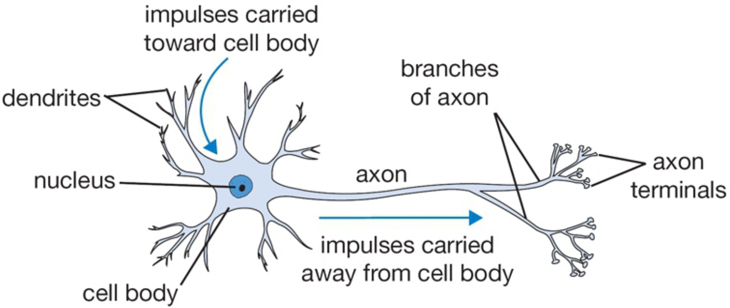


Figure: Biological neuron (Source: `https://cs231n.github.io/neural-networks-1/`)

- $\sim$86 billion neurons are found in the human nervous system
- These neurons are connected by 10^{14} to 10^{15} synapses
- Each neuron receives input signals from its dendrites and outputs signals along a single axon

Biological neuron

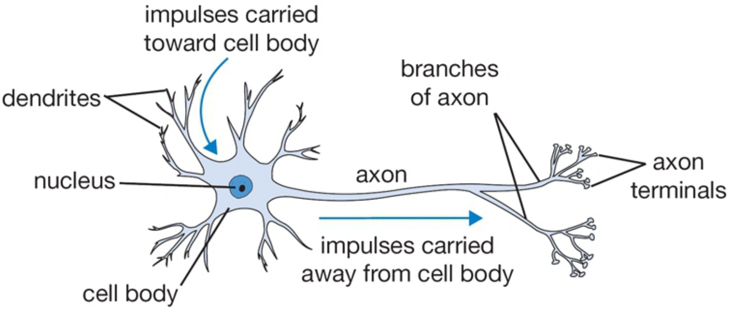


Figure: Biological neuron (Source: `https://cs231n.github.io/neural-networks-1/`)

- ∼86 billion neurons are found in the human nervous system
- These neurons are connected by 10^{14} to 10^{15} synapses
- Each neuron receives input signals from its dendrites and outputs signals along a single axon
- The axon in turn connects to other neurons via synapses

Artificial neural networks

Artificial neural networks

[Artificial] Neural networks (ANNs) are modeled as connected layers of neuron in an acyclic graph (no loops).

- ANNs are organized into layers of neurons (or "units")

Artificial neural networks

[Artificial] Neural networks (ANNs) are modeled as connected layers of neuron in an acyclic graph (no loops).

- ANNs are organized into layers of neurons (or "units")
- Fully-connected layers are common

Artificial neural networks

[Artificial] Neural networks (ANNs) are modeled as connected layers of neuron in an acyclic graph (no loops).

- ANNs are organized into layers of neurons (or "units")
- Fully-connected layers are common
- The basic ANN architecture with multiple hidden layers is called the **multilayer perceptron (MLP)**

Artificial neural networks

[Artificial] Neural networks (ANNs) are modeled as connected layers of neuron in an acyclic graph (no loops).

- ANNs are organized into layers of neurons (or "units")
- Fully-connected layers are common
- The basic ANN architecture with multiple hidden layers is called the **multilayer perceptron (MLP)**
  - An ANN with only one hidden layer is called the **single layer perceptron**

Artificial neural networks

[Artificial] Neural networks (ANNs) are modeled as connected layers of neuron in an acyclic graph (no loops).

- ANNs are organized into layers of neurons (or "units")
- Fully-connected layers are common
- The basic ANN architecture with multiple hidden layers is called the **multilayer perceptron (MLP)**
  - An ANN with only one hidden layer is called the **single layer perceptron**
  - N-layer neural network (number of hidden layers + output layer)

Artificial neural networks

[Artificial] Neural networks (ANNs) are modeled as connected layers of neuron in an acyclic graph (no loops).

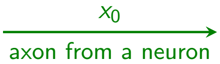
- ANNs are organized into layers of neurons (or "units")
- Fully-connected layers are common
- The basic ANN architecture with multiple hidden layers is called the **multilayer perceptron (MLP)**
  - An ANN with only one hidden layer is called the **single layer perceptron**
  - N-layer neural network (number of hidden layers + output layer)
- The output neurons have no activation function. Instead, they perform a final transformation of outputs from the penultimate layer

Computational neuron model

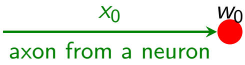
Computational neuron model

axon from a neuron

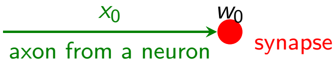
Computational neuron model



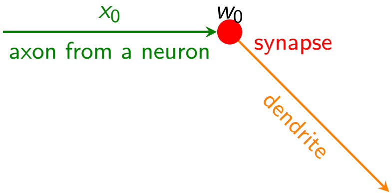
Computational neuron model



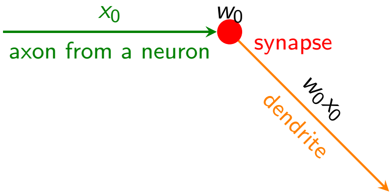
Computational neuron model



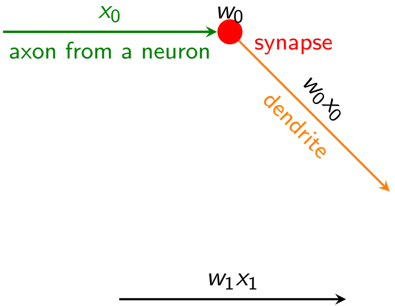
Computational neuron model



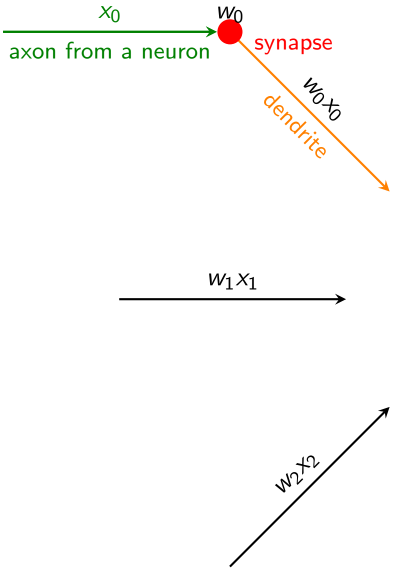
Computational neuron model



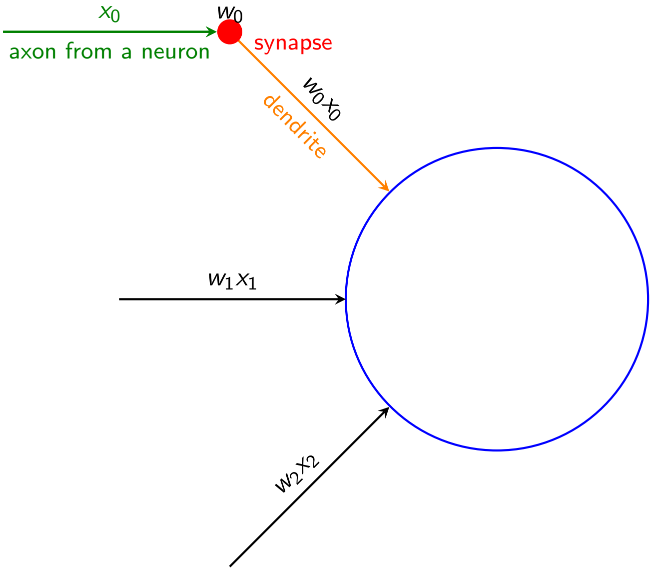
Computational neuron model



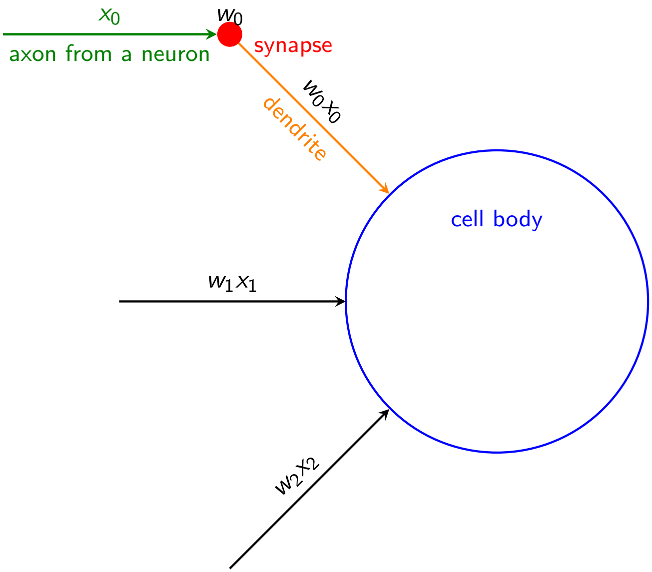
Computational neuron model



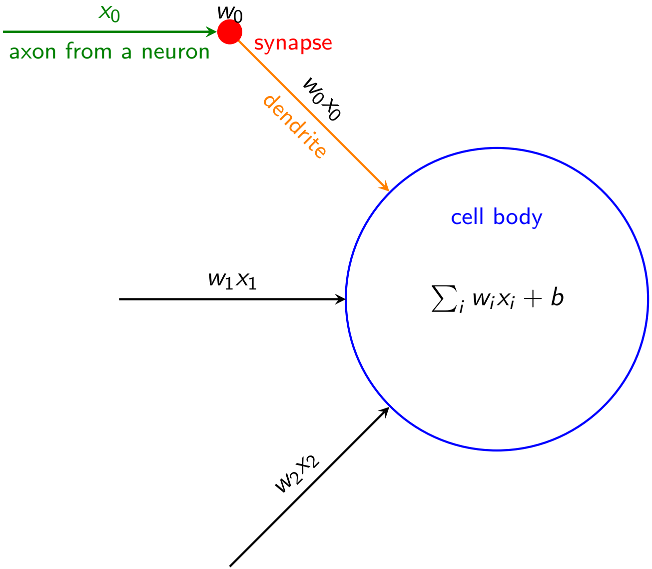
Computational neuron model



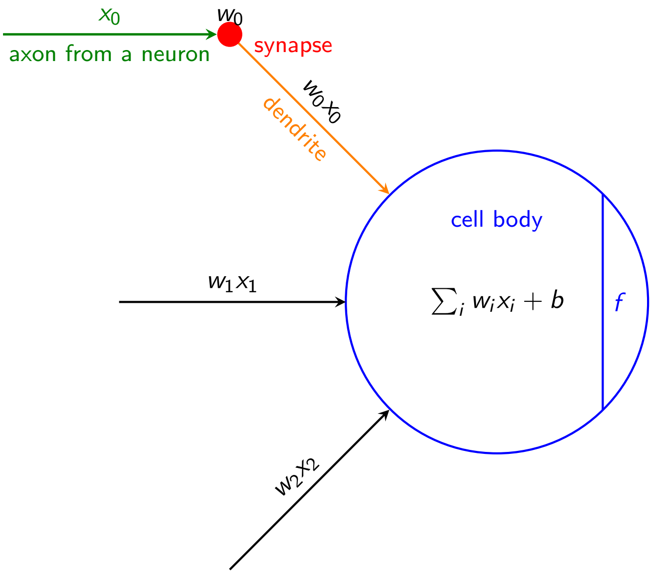
Computational neuron model



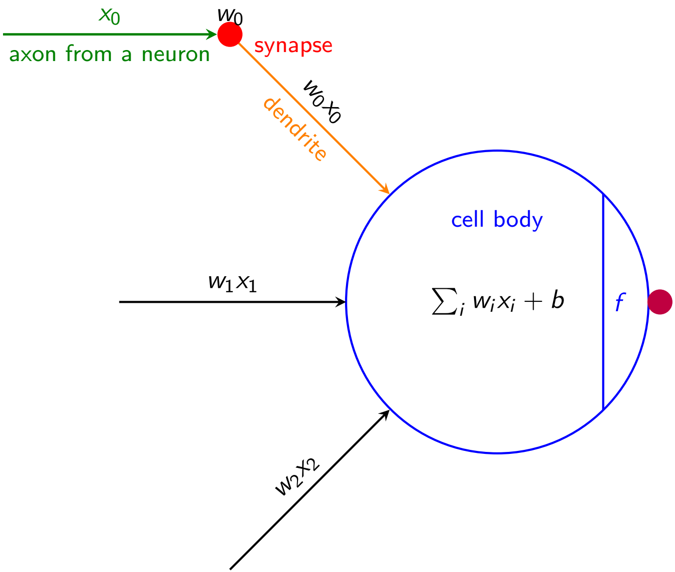
Computational neuron model



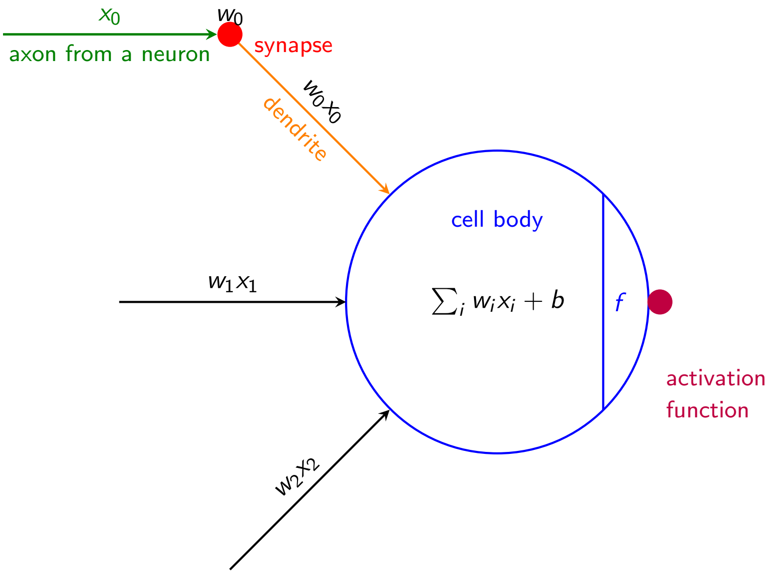
Computational neuron model



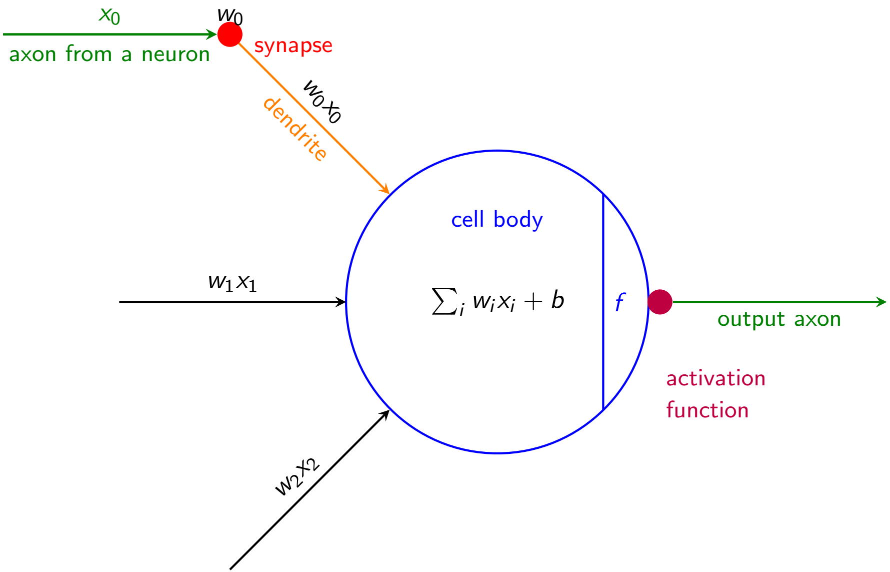
Computational neuron model



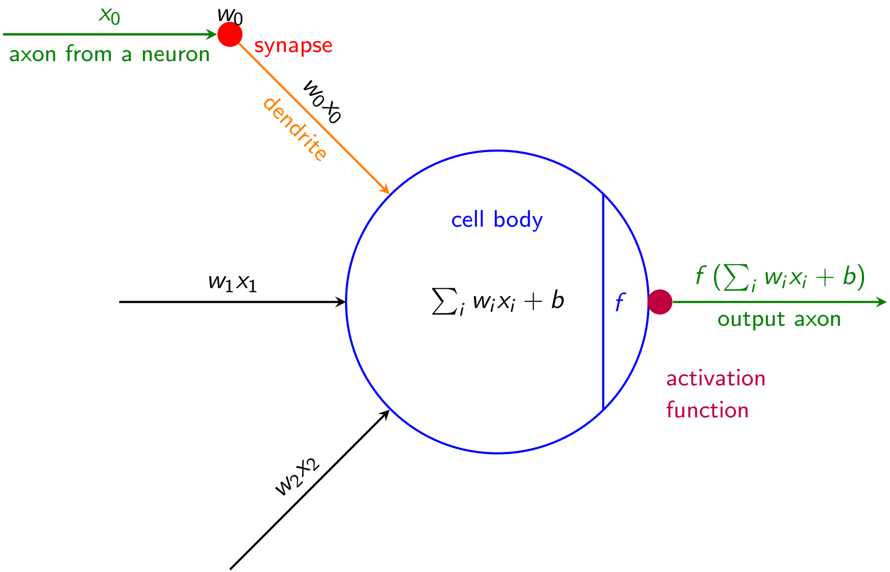
Computational neuron model



Computational neuron model



Computational neuron model



Computational neuron model (cont.)

[1]intercept *b* is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i:

[1]intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i:

[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i:

[1]intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i:

[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;

[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;
  - $w_i > 0 \rightarrow$ excitory influence

[1]intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;
  - $w_i > 0 \rightarrow$ excitory influence
  - $w_i < 0 \rightarrow$ inhibitory influence

[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;
  - $w_i > 0 \rightarrow$ excitory influence
  - $w_i < 0 \rightarrow$ inhibitory influence
- Dendrites carry signals $w_i x_i$ to the cell body, where they are summed.

[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;
  - $w_i > 0 \rightarrow$ excitory influence
  - $w_i < 0 \rightarrow$ inhibitory influence
- Dendrites carry signals $w_i x_i$ to the cell body, where they are summed.
- If the final sum $w_i x_i + b > t$ where t is a threshold[1], the neuron sends a spike along its axon (i.e. fires)

[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;
  - $w_i > 0 \rightarrow$ excitory influence
  - $w_i < 0 \rightarrow$ inhibitory influence
- Dendrites carry signals $w_i x_i$ to the cell body, where they are summed.
- If the final sum $w_i x_i + b > t$ where t is a threshold[1], the neuron sends a spike along its axon (i.e. fires)
- Computationally, the firing rate of a neuron is represented by an **activation function** f

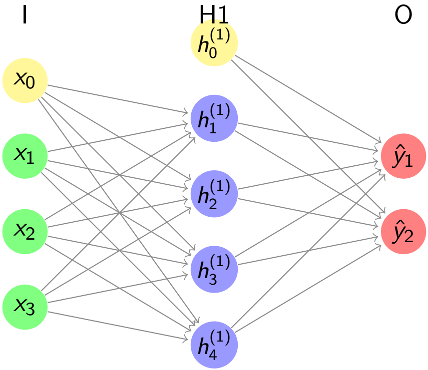
[1] intercept b is referred to as the "bias" in ML literature

Computational neuron model (cont.)

- x_i: signals traveling along axons (inputs)
- w_i: measure of synaptic strength, which is learned;
  - $w_i > 0 \rightarrow$ excitory influence
  - $w_i < 0 \rightarrow$ inhibitory influence
- Dendrites carry signals $w_i x_i$ to the cell body, where they are summed.
- If the final sum $w_i x_i + b > t$ where t is a threshold[1], the neuron sends a spike along its axon (i.e. fires)
- Computationally, the firing rate of a neuron is represented by an **activation function** f
- The output of a neuron is also called the *activation*

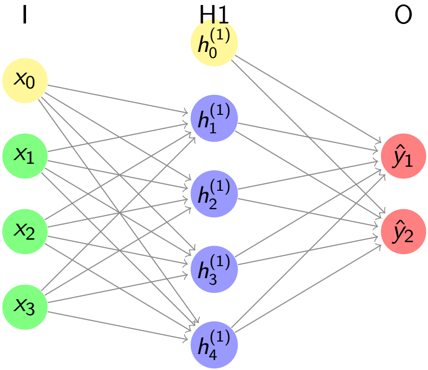
[1] intercept b is referred to as the "bias" in ML literature

Two-layer neural network (with bias neurons)



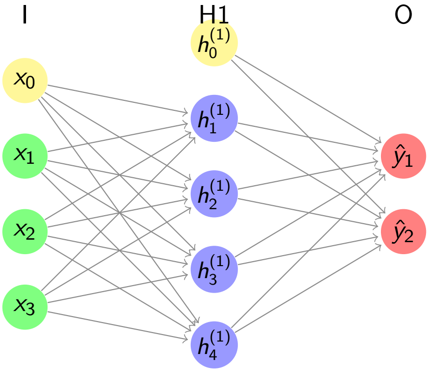
- **Layers**: 2 (input layer not counted);

Two-layer neural network (with bias neurons)



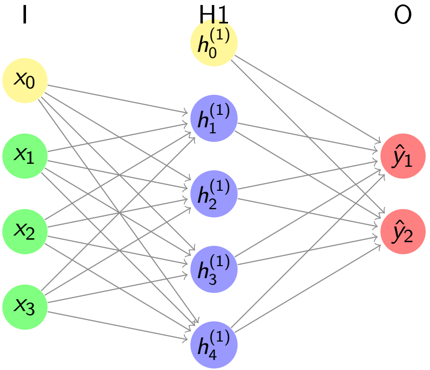
- **Layers**: 2 (input layer not counted);

Two-layer neural network (with bias neurons)



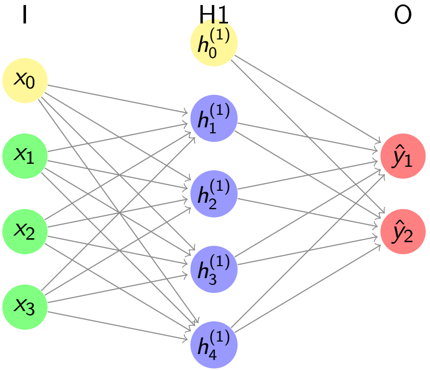
- **Layers**: 2 (input layer not counted); **Hidden layers:** 1
- **Neurons**: 7 (inputs not counted)

Two-layer neural network (with bias neurons)



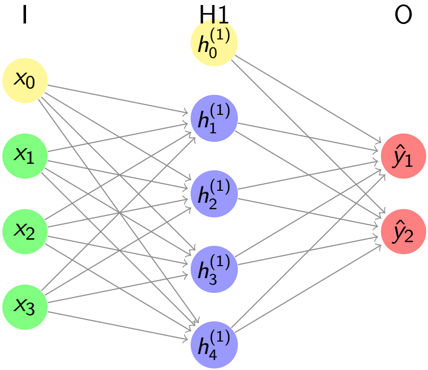
- **Layers**: 2 (input layer not counted); **Hidden layers:** 1
- **Neurons**: 7 (inputs not counted)
- **Learnable parameters:** $(4 \times 4) + (5 \times 2)$;

Two-layer neural network (with bias neurons)



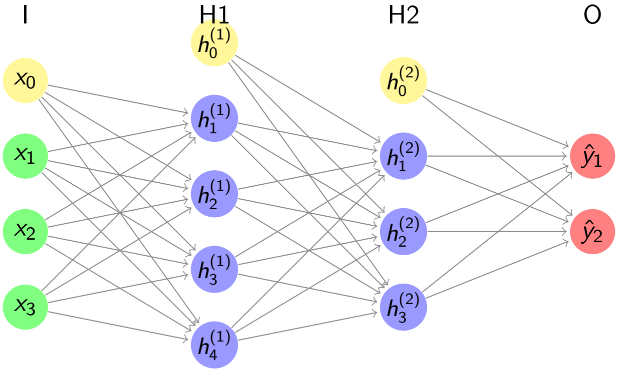
- **Layers**: 2 (input layer not counted); **Hidden layers:** 1
- **Neurons**: 7 (inputs not counted)
- **Learnable parameters:** $(4 \times 4) + (5 \times 2)$;

Two-layer neural network (with bias neurons)

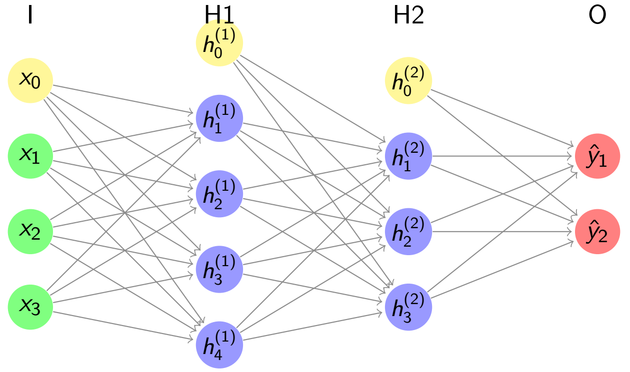


- **Layers**: 2 (input layer not counted); **Hidden layers:** 1
- **Neurons**: 7 (inputs not counted)
- **Learnable parameters:** $(4 \times 4) + (5 \times 2)$; total $= 26$

Three-layer neural network (with bias neurons)

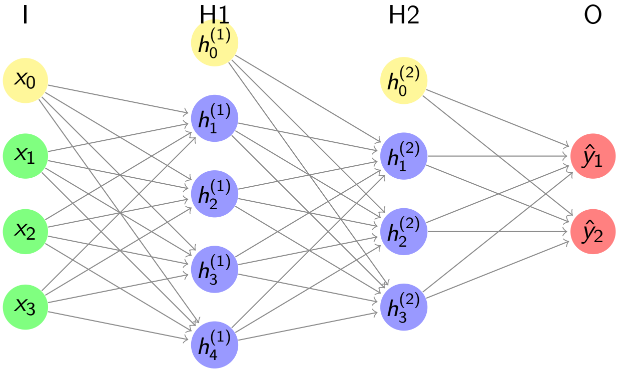


Three-layer neural network (with bias neurons)



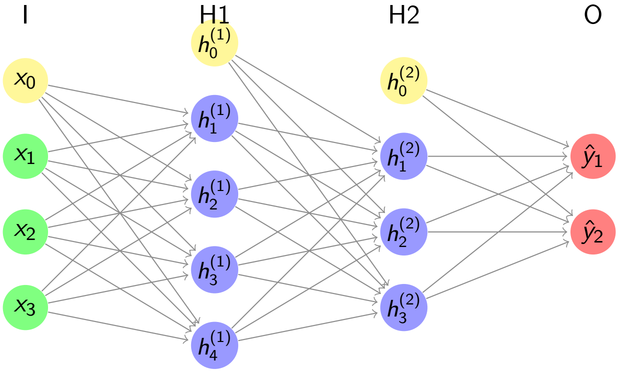
- **Layers**: 3;

Three-layer neural network (with bias neurons)



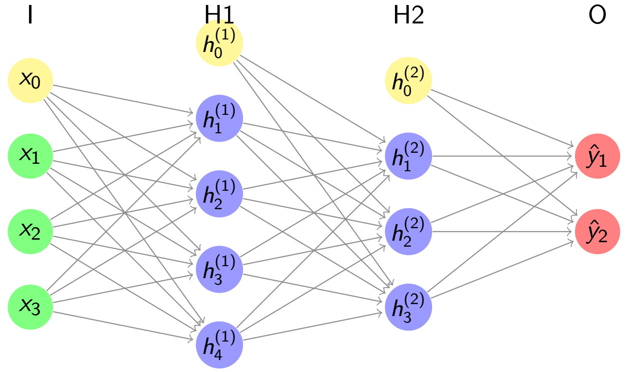
- **Layers**: 3;

Three-layer neural network (with bias neurons)



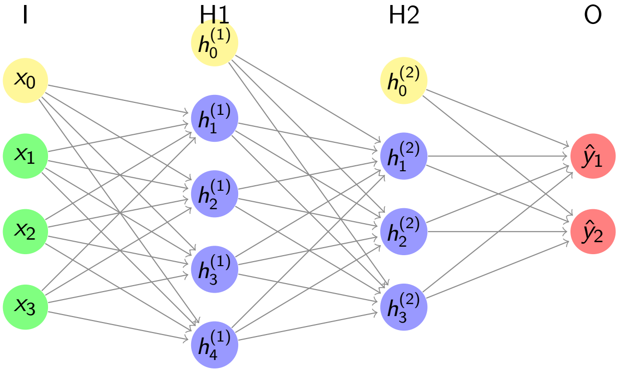
- **Layers**: 3; **Hidden layers:** 2
- **Neurons**: 9

Three-layer neural network (with bias neurons)



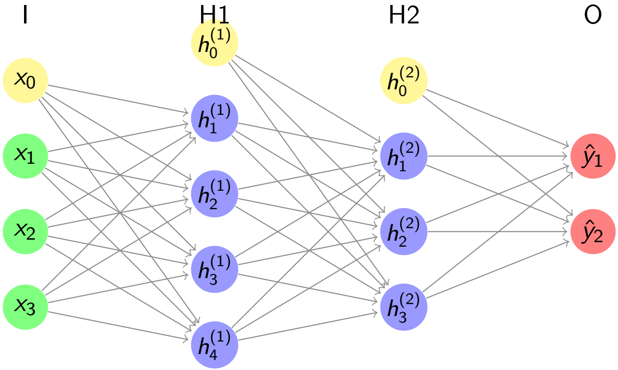
- **Layers**: 3; **Hidden layers:** 2
- **Neurons**: 9
- **Learnable parameters:** $(4 \times 4) + (5 \times 3) + (4 \times 2) =$

Three-layer neural network (with bias neurons)



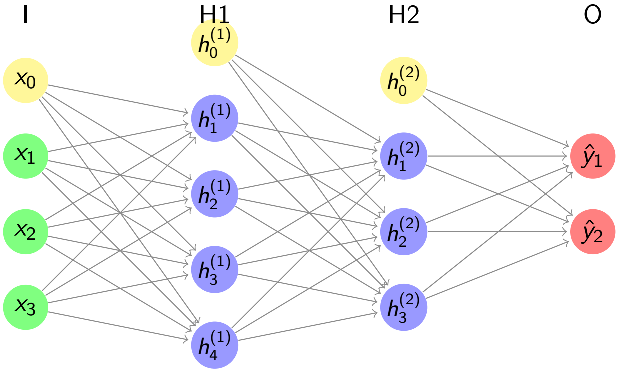
- **Layers**: 3; **Hidden layers:** 2
- **Neurons**: 9
- **Learnable parameters:** $(4 \times 4) + (5 \times 3) + (4 \times 2) =$

Three-layer neural network (with bias neurons)



- **Layers**: 3; **Hidden layers:** 2
- **Neurons**: 9
- **Learnable parameters:** $(4 \times 4) + (5 \times 3) + (4 \times 2) = 39$ weights;

Three-layer neural network (with bias neurons)



- **Layers**: 3; **Hidden layers:** 2
- **Neurons**: 9
- **Learnable parameters:** $(4 \times 4) + (5 \times 3) + (4 \times 2) = 39$ weights; total $= 39$

Activation functions

Activation functions

In an ANN, the activation function f_ℓ modulates determines whether a certain neuron "fires" or passes information (hidden units $\boldsymbol{z}_\ell$ at layer ℓ) to the subsequent layer $\ell + 1$.

Activation functions

In an ANN, the activation function f_ℓ modulates determines whether a certain neuron "fires" or passes information (hidden units $\boldsymbol{z}_\ell$ at layer ℓ) to the subsequent layer $\ell + 1$.

$$\boldsymbol{z}_\ell = f_\ell(\boldsymbol{z}_{\ell-1}) = \varphi_\ell(\boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1}) \tag{5}$$

- The input to the activation function $\boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1}$ is termed the **pre-activations**:

Activation functions

In an ANN, the activation function f_ℓ modulates determines whether a certain neuron "fires" or passes information (hidden units $\boldsymbol{z}_\ell$ at layer ℓ) to the subsequent layer $\ell + 1$.

$$\boldsymbol{z}_\ell = f_\ell(\boldsymbol{z}_{\ell-1}) = \varphi_\ell(\boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1}) \tag{5}$$

- The input to the activation function $\boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1}$ is termed the **pre-activations**:

$$\boldsymbol{a}_\ell = \boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1} \tag{6}$$

Activation functions

In an ANN, the activation function f_ℓ modulates determines whether a certain neuron "fires" or passes information (hidden units $\boldsymbol{z}_\ell$ at layer ℓ) to the subsequent layer $\ell + 1$.

$$\boldsymbol{z}_\ell = f_\ell(\boldsymbol{z}_{\ell-1}) = \varphi_\ell(\boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1}) \tag{5}$$

- The input to the activation function $\boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1}$ is termed the **pre-activations**:

$$\boldsymbol{a}_\ell = \boldsymbol{b}_\ell + \boldsymbol{W}_\ell \boldsymbol{z}_{\ell-1} \tag{6}$$

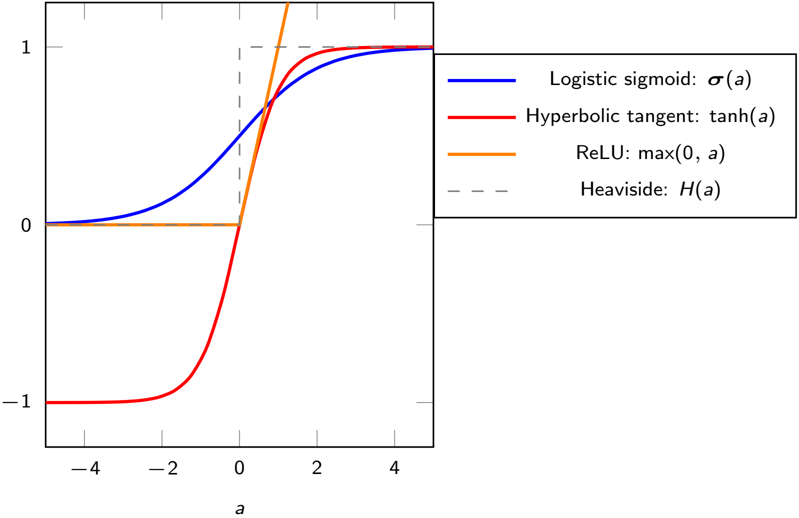
  Thus

$$\boldsymbol{z}_\ell = \varphi_\ell(\boldsymbol{a}_\ell) \tag{7}$$

- In the historic MLP, the activation function was the non-differentiable Heaviside function (difficult to train)
- Later on, the sigmoid was introduced (smooth, trainable/differentiable)

Examples of activation functions

Examples of activation functions



Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) =$$

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

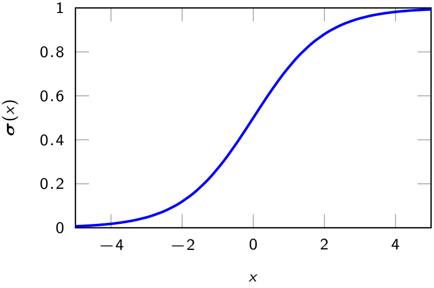
$$\boldsymbol{\sigma}(x) =$$

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.

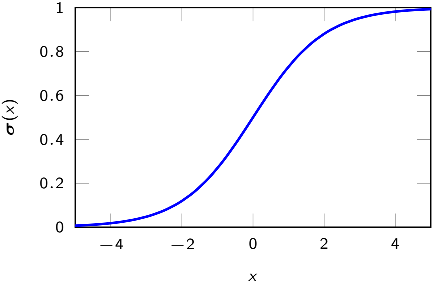


Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.



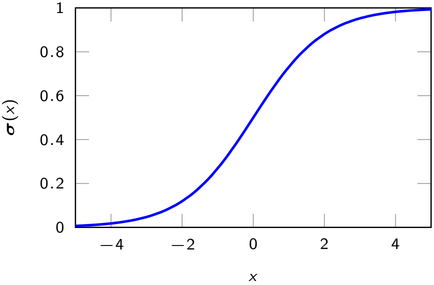
- Historically, it was used as it nicely represents the firing rate

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.



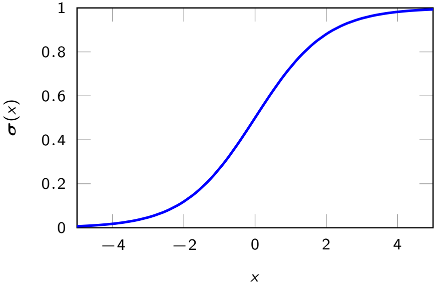
- Historically, it was used as it nicely represents the firing rate
- Recently, it has been superseded by the hyperbolic tangent due to its

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.



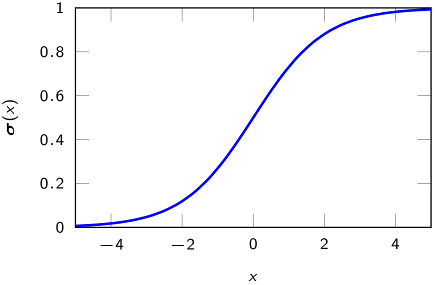
- Historically, it was used as it nicely represents the firing rate
- Recently, it has been superseded by the hyperbolic tangent due to its

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.



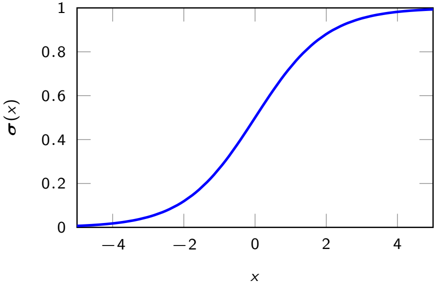
- Historically, it was used as it nicely represents the firing rate
- Recently, it has been superseded by the hyperbolic tangent due to its (a) gradient saturation

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.



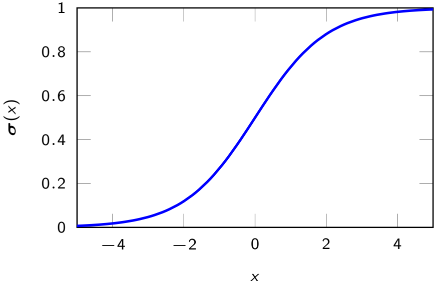
- Historically, it was used as it nicely represents the firing rate
- Recently, it has been superseded by the hyperbolic tangent due to its (a) gradient saturation and (b)

Logistic sigmoid function

- The form of the logistic sigmoid function is given by:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

- It transforms a real-valued input in the interval $[0, 1]$.



- Historically, it was used as it nicely represents the firing rate
- Recently, it has been superseded by the hyperbolic tangent due to its (a) gradient saturation and (b) non-zero-centeredness.

Hyperbolic tangent (tanh)

- The hyperbolic tangent function is given by:

$$\tanh(x) =$$

Hyperbolic tangent (tanh)

- The hyperbolic tangent function is given by:

$$\tanh(x) =$$

Hyperbolic tangent (tanh)

- The hyperbolic tangent function is given by:

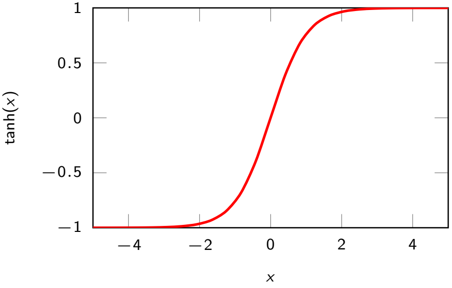
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} =$$

Hyperbolic tangent (tanh)

- The hyperbolic tangent function is given by:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\boldsymbol{\sigma}(2x) - 1 \tag{9}$$

- It transforms a real-valued input in the interval $[-1, 1]$.

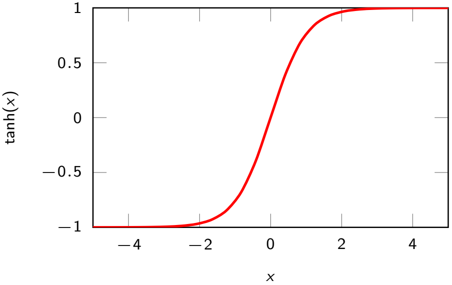


Hyperbolic tangent (tanh)

- The hyperbolic tangent function is given by:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = 2\boldsymbol{\sigma}(2x) - 1 \quad (9)$$

- It transforms a real-valued input in the interval $[-1, 1]$.



- Preferred to sigmoid activation function due to its zero-centeredness.

Rectified linear unit (ReLU)

Rectified linear unit (ReLU)

- The ReLU is given by

$$\mathrm{ReLU}(x) =$$

Rectified linear unit (ReLU)

- The ReLU is given by

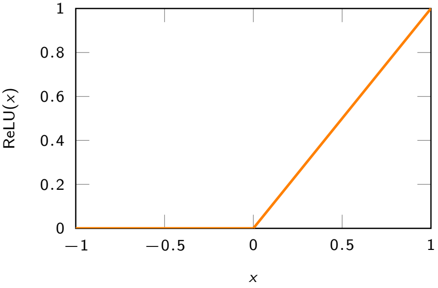
$$\text{ReLU}(x) =$$

Rectified linear unit (ReLU)

- The ReLU is given by

$$\mathrm{ReLU}(x) = \max(0, x) \tag{10}$$

- Performs a simple thresholding of input at 0.

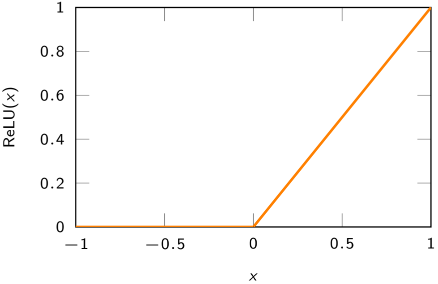


Rectified linear unit (ReLU)

- The ReLU is given by

$$\text{ReLU}(x) = \max(0, x) \tag{10}$$

- Performs a simple thresholding of input at 0.



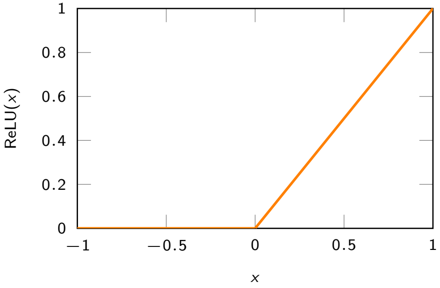
- Demonstrates faster convergence than $\boldsymbol{\sigma}(x)$ and $\tanh(x)$

Rectified linear unit (ReLU)

- The ReLU is given by

$$\mathrm{ReLU}(x) = \max(0, x) \tag{10}$$

- Performs a simple thresholding of input at 0.



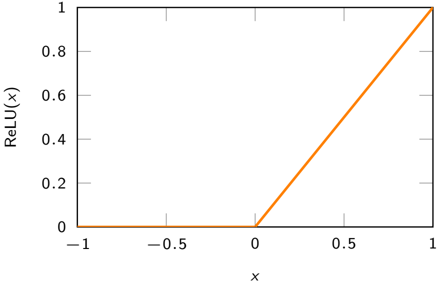
- Demonstrates faster convergence than $\sigma(x)$ and $\tanh(x)$
- Popular for deep convolutional networks (several hidden layers)

Rectified linear unit (ReLU)

- The ReLU is given by

$$\text{ReLU}(x) = \max(0, x) \tag{10}$$

- Performs a simple thresholding of input at 0.



- Demonstrates faster convergence than $\sigma(x)$ and $\tanh(x)$
- Popular for deep convolutional networks (several hidden layers)
- Neurons can be fragile, however, requiring care in selection of learning rate

Neural network notation

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Weights are denoted as $w_{to,from}$, e.g.

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Weights are denoted as $w_{to,from}$, e.g.

- $w_{2,3}^2$: from the third neuron in the layer 1 to the second neuron in layer 2

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Weights are denoted as $w_{to,from}$, e.g.

- $w_{2,3}^2$: from the third neuron in the layer 1 to the second neuron in layer 2
- The superscript is not often used, as it is clear from the context which layer we are dealing with

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Weights are denoted as $w_{to,from}$, e.g.

- $w_{2,3}^2$: from the third neuron in the layer 1 to the second neuron in layer 2
- The superscript is not often used, as it is clear from the context which layer we are dealing with

Neural network notation

The sigmoid **activation (output)** of a neuron is denoted:

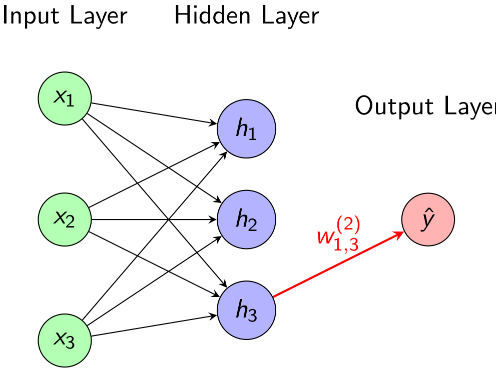
$$\varphi(w_0 z_1 + w_1 z_2 + \cdots + w_{m-1} z_{m-1} + b) = \varphi\left(\sum w_i z_i + b\right) = \text{new neuron} \quad (11)$$

Further, we denote each hidden unit as $z_{neuron}^{(layer)}$, e.g.

- $z_4^{(1)}$: fourth neuron in first layer (layers are counted from first hidden layer)

Weights are denoted as $w_{to,from}$, e.g.

- $w_{2,3}^2$: from the third neuron in the layer 1 to the second neuron in layer 2
- The superscript is not often used, as it is clear from the context which layer we are dealing with



Matrix operations in neural networks

Matrix operations in neural networks

Given the activation vector (D neurons) in the zeroth (input) layer:

$$\boldsymbol{x} \in \mathbb{R}^D = \boldsymbol{z}^{(0)} = \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} \tag{12}$$

Matrix operations in neural networks

Given the activation vector (D neurons) in the zeroth (input) layer:

$$\boldsymbol{x} \in \mathbb{R}^D = \boldsymbol{z}^{(0)} = \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} \tag{12}$$

Then the activations in the next layer (M neurons) are given by:

Matrix operations in neural networks

Given the activation vector (D neurons) in the zeroth (input) layer:

$$\boldsymbol{x} \in \mathbb{R}^D = \boldsymbol{z}^{(0)} = \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} \tag{12}$$

Then the activations in the next layer (M neurons) are given by:

$$\boldsymbol{z}^{(1)} = \varphi\left(\boldsymbol{W}\boldsymbol{z}^{(0)} + \boldsymbol{b}\right) =$$

Matrix operations in neural networks

Given the activation vector (D neurons) in the zeroth (input) layer:

$$\boldsymbol{x} \in \mathbb{R}^D = \boldsymbol{z}^{(0)} = \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} \tag{12}$$

Then the activations in the next layer (M neurons) are given by:

$$\boldsymbol{z}^{(1)} = \varphi\left(\boldsymbol{W}\boldsymbol{z}^{(0)} + \boldsymbol{b}\right) = \varphi\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,D} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M,1} & w_{M,2} & \cdots & w_{M,D} \end{bmatrix} \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix}\right) \tag{13}$$

Matrix operations in neural networks

Given the activation vector (D neurons) in the zeroth (input) layer:

$$\boldsymbol{x} \in \mathbb{R}^D = \boldsymbol{z}^{(0)} = \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} \tag{12}$$

Then the activations in the next layer (M neurons) are given by:

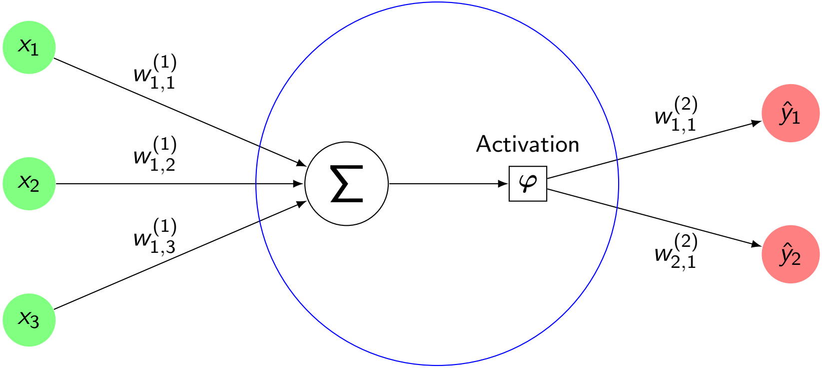
$$\boldsymbol{z}^{(1)} = \varphi\left(\boldsymbol{W}\boldsymbol{z}^{(0)} + \boldsymbol{b}\right) = \varphi\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,D} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ w_{M,1} & w_{M,2} & \cdots & w_{M,D} \end{bmatrix} \begin{bmatrix} z_1^0 \\ z_2^0 \\ \vdots \\ z_D^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_M \end{bmatrix}\right) \tag{13}$$

Example: If Layer 1 had only two neurons, then the weight matrix $\boldsymbol{W}$ would have only 2 rows.

Example: MLP with two outputs

Example: MLP with two outputs

This simple MLP has 2 layers (1 hidden, one outer), and



Example: 2-layer regression MLP

Example: 2-layer regression MLP

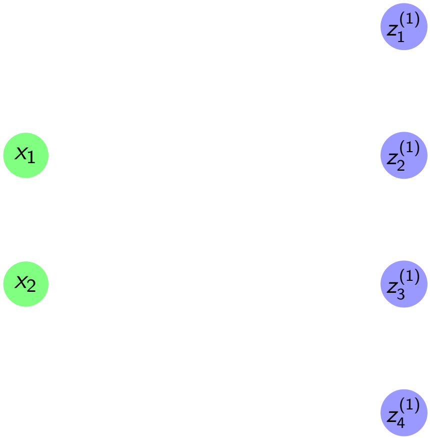
Two-layer MLP for regression

Example: 2-layer regression MLP

Two-layer MLP for regression

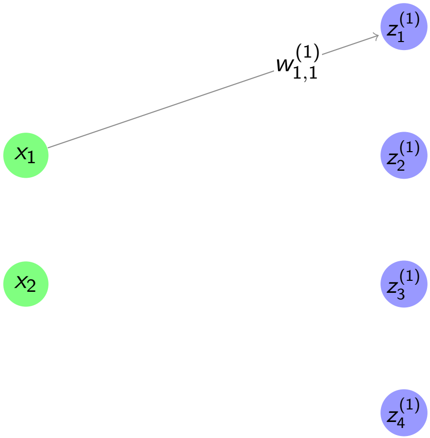
Example: 2-layer regression MLP

Two-layer MLP for regression



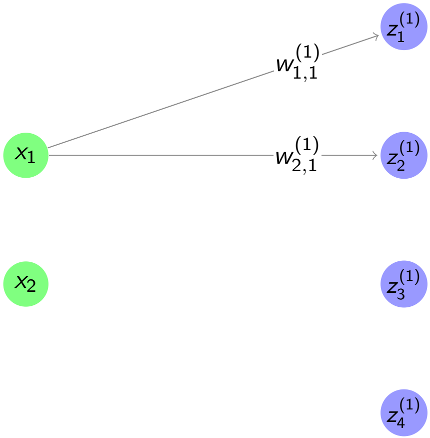
Example: 2-layer regression MLP

Two-layer MLP for regression



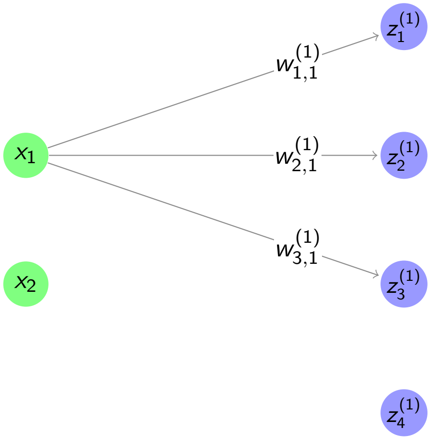
Example: 2-layer regression MLP

Two-layer MLP for regression



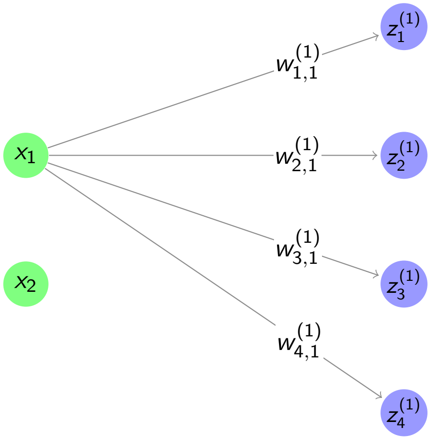
Example: 2-layer regression MLP

Two-layer MLP for regression



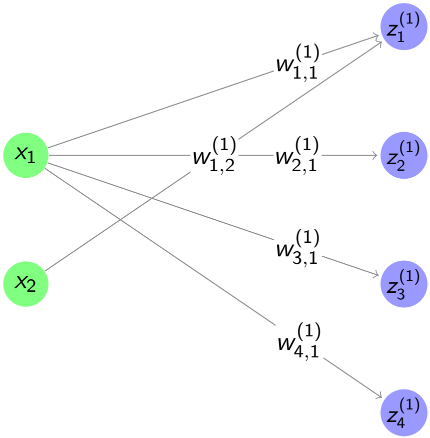
Example: 2-layer regression MLP

Two-layer MLP for regression



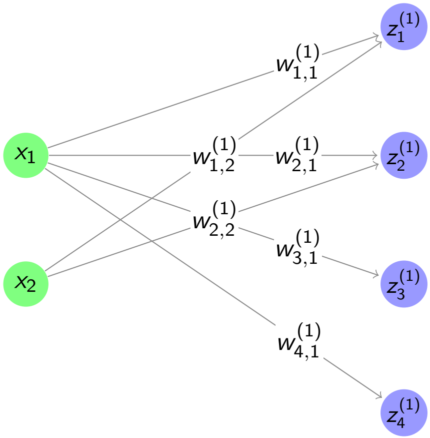
Example: 2-layer regression MLP

Two-layer MLP for regression



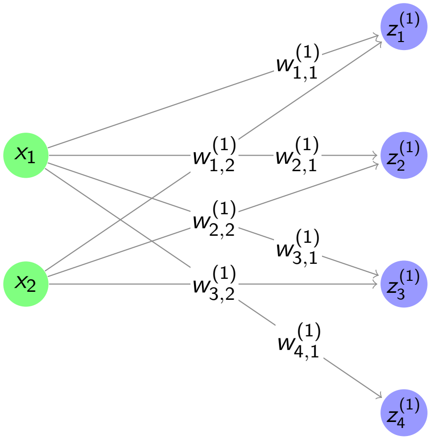
Example: 2-layer regression MLP

Two-layer MLP for regression



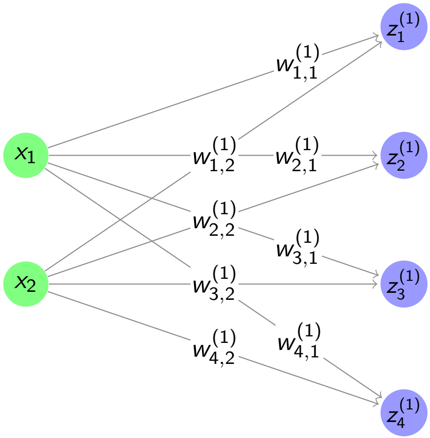
Example: 2-layer regression MLP

Two-layer MLP for regression



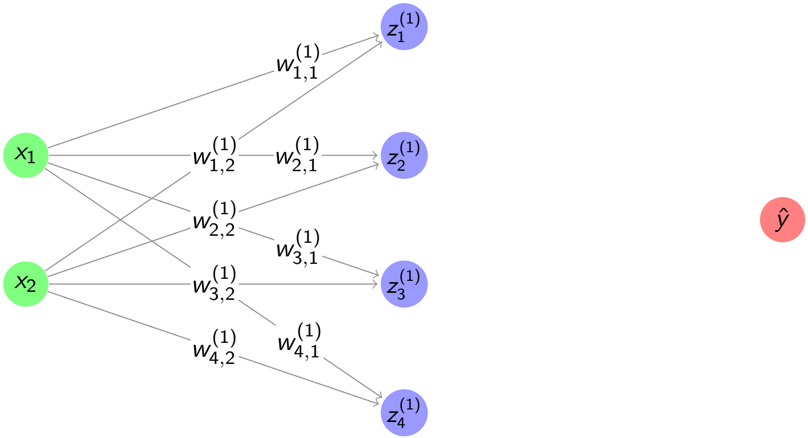
Example: 2-layer regression MLP

Two-layer MLP for regression



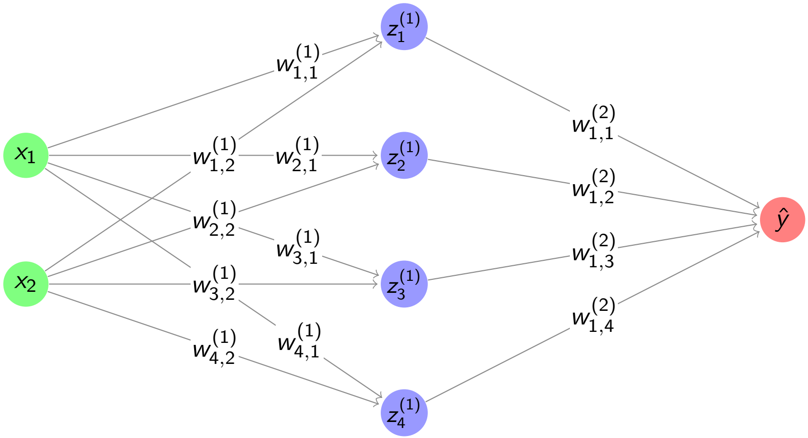
Example: 2-layer regression MLP

Two-layer MLP for regression



Example: 2-layer regression MLP

Two-layer MLP for regression



Example: 2-layer regression MLP: scalar form equations

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features,

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$z_m^{(1)} = \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right)$$

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$
\begin{aligned}
z_m^{(1)} &= \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right) \\
y_i(x_i) &=
\end{aligned}
$$

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$
\begin{aligned}
z_m^{(1)} &= \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right) \\
y_i(x_i) &= \sum_{m=1}^{M} w_{1,m}^{(2)} z_m^{(1)} + b^{(2)}
\end{aligned}
$$

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$
\begin{aligned}
z_m^{(1)} &= \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right) \\
y_i(x_i) &= \sum_{m=1}^{M} w_{1,m}^{(2)} z_m^{(1)} + b^{(2)}
\end{aligned}
$$

- D is number of input neurons

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$
\begin{aligned}
z_m^{(1)} &= \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right) \\
y_i(x_i) &= \sum_{m=1}^{M} w_{1,m}^{(2)} z_m^{(1)} + b^{(2)}
\end{aligned}
$$

- D is number of input neurons
- M is number of hidden neurons

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$z_m^{(1)} = \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right)$$

$$y_i(x_i) = \sum_{m=1}^{M} w_{1,m}^{(2)} z_m^{(1)} + b^{(2)}$$

- D is number of input neurons
- M is number of hidden neurons
- Total number of learnable parameters:

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$
\begin{aligned}
z_m^{(1)} &= \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right) \\
y_i(x_i) &= \sum_{m=1}^{M} w_{1,m}^{(2)} z_m^{(1)} + b^{(2)}
\end{aligned}
$$

- D is number of input neurons
- M is number of hidden neurons
- Total number of learnable parameters:

Example: 2-layer regression MLP: scalar form equations

Given an observation x_{nd} with $d = 1, \ldots, D$ features, these equations describe the output from a 2-layer network:

$$
\begin{aligned}
z_m^{(1)} &= \varphi\left(\sum_{d=1}^{D} w_{1,d}^{(1)} x_{nd} + b_d^{(1)}\right) \\
y_i(x_i) &= \sum_{m=1}^{M} w_{1,m}^{(2)} z_m^{(1)} + b^{(2)}
\end{aligned}
$$

- D is number of input neurons
- M is number of hidden neurons
- Total number of learnable parameters: $M(D+1)$ weights and $(D+1)$ biases
- Linear/identity activation is used in output

Neural network loss function

Neural network loss function

Given K output neurons and N observations (where f_k is the output), we can compute the loss (cost) functions C as follows.

Neural network loss function

Given K output neurons and N observations (where f_k is the output), we can compute the loss (cost) functions C as follows.

For regression:

$$\mathcal{L} =$$

Neural network loss function

Given K output neurons and N observations (where f_k is the output), we can compute the loss (cost) functions C as follows.

For regression:

$$\mathcal{L} = \sum_{k=1}^{K} \sum_{n=1}^{N} (y_{nk} - f_k(x_n))^2 \tag{14}$$

Neural network loss function

Given K output neurons and N observations (where f_k is the output), we can compute the loss (cost) functions C as follows.

For regression:

$$\mathcal{L} = \sum_{k=1}^{K} \sum_{n=1}^{N} (y_{nk} - f_k(x_n))^2 \tag{14}$$

Thus, we can write, where $K = 1$ (univariate output):

$$\mathcal{L} = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 \tag{15}$$

For classification, we use the cross-entropy (deviance) given K classes:

Neural network loss function

Given K output neurons and N observations (where f_k is the output), we can compute the loss (cost) functions C as follows.

For regression:

$$\mathcal{L} = \sum_{k=1}^{K} \sum_{n=1}^{N} (y_{nk} - f_k(x_n))^2 \tag{14}$$

Thus, we can write, where $K = 1$ (univariate output):

$$\mathcal{L} = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 \tag{15}$$

For classification, we use the cross-entropy (deviance) given K classes:

$$\mathcal{L} = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log f_k(x_n)) \tag{16}$$

Neural network loss function

Given K output neurons and N observations (where f_k is the output), we can compute the loss (cost) functions C as follows.

For regression:

$$\mathcal{L} = \sum_{k=1}^{K} \sum_{n=1}^{N} (y_{nk} - f_k(x_n))^2 \tag{14}$$

Thus, we can write, where $K = 1$ (univariate output):

$$\mathcal{L} = \sum_{n=1}^{N} (y_n - \hat{y}_n)^2 \tag{15}$$

For classification, we use the cross-entropy (deviance) given K classes:

$$\mathcal{L} = -\sum_{n=1}^{N} \sum_{k=1}^{K} y_{nk} \log f_k(x_n)) \tag{16}$$

Training a neural network

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

$$w^{\text{new}} = w^{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial w^{\text{old}}} \tag{17}$$

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

$$w^{\text{new}} = w^{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial w^{\text{old}}} \tag{17}$$

where:

- η is the learning rate

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

$$w^{\text{new}} = w^{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial w^{\text{old}}} \tag{17}$$

where:

- η is the learning rate
- $\mathcal{L}$ is the cost function (e.g. residual sum of squares)

Training a neural network

- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

$$w^{\text{new}} = w^{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial w^{\text{old}}} \tag{17}$$

where:

- η is the learning rate
- $\mathcal{L}$ is the cost function (e.g. residual sum of squares)
- w the weight

Training a neural network

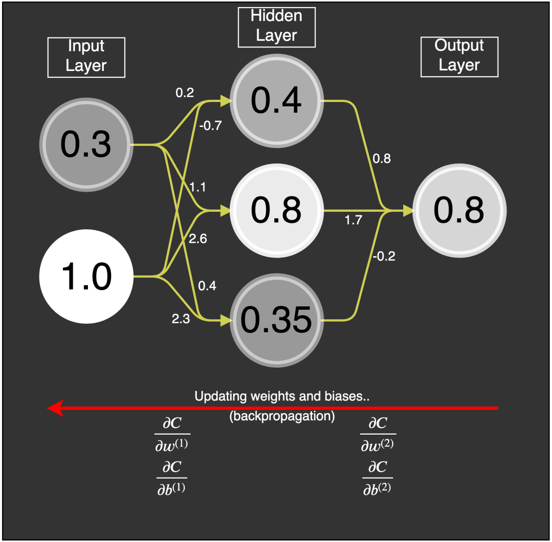
- A neural network is trained or fitted by *learning* the optimal values of the weights (and biases).
- This learning is done via optimization (e.g. gradient descent)
- Gradient descent update:

$$w^{\text{new}} = w^{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial w^{\text{old}}} \tag{17}$$

  where:
  - η is the learning rate
  - $\mathcal{L}$ is the cost function (e.g. residual sum of squares)
  - w the weight
- In neural networks, the gradients are computed via **backpropagation**

Backpropagation overview

Backpropagation overview



Training procedure

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$\boldsymbol{z}^{(\ell)} = \varphi(\boldsymbol{W}^{\ell}\boldsymbol{z}^{\ell-1} + \boldsymbol{b}^{\ell}) \tag{18}$$

- At the output layer, we compute the cost (loss) function $\mathcal{L}$ (what we want to minimize)

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$\boldsymbol{z}^{(\ell)} = \varphi(\boldsymbol{W}^{\ell}\boldsymbol{z}^{\ell-1} + \boldsymbol{b}^{\ell}) \tag{18}$$

- At the output layer, we compute the cost (loss) function $\mathcal{L}$ (what we want to minimize)
- Then, we *backpropagate* the errors through each layer in order to compute the gradients for the weight updates:

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations *a* (outputs) of each layer *l* as:

$$\boldsymbol{z}^{(\ell)} = \varphi(\boldsymbol{W}^{\ell}\boldsymbol{z}^{\ell-1} + \boldsymbol{b}^{\ell}) \tag{18}$$

- At the output layer, we compute the cost (loss) function $\mathcal{L}$ (what we want to minimize)
- Then, we *backpropagate* the errors through each layer in order to compute the gradients for the weight updates:

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$\boldsymbol{z}^{(\ell)} = \varphi(\boldsymbol{W}^{\ell}\boldsymbol{z}^{\ell-1} + \boldsymbol{b}^{\ell}) \tag{18}$$

- At the output layer, we compute the cost (loss) function $\mathcal{L}$ (what we want to minimize)
- Then, we *backpropagate* the errors through each layer in order to compute the gradients for the weight updates:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{(L)}} \frac{\partial \boldsymbol{z}^{(L)}}{\partial \boldsymbol{a}^{(L)}} \frac{\partial \boldsymbol{a}^{(L)}}{\partial \boldsymbol{W}^{(L)}} \tag{19}$$

Training procedure

- Fix initial weights and perform a forward sweep/pass through the network computing the activations *a* (outputs) of each layer *l* as:

$$\boldsymbol{z}^{(\ell)} = \varphi(\boldsymbol{W}^{\ell}\boldsymbol{z}^{\ell-1} + \boldsymbol{b}^{\ell}) \tag{18}$$

- At the output layer, we compute the cost (loss) function $\mathcal{L}$ (what we want to minimize)
- Then, we *backpropagate* the errors through each layer in order to compute the gradients for the weight updates:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}^{(L)}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{z}^{(L)}} \frac{\partial \boldsymbol{z}^{(L)}}{\partial \boldsymbol{a}^{(L)}} \frac{\partial \boldsymbol{a}^{(L)}}{\partial \boldsymbol{W}^{(L)}} \tag{19}$$

  where L is the last layer and $\boldsymbol{a} = \boldsymbol{W}\boldsymbol{z}^{\ell-1} + \boldsymbol{b}$
- Repeat the forward and backward passes until cost is sufficiently minimized

Equation summary: outer layer (regression case)

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top}\boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

$$o^{(L)} = a^{(L)} \quad \text{(linear activation or } \textit{no} \text{ activation)} \tag{21}$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

$$o^{(L)} = a^{(L)} \quad \text{(linear activation or \textit{no} activation)} \tag{21}$$

$$\mathcal{L} = (o - y)^2 \tag{22}$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

$$o^{(L)} = a^{(L)} \quad \text{(linear activation or \textit{no} activation)} \tag{21}$$

$$\mathcal{L} = (o - y)^2 \tag{22}$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top}\boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

$$o^{(L)} = a^{(L)} \quad \text{(linear activation or \textit{no} activation)} \tag{21}$$

$$\mathcal{L} = (o - y)^2 \tag{22}$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} =$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$
\begin{aligned}
a^{(L)} &= \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} && (20) \\
o^{(L)} &= a^{(L)} \quad \text{(linear activation or \textit{no} activation)} && (21) \\
\mathcal{L} &= (o - y)^2 && (22)
\end{aligned}
$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial \boldsymbol{w}^{(L)}} =
$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$
\begin{aligned}
a^{(L)} &= \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} && (20)\\
o^{(L)} &= a^{(L)} \quad \text{(linear activation or \textit{no} activation)} && (21)\\
\mathcal{L} &= (o - y)^2 && (22)
\end{aligned}
$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial \boldsymbol{w}^{(L)}} = 2\left(a^{(L)} - y\right) \boldsymbol{z}^{(L-1)} \qquad (23)
$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$
\begin{aligned}
a^{(L)} &= \boldsymbol{w}^{(L)\top}\boldsymbol{z}^{(L-1)} + b^{(L)} && (20)\\
o^{(L)} &= a^{(L)} \quad \text{(linear activation or \textit{no} activation)} && (21)\\
\mathcal{L} &= (o-y)^2 && (22)
\end{aligned}
$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} = \frac{\partial \mathcal{L}}{\partial o}\frac{\partial o^{(L)}}{\partial a^{(L)}}\frac{\partial a^{(L)}}{\partial \boldsymbol{w}^{(L)}} = 2\left(a^{(L)} - y\right)\boldsymbol{z}^{(L-1)} \tag{23}
$$

Thus, we see that this gradient depends on the activation from the previous layer $a^{(L-1)}$.

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

$$o^{(L)} = a^{(L)} \quad \text{(linear activation or \textit{no} activation)} \tag{21}$$

$$\mathcal{L} = (o - y)^2 \tag{22}$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial \boldsymbol{w}^{(L)}} = 2\left(a^{(L)} - y\right) \boldsymbol{z}^{(L-1)} \tag{23}$$

Thus, we see that this gradient depends on the activation from the previous layer $a^{(L-1)}$. Also wrt to the bias:

$$\frac{\partial \mathcal{L}}{\partial b^{(L)}} =$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$a^{(L)} = \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} \tag{20}$$

$$o^{(L)} = a^{(L)} \quad \text{(linear activation or \textit{no} activation)} \tag{21}$$

$$\mathcal{L} = (o - y)^2 \tag{22}$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial \boldsymbol{w}^{(L)}} = 2\left(a^{(L)} - y\right) \boldsymbol{z}^{(L-1)} \tag{23}$$

Thus, we see that this gradient depends on the activation from the previous layer $a^{(L-1)}$. Also wrt to the bias:

$$\frac{\partial \mathcal{L}}{\partial b^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} =$$

Equation summary: outer layer (regression case)

At the outer layer L (without indexing by neuron):

$$
\begin{aligned}
a^{(L)} &= \boldsymbol{w}^{(L)\top} \boldsymbol{z}^{(L-1)} + b^{(L)} && (20) \\
o^{(L)} &= a^{(L)} \quad \text{(linear activation or } \textit{no} \text{ activation)} && (21) \\
\mathcal{L} &= (o-y)^2 && (22)
\end{aligned}
$$

The gradient of the cost function with respect to $\boldsymbol{w}^{(L)}$ is:

$$
\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial \boldsymbol{w}^{(L)}} = 2\left(a^{(L)} - y\right) \boldsymbol{z}^{(L-1)} \tag{23}
$$

Thus, we see that this gradient depends on the activation from the previous layer $a^{(L-1)}$. Also wrt to the bias:

$$
\frac{\partial \mathcal{L}}{\partial b^{(L)}} = \frac{\partial \mathcal{L}}{\partial o} \frac{\partial o}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} = 2\left(a^{(L)} - y\right)(1) \tag{24}
$$

Updating weights

Updating weights

We can then update the weights for the last layer for the next iteration $r + 1$:

Updating weights

We can then update the weights for the last layer for the next iteration $r+1$:

$$w^{(L),r+1} = w^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L)}} \tag{25}$$

Updating weights

We can then update the weights for the last layer for the next iteration $r + 1$:

$$w^{(L),r+1} = w^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L)}} \tag{25}$$

$$b^{(L),r+1} = b^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L)}} \tag{26}$$

Updating weights

We can then update the weights for the last layer for the next iteration $r + 1$:

$$w^{(L),r+1} = w^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L)}} \quad (25)$$

$$b^{(L),r+1} = b^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L)}} \quad (26)$$

To update the weights for layer $L - 1$, we need to find the gradients $\frac{\partial \mathcal{L}}{\partial w^{(L-1)}}$ and $\frac{\partial \mathcal{L}}{\partial b^{(L-1)}}$.

Updating weights

We can then update the weights for the last layer for the next iteration $r+1$:

$$w^{(L),r+1} = w^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L)}} \quad (25)$$

$$b^{(L),r+1} = b^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L)}} \quad (26)$$

To update the weights for layer $L-1$, we need to find the gradients $\frac{\partial \mathcal{L}}{\partial w^{(L-1)}}$ and $\frac{\partial \mathcal{L}}{\partial b^{(L-1)}}$.

Using the chain rule again, we write:

Updating weights

We can then update the weights for the last layer for the next iteration $r+1$:

$$w^{(L),r+1} = w^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L)}} \quad (25)$$

$$b^{(L),r+1} = b^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L)}} \quad (26)$$

To update the weights for layer $L-1$, we need to find the gradients $\frac{\partial \mathcal{L}}{\partial w^{(L-1)}}$ and $\frac{\partial \mathcal{L}}{\partial b^{(L-1)}}$.

Using the chain rule again, we write:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \quad (27)$$

Updating weights

We can then update the weights for the last layer for the next iteration $r + 1$:

$$w^{(L),r+1} = w^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L)}} \tag{25}$$

$$b^{(L),r+1} = b^{(L),r} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L)}} \tag{26}$$

To update the weights for layer $L - 1$, we need to find the gradients $\frac{\partial \mathcal{L}}{\partial w^{(L-1)}}$ and $\frac{\partial \mathcal{L}}{\partial b^{(L-1)}}$.

Using the chain rule again, we write:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{27}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{28}$$

Backward pass

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$.

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$.
However, it is *implicitly* dependent, since

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

$$a^{(L)} \propto z^{(L)} \tag{30}$$

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

$$a^{(L)} \propto z^{(L)} \tag{30}$$

and

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

$$a^{(L)} \propto z^{(L)} \tag{30}$$

and

$$z^{(L)} \propto a^{(L-1)} \tag{31}$$

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

$$a^{(L)} \propto z^{(L)} \tag{30}$$

and

$$z^{(L)} \propto a^{(L-1)} \tag{31}$$

So, we use the chain rule to expand $\frac{\partial \mathcal{L}}{\partial a^{(L-1)}}$ as follows:

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

$$a^{(L)} \propto z^{(L)} \tag{30}$$

and

$$z^{(L)} \propto a^{(L-1)} \tag{31}$$

So, we use the chain rule to expand $\frac{\partial \mathcal{L}}{\partial a^{(L-1)}}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial a^{(L-1)}} =$$

Backward pass

But we recall that $\mathcal{L}$ is not *explicitly* dependent on $a^{(L-1)}$ as $C = (a^{(L)} - y)^2$. However, it is *implicitly* dependent, since

$$C \propto a^{(L)}, \tag{29}$$

$$a^{(L)} \propto z^{(L)} \tag{30}$$

and

$$z^{(L)} \propto a^{(L-1)} \tag{31}$$

So, we use the chain rule to expand $\frac{\partial \mathcal{L}}{\partial a^{(L-1)}}$ as follows:

$$\frac{\partial \mathcal{L}}{\partial a^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \tag{32}$$

Backward pass (cont.)

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L - 1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \quad (33)$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \quad (34)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \quad (35)$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \quad (36)$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

$$w^{(L-1),r+1} =$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

$$w^{(L-1),r+1} = w^{(L-1),r} -$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

$$w^{(L-1),r+1} = w^{(L-1),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{37}$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

$$w^{(L-1),r+1} = w^{(L-1),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{37}$$

$$b^{(L-1),r+1} =$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

$$w^{(L-1),r+1} = w^{(L-1),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{37}$$

$$b^{(L-1),r+1} = b^{(L-1),r} -$$

Backward pass (cont.)

We can then expand the cost function gradient wrt to weights for layer $L-1$ as:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{33}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial w^{(L-1)}} \tag{34}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{35}$$

$$= \frac{\partial \mathcal{L}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial z^{(L)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial b^{(L-1)}} \tag{36}$$

Once these gradients are computed, we update the weights for the $(r+1)$th iteration using:

$$w^{(L-1),r+1} = w^{(L-1),r} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{37}$$

$$b^{(L-1),r+1} = b^{(L-1),r} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L-1)}} \tag{38}$$

Summary: forward pass

1. ($r = 0$): Initialize weights and biases: $w^{(l),0}$, $b^{(l),0}$

Summary: forward pass

1. ($r = 0$): Initialize weights and biases: $w^{(l),0}$, $b^{(l),0}$
2. Perform forward pass to compute activations:

$$a^{(l),0} = w^{(l),0} \times z^{(l-1),0} + b^{(l),0} \tag{39}$$

Summary: forward pass

1. ($r = 0$): Initialize weights and biases: $w^{(l),0}$, $b^{(l),0}$
2. Perform forward pass to compute activations:

$$a^{(l),0} = w^{(l),0} \times z^{(l-1),0} + b^{(l),0} \tag{39}$$

Summary: forward pass

1. ($r = 0$): Initialize weights and biases: $w^{(l),0}$, $b^{(l),0}$
2. Perform forward pass to compute activations:

$$a^{(l),0} = w^{(l),0} \times z^{(l-1),0} + b^{(l),0} \tag{39}$$

$$z^{(l)} = \varphi(a^{(l),0}) \tag{40}$$

Summary: forward pass

1. ($r = 0$): Initialize weights and biases: $w^{(l),0}$, $b^{(l),0}$
2. Perform forward pass to compute activations:

$$a^{(l),0} = w^{(l),0} \times z^{(l-1),0} + b^{(l),0} \quad (39)$$

$$z^{(l)} = \varphi(a^{(l),0}) \quad (40)$$

At output layer:

$$a^{(L),0} = w^{(L),0} \times z^{(L-1),0} + b^{(L),0} \quad (41)$$

Summary: forward pass

1. $(r = 0)$: Initialize weights and biases: $w^{(l),0}$, $b^{(l),0}$
2. Perform forward pass to compute activations:

$$a^{(l),0} = w^{(l),0} \times z^{(l-1),0} + b^{(l),0} \quad (39)$$
$$z^{(l)} = \varphi(a^{(l),0}) \quad (40)$$

At output layer:

$$a^{(L),0} = w^{(L),0} \times z^{(L-1),0} + b^{(L),0} \quad (41)$$
$$o = \varphi(a^{(L),0}) \quad (42)$$
$$C = (o - y)^2 \quad (43)$$

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
    1. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L)}} =$$

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \qquad (44)$$

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
    1. Compute gradients:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w^{(L)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \\
\frac{\partial \mathcal{L}}{\partial b^{(L)}} &=
\end{aligned}
\tag{44}
$$

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w^{(L)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \\ \frac{\partial \mathcal{L}}{\partial b^{(L)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} \end{aligned} \tag{44}$$

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \tag{44}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} \tag{45}$$

   2. Update weights:

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \quad (44)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} \quad (45)$$

   2. Update weights:

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \tag{44}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} \tag{45}$$

   2. Update weights:

$$w^{(L),1} = w^{(L),0} - \eta \frac{\partial C^0}{\partial w^{(L)}} \tag{46}$$

Summary: backward pass—outer layer

3. Backward pass, outer layer (L):
   1. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial w^{(L)}} \tag{44}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial b^{(L)}} \tag{45}$$

   2. Update weights:

$$w^{(L),1} = w^{(L),0} - \eta \frac{\partial C^0}{\partial w^{(L)}} \tag{46}$$

$$b^{(L),1} = b^{(L),0} - \eta \frac{\partial C^0}{\partial b^{(L)}} \tag{47}$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
    3. Compute gradients:

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
    3. Compute gradients:

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
    3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \tag{48}$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \quad (48)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \quad (49)$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$
\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \qquad (48) \\
\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \qquad (49)
\end{aligned}
$$

   4. Update weights:

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \quad (48)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \quad (49)$$

   4. Update weights:

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \tag{48}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \tag{49}$$

   4. Update weights:

$$w^{(L-1),1} =$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \quad (48)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \quad (49)$$

   4. Update weights:

$$w^{(L-1),1} = w^{(L-1),0} -$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \quad (48)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \quad (49)$$

   4. Update weights:

$$w^{(L-1),1} = w^{(L-1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \quad (50)$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \tag{48}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \tag{49}$$

   4. Update weights:

$$w^{(L-1),1} = w^{(L-1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{50}$$

$$b^{(L-1),1} =$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \tag{48}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \tag{49}$$

   4. Update weights:

$$w^{(L-1),1} = w^{(L-1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{50}$$

$$b^{(L-1),1} = b^{(L-1),0} -$$

Summary: backward pass—last hidden layer

3. Backward pass, layer $(L-1)$:
   3. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial w^{(L-1)}} \tag{48}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial b^{(L-1)}} \tag{49}$$

   4. Update weights:

$$w^{(L-1),1} = w^{(L-1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-1)}} \tag{50}$$

$$b^{(L-1),1} = b^{(L-1),0} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L-1)}} \tag{51}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
    5. Compute gradients:

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
    5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \tag{52}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \tag{52}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \tag{52}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \tag{53}$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \tag{52}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \tag{53}$$

   6. Update weights:

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
    5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \quad (53)$$

    6. Update weights:

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \quad (53)$$

   6. Update weights:

$$w^{(L-2),1} =$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \quad (53)$$

   6. Update weights:

$$w^{(L-2),1} = w^{(L-2),0} -$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \quad (53)$$

   6. Update weights:

$$w^{(L-2),1} = w^{(L-2),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-2)}} \quad (54)$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \tag{52}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \tag{53}$$

   6. Update weights:

$$w^{(L-2),1} = w^{(L-2),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-2)}} \tag{54}$$

$$b^{(L-2),1} =$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \quad (53)$$

   6. Update weights:

$$w^{(L-2),1} = w^{(L-2),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-2)}} \quad (54)$$

$$b^{(L-2),1} = b^{(L-2),0} -$$

Summary: backward pass—second-to-last hidden layer

3. Backward pass, layer $(L-2)$:
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial a^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial w^{(L-2)}} \quad (52)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(L-2)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \frac{\partial a^{(L-1)}}{\partial z^{(L-2)}} \frac{\partial z^{(L-2)}}{\partial b^{(L-2)}} \frac{\partial a^{(L-2)}}{\partial b^{(L-2)}} \quad (53)$$

   6. Update weights:

$$w^{(L-2),1} = w^{(L-2),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(L-2)}} \quad (54)$$

$$b^{(L-2),1} = b^{(L-2),0} - \eta \frac{\partial \mathcal{L}}{\partial b^{(L-2)}} \quad (55)$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
    5. Compute gradients:

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
    5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}}$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}}$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{56}$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w^{(1)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \qquad (56) \\ \frac{\partial \mathcal{L}}{\partial b^{(1)}} &= \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \end{aligned}$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
    5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{56}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}}$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{56}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \tag{57}$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \quad (56)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \quad (57)$$

   6. Update weights:

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{56}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \tag{57}$$

   6. Update weights:

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \quad (56)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \quad (57)$$

   6. Update weights:

$$w^{(1),1} =$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \quad (56)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \quad (57)$$

   6. Update weights:

$$w^{(1),1} = w^{(1),0} -$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \quad (56)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \quad (57)$$

   6. Update weights:

$$w^{(1),1} = w^{(1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(1)}} \quad (58)$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
    5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \quad (56)$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \quad (57)$$

    6. Update weights:

$$w^{(1),1} = w^{(1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(1)}} \quad (58)$$

$$b^{(1),1} =$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{56}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \tag{57}$$

   6. Update weights:

$$w^{(1),1} = w^{(1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(1)}} \tag{58}$$

$$b^{(1),1} = b^{(1),0} -$$

Summary: backward pass—first hidden layer

3. Backward pass, layer (1):
   5. Compute gradients:

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{56}$$

$$\frac{\partial \mathcal{L}}{\partial b^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(L)}} \frac{\partial o^{(L)}}{\partial a^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L-1)}} \frac{\partial z^{(L-1)}}{\partial a^{(L-1)}} \cdots \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial b^{(1)}} \tag{57}$$

   6. Update weights:

$$w^{(1),1} = w^{(1),0} - \eta \frac{\partial \mathcal{L}}{\partial w^{(1)}} \tag{58}$$

$$b^{(1),1} = b^{(1),0} - \eta \frac{\partial \mathcal{L}}{\partial b^{(1)}} \tag{59}$$

Summary of backpropagation

Summary of backpropagation

1. Fix initial weights $w^{(l),0}$, $b^{(l),0}$ and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

Summary of backpropagation

1. Fix initial weights $w^{(l),0}$, $b^{(l),0}$ and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

Summary of backpropagation

1. Fix initial weights $w^{(l),0}$, $b^{(l),0}$ and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$a^{(l)} = \varphi(\boldsymbol{W}^{(l)} z^{(l-1)} + b^{(l)}) \tag{60}$$

2. At the output layer, we compute the cost function C (what we want to minimize)

Summary of backpropagation

1. Fix initial weights $w^{(l),0}$, $b^{(l),0}$ and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$a^{(l)} = \varphi(\boldsymbol{W}^{(l)} z^{(l-1)} + b^{(l)}) \tag{60}$$

2. At the output layer, we compute the cost function C (what we want to minimize)
3. Then, we *backpropagate* the errors through each layer in order to compute the gradients $\frac{\partial \mathcal{L}}{\partial w^{(l)}}$, $\frac{\partial \mathcal{L}}{\partial b^{(l)}}$

Summary of backpropagation

1. Fix initial weights $w^{(l),0}$, $b^{(l),0}$ and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$a^{(l)} = \varphi(\boldsymbol{W}^{(l)} z^{(l-1)} + b^{(l)}) \tag{60}$$

2. At the output layer, we compute the cost function C (what we want to minimize)
3. Then, we *backpropagate* the errors through each layer in order to compute the gradients $\frac{\partial \mathcal{L}}{\partial w^{(l)}}$, $\frac{\partial \mathcal{L}}{\partial b^{(l)}}$

Summary of backpropagation

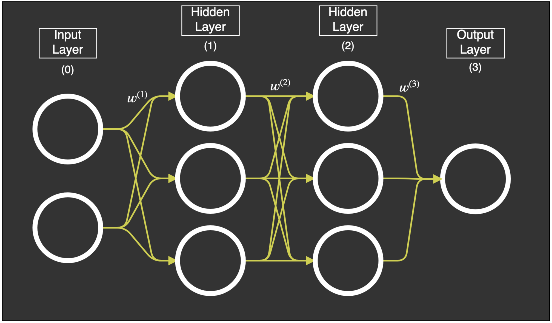
1. Fix initial weights $w^{(l),0}$, $b^{(l),0}$ and perform a forward sweep/pass through the network computing the activations a (outputs) of each layer l as:

$$a^{(l)} = \varphi(\boldsymbol{W}^{(l)} z^{(l-1)} + b^{(l)}) \tag{60}$$

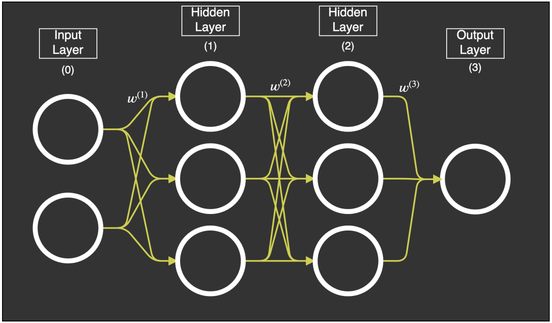
2. At the output layer, we compute the cost function C (what we want to minimize)
3. Then, we *backpropagate* the errors through each layer in order to compute the gradients $\frac{\partial \mathcal{L}}{\partial w^{(l)}}$, $\frac{\partial \mathcal{L}}{\partial b^{(l)}}$ and weight updates $w^{(l),r+1}$ and $b^{(l),r+1}$
4. Repeat the forward and backward passes until cost is sufficiently minimized

Example: backpropagation for 3-layer network

Example: backpropagation for 3-layer network

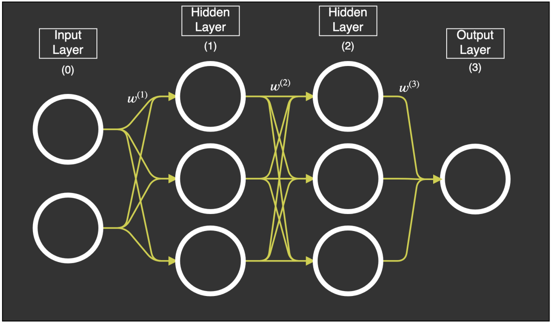


Example: backpropagation for 3-layer network



$$\frac{\partial \mathcal{L}}{\partial w^{(3)}} = \frac{\partial \mathcal{L}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial w^{(3)}} \tag{61}$$

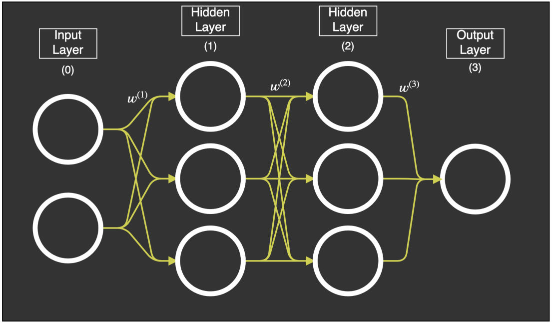
Example: backpropagation for 3-layer network



$$\frac{\partial \mathcal{L}}{\partial w^{(3)}} = \frac{\partial \mathcal{L}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial w^{(3)}} \tag{61}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial w^{(2)}} \tag{62}$$

Example: backpropagation for 3-layer network



$$\frac{\partial \mathcal{L}}{\partial w^{(3)}} = \frac{\partial \mathcal{L}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial w^{(3)}} \tag{61}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(2)}} = \frac{\partial \mathcal{L}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial w^{(2)}} \tag{62}$$

$$\frac{\partial \mathcal{L}}{\partial w^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(3)}} \frac{\partial o^{(3)}}{\partial a^{(3)}} \frac{\partial a^{(3)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial w^{(1)}} \tag{63}$$

Regression MLP architecture

Regression MLP architecture

Typical hyperparameter values are:

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	**Value**
# input neurons	1 per input feature

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	Value
# input neurons	1 per input feature
# hidden layers	Usually 1 – 5

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	Value
# input neurons	1 per input feature
# hidden layers	Usually 1 – 5
# neurons per hidden layer	Usually 10 – 100

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	Value
# input neurons	1 per input feature
# hidden layers	Usually 1 – 5
# neurons per hidden layer	Usually 10 – 100
# output neurons	1 per prediction dimension

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	Value
# input neurons	1 per input feature
# hidden layers	Usually 1 – 5
# neurons per hidden layer	Usually 10 – 100
# output neurons	1 per prediction dimension
hidden layer activation	ReLU

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	Value
# input neurons	1 per input feature
# hidden layers	Usually 1 – 5
# neurons per hidden layer	Usually 10 – 100
# output neurons	1 per prediction dimension
hidden layer activation	ReLU
output activation	None (if unbounded)

Regression MLP architecture

Typical hyperparameter values are:

Hyperparameter	**Value**
# input neurons	1 per input feature
# hidden layers	Usually 1 – 5
# neurons per hidden layer	Usually 10 – 100
# output neurons	1 per prediction dimension
hidden layer activation	ReLU
output activation	None (if unbounded)
loss function	MSE or MAE/Huber

Classification MLP architecture

Classification MLP architecture

- For classifcation, input and hidden layers are chosen in similar fashion to the regression case

Classification MLP architecture

- For classifcation, input and hidden layers are chosen in similar fashion to the regression case
- However, the number of output neurons is given by the name of classes/labels
- The output layer activation is typically the softmax function:

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}} \tag{64}$$

where z_k is the unnormalized log probability of each class k

Classification MLP architecture

- For classifcation, input and hidden layers are chosen in similar fashion to the regression case
- However, the number of output neurons is given by the name of classes/labels
- The output layer activation is typically the softmax function:

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}} \tag{64}$$

where z_k is the unnormalized log probability of each class k

- The loss function is taken as the cross entropy

Other types of neural networks

Other types of neural networks

The standard ANN architecture (MLP) we have studied is also called the feed-forward network.

Other architectures have been shown to give better performance for various applications:

Other types of neural networks

The standard ANN architecture (MLP) we have studied is also called the feed-forward network.

Other architectures have been shown to give better performance for various applications:

- Recurrent neural networks (RNNs): time-series forecasting

Other types of neural networks

The standard ANN architecture (MLP) we have studied is also called the feed-forward network.

Other architectures have been shown to give better performance for various applications:

- Recurrent neural networks (RNNs): time-series forecasting
- Convolutional neural networks (CNNs): image classification

Other types of neural networks

The standard ANN architecture (MLP) we have studied is also called the feed-forward network.

Other architectures have been shown to give better performance for various applications:

- Recurrent neural networks (RNNs): time-series forecasting
- Convolutional neural networks (CNNs): image classification
- Long short-term memory networks (LSTMs): time-series, pattern identification, etc.

Reading

We will discuss the CNN on Wednesday, along with examples in Python.

- **PMLI**: 13.1-3

Reading

We will discuss the CNN on Wednesday, along with examples in Python.

- **PMLI**: 13.1-3
- **PML**: 8.3, 9.4

Reading

We will discuss the CNN on Wednesday, along with examples in Python.

- **PMLI**: 13.1-3
- **PML**: 8.3, 9.4
- **ESL**: 11

Reading

We will discuss the CNN on Wednesday, along with examples in Python.

- **PMLI**: 13.1-3
- **PML**: 8.3, 9.4
- **ESL**: 11
- **DL**: 6-8, 11, 12

Reading

We will discuss the CNN on Wednesday, along with examples in Python.

- **PMLI**: 13.1-3
- **PML**: 8.3, 9.4
- **ESL**: 11
- **DL**: 6-8, 11, 12
- Experiment in this **playground**