Neural Networks

Lecture 1 - Introduction

Igor Janos

Agenda

1. Introduction to AI
    - AI, Machine Learning, Representation Learning, Deep Learning
    - Brief History
2. Introduction to Neural Networks
    - Perceptron, Multi-layer Perceptron
    - Activation Functions
3. Training of Neural Networks
    - Gradient Descent
    - Back-propagation of Errors

1 Introduction to AI

Artificial Intelligence

Artificial Intelligence

- "*The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings”*

Encyclopedia Britannica

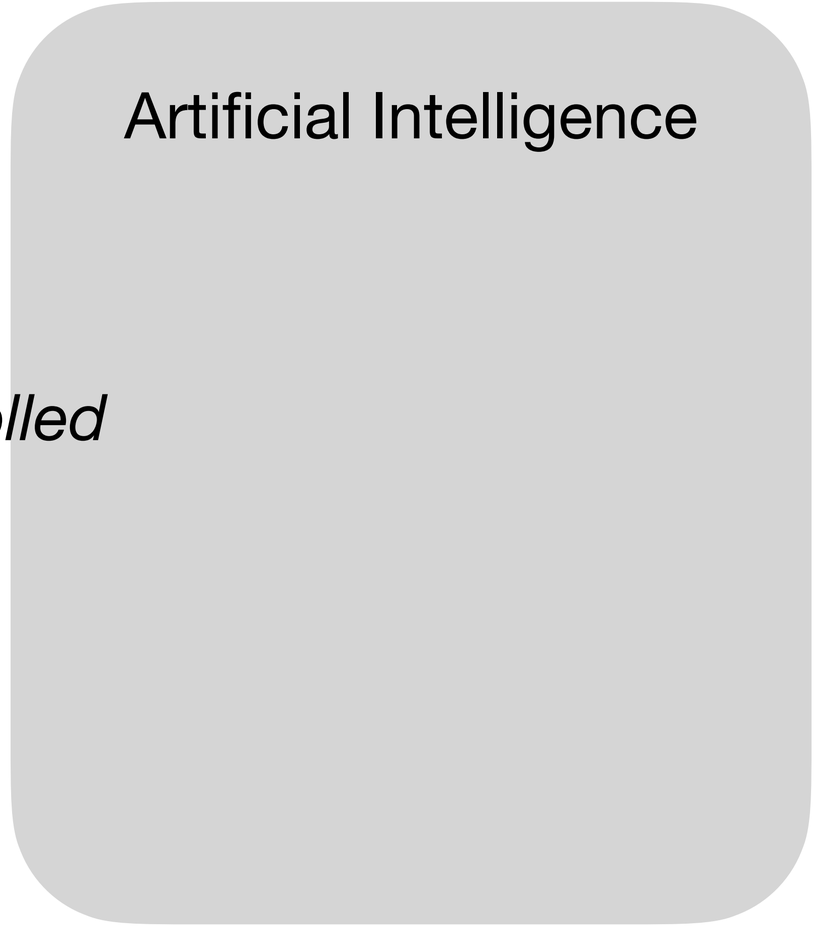
Artificial Intelligence

Artificial Intelligence

- "*The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings”*
- *1956 - Founded as an academic discipline*

Encyclopedia Britannica

Artificial Intelligence

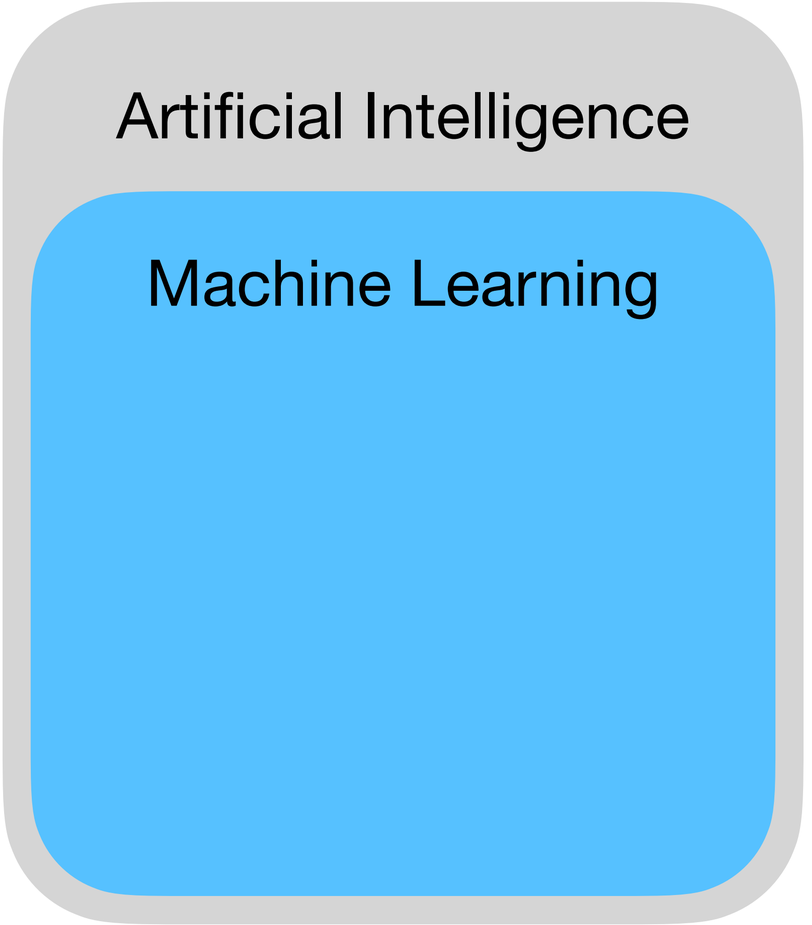


- *"The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings"*
- *1956 - Founded as an academic discipline*
- *Problem-solving techniques*
    - *Search & mathematical optimization*
    - *Formal logic, Artificial neural networks, …*

Encyclopedia Britannica

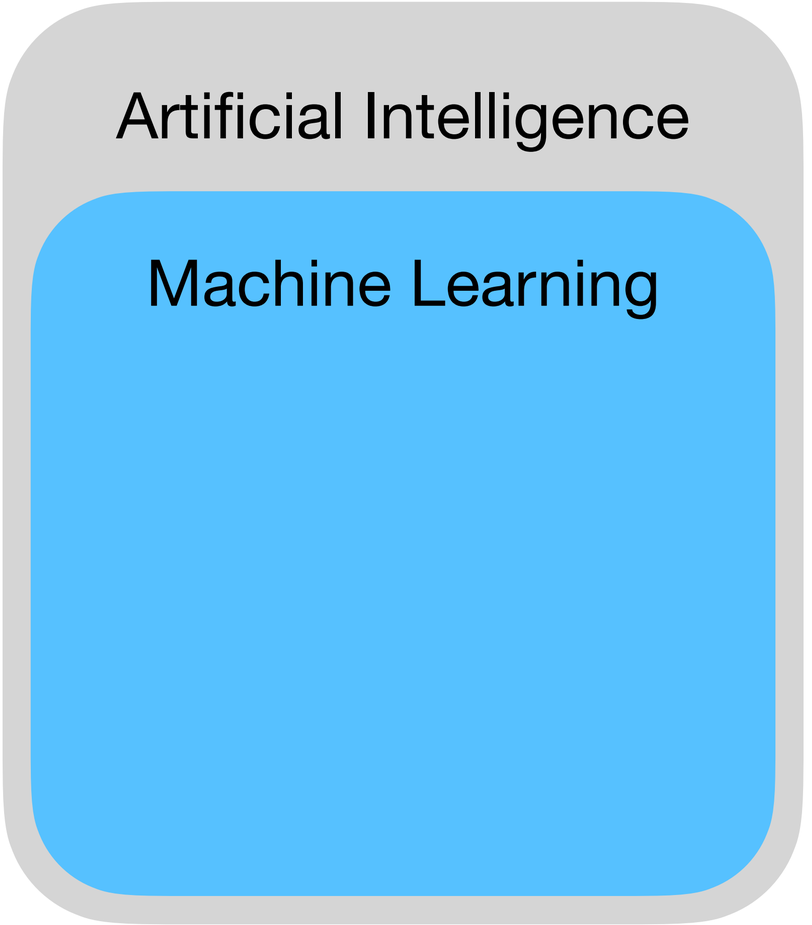
Machine Learning

- **Field of study** in Artificial Intelligence



Machine Learning

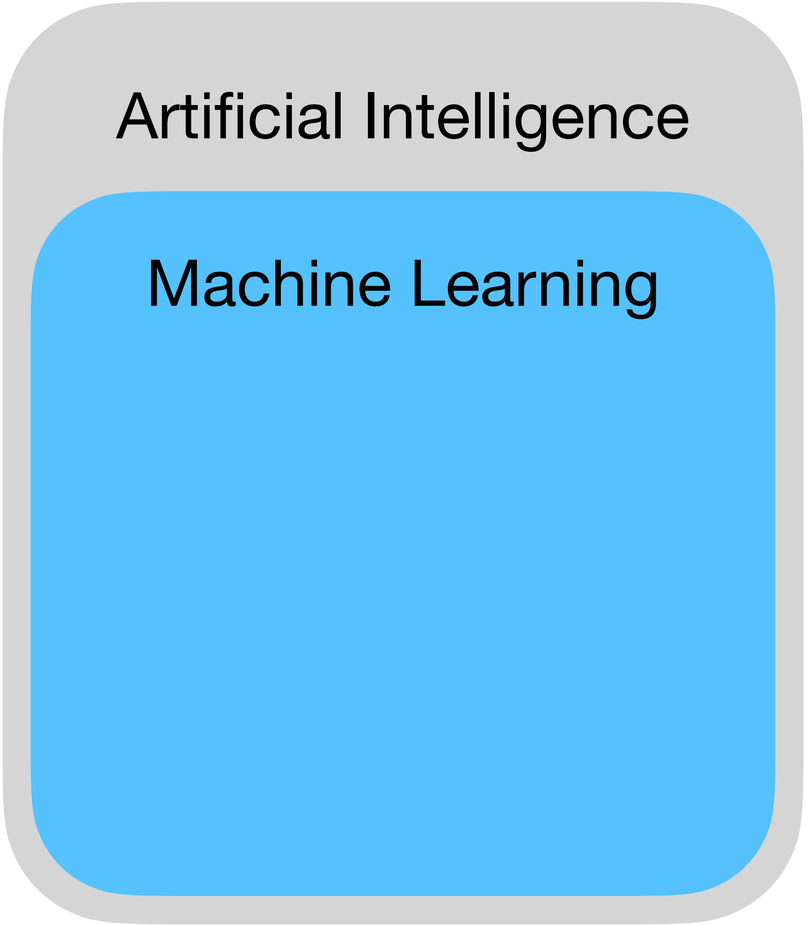
- **Field of study** in Artificial Intelligence
  - Learn from data
  - Generalize to unseen data



Machine Learning

Paradigms

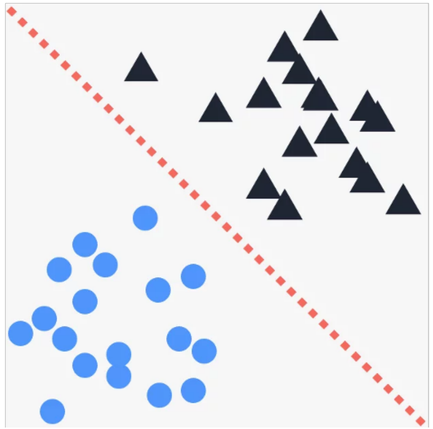
1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning



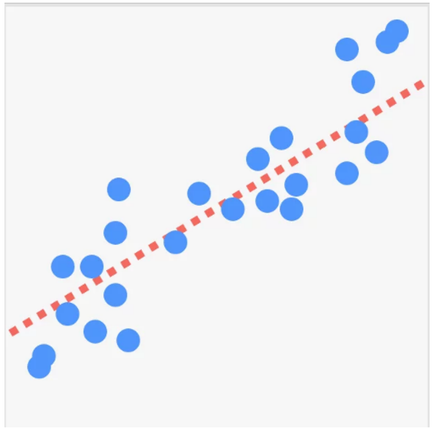
Machine Learning

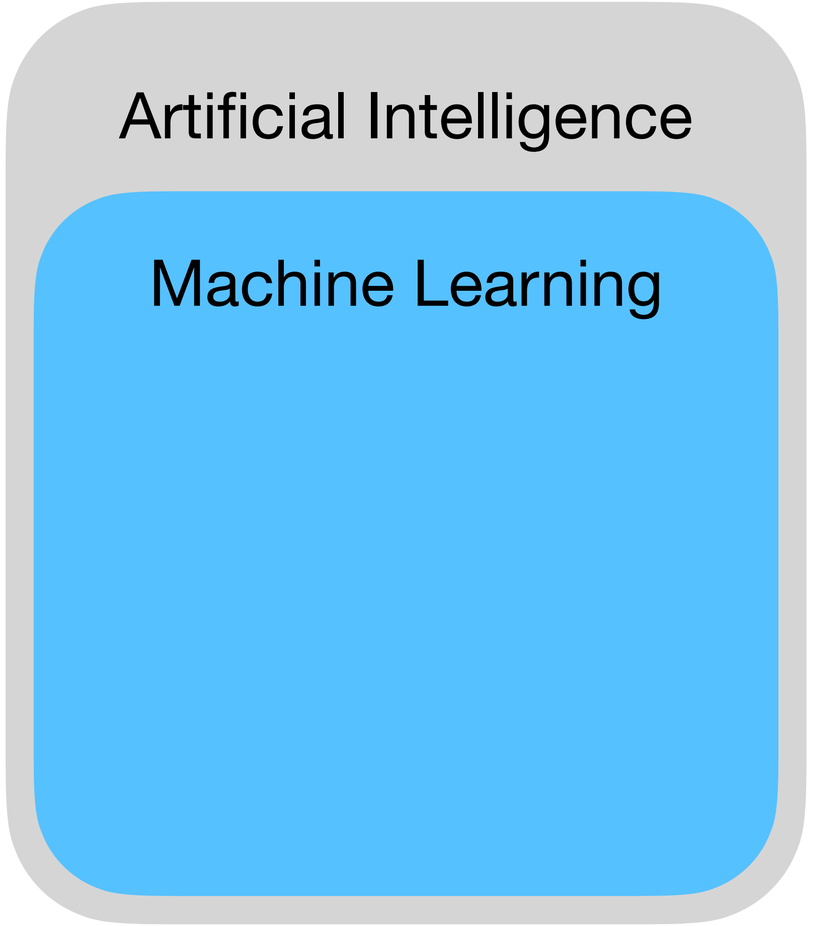
Basic Tasks

1. Classification



2. Regression

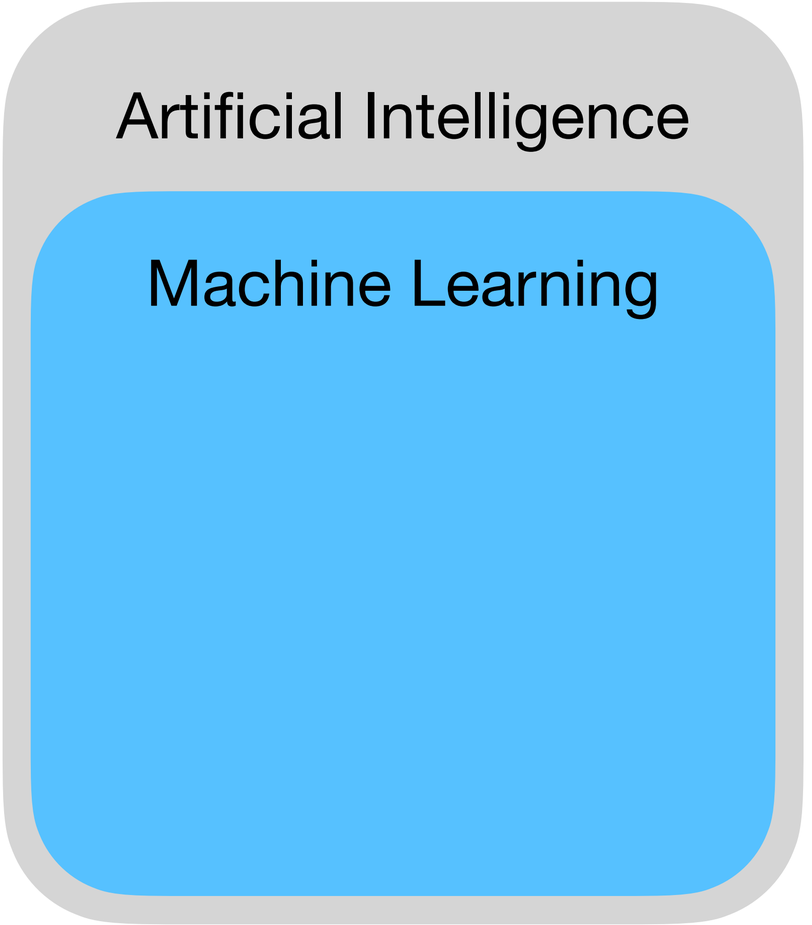




Machine Learning

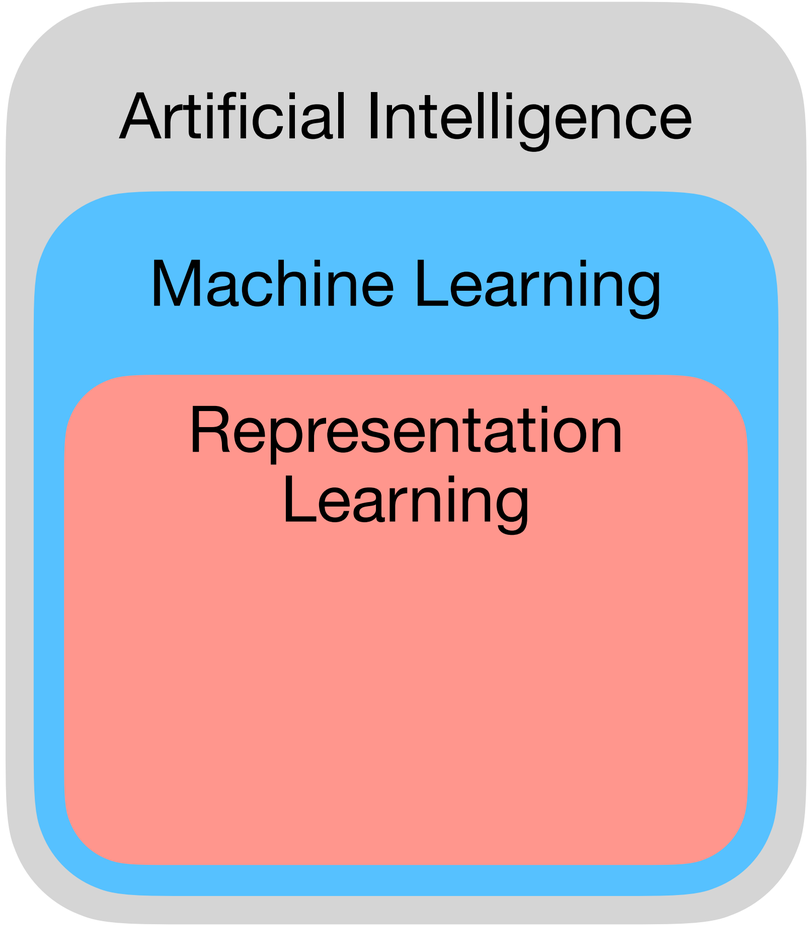
Algorithms

- Regression - Linear, Logistic, …
- Clustering - K-Means, K-Medians, Hierarchical, …
- Dimensionality Reduction - PCA, PCR, LDA, …
- Decision Trees
- Support Vector Machines
- **Artificial Neural Networks**
- …



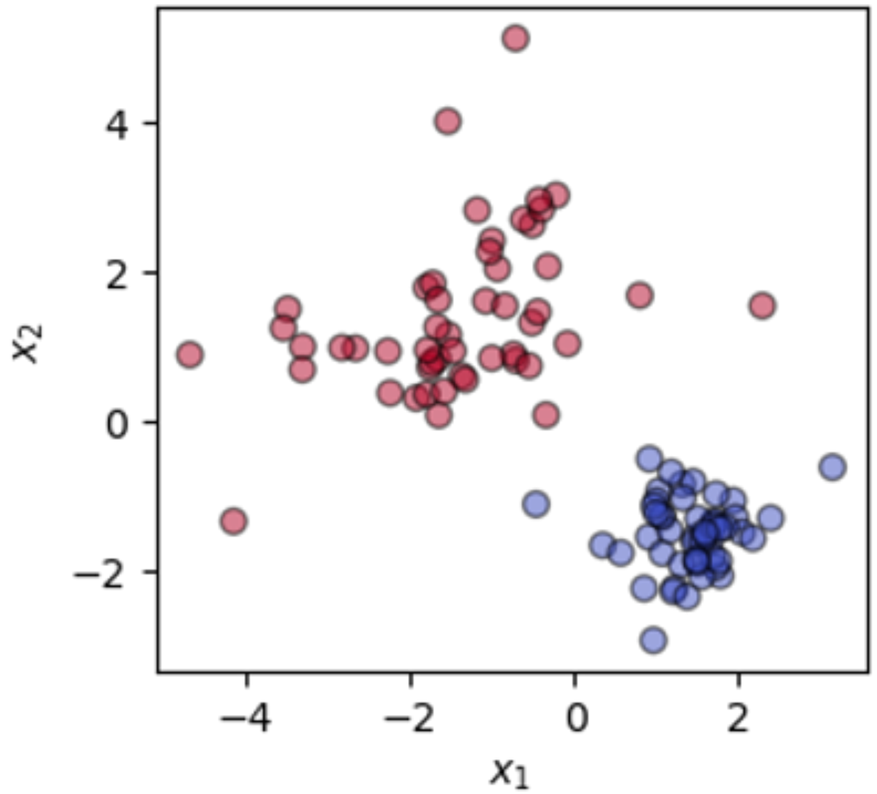
Representation Learning

- Representation = **Transformation** of input features
- **Better suited** for a given task than the original input



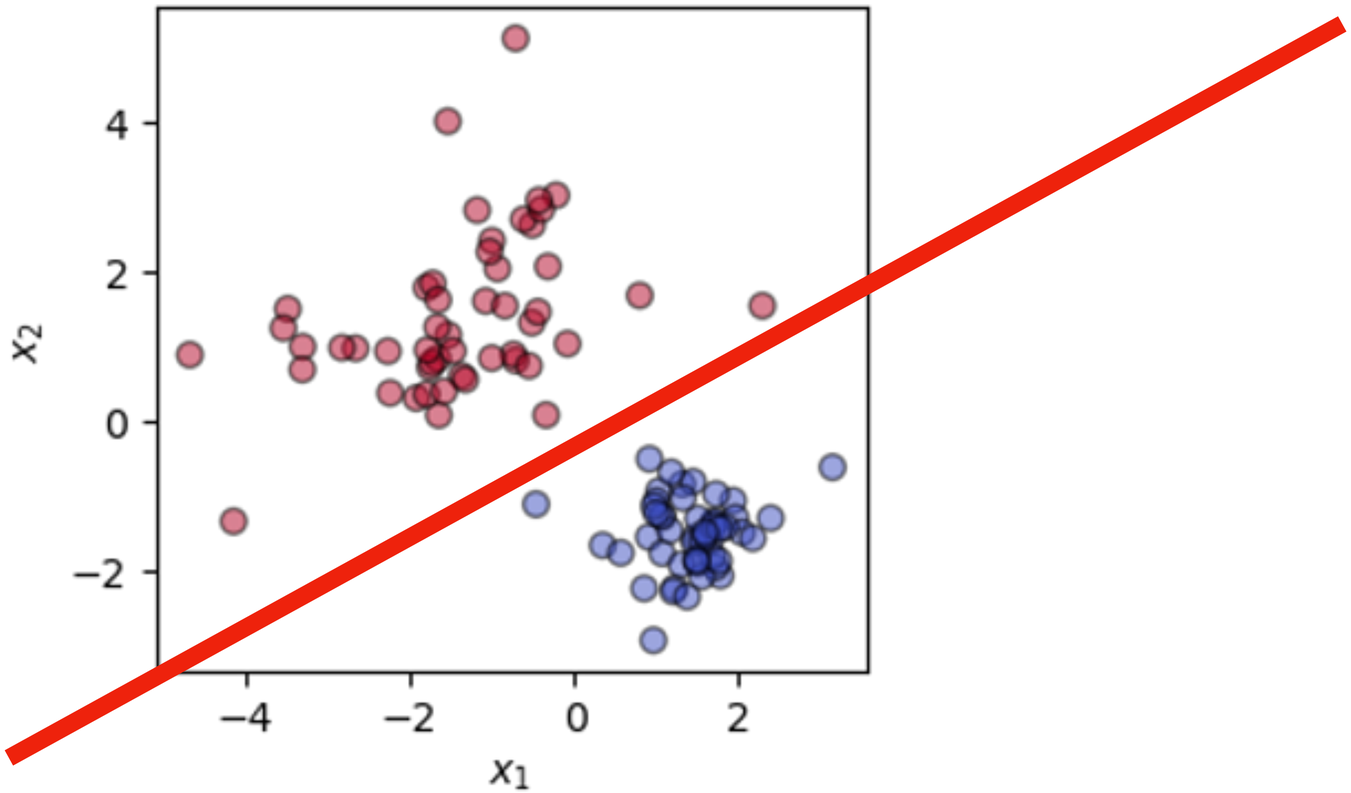
Representation Learning

Task - Find The Simplest Classifier



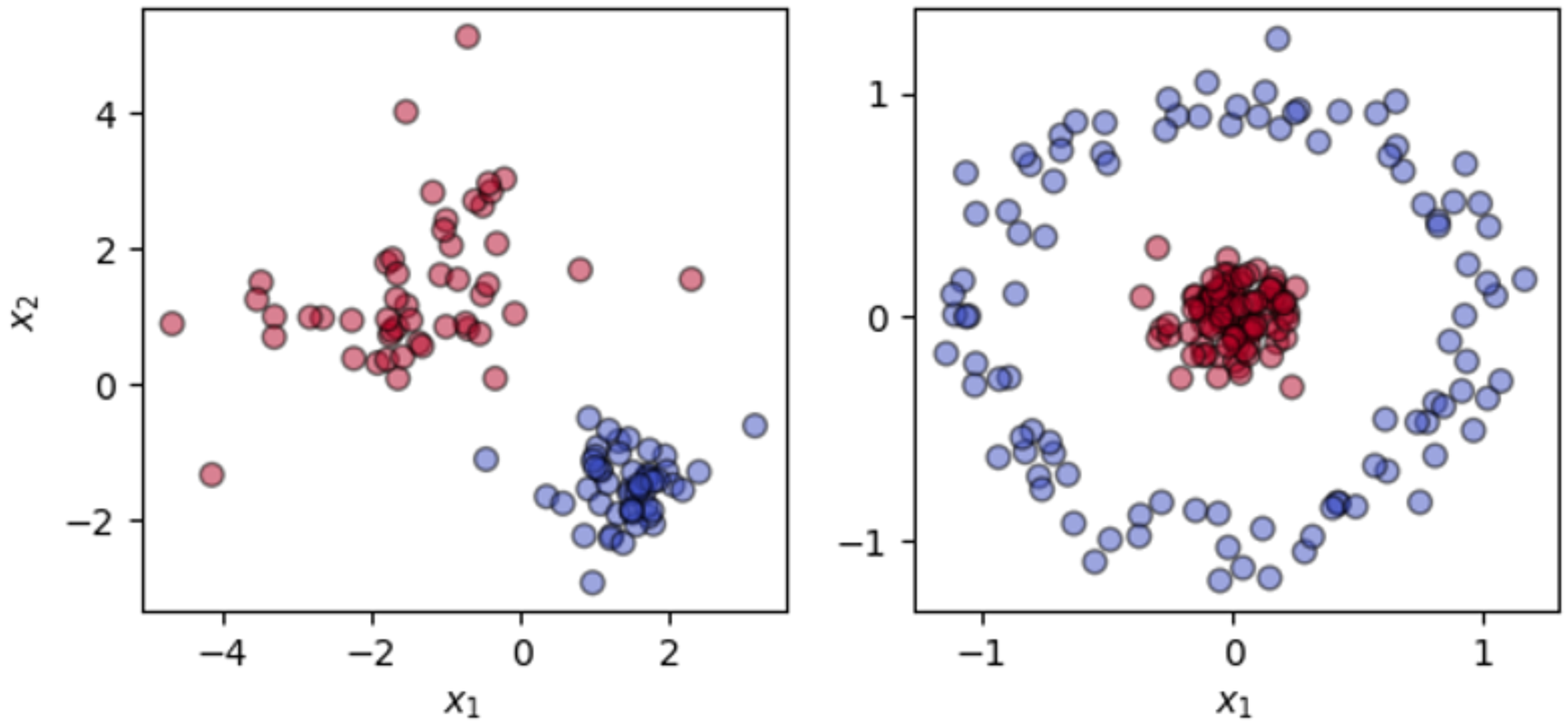
Representation Learning

Task - Find The Simplest Classifier



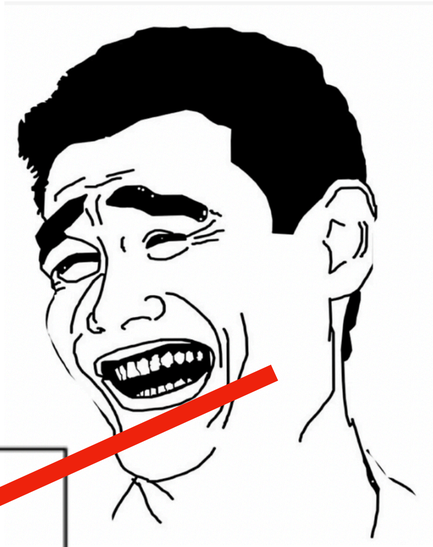
Representation Learning

How About Now ?



Representation Learning

How About Now ?



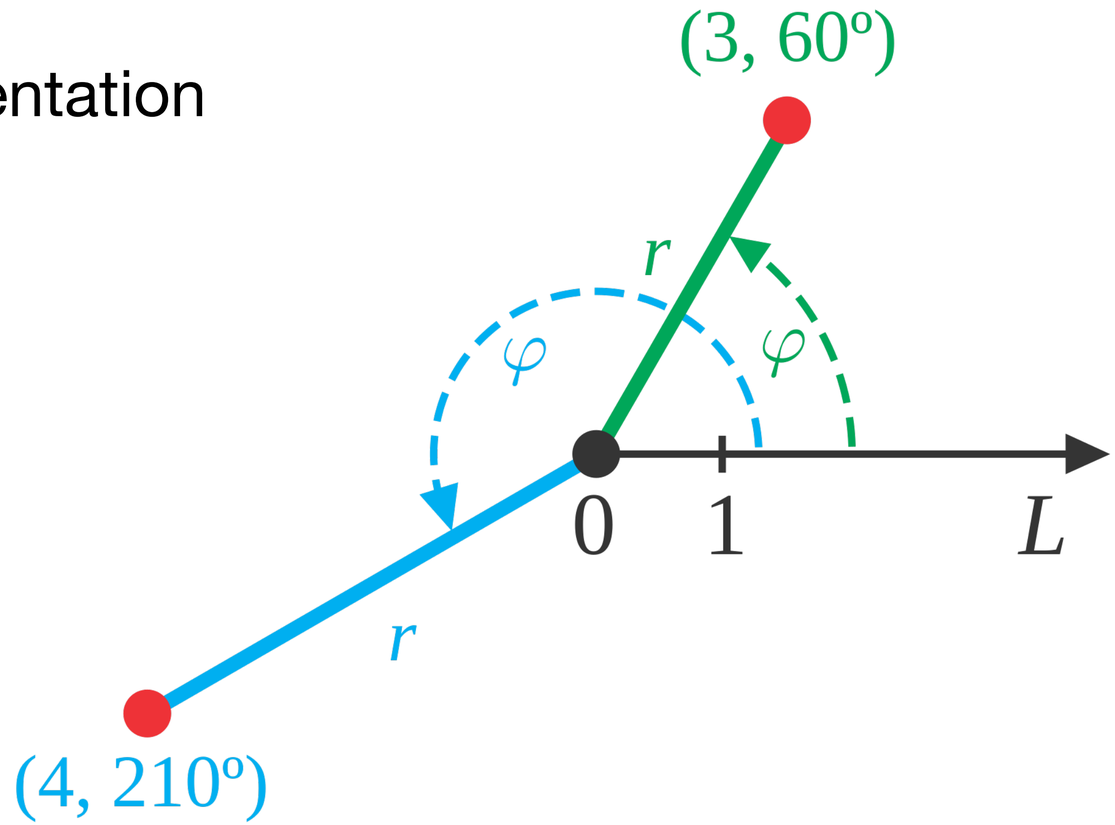
Representation Learning

How About Now ?

- Polar Coordinates Representation

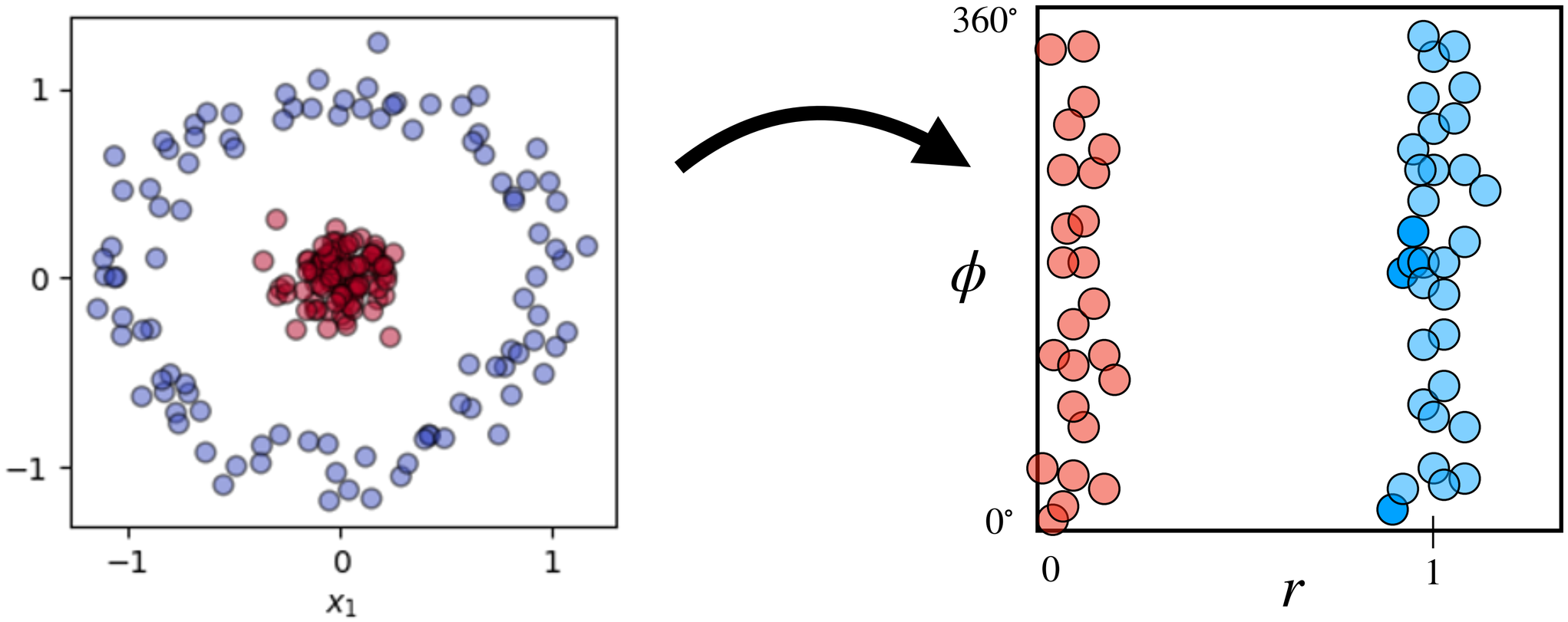
$$r = \sqrt{x^2 + y^2}$$

$$\tan\phi = \frac{y}{x}$$



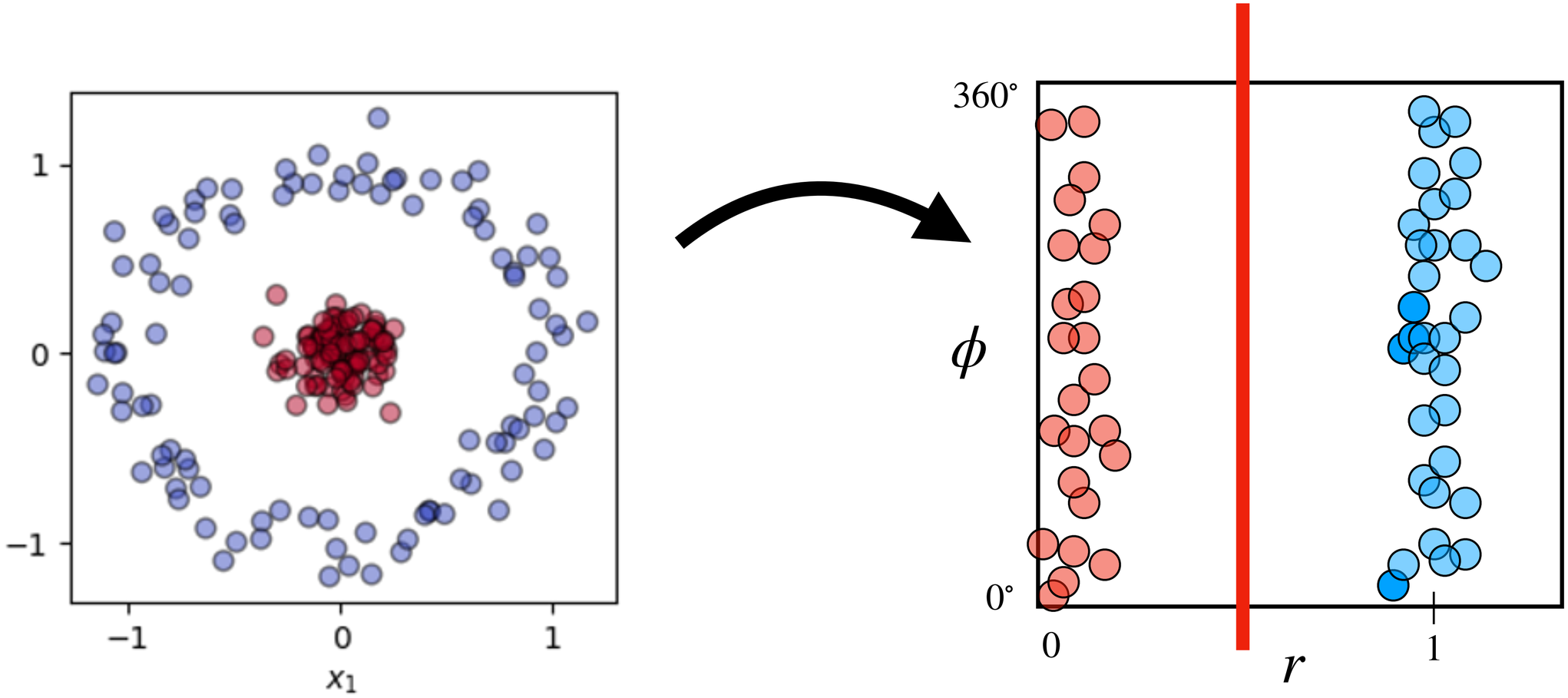
Representation Learning

How About Now ?



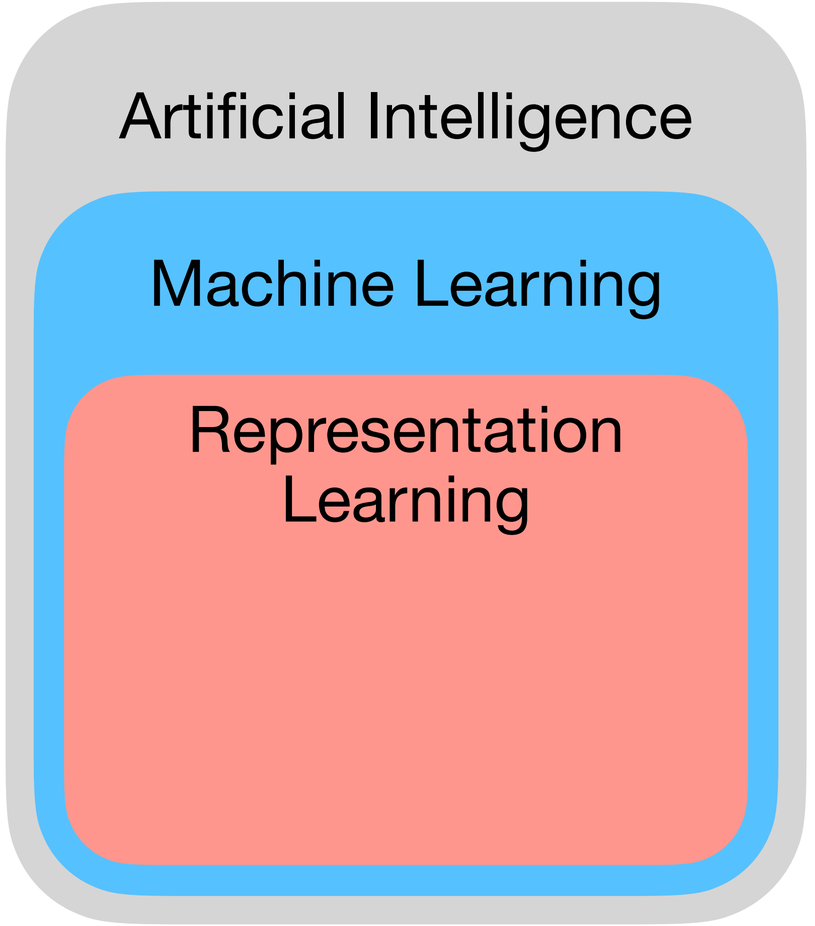
Representation Learning

How About Now ?



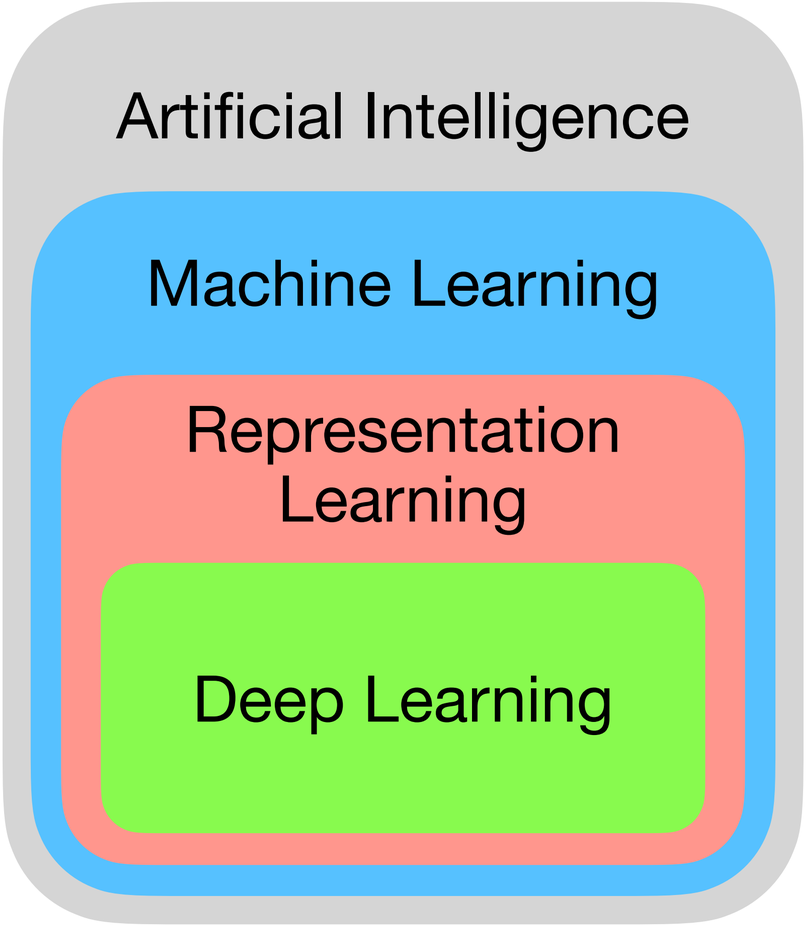
Representation Learning

- Representation = **Transformation** of input features
- **Better suited** for a given task than the original input
- Hand-crafted or **Learned !**



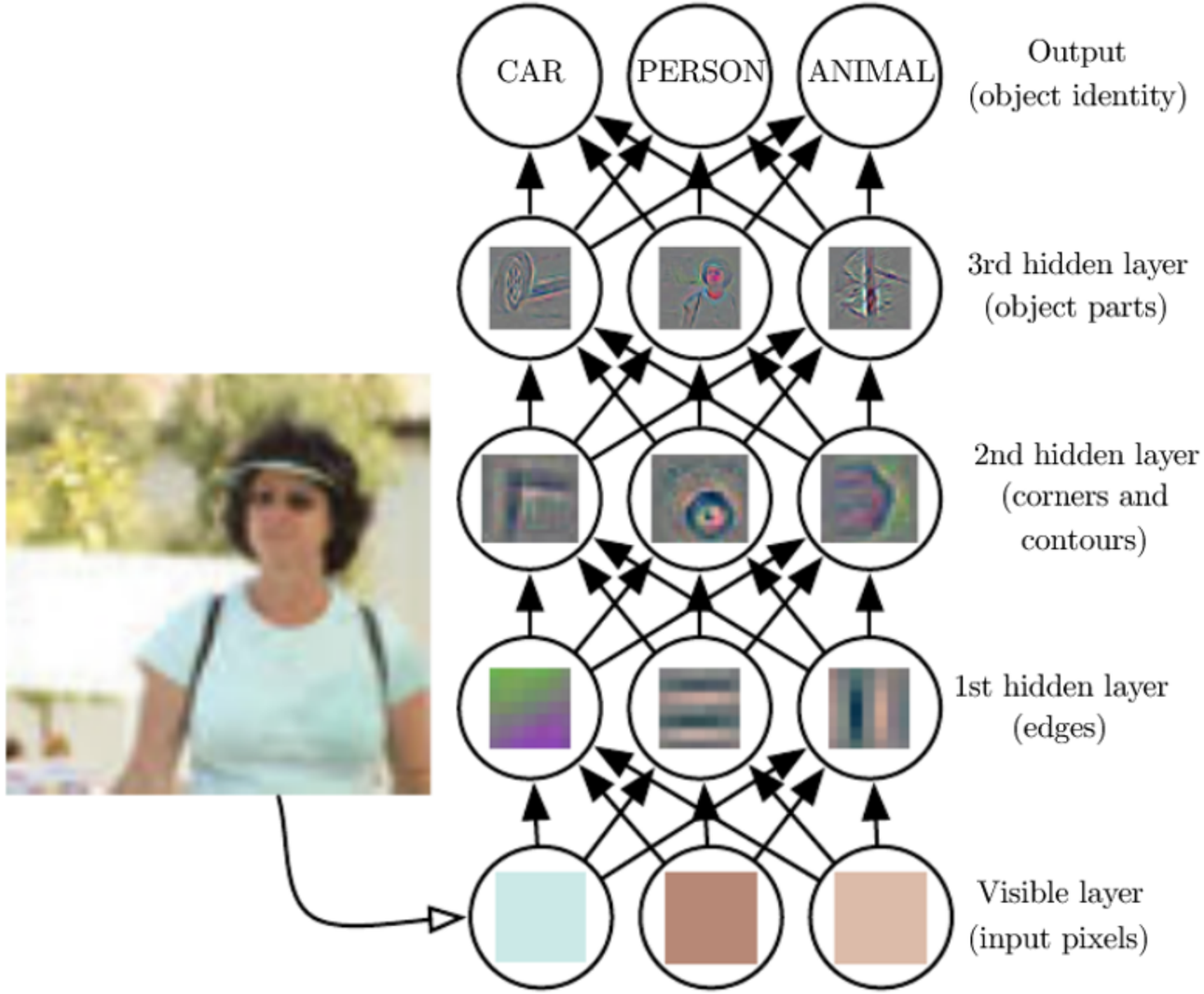
Deep Learning

- **Subset** of Machine Learning
- Based on Deep Neural Networks and Representation Learning



Deep Learning

- Each layer - different level of representation
- Deeper layers - more abstract features



deeplearningbook.com

History

- **1943** - McCulloch & Pitts - Mathematical model of biological neuron
- **1958** - Rosenblatt - Perceptron
- **1969** - Minsky - Proof, that perceptron cannot solve complex problems
- **1985** - Ackley, Hinton, Sejnowski - Boltzman Machine
- **1986** - Hinton, Rumelhart, Williams - Back-propagation of errors
- **2008** - Andrew Ng - Possibility of using GPUs for training
- **2009** - Fei-Fei Li - ImageNet dataset, ILSVRC challenge

History

- **2012** - Alex Krizhevsky - CNN beats the ILSVRC challenge
- **2014** - Ian Goodfellow - Generative Adversarial Networks
- **2017** - Ashish Vaswani - Transformer Architecture
- **2020** - Johnathan Ho - Denoising Diffusion Probabilistic Models
- **2022** - OpenAI - ChatGPT

Heroes

Godfathers of AI

Geoffrey Hinton

Yoshua Bengio

Yann LeCun

Heroes

Fei-Fei Li

Ian Goodfellow

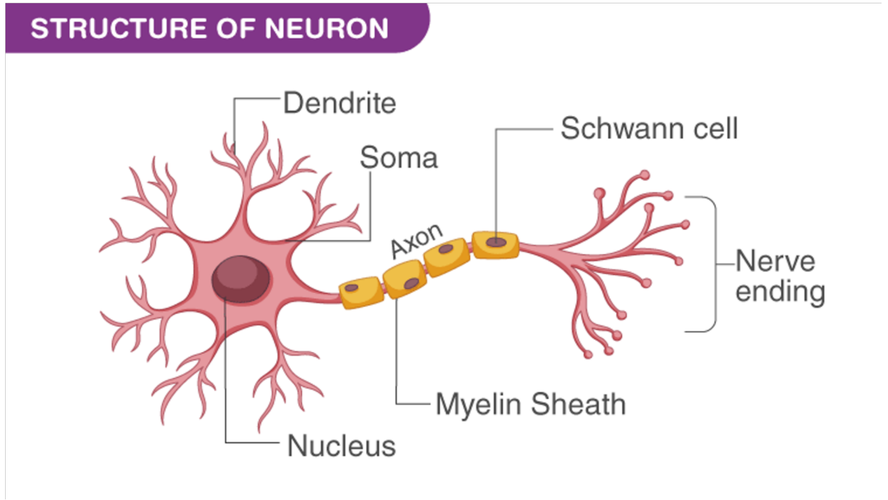
Andrew Ng

And others ...

2 Introduction to Neural Nets

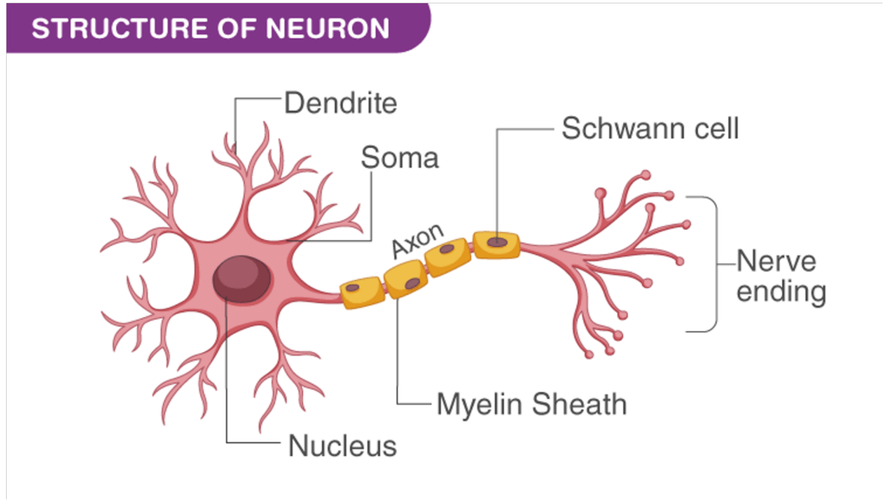
Artificial Neuron

- Inspired by biological neuron



Artificial Neuron

- Inspired by biological neuron



$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \longrightarrow \boxed{z = \sum_i^n w_i x_i + b} \longrightarrow \boxed{\hat{y} = g(z)} \longrightarrow \hat{y}$$

Input feature vector

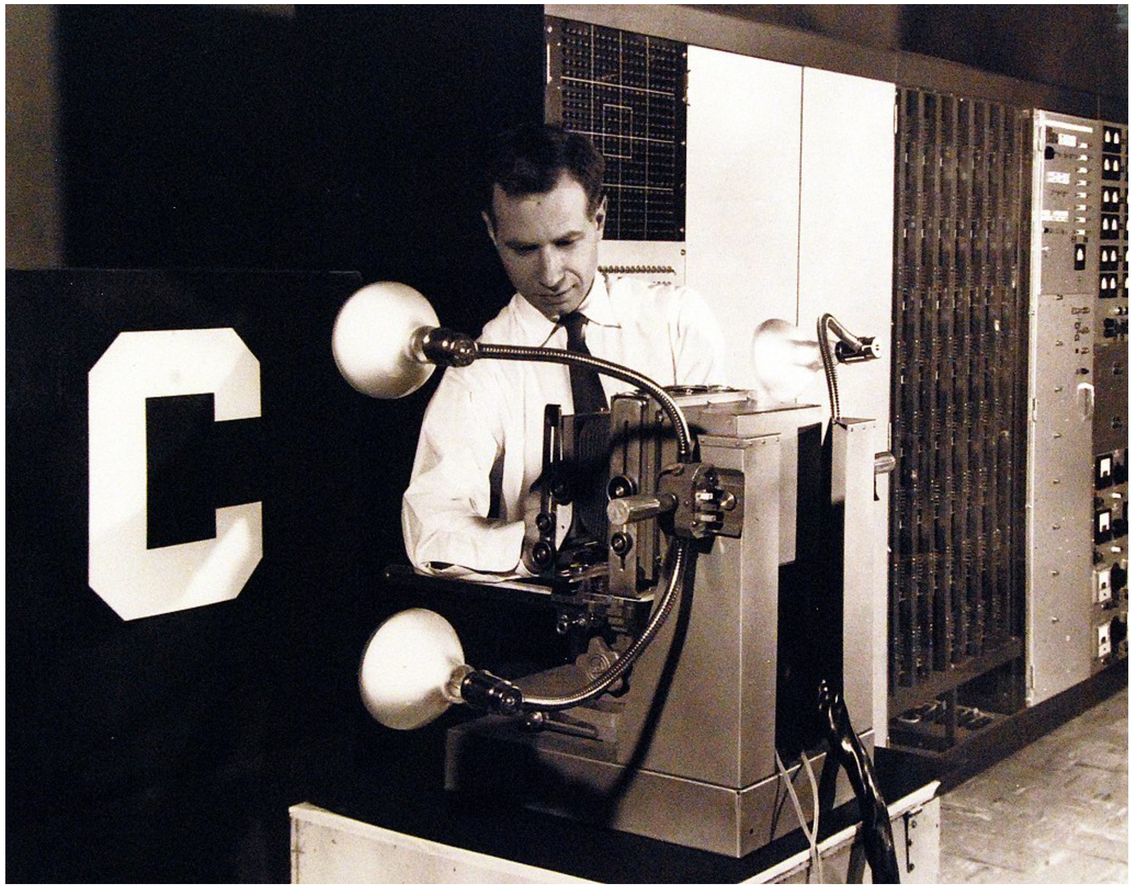
Linear function

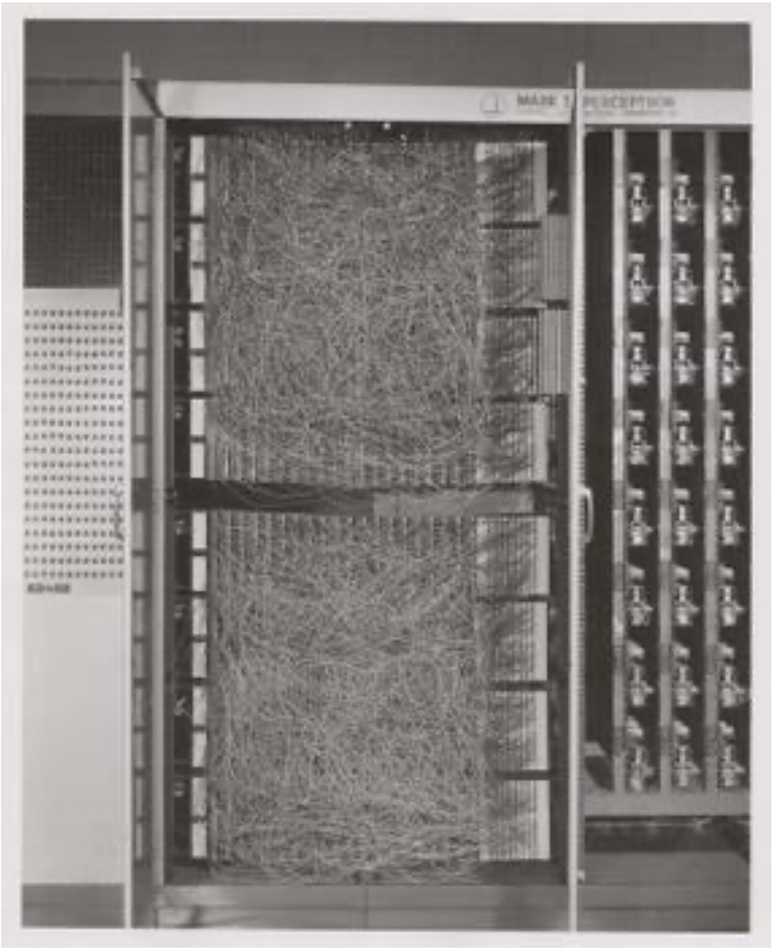
Activation function

Output value

Perceptron

1958 - Rosenblatt - Discrete Perceptron



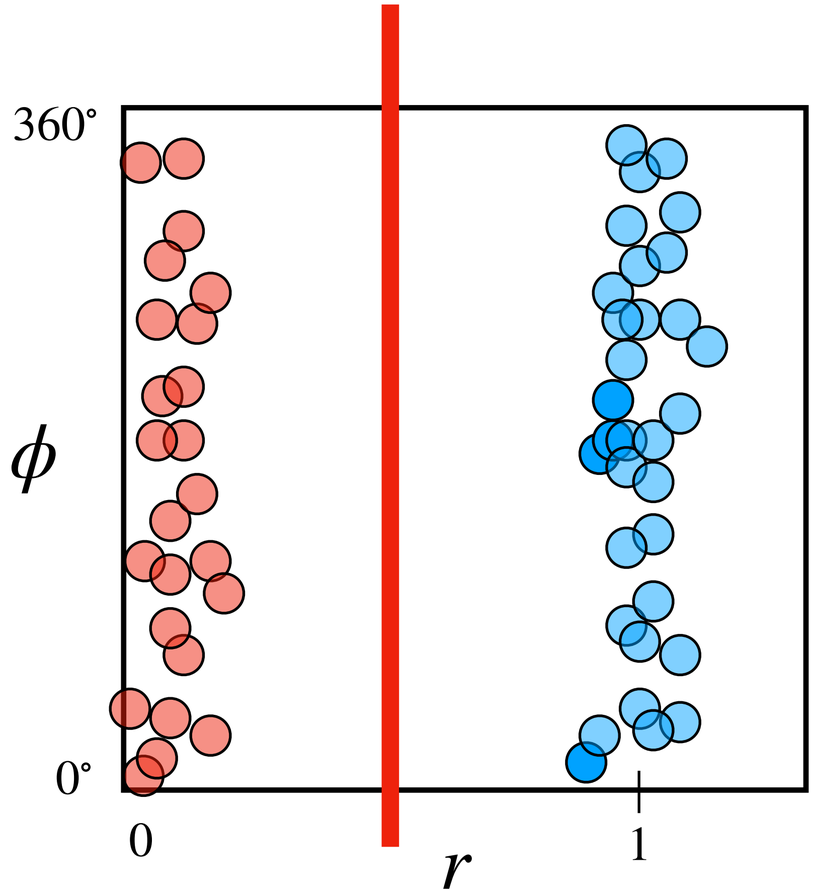


Images - https://en.wikipedia.org/wiki/Perceptron

Perceptron

1958 - Rosenblatt - Discrete Perceptron

- Step activation function - Classification of **linearly separable** patterns

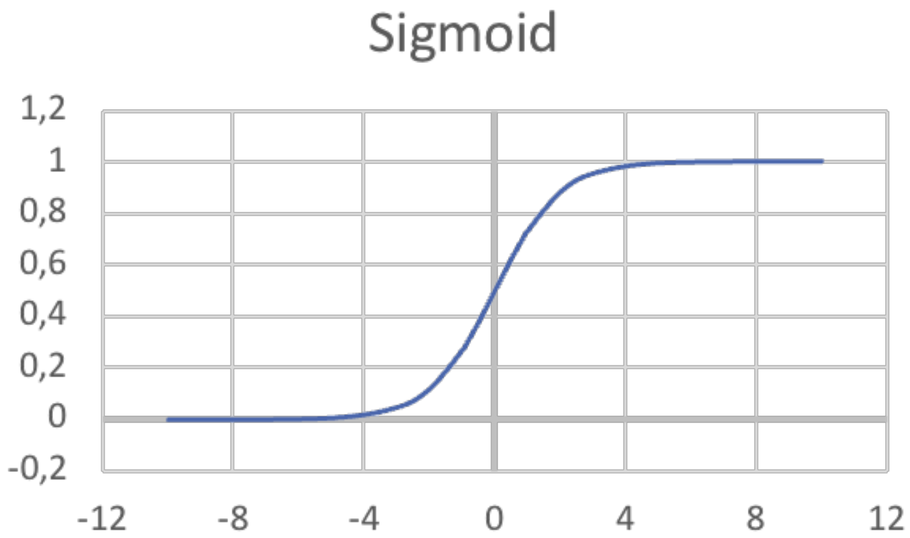


Perceptron

Continuous Perceptron

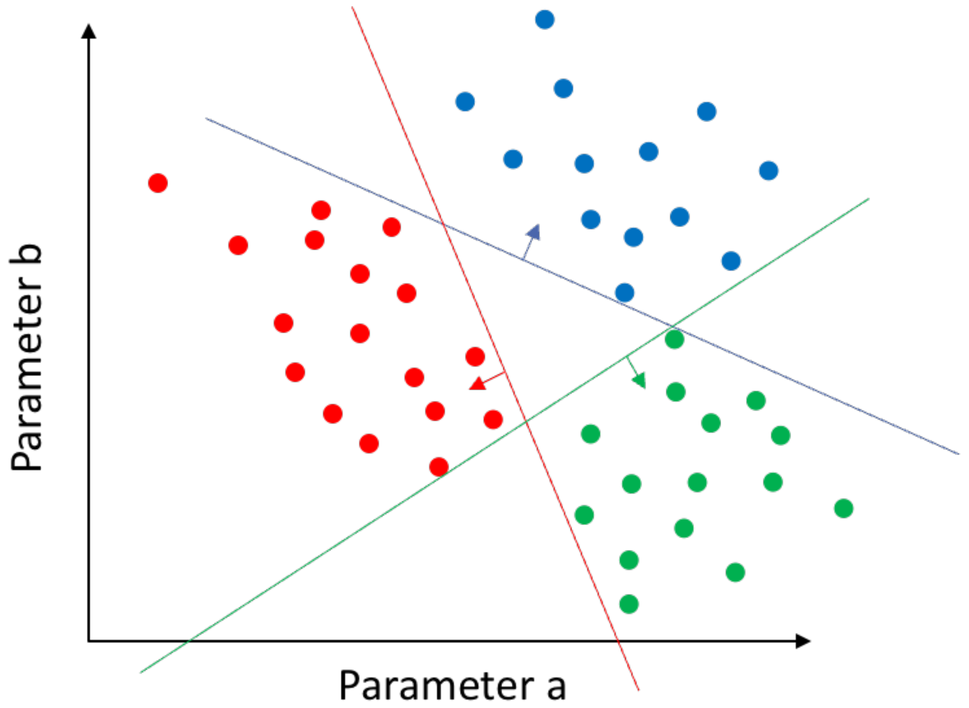
- Sigmoid activation function
- Output - interpreted as probability

$$g(z) = \frac{1}{1 + e^{-z}}$$



Single-Layer Perceptron

- Multiple classes - Multiple parallel perceptrons



Multi-Layer Perceptron

1985 - Rumelhart

- Idea - Sequence of Single-Layer Perceptrons can learn complex patterns
- **We need non-linear activations between successive SLPs**

Multi-Layer Perceptron

1985 - Rumelhart

- Idea - Sequence of Single-Layer Perceptrons can learn complex patterns
- **We need non-linear activations between successive SLPs**

$$f(x) = ax + b$$

$$g(x) = cx + d$$

$$g(f(x)) = c(ax + b) + d$$

$$= acx + (cb + d)$$

Linear function

Activation Functions

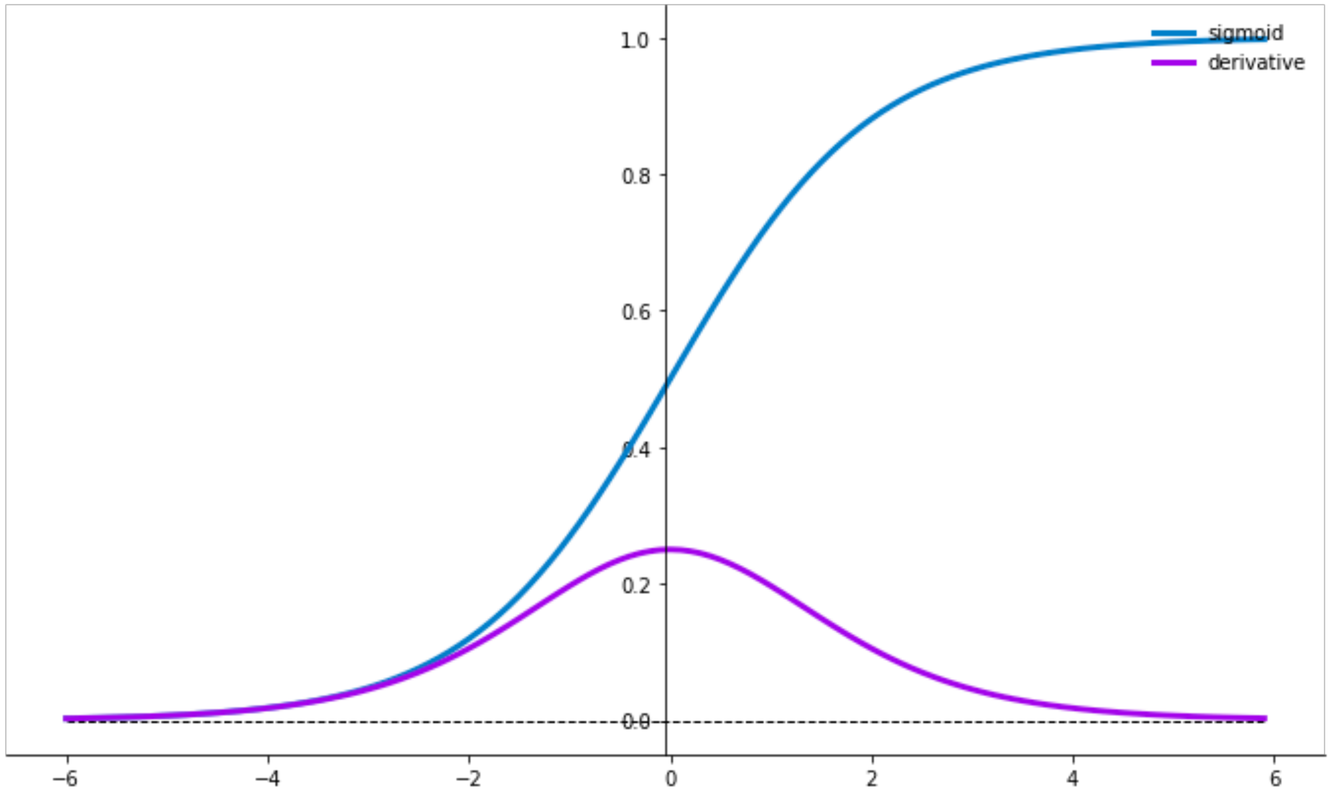
- **1975 - Fukushima - ReLU !**
    - was still not generally used back then

- 2010 - Nair & Hinton - Rectified Linear Units Improve Restricted Boltzmann Machines

Fukushima, Kunihiko. "Cognitron: A self-organizing multilayered neural network." *Biological cybernetics* 20.3-4 (1975): 121-136.

Activation Functions

Sigmoid

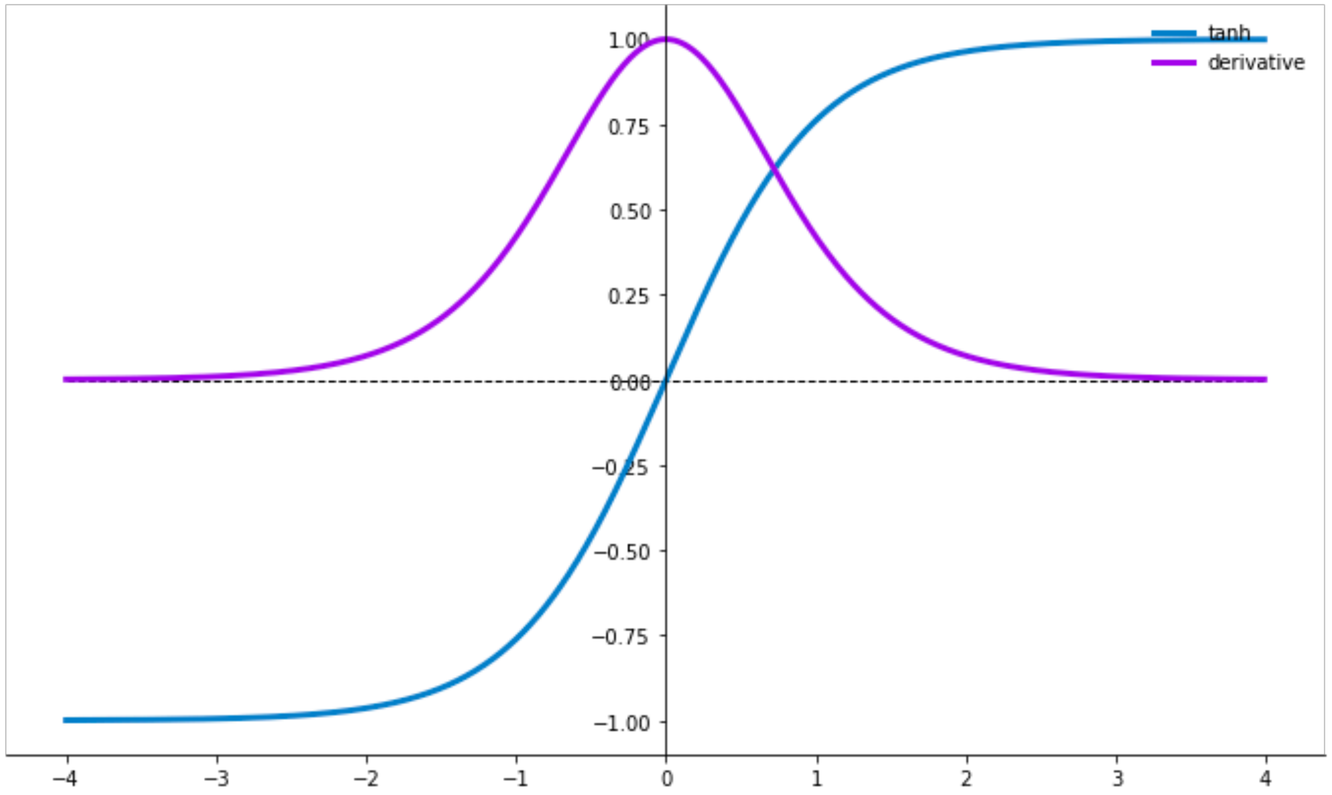


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

Activation Functions

TanH

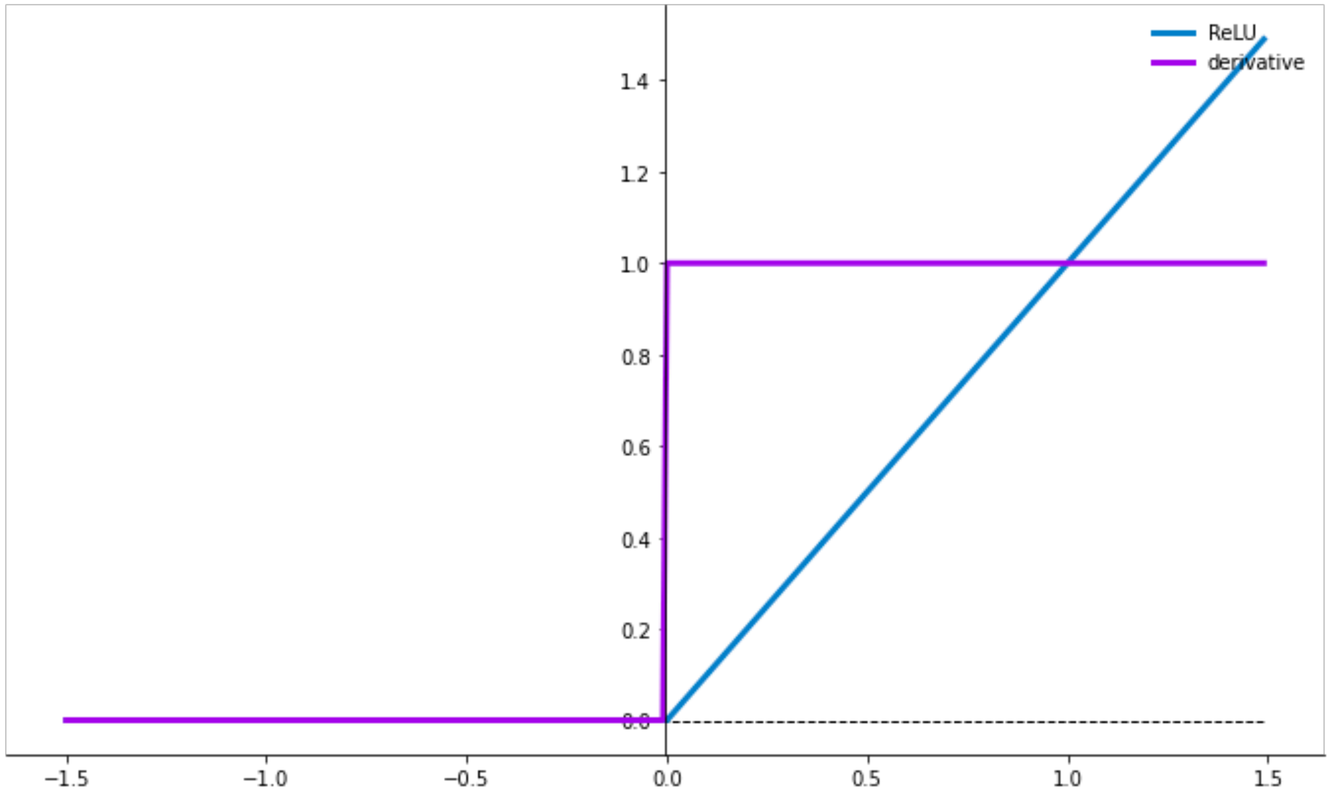


$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\tanh'(z) = 1 - \tanh^2(z)$$

Activation Functions

ReLU

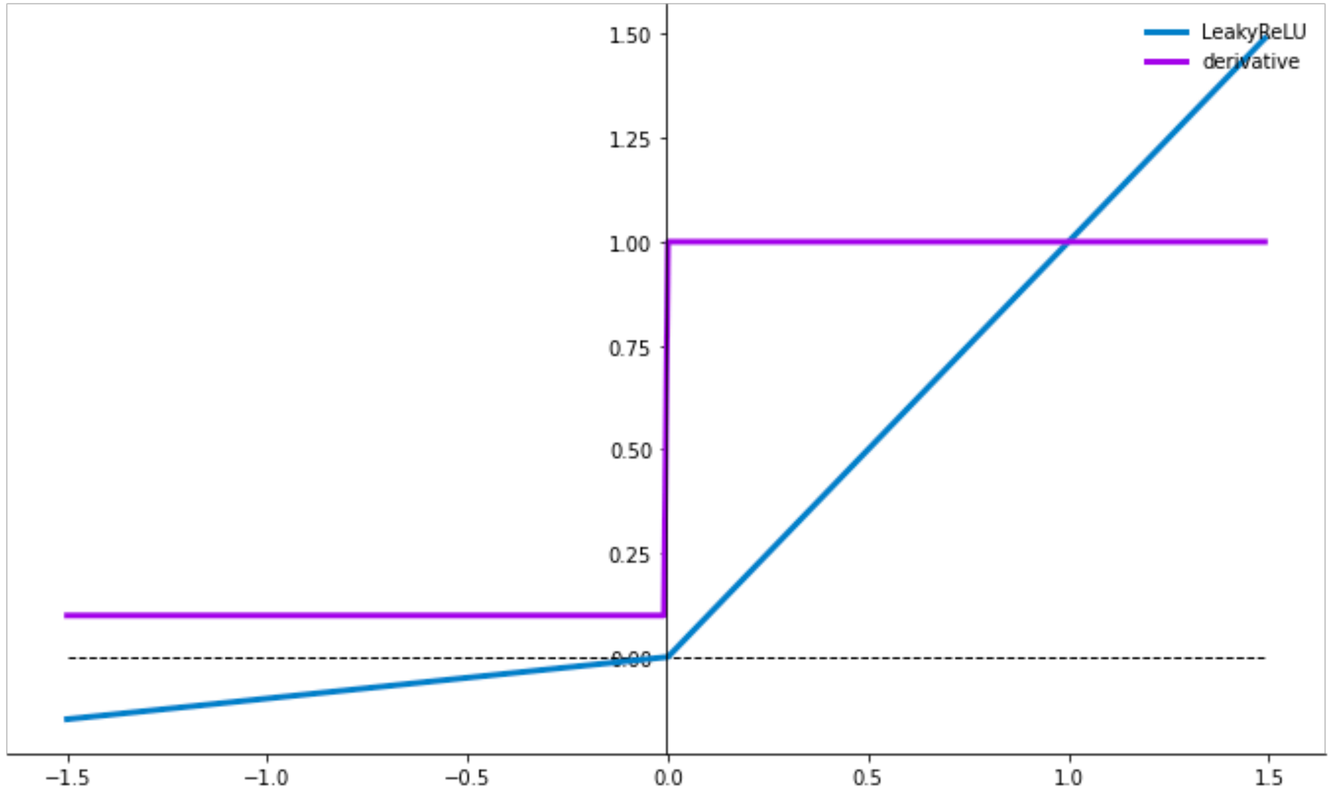


$$ReLU(z) = \max(0, z)$$

$$ReLU'(z) = \begin{cases} 1 & if\ z > 0 \\ 0 & else \end{cases}$$

Activation Functions

LeakyReLU



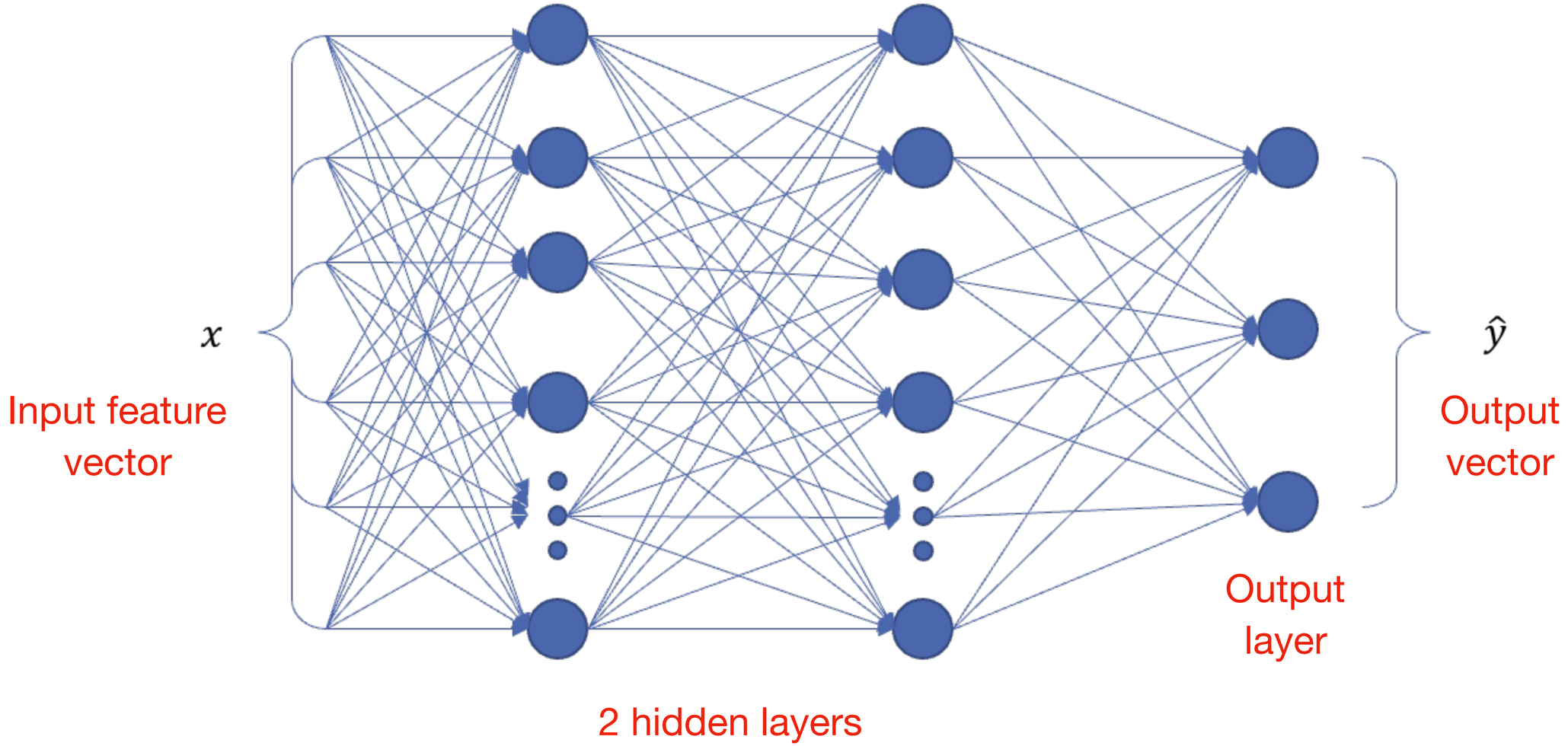
$$\alpha = 0.1$$

$$lReLU(z) = \max(\alpha z, z)$$

$$lReLU'(z) = \begin{cases} 1 & if\ z > 0 \\ \alpha & else \end{cases}$$

Multi-Layer Perceptron

1985 - Rumelhart



Multi-Layer Perceptron

Properties

- Sequence of multiple layers
    - Several hidden layers + final output layer
- Non-linear activation functions between successive layers
- Fully-connected layers - all units between two layers
- Number of output perceptrons - number of classes for classification
- **Continuous & Differentiable function**

Training of MLPs

1986 - Rumelhart, Hinton, Williams - Back-propagation of errors

Abstract

We describe a new learning procedure, back-propagation, for networks of neurone-like units. The procedure repeatedly adjusts the weights of the connections in the network so as to minimize a measure of the difference between the actual output vector of the net and the desired output vector. As a result of the weight adjustments, internal 'hidden' units which are not part of the input or output come to represent important features of the task domain, and the regularities in the task are captured by the interactions of these units. The ability to create useful new features distinguishes back-propagation from earlier, simpler methods such as the perceptron-convergence procedure[1].

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors." *nature* 323.6088 (1986): 533-536.

3 Training of Neural Networks

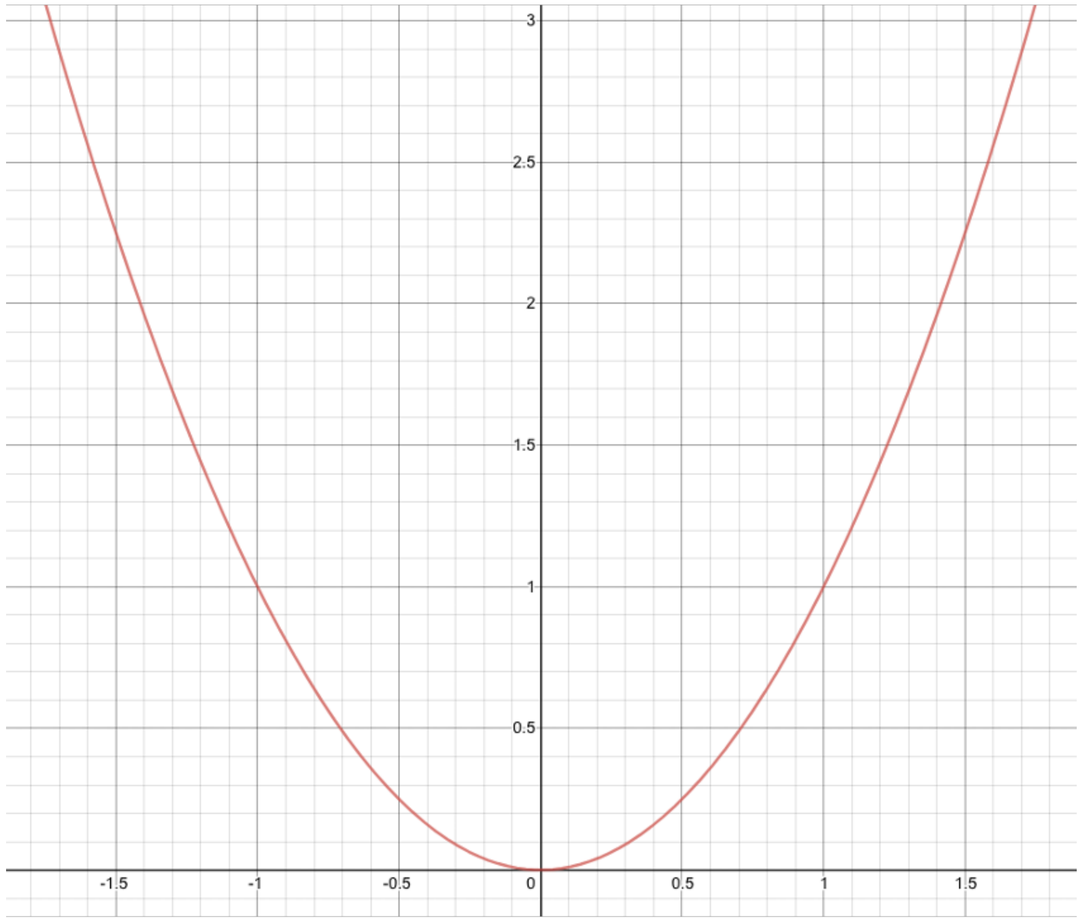
3 Training of Neural Networks

Content

1. Gradient Descent
2. Forward Pass
3. Back-propagation Algorithm - Examples

3.1 Gradient Descent

Gradient Descent

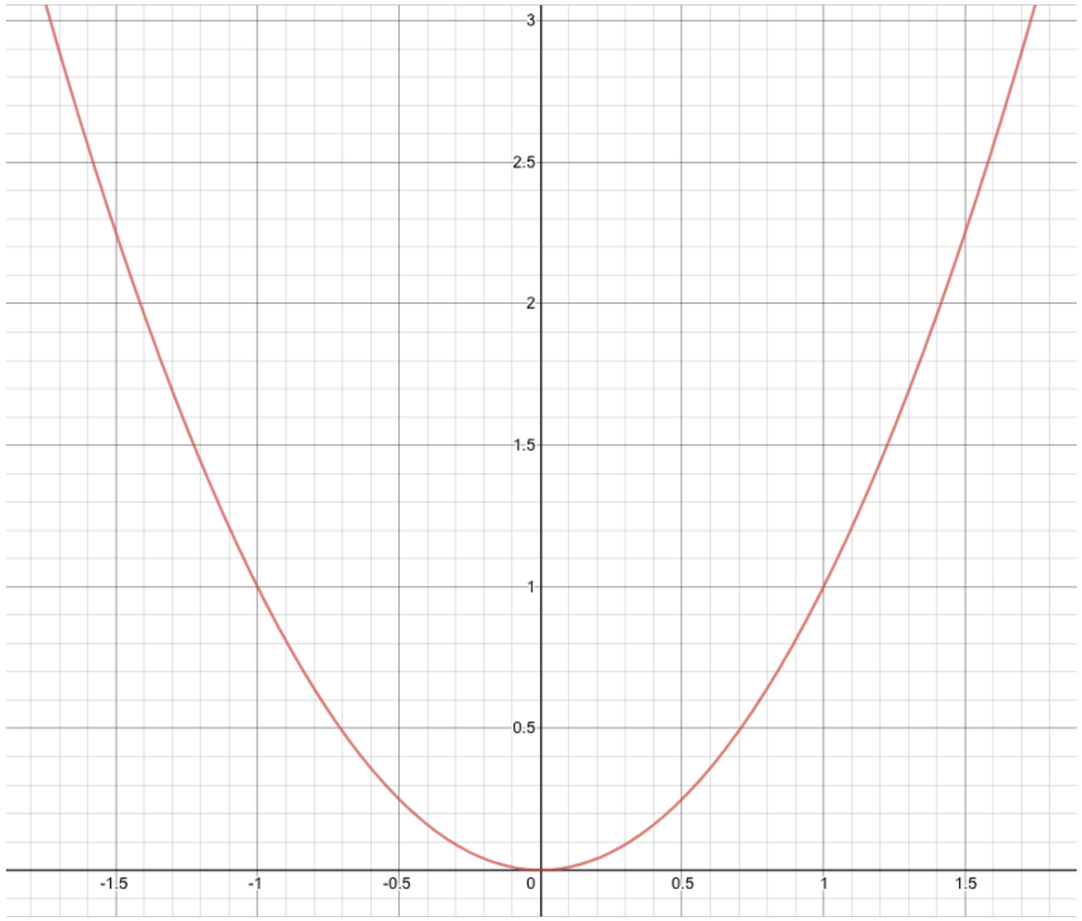


$$f(x) = x^2$$

$$f(x) \geq f(x_{min}); x \in R$$

$$x_{min} = ?$$

Gradient Descent



$$f(x) = x^2$$

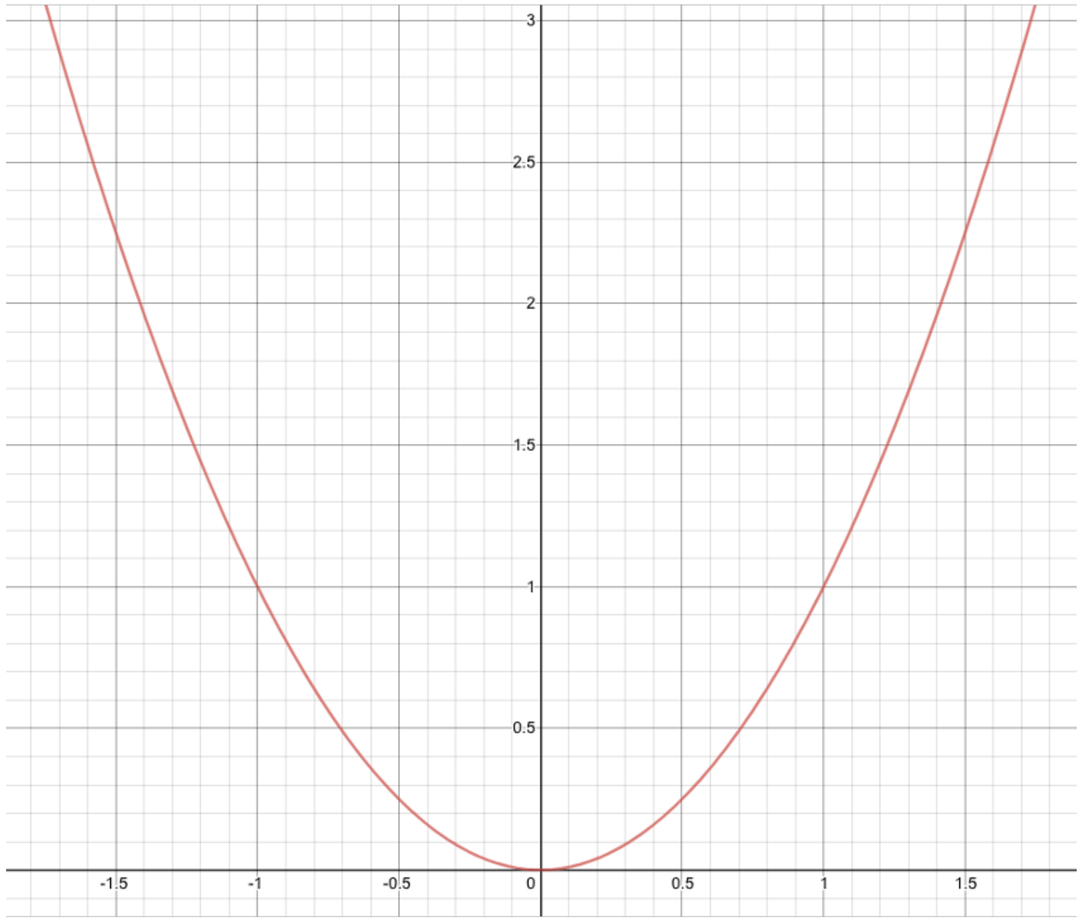
$$x_{min} = ?$$

$$f'(x) = 0$$

$$f'(x) = 2x = 0$$

$$x_{min} = 0$$

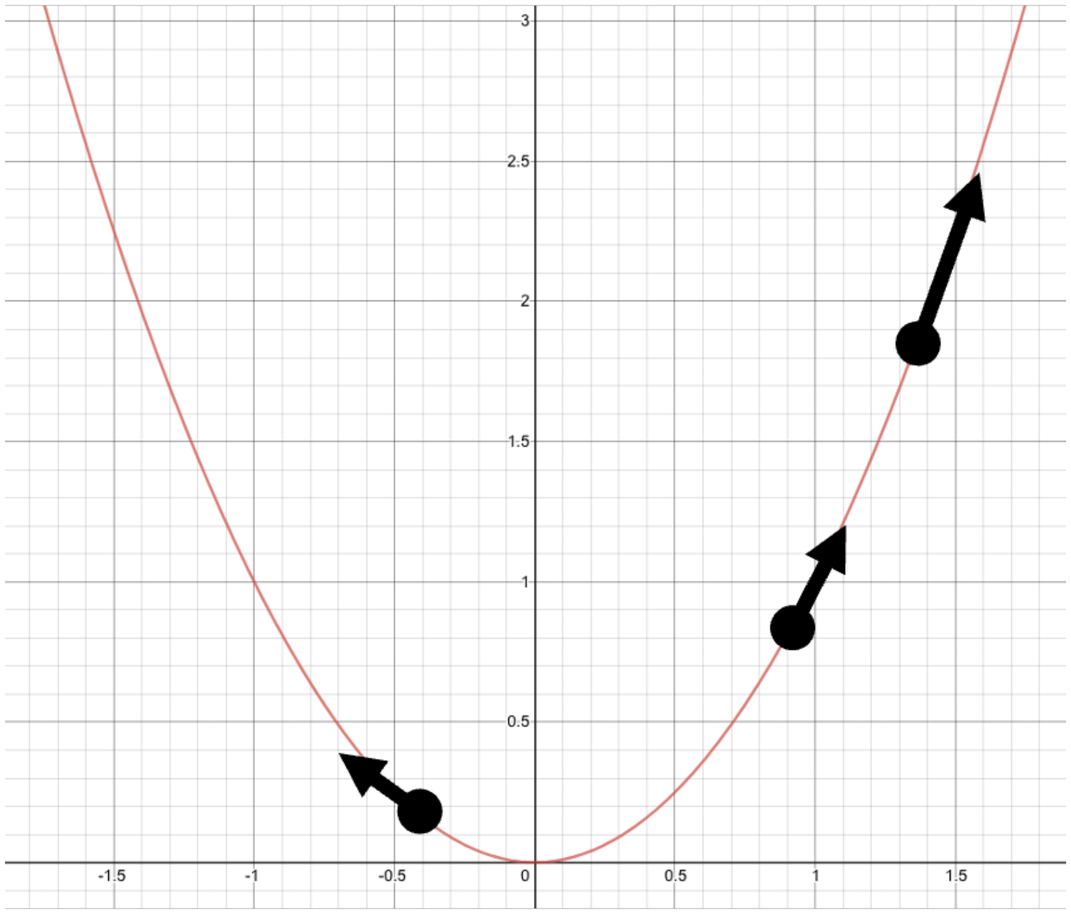
Gradient Descent



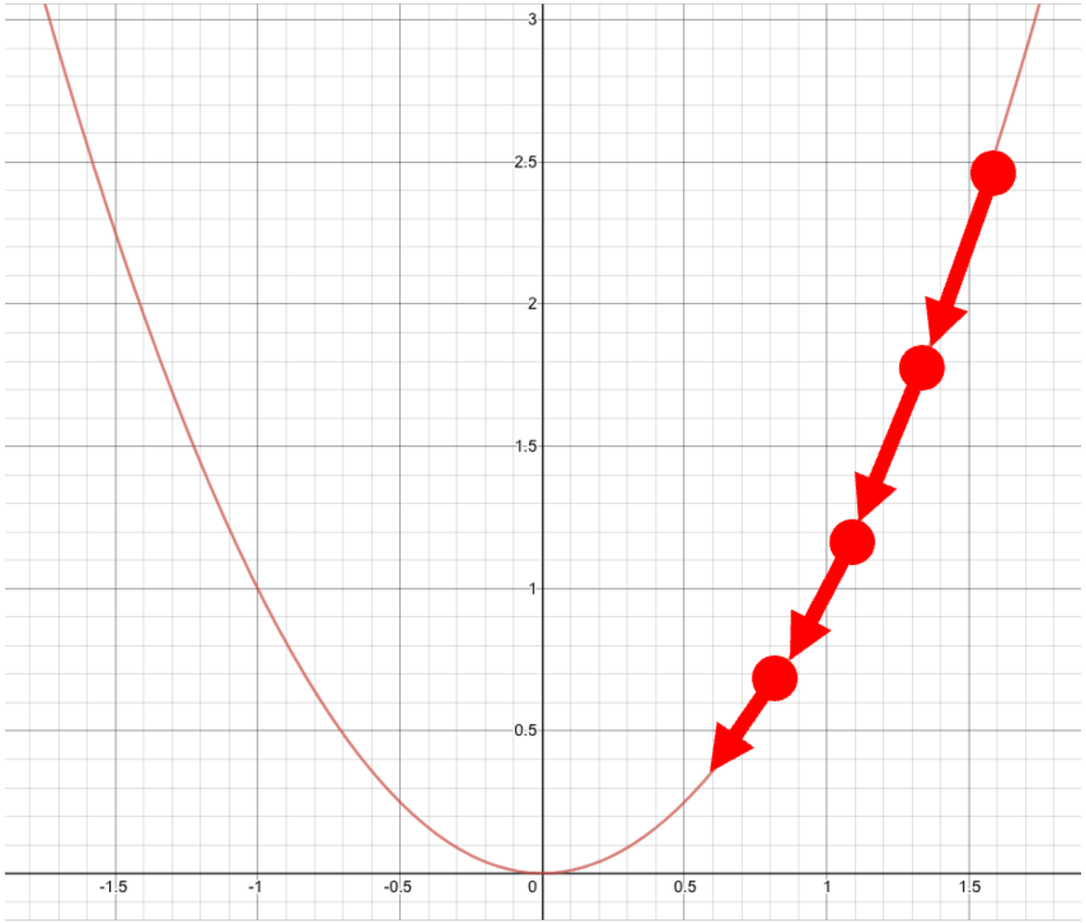
$$p = (x_1, x_2, x_3)$$
$$f(p) = \ldots$$

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f(p)}{\partial x_1} \\ \frac{\partial f(p)}{\partial x_2} \\ \frac{\partial f(p)}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Gradient Descent

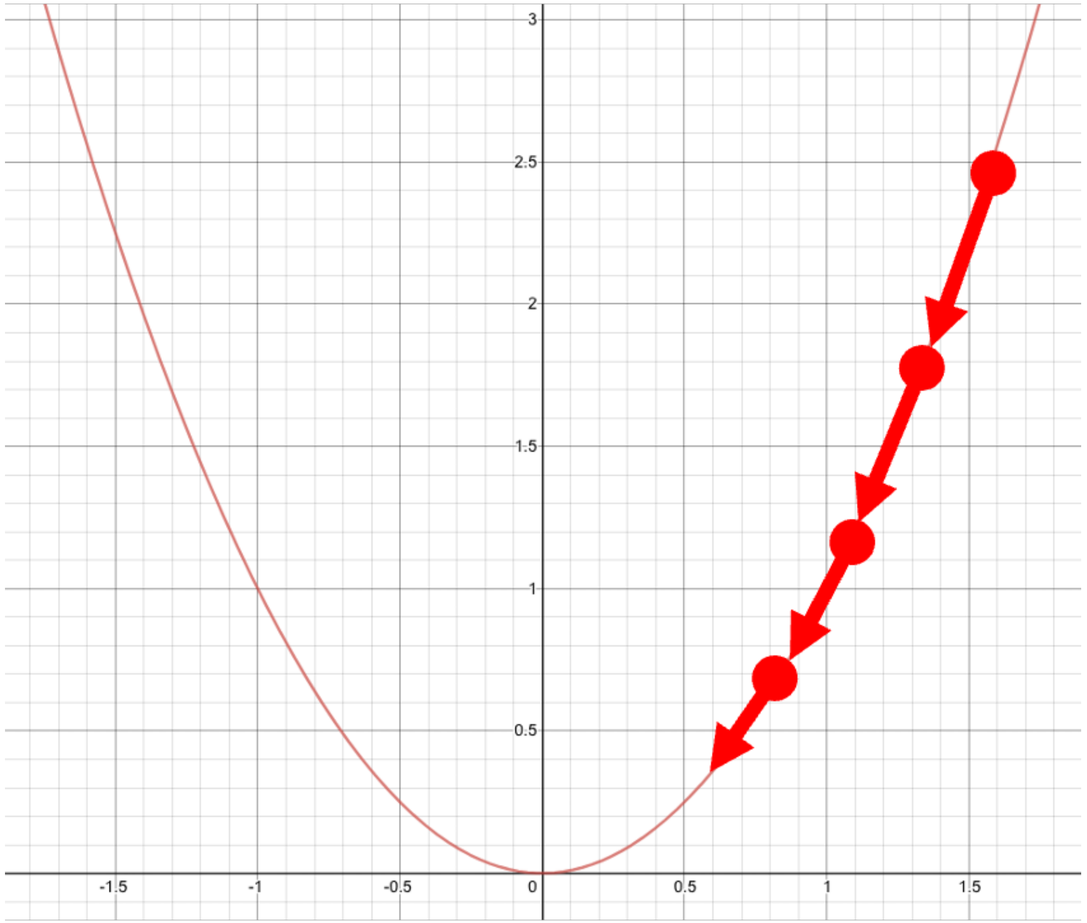


Gradient Descent



- Regardless of initial position, we can make **iterative steps in the opposite direction** of the gradient
- Repeat until we reach the minimum
- **Function minimization**

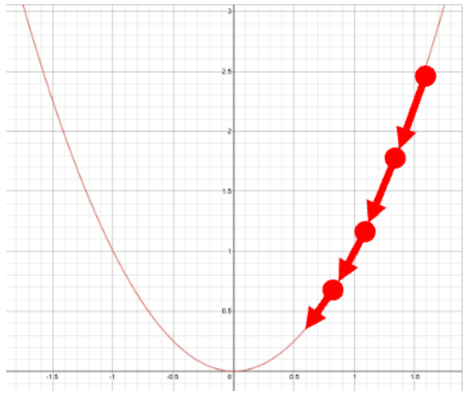
Gradient Descent



- Regardless of initial position, we can make **iterative steps in the opposite direction** of the gradient
- Repeat until we reach the minimum
- **Function minimization**

$$x := x - \alpha \nabla f(x)$$

Gradient Descent

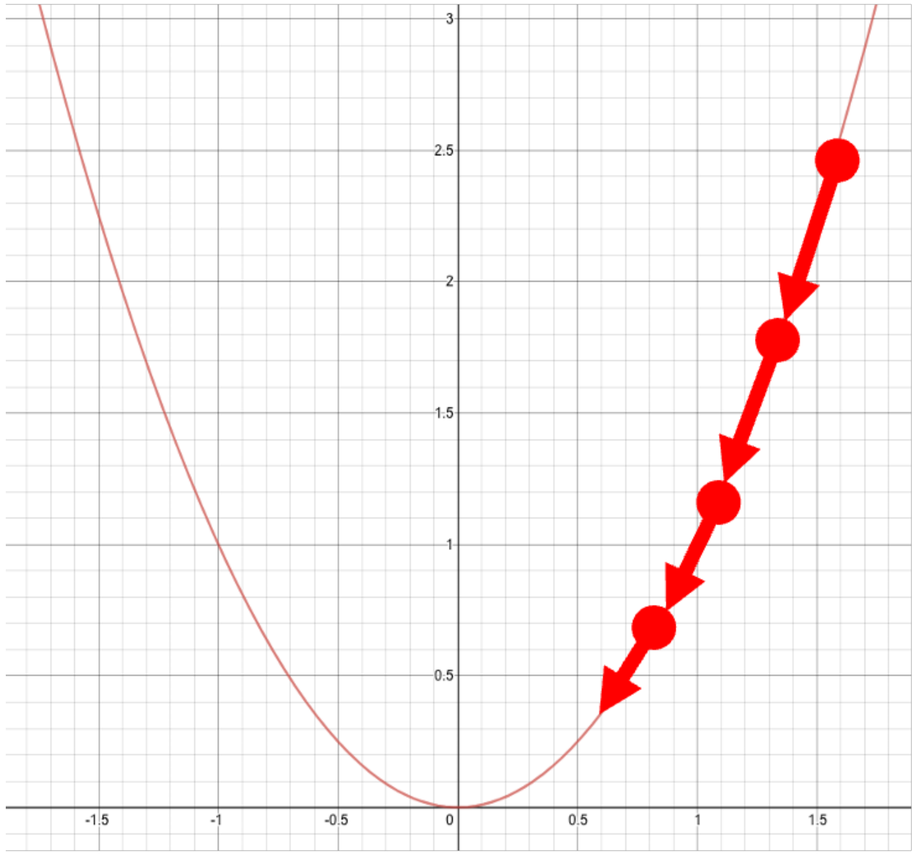


$$x := x - \alpha \nabla f(x)$$

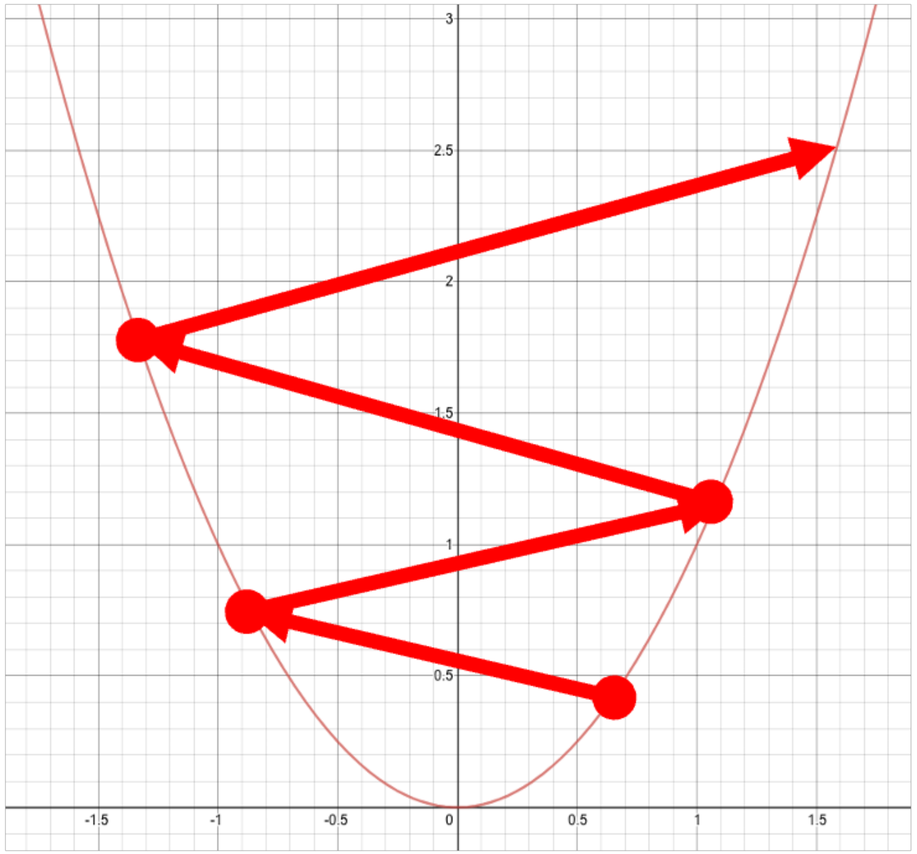
$-\nabla f(x)$ - direction of descent

α - step size, learning rate

Gradient Descent



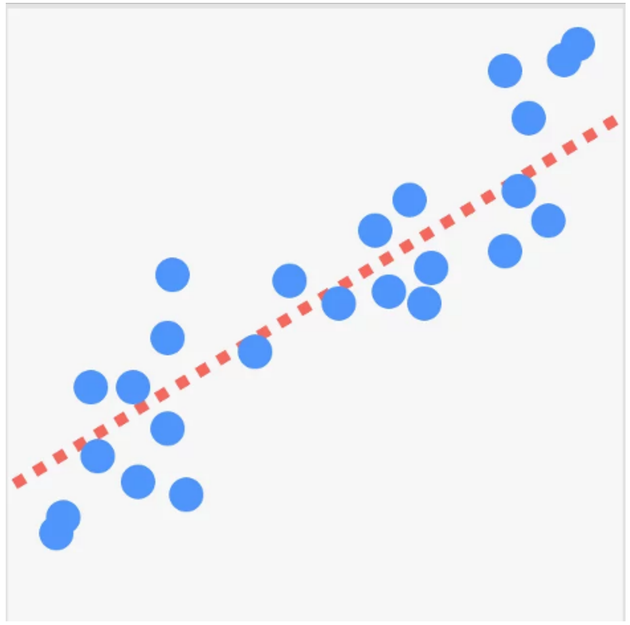
$\alpha = 0.3$



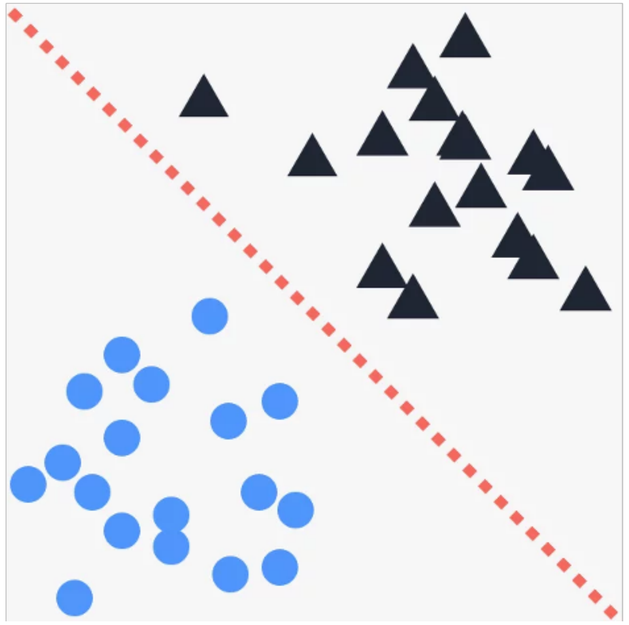
$\alpha = 1.5$

Key Tasks of Machine Learning

Regression



Classification



Regression

- **Estimate** the relationship between the **dependent variable (outcome)** and one or more **independent variables (features)**

$$y \in R \longleftarrow \text{Outcome}$$

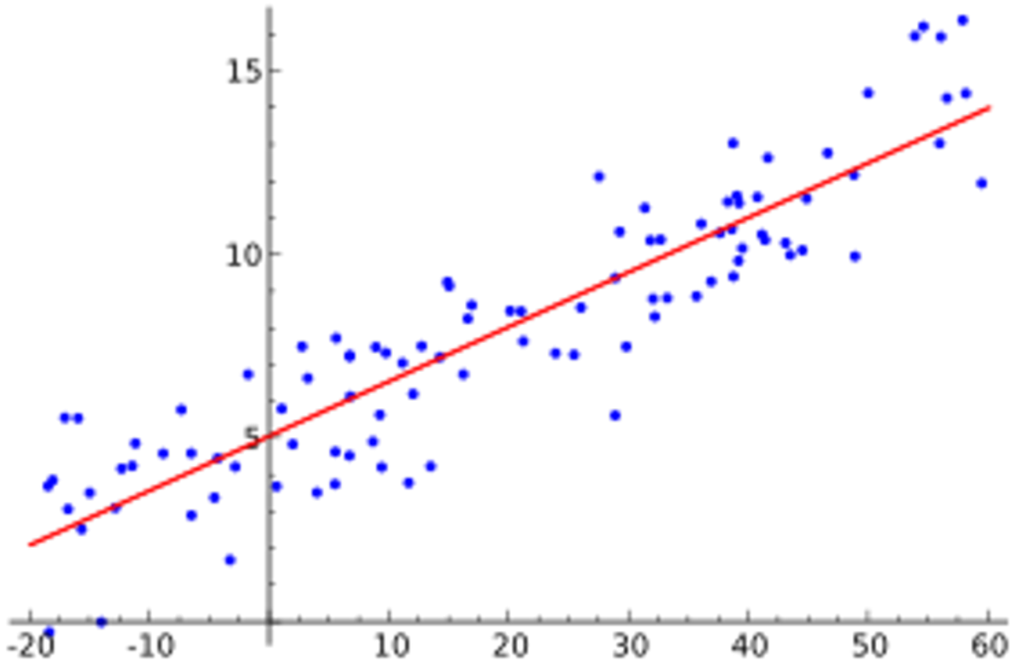
- **Data** - set of pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$

$$h(x) \approx y$$

Regression

Linear Regression

- The most common case of regression

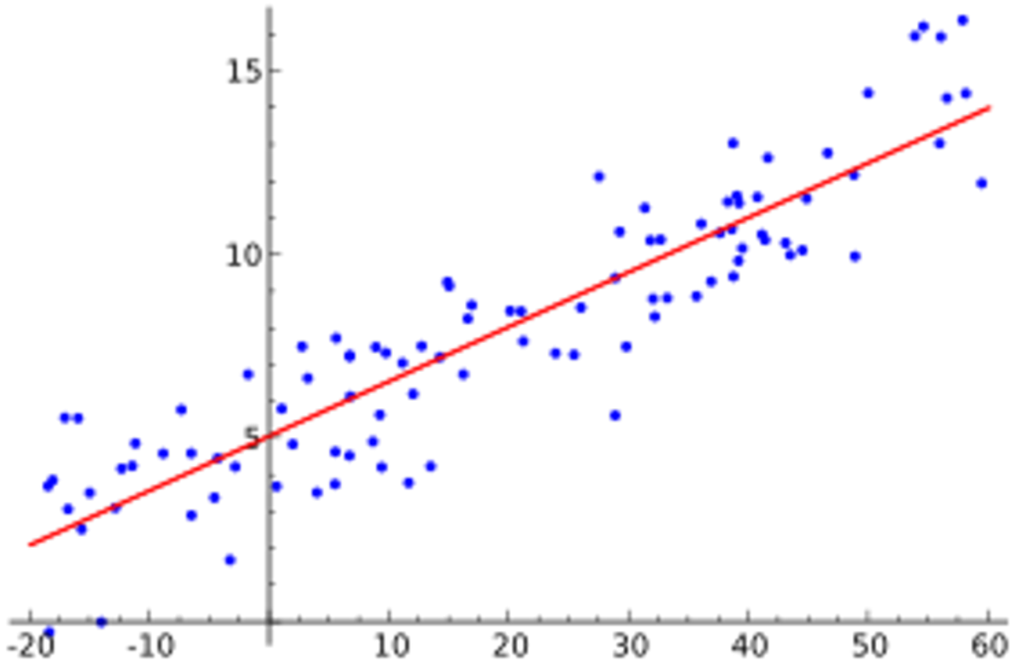


$$y = w_1 x + w_0$$

Regression

Linear Regression

- The most common case of regression

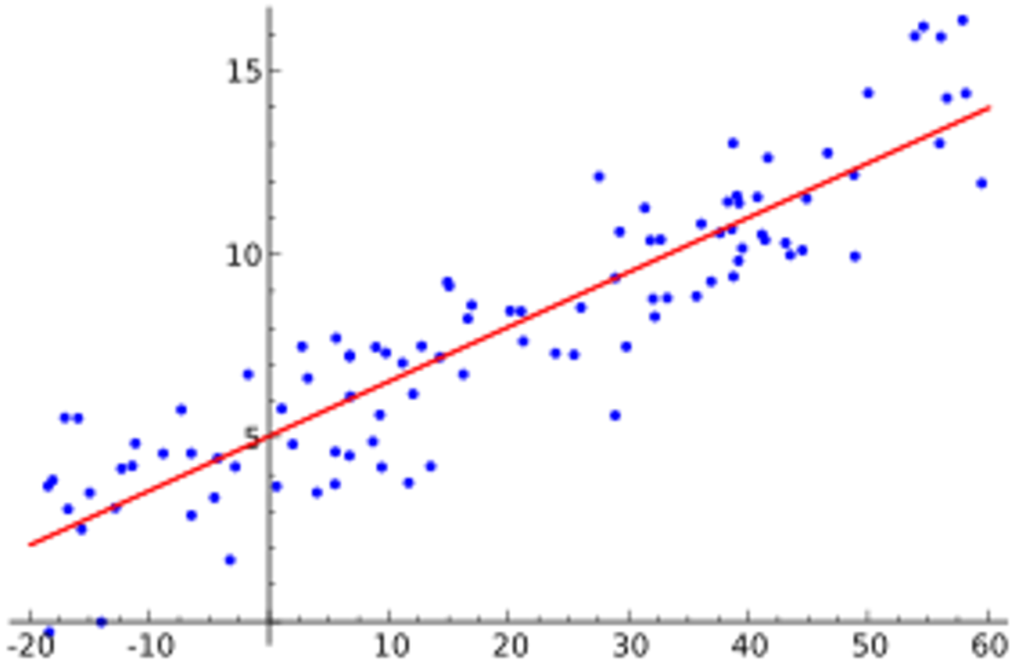


$$y = \sum_{i=1}^{n} w_i x_i + w_0$$

Regression

Linear Regression

- The most common case of regression



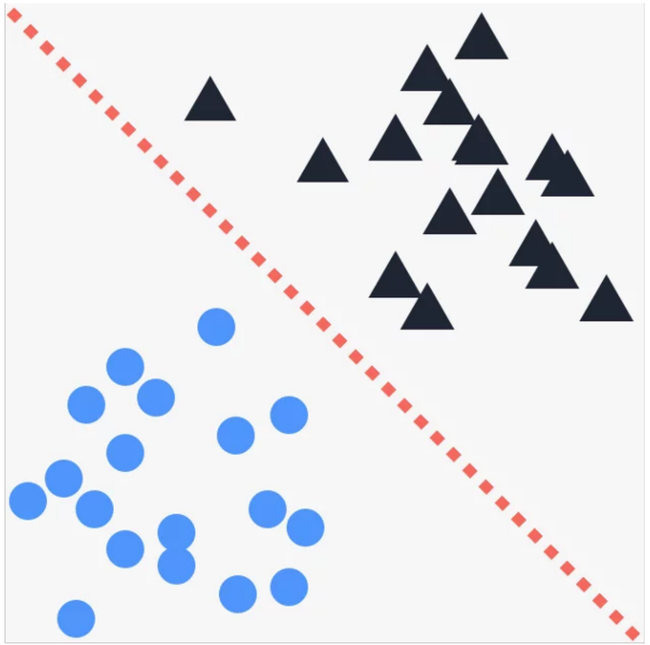
$$y = \sum_{i=1}^{n} w_i x_i + w_0$$

$$y = \mathrm{w}^T \mathrm{x} + w_0$$

n-dimensional vectors

Classification

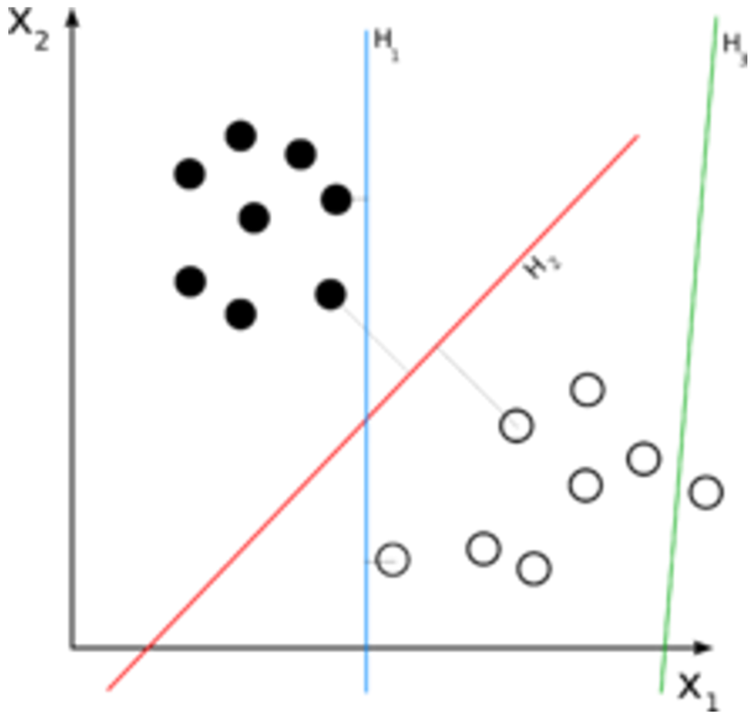
- Assign **exactly 1 out of m possible** classes for each input feature vector x



Classification

Linear Classification

- Linear classifier separates the feature space using a line or a hyper-plane



Classification

Linear Classification - Example

- Input feature vector x $= (x_1, x_2, \ldots, x_n)$

$$h(\mathrm{x}) = \sum_{i=1}^{n} w_i x_i \; + \; w_0$$

Regression

$$f(h) = \begin{cases} 1 & \text{if } h > T \\ 0 & \text{otherwise} \end{cases}$$

Step function

$$\hat{y} = f(h(x))$$

Loss & Cost

- Let

  $\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ be the training data set

  $\hat{y} = h(x)$ be a function (model) estimating y for a given feature vector x

- Loss function
  - Measures, how good does the function $h(x)$ estimate y for the **given sample** x

$$\mathcal{L}(\hat{y}, y) = \mathcal{L}(h(x), y)$$

Loss & Cost

- Let

  $\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m)\}$ be the training data set

  $\hat{y} = h(x)$ be a function (model) estimating y for a given feature vector x

- Cost function
  - Measures, how good does the function $h(x)$ estimate y for the **entire data set** $\mathcal{X}$

$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathcal{L}(h(x_i), y_i)$$

Loss & Cost

- Cost function
  - Measures, how good does the function $h(x)$ estimate y for the **entire data set** $\mathcal{X}$

$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathcal{L}(h(x_i), y_i)$$

$$\Theta = \{w_0, w_1\}$$

$$h(x) = w_1 x + w_0$$

Loss & Cost

- Cost function
  - Measures, how good does the function $h(x)$ estimate y for the **entire data set** $\mathcal{X}$

$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathcal{L}(h(x_i), y_i)$$

$$\Theta = \{w_0, w_1\}$$

$$h(x) = w_1 x + w_0$$

From the “Loss perspective” the “x” are variables, and the “w” are constants

Loss & Cost

- Cost function
  - Measures, how good does the function $h(x)$ estimate y for the **entire data set** $\mathcal{X}$

$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathcal{L}(h(x_i), y_i)$$

$$\Theta = \{w_0, w_1\}$$

$$h(x) = w_1 x + w_0$$

From the “Cost perspective”
the “w” are variables, and the
“x” are constants

Loss & Cost

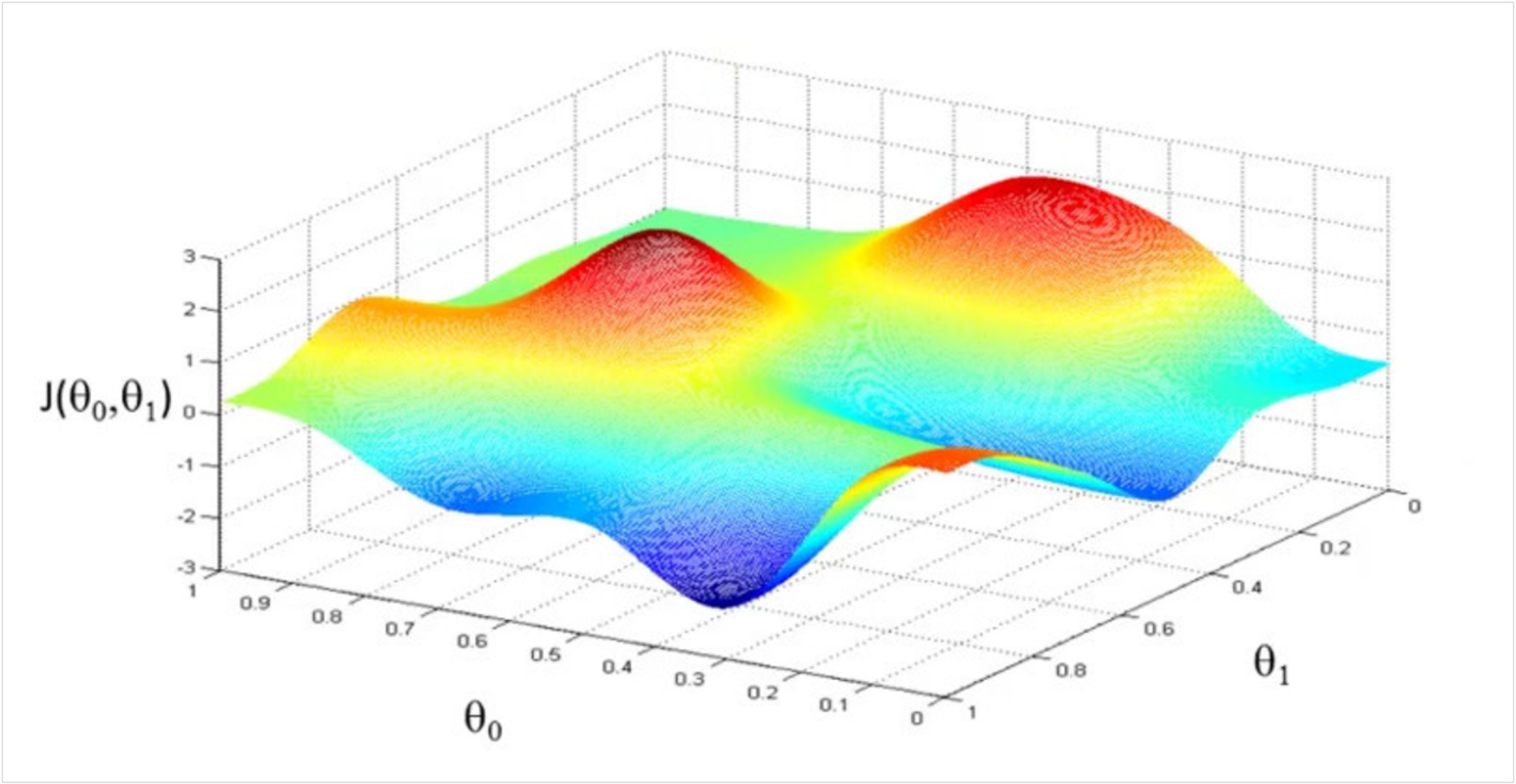
- Cost function
  - Measures, how good does the function $h(x)$ estimate y for the **entire data set** $\mathcal{X}$

$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathcal{L}(h(x_i), y_i)$$

$$\Theta = \{w_0, w_1\} \qquad h(x) = w_1 x + w_0$$

- To train the model $h(x)$, we need to **minimize the function** $J(\Theta)$
  - Find the the weights w_0, w_1 which yield the lowest value of $J(\Theta)$ for the entire data set $\mathcal{X}$

Loss & Cost

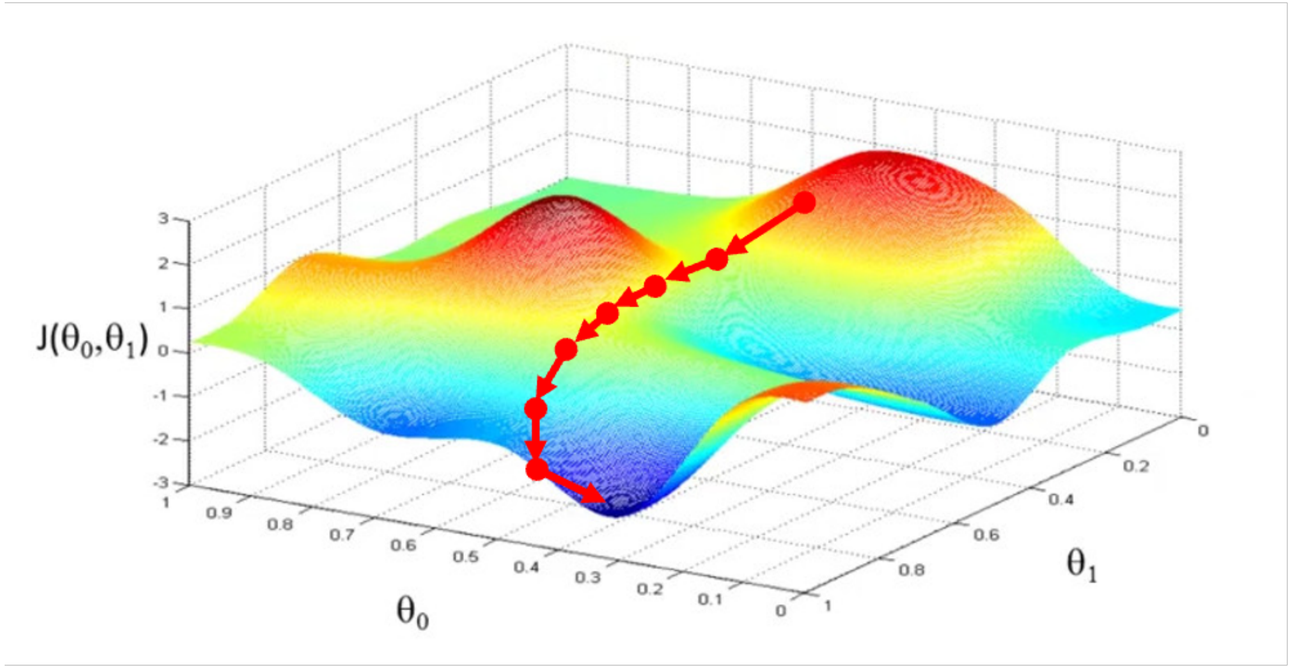


Loss & Cost

- Minimizing Cost

$$\Theta := \Theta - \alpha \nabla J(\Theta)$$

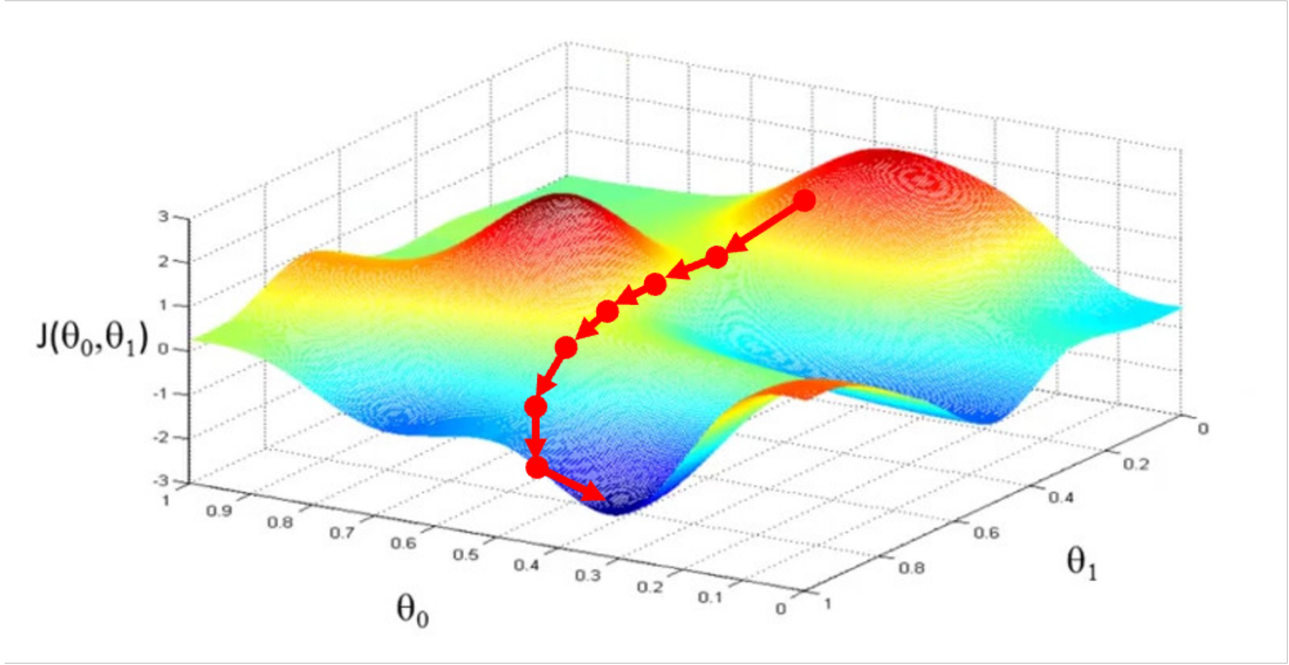
$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathscr{L}(h(x_i), y_i)$$



Loss & Cost

- Minimizing Cost

$$\Theta := \Theta - \alpha \nabla J(\Theta)$$

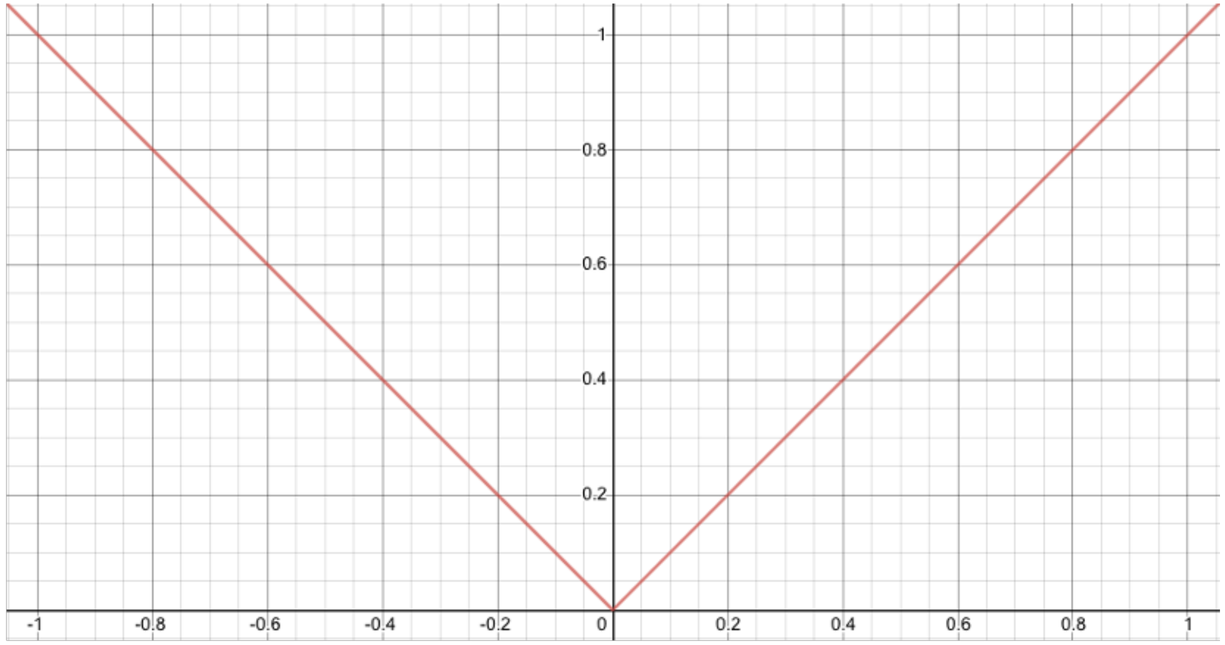


$$J(\Theta) = \frac{1}{m} \sum_{i}^{m} \mathscr{L}(h(x_i), y_i)$$

$$\nabla J(\Theta) \rightarrow \begin{pmatrix} \frac{\partial \mathscr{L}}{\partial w_0} \\ \frac{\partial \mathscr{L}}{\partial w_1} \end{pmatrix}$$

Loss Functions

Absolute Error - L1 Loss

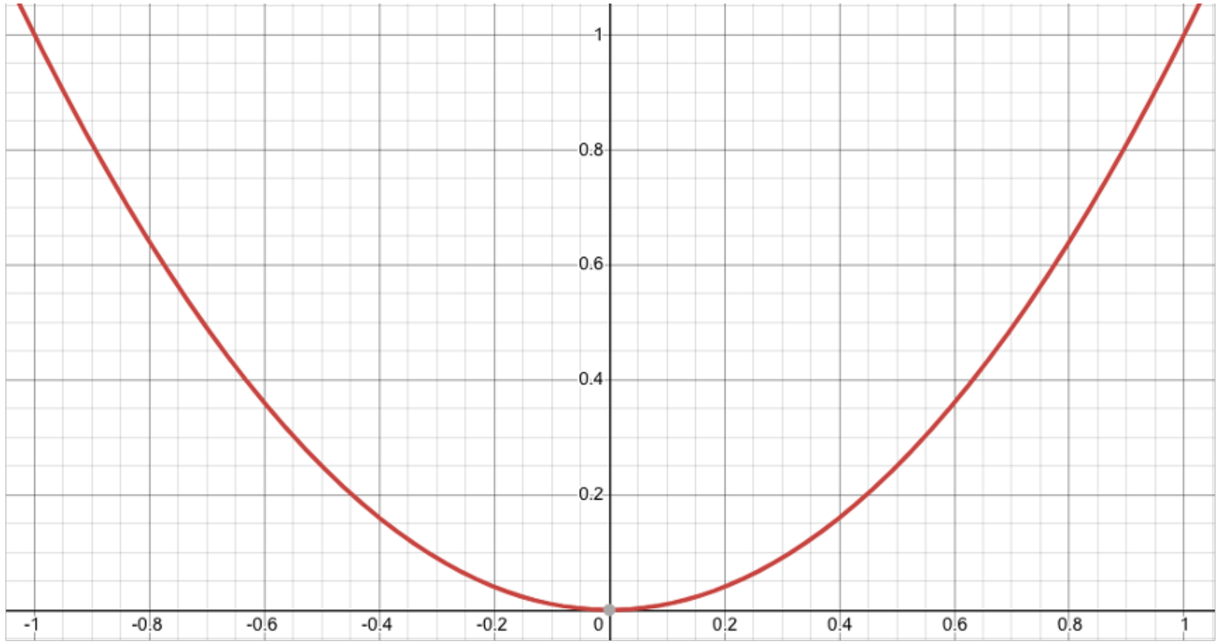


$$\mathscr{L}(\hat{y}, y) = |y - \hat{y}|$$

$$\frac{\partial \mathscr{L}}{\partial \hat{y}} = \begin{cases} -1 & y < \hat{y} \\ 1 & \text{otherwise} \end{cases}$$

Loss Functions

Square Error - L2 Loss

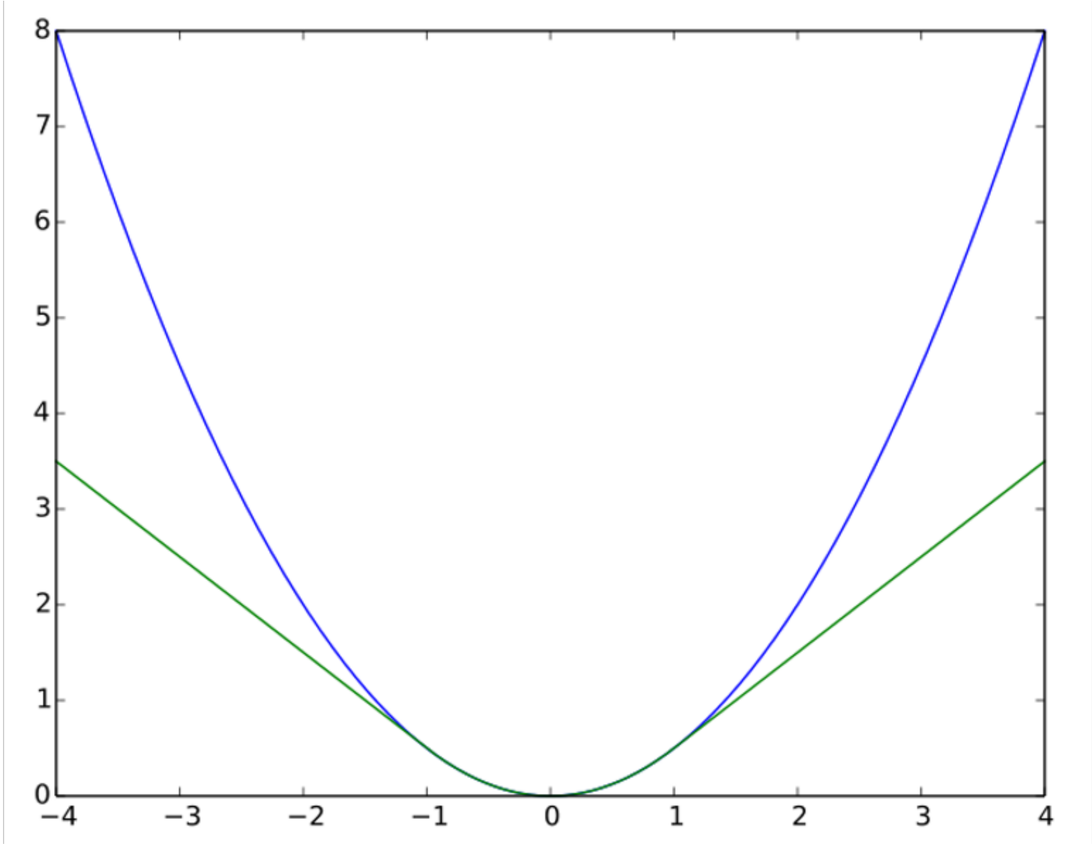


$$\mathscr{L}(\hat{y}, y) = (y - \hat{y})^2$$

$$\frac{\partial \mathscr{L}}{\partial \hat{y}} = -2(y - \hat{y})$$

Loss Functions

Huber Loss



$$\mathscr{L}(\hat{y}, y) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & if\,|y - \hat{y}| \leq \delta \\ \delta\,|y - \hat{y}| - \frac{1}{2}\delta^2 & if\,|y - \hat{y}| > \delta \end{cases}$$

$$\frac{\partial \mathscr{L}}{\partial \hat{y}} = \begin{cases} -\delta & if(y - \hat{y}) < -\delta \\ -(y - \hat{y}) & if -\delta \leq (y - \hat{y}) \leq \delta \\ \delta & if\delta < (y - \hat{y}) \end{cases}$$

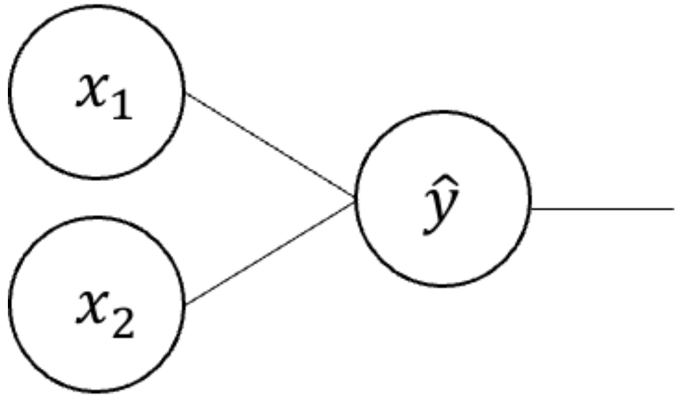
3.2 Forward Pass

Vectorized Form

Single Unit

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\hat{y} = a = g(z)$$

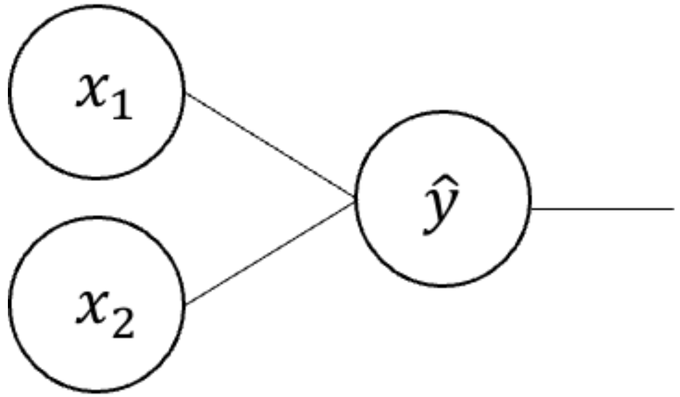


Vectorized Form

Single Unit

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\hat{y} = a = g(z)$$

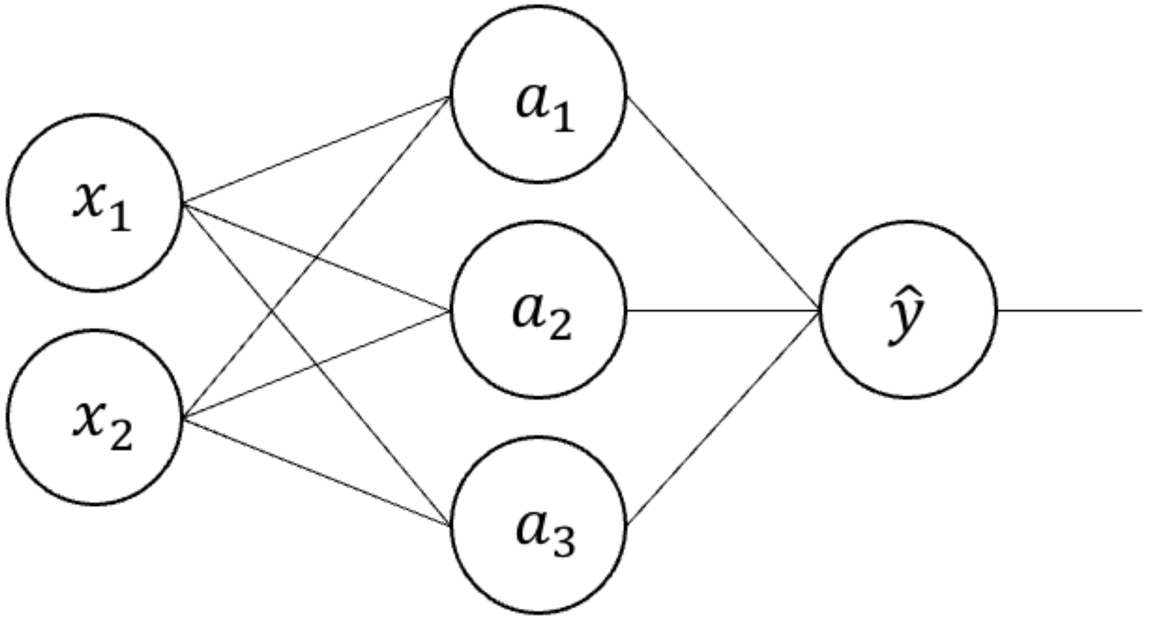


$$z = w^T x + b$$

$$z = \begin{pmatrix} - & w & - \end{pmatrix} \begin{pmatrix} | \\ x \\ | \end{pmatrix} + b$$

Vectorized Form

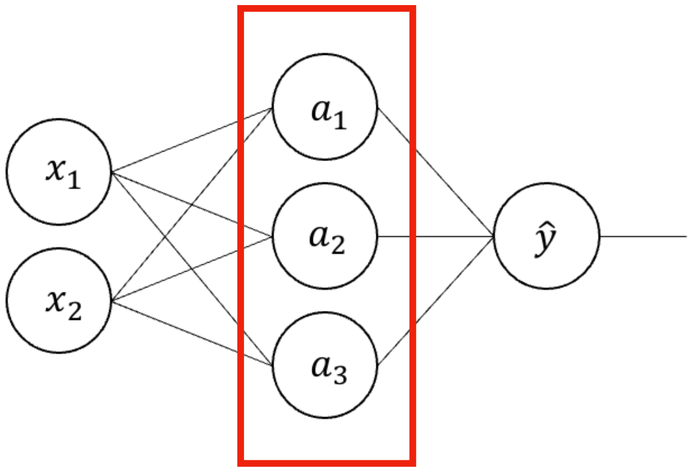
Hidden Layer



Vectorized Form

Hidden Layer

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} - & w_1 & - \\ - & w_2 & - \\ - & w_3 & - \end{pmatrix} \begin{pmatrix} | \\ x \\ | \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$



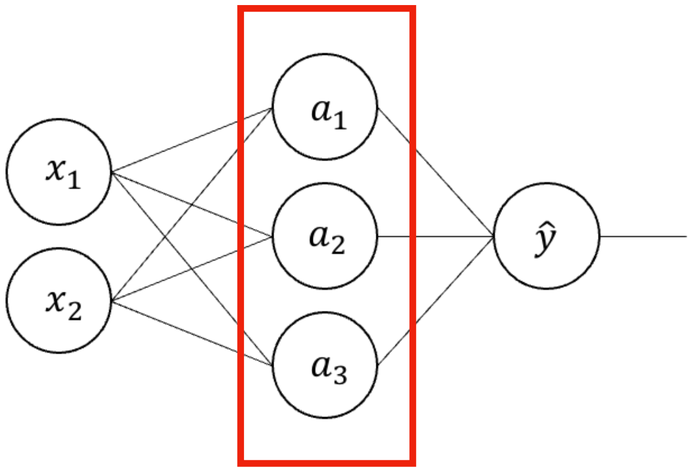
Vectorized Form

Hidden Layer

$$\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} - & w_1 & - \\ - & w_2 & - \\ - & w_3 & - \end{pmatrix} \begin{pmatrix} | \\ x \\ | \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$z = Wx + b$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = g\left(\begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}\right)$$



Vectorized Form

Multiple Samples

- Let us have a training set containing m samples

$$\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \qquad \mathcal{Y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$$

- Can we compute multiple samples at once ?

Vectorized Form

Multiple Samples

$$\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \qquad \mathcal{Y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$$

$$\begin{pmatrix} | & | & & | \\ z^{(1)} & z^{(2)} & \ldots & z^{(m)} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} - & w_1 & - \\ - & w_2 & - \\ - & w_3 & - \end{pmatrix} \begin{pmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\ | & | & & | \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Vectorized Form

Multiple Samples

$$\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \qquad \mathcal{Y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$$

$$\begin{pmatrix} | & | & & | \\ z^{(1)} & z^{(2)} & \ldots & z^{(m)} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} - & w_1 & - \\ - & w_2 & - \\ - & w_3 & - \end{pmatrix} \begin{pmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\ | & | & & | \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$Z = WX + b$$

Vectorized Form

Multiple Samples

$$\mathcal{X} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\} \qquad \mathcal{Y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$$

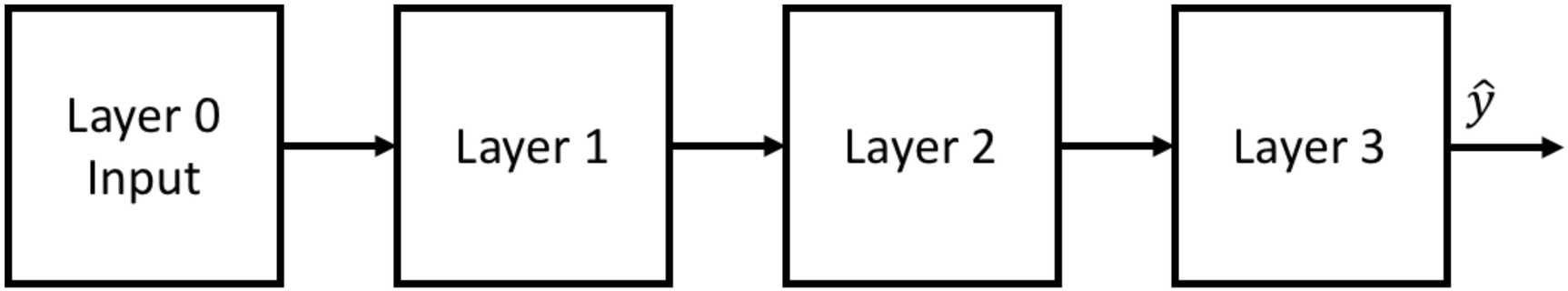
$$\begin{pmatrix} | & | & & | \\ z^{(1)} & z^{(2)} & \ldots & z^{(m)} \\ | & | & & | \end{pmatrix} = \begin{pmatrix} - & w_1 & - \\ - & w_2 & - \\ - & w_3 & - \end{pmatrix} \begin{pmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \ldots & x^{(m)} \\ | & | & & | \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

$$Z = WX + b$$

$$A = g(Z)$$

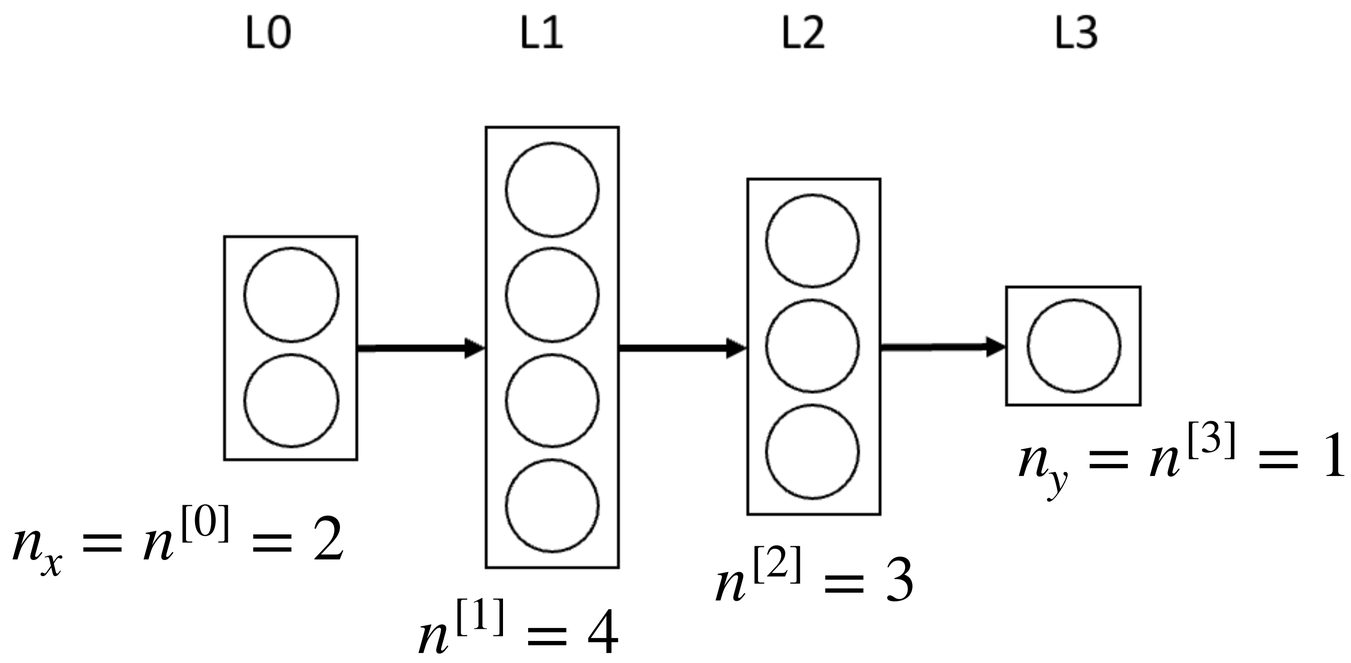
Artificial Neural Network

- Layer ~ Function
- Network ~ Compound Function



Artificial Neural Network

Notation

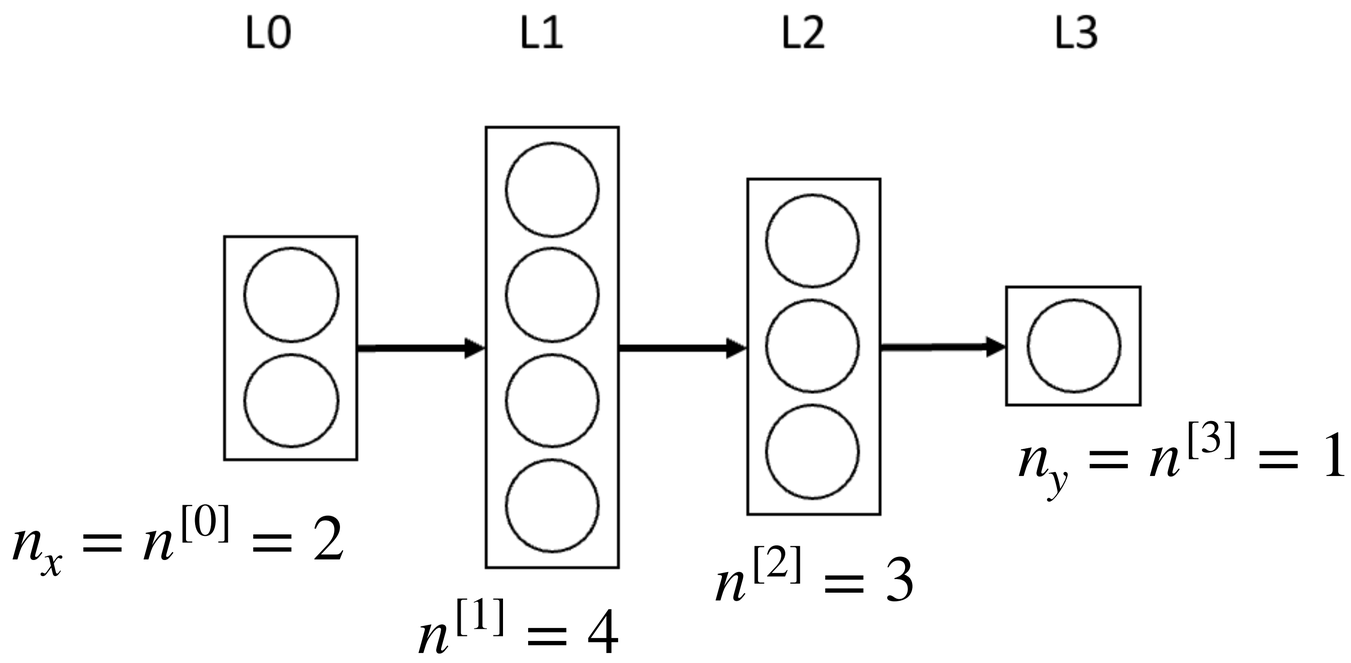


L=3

$n^{[l]}$

Artificial Neural Network

Notation



L=3

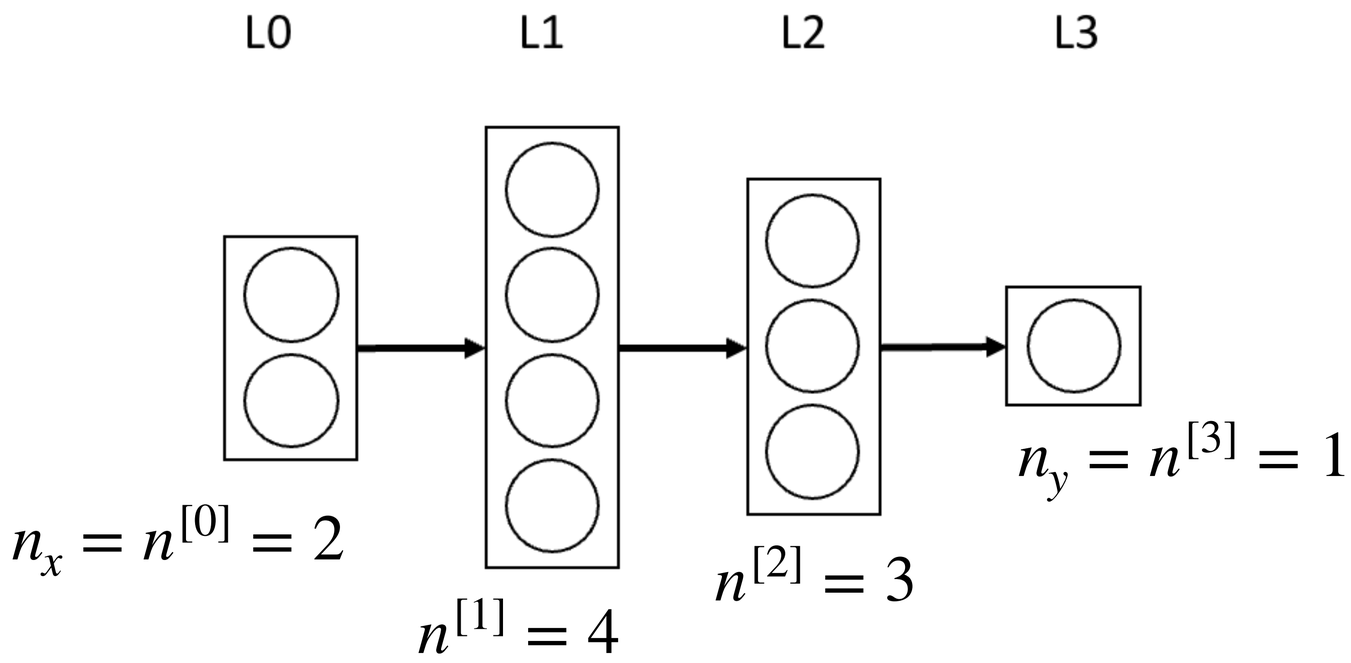
$n^{[l]}$

$a^{[l]}$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

Artificial Neural Network

Notation



L=3

$n^{[l]}$

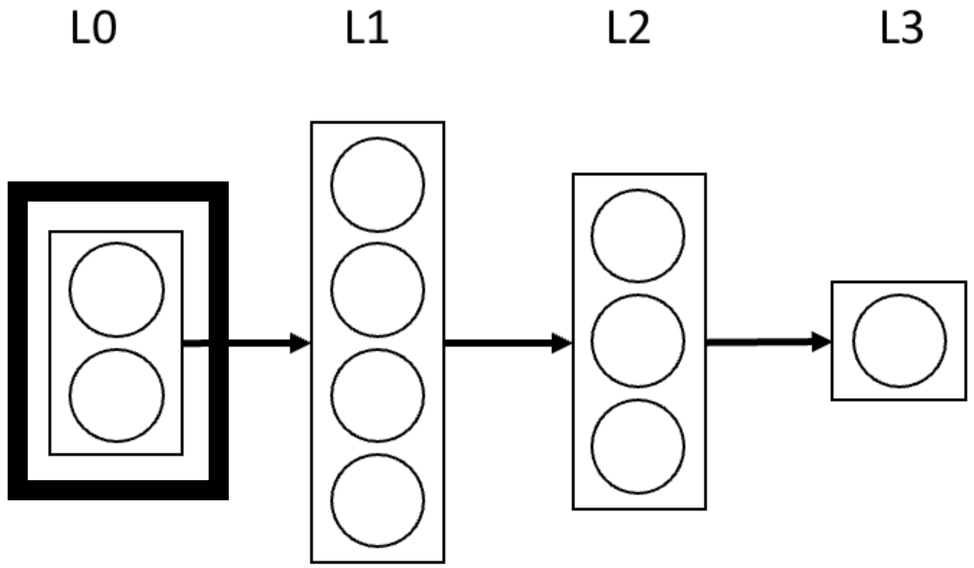
$A^{[l]}$

$$A^{[l]} = g^{[l]}(Z^{[l]})$$

$$W^{[l]}, b^{[l]}$$

Forward Pass

$$a^{[0]} = x$$

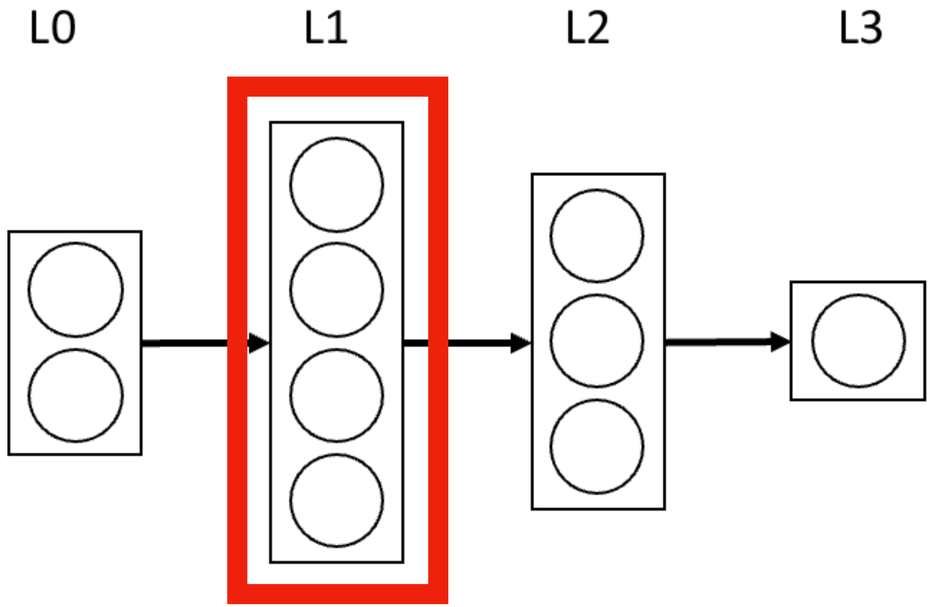


Forward Pass

$$a^{[0]} = x$$

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]})$$



Forward Pass

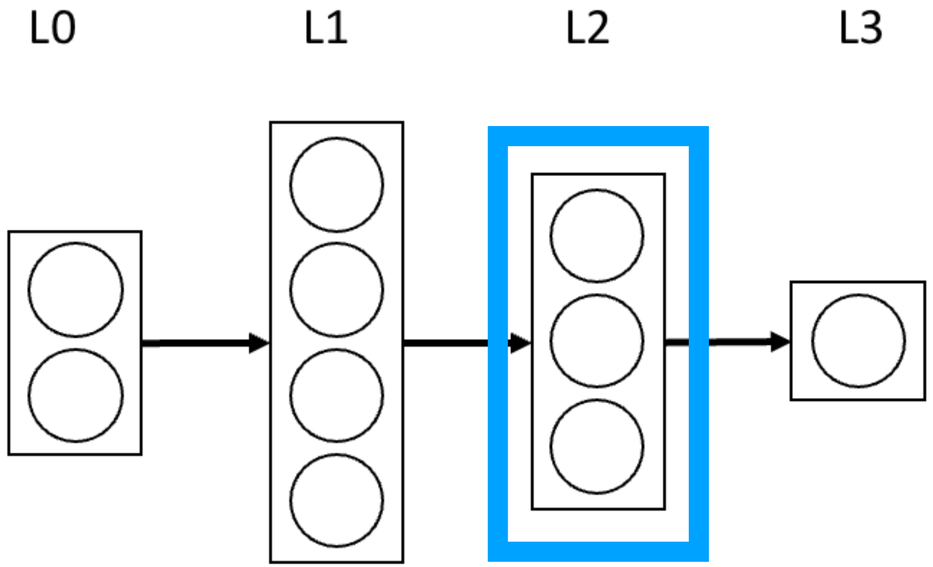
$$a^{[0]} = x$$

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g^{[2]}(z^{[2]})$$



Forward Pass

$$a^{[0]} = x$$

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

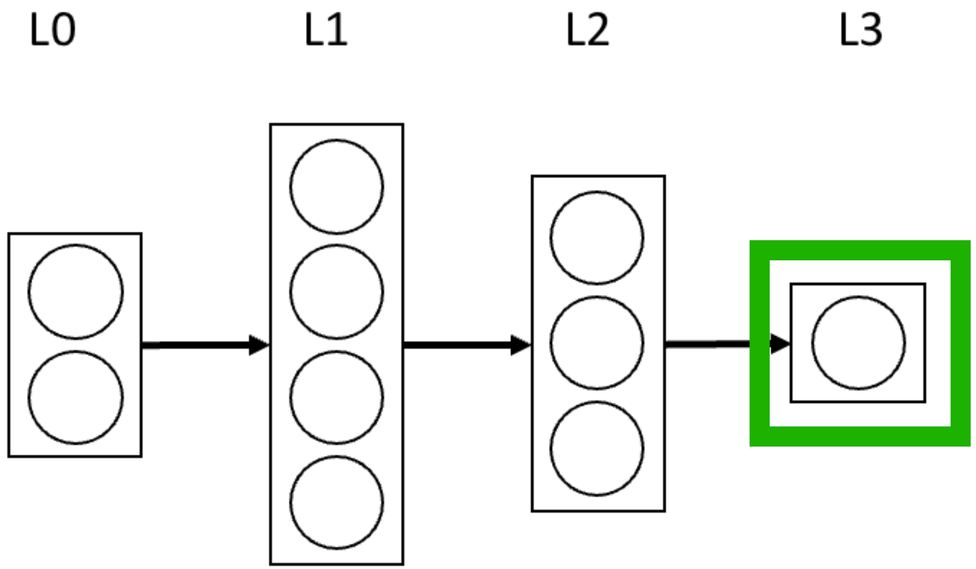
$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g^{[2]}(z^{[2]})$$

$$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$$

$$a^{[3]} = g^{[3]}(z^{[3]})$$



Forward Pass

$$a^{[0]} = x$$

$$z^{[1]} = W^{[1]}a^{[0]} + b^{[1]}$$

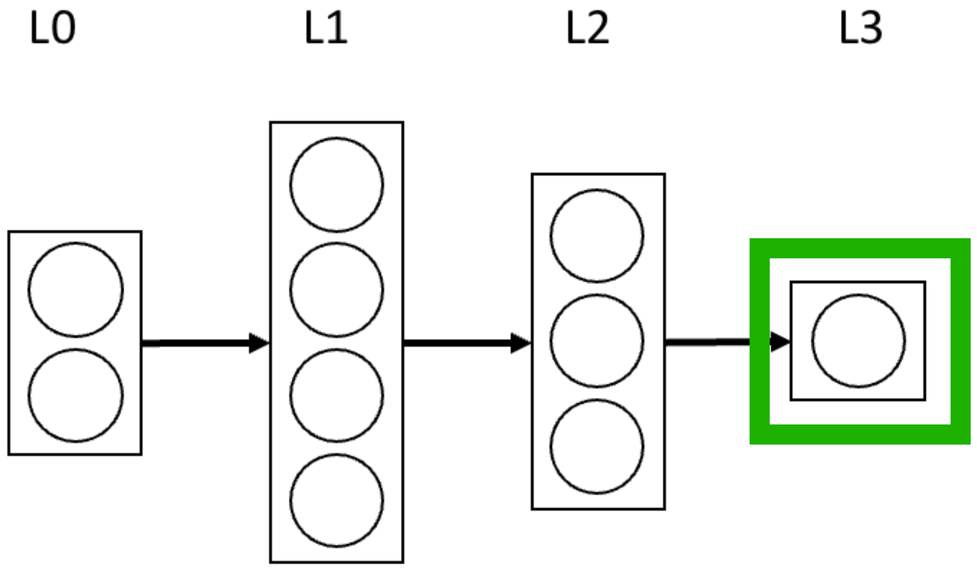
$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$

$$a^{[2]} = g^{[2]}(z^{[2]})$$

$$z^{[3]} = W^{[3]}a^{[2]} + b^{[3]}$$

$$a^{[3]} = g^{[3]}(z^{[3]})$$

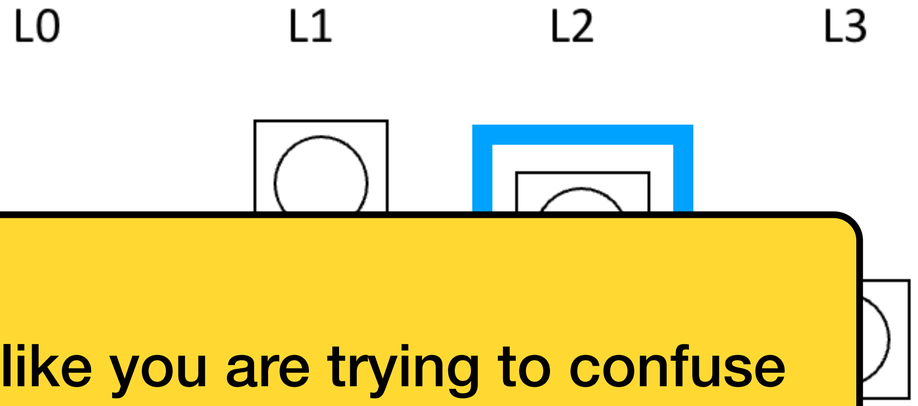


$$z^{[l]} = W^{[l]}a^{[l-1]} + b^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

Matrix Dimensions

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]}$$
$$a^{[2]} = g^{[2]}(z^{[2]})$$

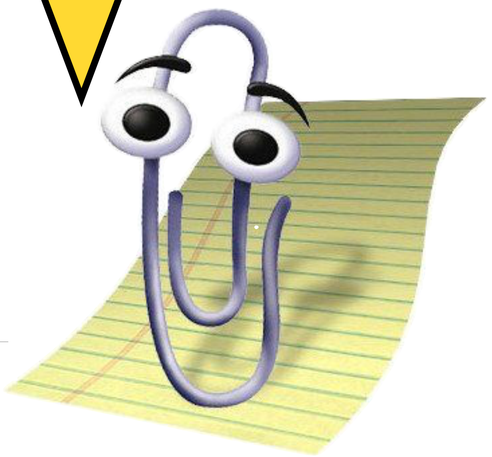


It looks like you are trying to confuse everyone.

Are there any questions ?

$$\underset{(3x1)}{\begin{bmatrix} z_1^{[2]} \\ z_2^{[2]} \\ z_3^{[2]} \end{bmatrix}} = \underset{(3x4)}{\begin{bmatrix} w_{11}^{[2]} & w_{12}^{[2]} & w_{13}^{[2]} & w_{14}^{[2]} \\ w_{21}^{[2]} & w_{22}^{[2]} & w_{23}^{[2]} & w_{24}^{[2]} \\ w_{31}^{[2]} & w_{32}^{[2]} & w_{33}^{[2]} & w_{34}^{[2]} \end{bmatrix}} \underset{(4x1)}{\begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix}} + \underset{(3x1)}{\begin{bmatrix} b_1^{[2]} \\ b_2^{[2]} \\ b_3^{[2]} \end{bmatrix}}$$

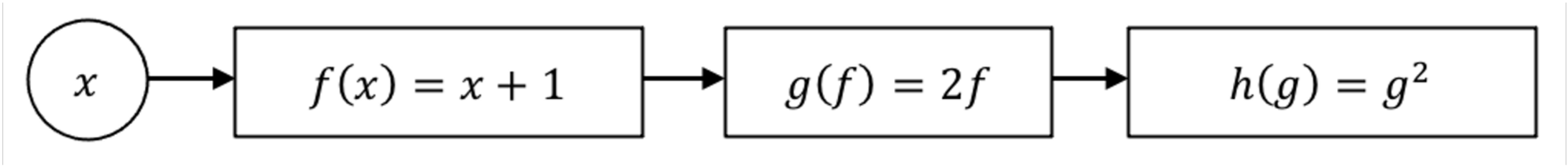
$$(n^{[l]} \; x \; n^{[l-1]})$$



3.3 Backward Pass

Computational Graph

- Represent compound function as a sequence of functions



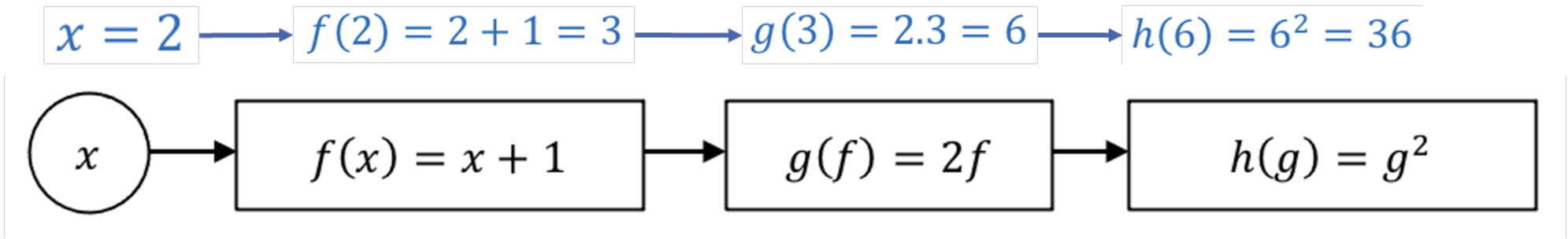
$$h(x) = h\Big(\, g\big(f(x)\big)\,\Big)$$

$$h(x) = \big(2(x+1)\big)^2$$

$$h(x) = 4x^2 + 8x + 4$$

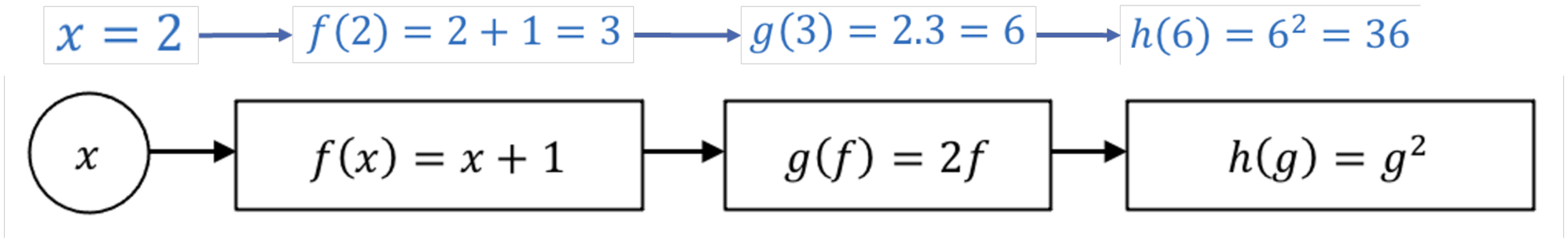
Computational Graph

Forward Pass



Computational Graph

Forward Pass



What is the derivative of compound function ?

Chain Rule

- Two differentiable functions

$$h(x), \quad g(y)$$

$$h'(x) = \frac{dh(x)}{dx}, \quad g'(y) = \frac{dg(y)}{dy}$$

Chain Rule

- Two differentiable functions

$$h(x), \quad g(y)$$

$$h'(x) = \frac{dh(x)}{dx}, \quad g'(y) = \frac{dg(y)}{dy}$$

- Compound function

$$f(y) = h\Big(g(y)\Big)$$

Chain Rule

- Two differentiable functions $h(x), \quad g(y)$

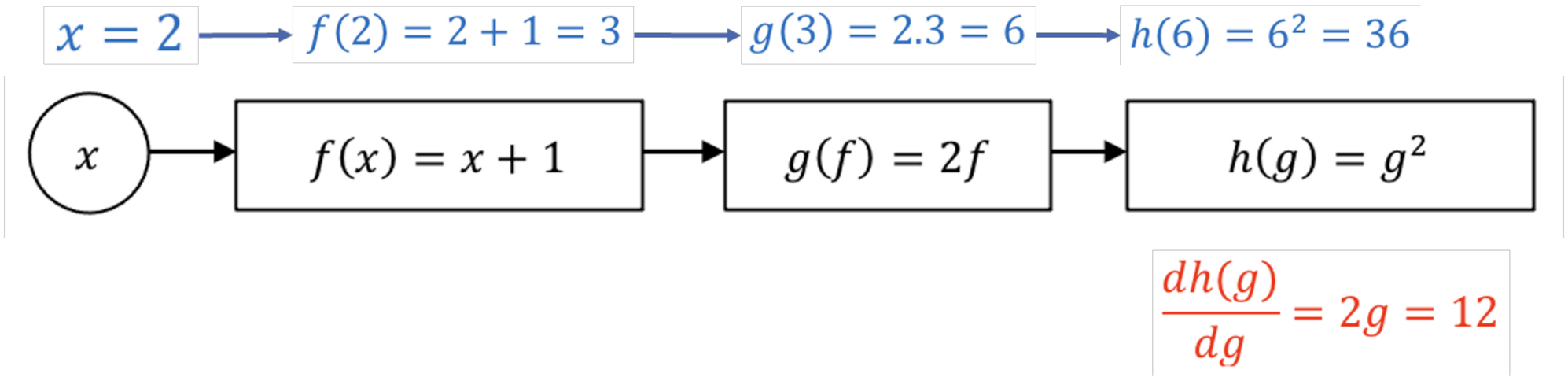
$$h'(x) = \frac{dh(x)}{dx}, \quad g'(y) = \frac{dg(y)}{dy}$$

- Compound function $f(y) = h\big(g(y)\big)$

$$f'(y) = h'\big(g(y)\big)g'(y)$$

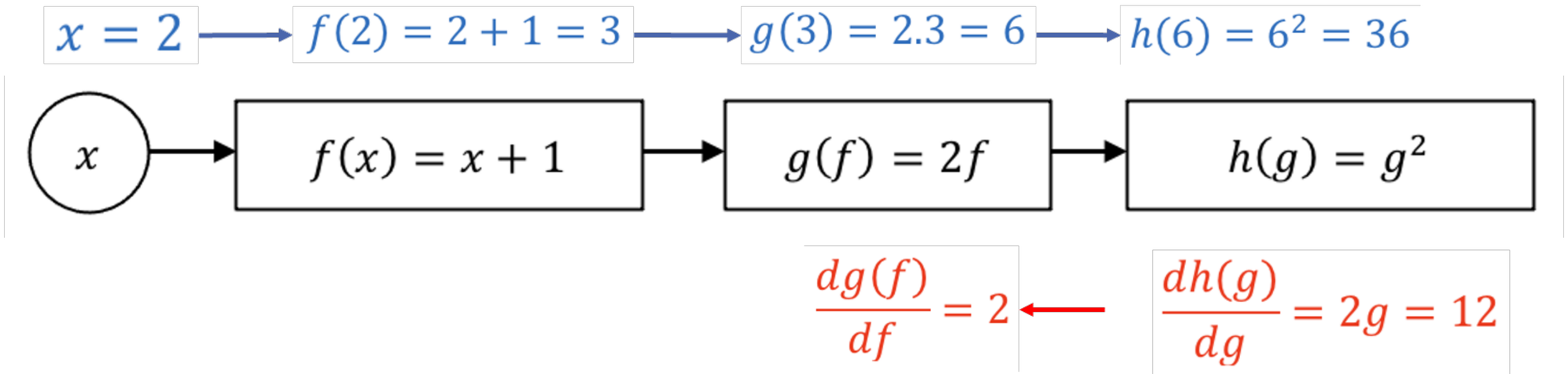
Computational Graph

Backward Pass



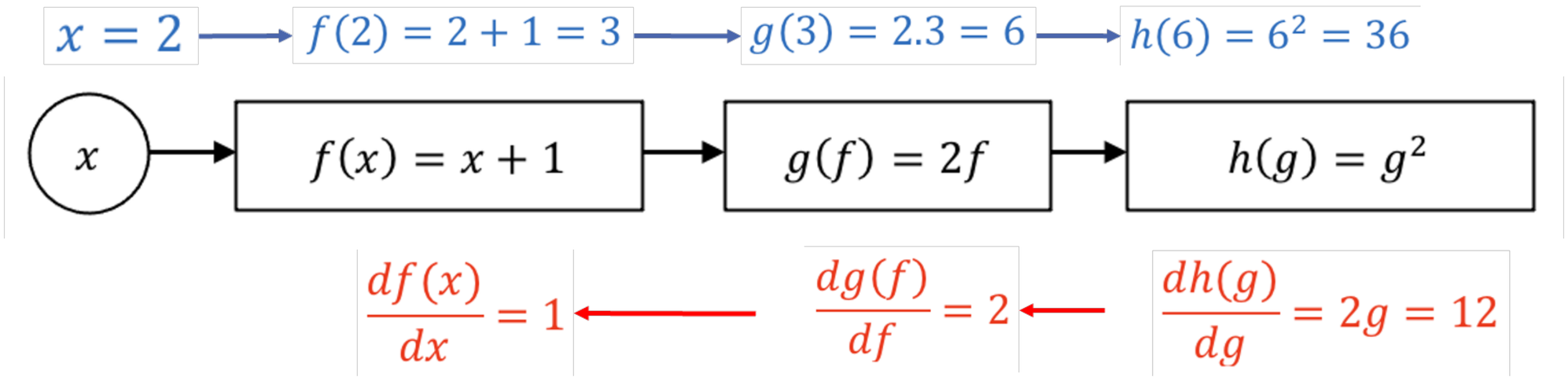
Computational Graph

Backward Pass



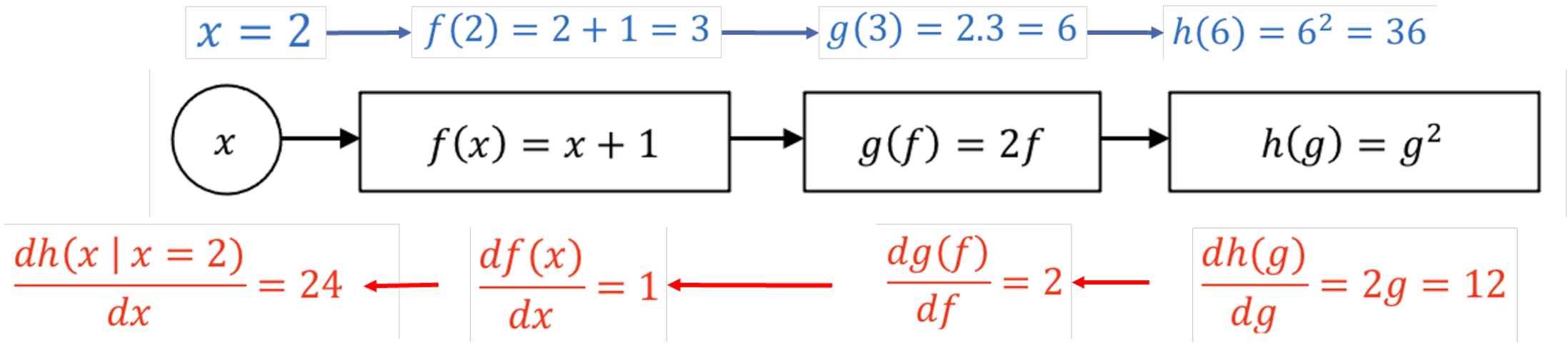
Computational Graph

Backward Pass



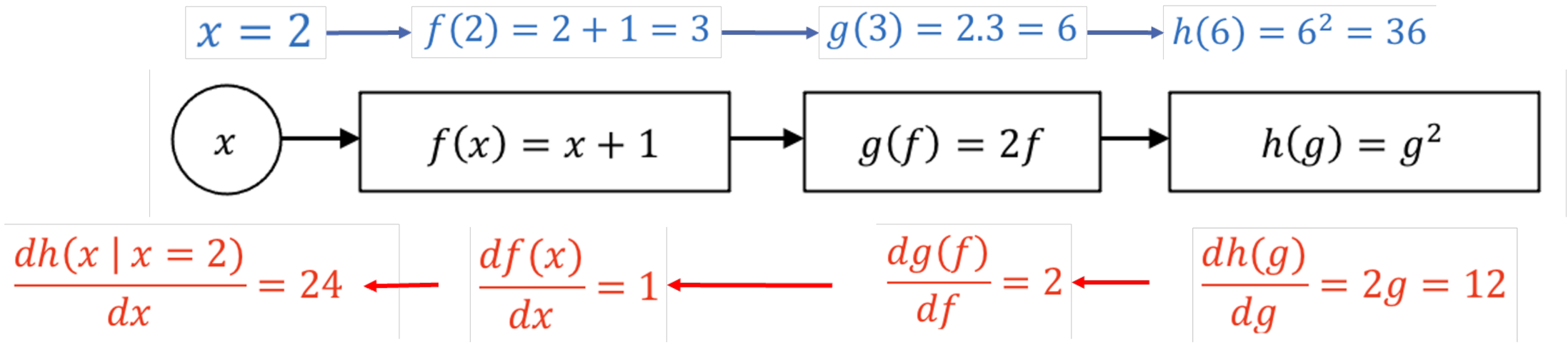
Computational Graph

Backward Pass



Computational Graph

Backward Pass



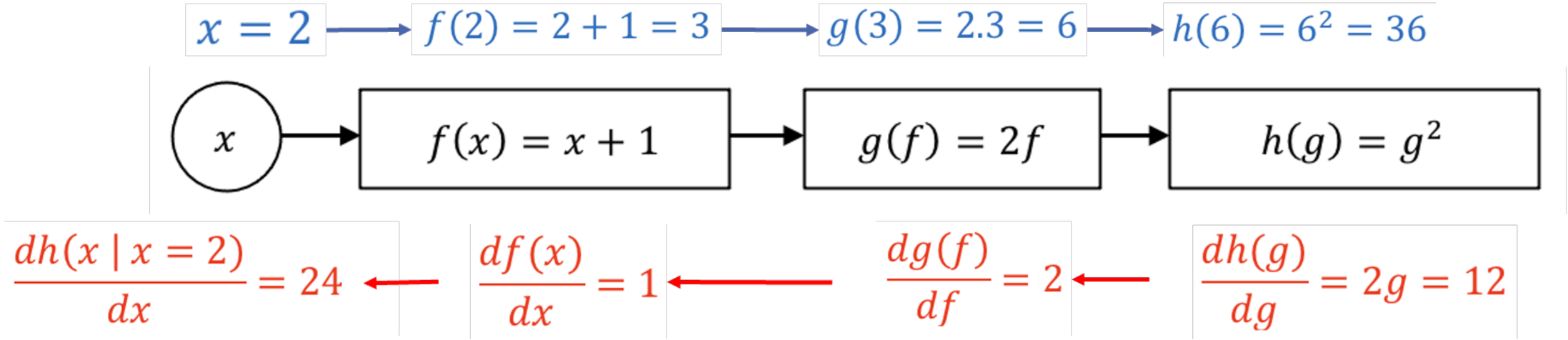
$$h(x) = h\Big(g\big(f(x)\big)\Big)$$

$$h(x) = \big(2(x+1)\big)^2$$

$$h(x) = 4x^2 + 8x + 4$$

Computational Graph

Backward Pass



$$h(x) = h\Big(g\big(f(x)\big)\Big)$$

$$h(x) = \big(2(x+1)\big)^2$$

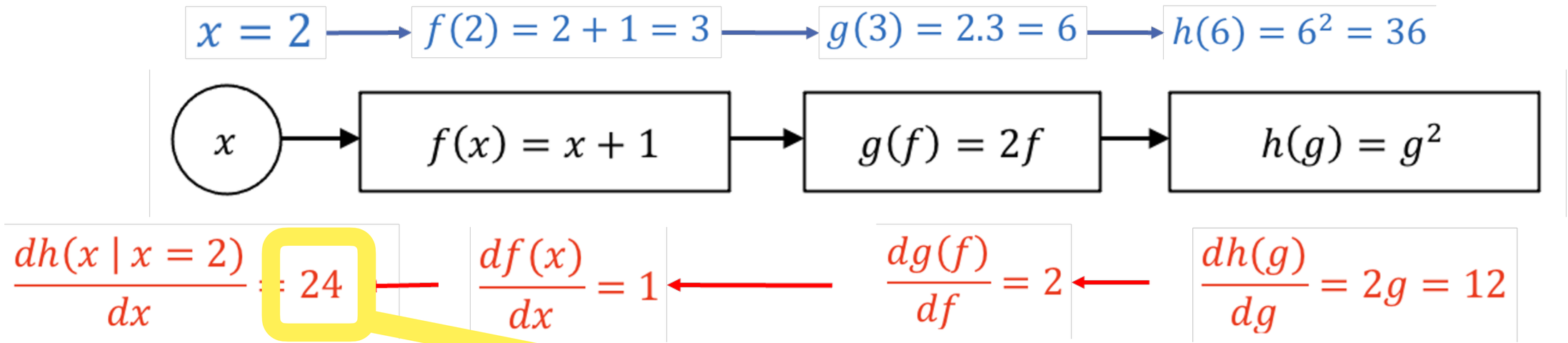
$$h(x) = 4x^2 + 8x + 4$$

$$h'(x) = 8x + 8$$

$$h'(x \mid x = 2) = 16 + 8 = 24$$

Computational Graph

Backward Pass



$$h(x) = h\Big(g\big(f(x)\big)\Big)$$

$$h(x) = \big(2(x+1)\big)^2$$

$$h(x) = 4x^2 + 8x + 4$$

$$h'(x) = 8x + 8$$

$$h'(x \mid x = 2) = 16 + 8 = 24$$

Example - Logistic Regression

Setup

- 1 training sample with 2 input features
- Sigmoid activation function
- Logistic loss function

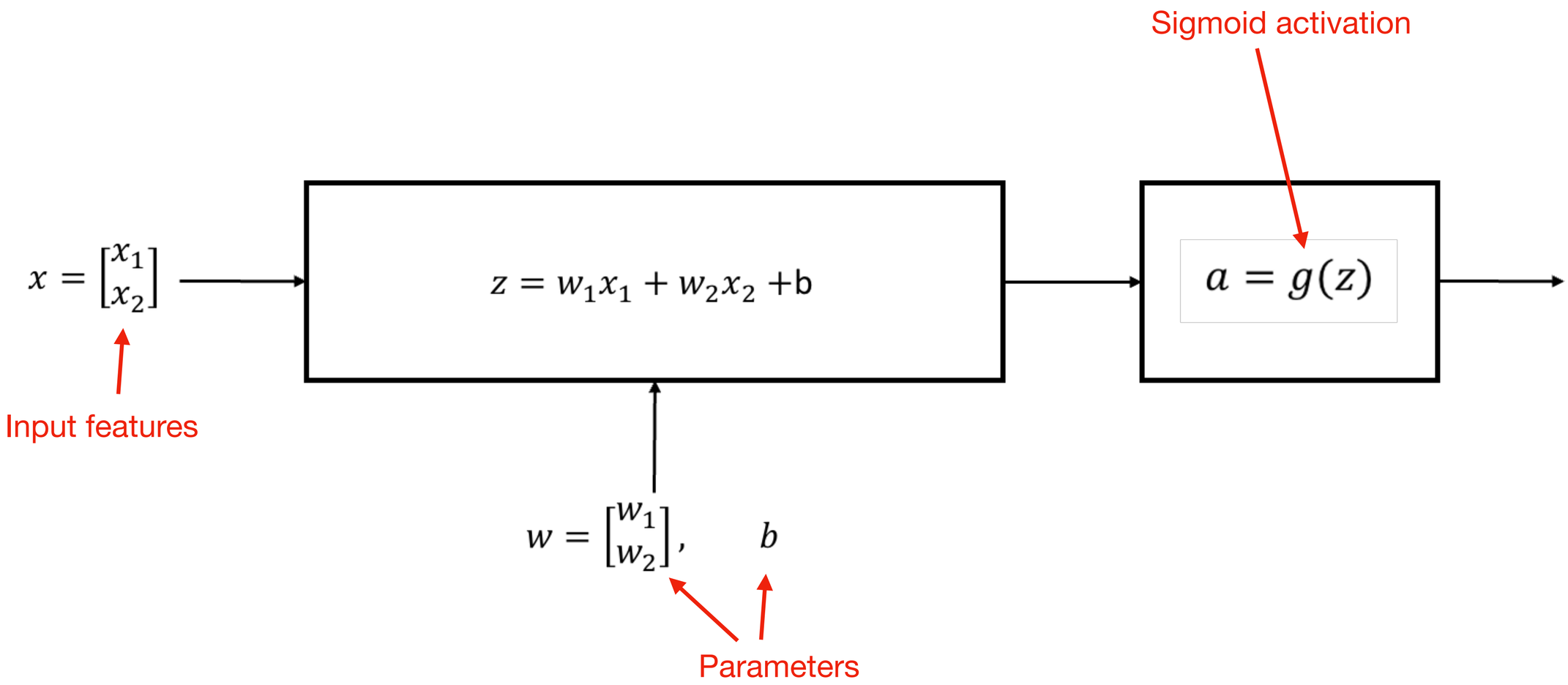
Example - Logistic Regression

Back-propagation of Errors Training Algorithm

1. Forward pass
    - compute loss
2. Backward pass
    - compute gradients
3. One step of the gradient descent
    - update parameters

Example - Logistic Regression

One Artificial Neuron



Example - Logistic Regression

Logistic Loss

- The true outcome can either be 0, or 1
- Model predicts the **probability** of

$$\hat{y} = p(y = 1 \mid x)$$

$$y \in \{0,1\}$$

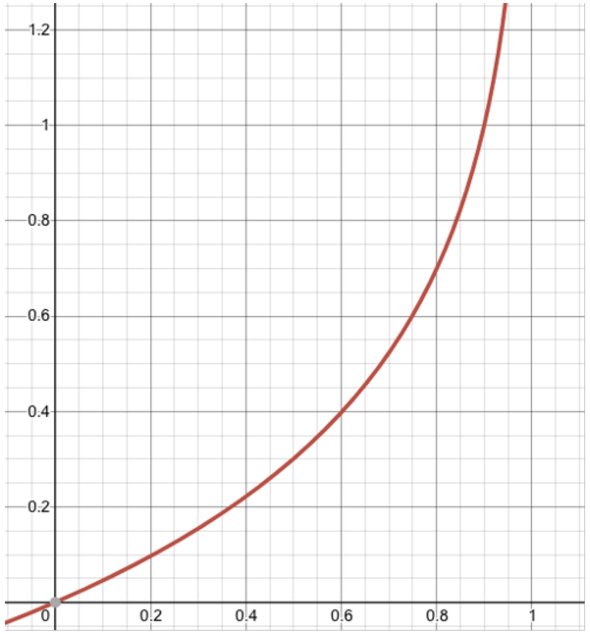
$$\hat{y} \in (0;1)$$

Sigmoid activation !!!

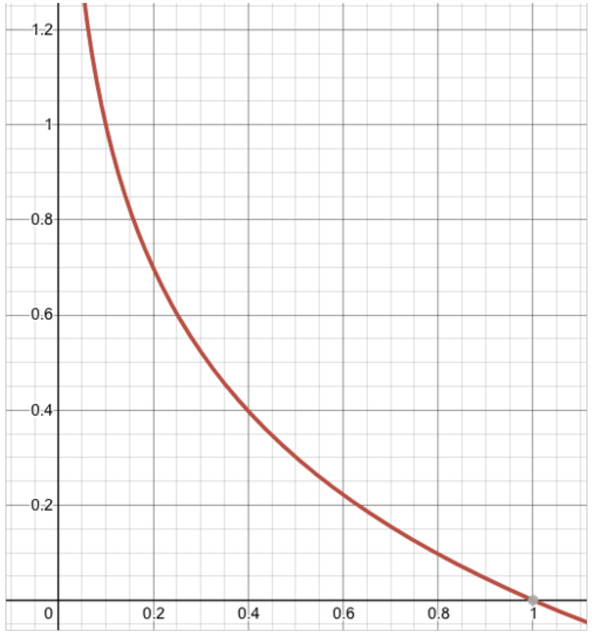
Example - Logistic Regression

Logistic Loss

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



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

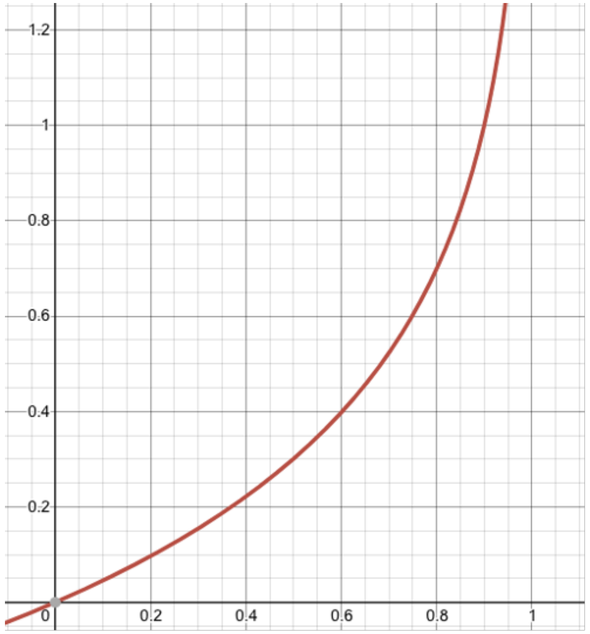


$-\log(\hat{y})$

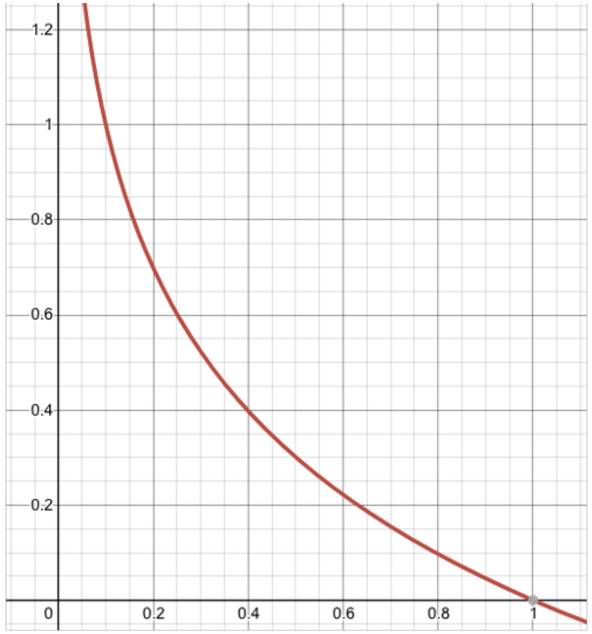
Example - Logistic Regression

Logistic Loss

$$\mathscr{L}(\hat{y}, y) = -\left(y \log(\hat{y}) + (1 - y)\log(1 - \hat{y})\right)$$



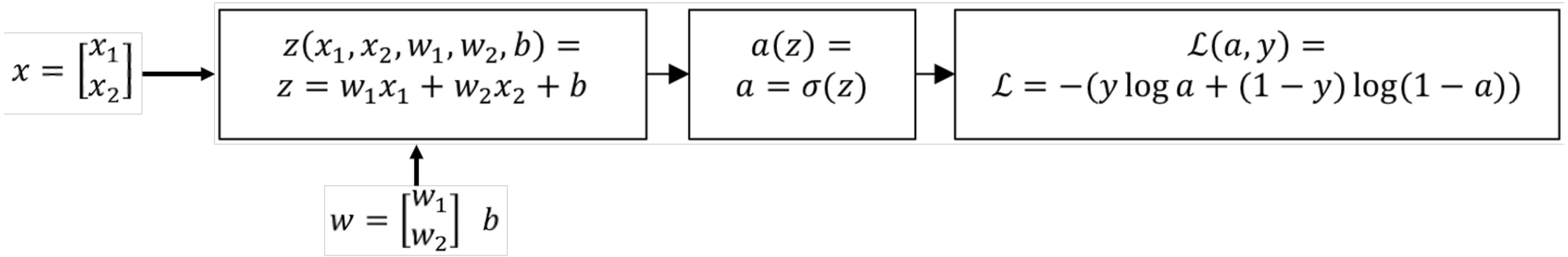
$-\log(1 - \hat{y})$



$-\log(\hat{y})$

Example - Logistic Regression

Computational Graph



Example - Logistic Regression

Functions and Their Derivatives

- Linear function - trivial
- Sigmoid
- Logistic loss

Example - Logistic Regression

Functions and Their Derivatives

- Sigmoid

$$a = \sigma(z)$$
$$a' = a(1 - a)$$

$$\begin{aligned} g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\ &= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\ &= \frac{1}{(1+e^{-z})} \cdot \left(1 - \frac{1}{(1+e^{-z})}\right) \\ &= g(z)(1-g(z)). \end{aligned}$$

Example - Logistic Regression

Functions and Their Derivatives

- Logistic loss

$$\frac{d}{dx}\ln(x) = \frac{1}{x}$$

$$\frac{d}{dx}\ln\left[f(x)\right] = \frac{1}{f(x)}f'(x)$$

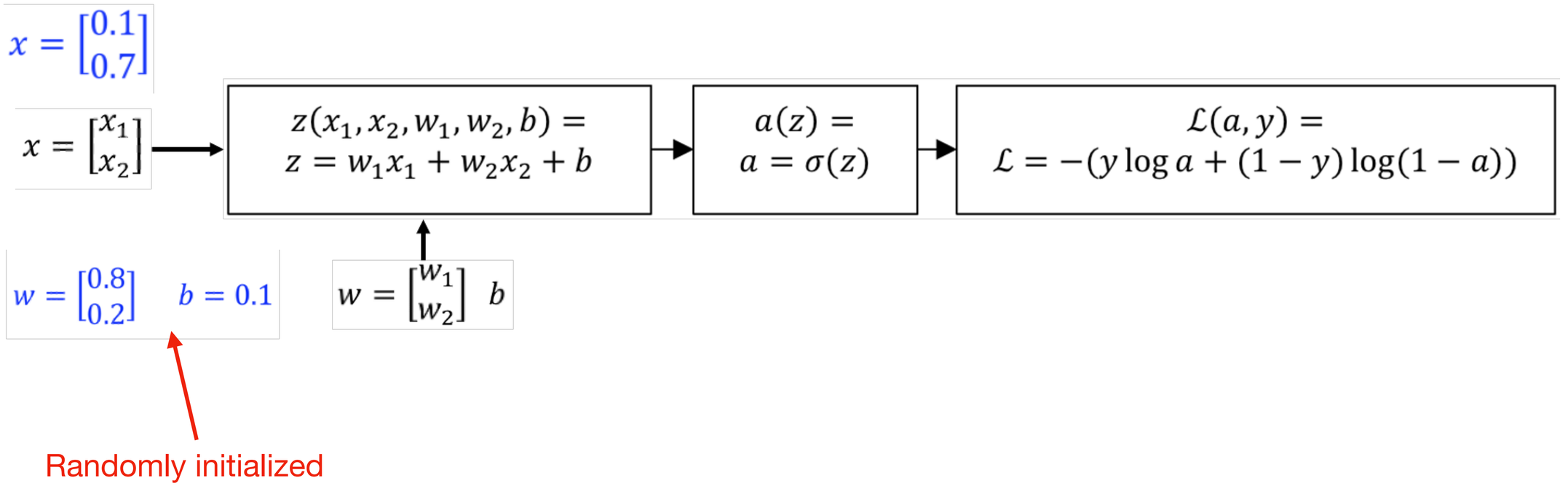
$$\mathcal{L}(a, y) = -(y \log a + (1 - y) \log(1 - a))$$

$$\frac{\partial \mathcal{L}(a, y)}{\partial a} = \frac{-y}{a} + \frac{(1 - y)}{(1 - a)}$$

Example - Logistic Regression

Forward Pass

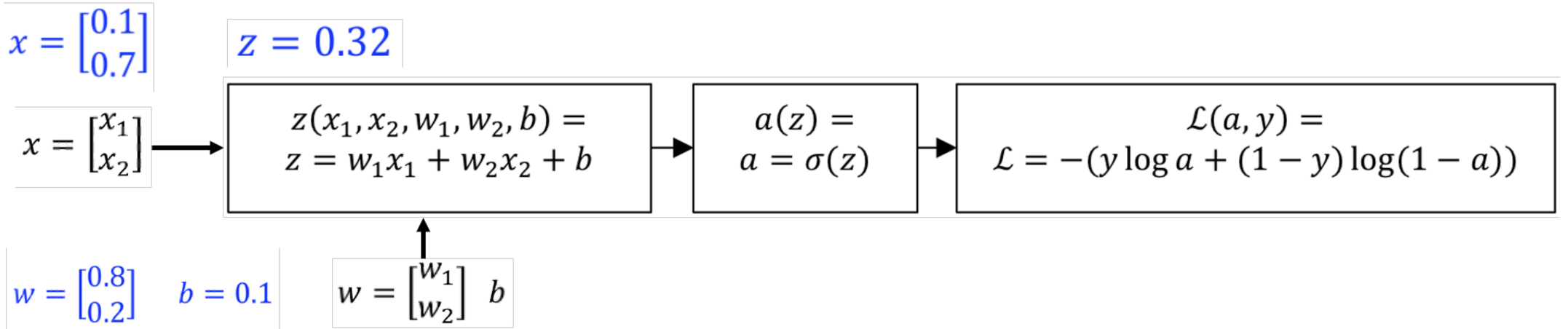
$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



Example - Logistic Regression

Forward Pass

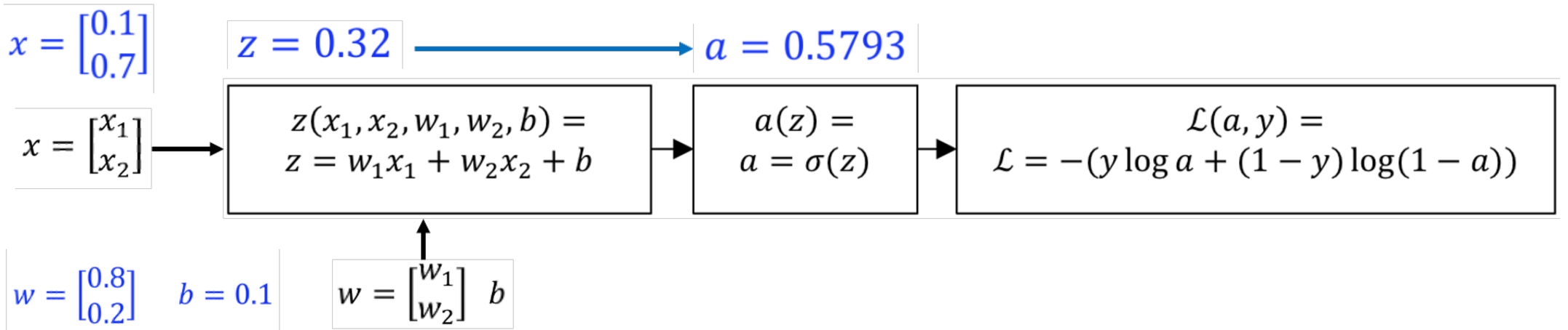
$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



Example - Logistic Regression

Forward Pass

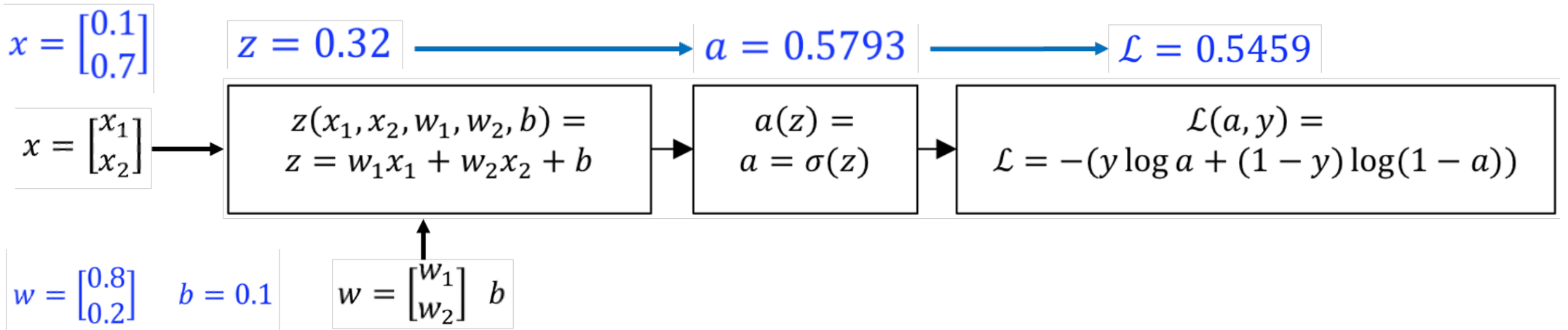
$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



Example - Logistic Regression

Forward Pass

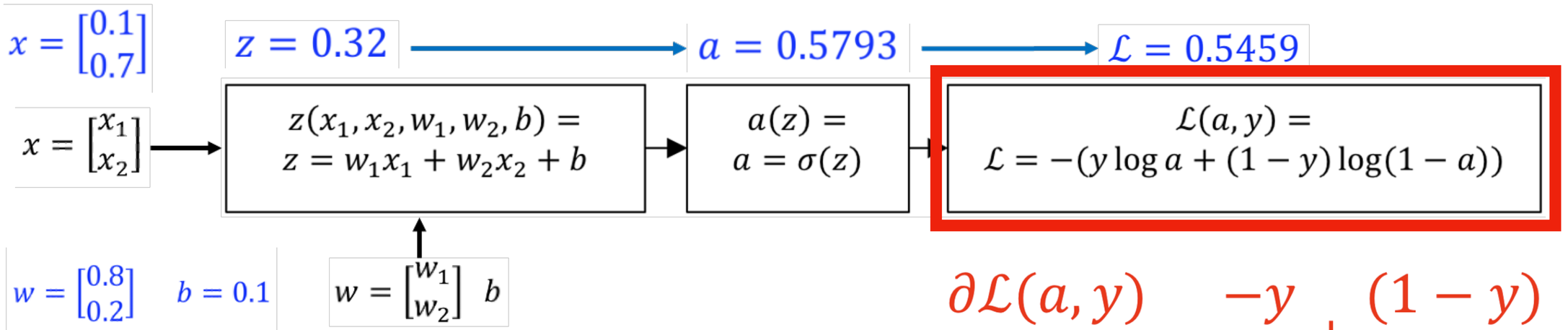
$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$

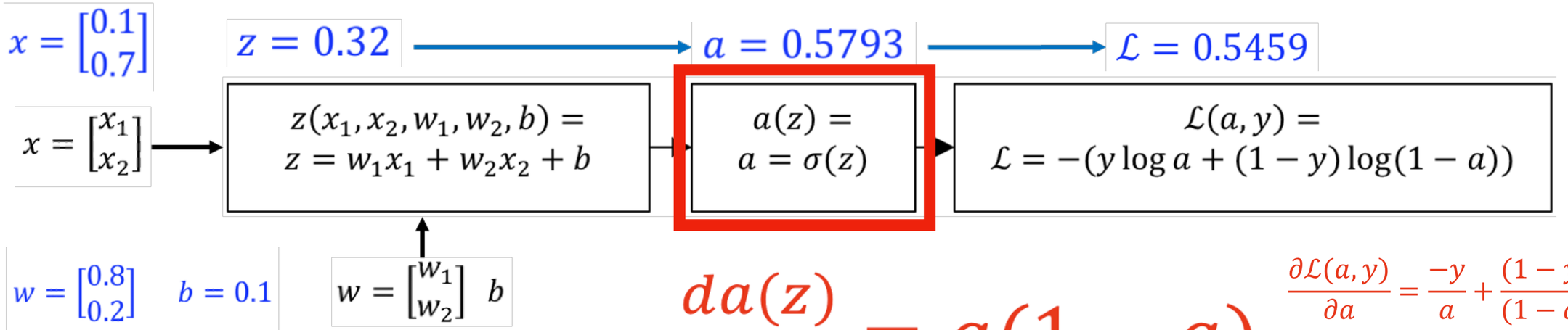


$$\frac{\partial \mathcal{L}(a, y)}{\partial a} = \frac{-y}{a} + \frac{(1 - y)}{(1 - a)}$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



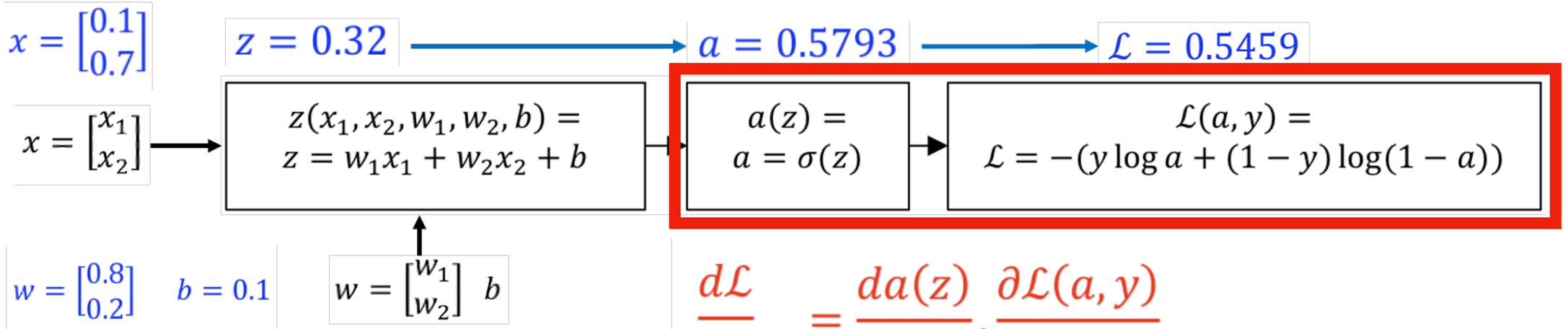
$$\frac{da(z)}{dz} = a(1 - a)$$

$$\frac{\partial \mathcal{L}(a, y)}{\partial a} = \frac{-y}{a} + \frac{(1 - y)}{(1 - a)}$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



$$\frac{d\mathcal{L}}{dz} = \frac{da(z)}{dz} \cdot \frac{\partial \mathcal{L}(a, y)}{\partial a}$$

$$= a(1 - a)\left(\frac{-y}{a} + \frac{(1 - y)}{(1 - a)}\right)$$

$$= \frac{-ya(1 - a)}{a} + \frac{a(1 - a)(1 - y)}{(1 - a)}$$

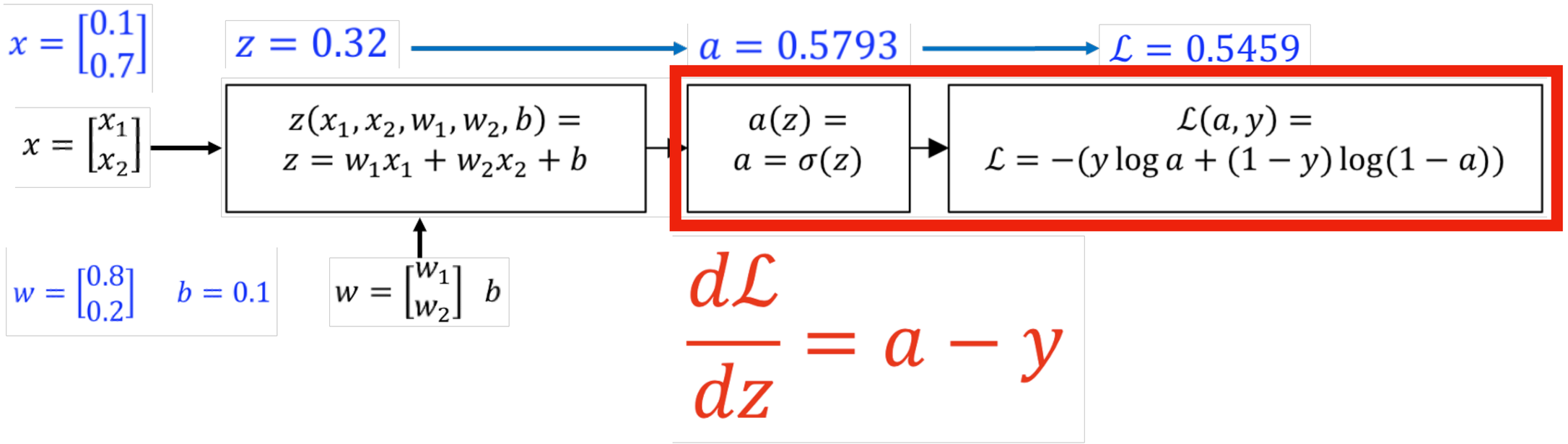
$$= -y(1 - a) + a(1 - y)$$

$$= a - y$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$

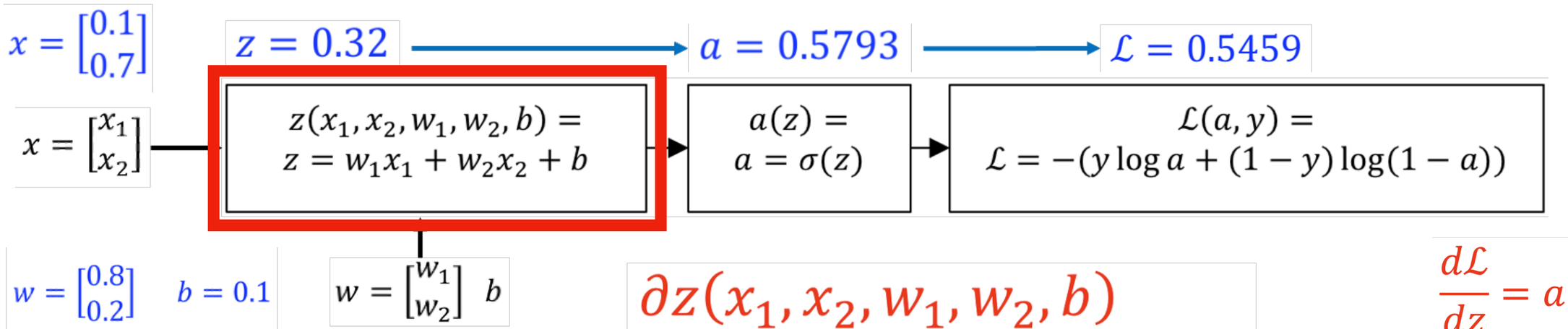


$$\frac{d\mathcal{L}}{dz} = a - y$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



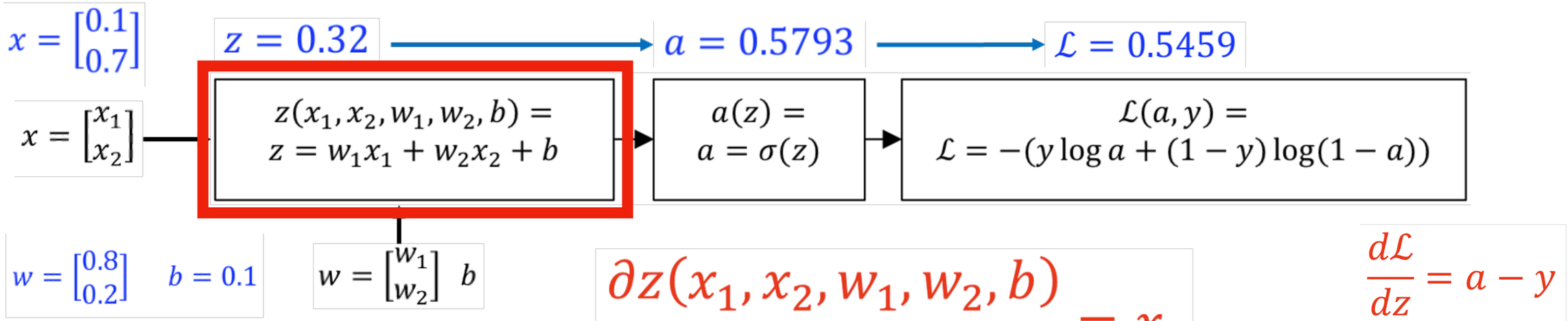
$$\frac{\partial z(x_1, x_2, w_1, w_2, b)}{\partial w_1} = x_1$$

$$\frac{d\mathcal{L}}{dz} = a - y$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



$$\frac{\partial z(x_1, x_2, w_1, w_2, b)}{\partial w_1} = x_1$$

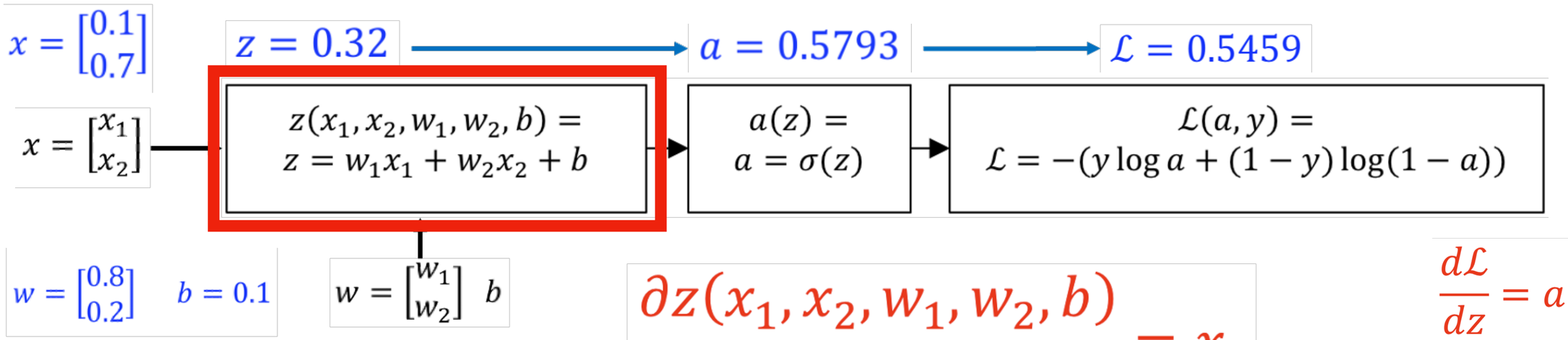
$$\frac{d\mathcal{L}}{dz} = a - y$$

$$\frac{\partial z(x_1, x_2, w_1, w_2, b)}{\partial w_2} = x_2$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



$$\frac{\partial z(x_1, x_2, w_1, w_2, b)}{\partial w_1} = x_1$$

$$\frac{d\mathcal{L}}{dz} = a - y$$

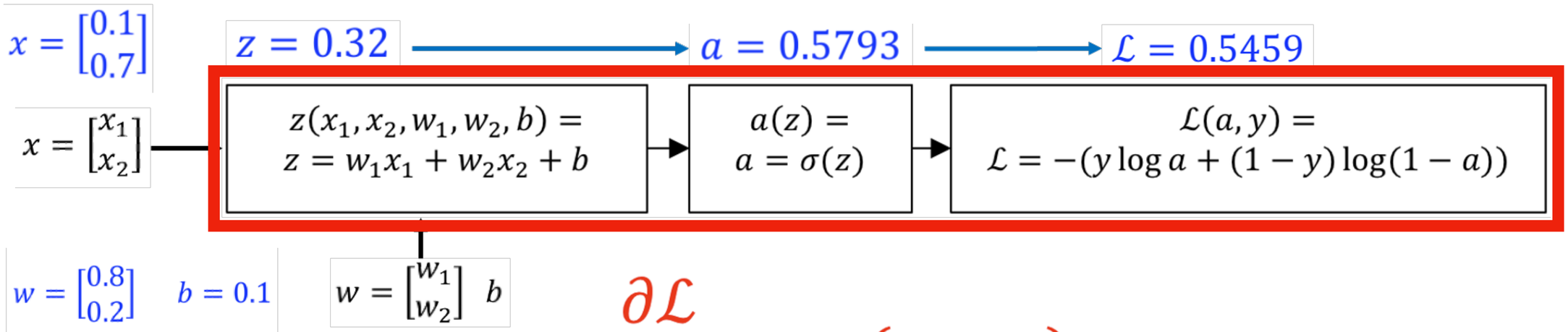
$$\frac{\partial z(x_1, x_2, w_1, w_2, b)}{\partial b} = 1$$

$$\frac{\partial z(x_1, x_2, w_1, w_2, b)}{\partial w_2} = x_2$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



$$\frac{\partial \mathcal{L}}{\partial w_1} = x_1(a - y)$$

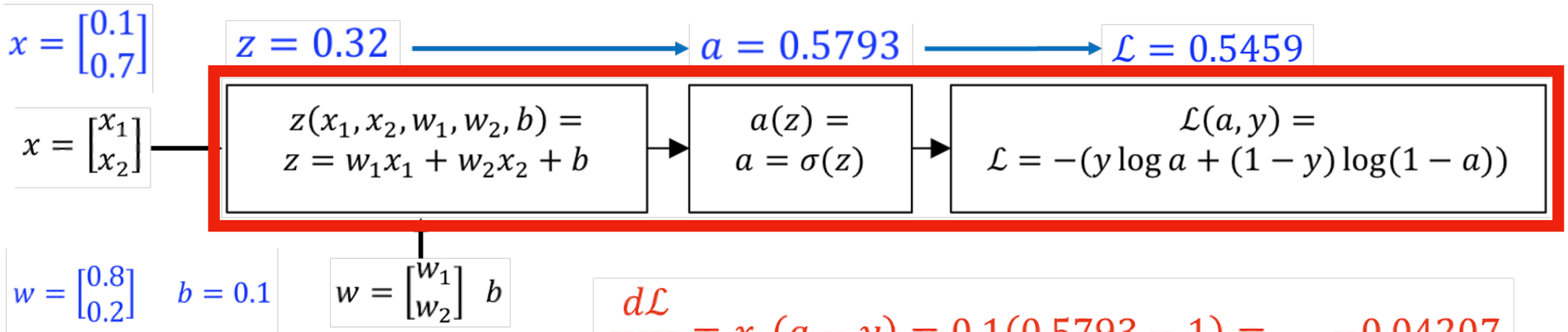
$$\frac{\partial \mathcal{L}}{\partial w_2} = x_2(a - y)$$

$$\frac{\partial \mathcal{L}}{\partial b} = (a - y)$$

Example - Logistic Regression

Backward Pass

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



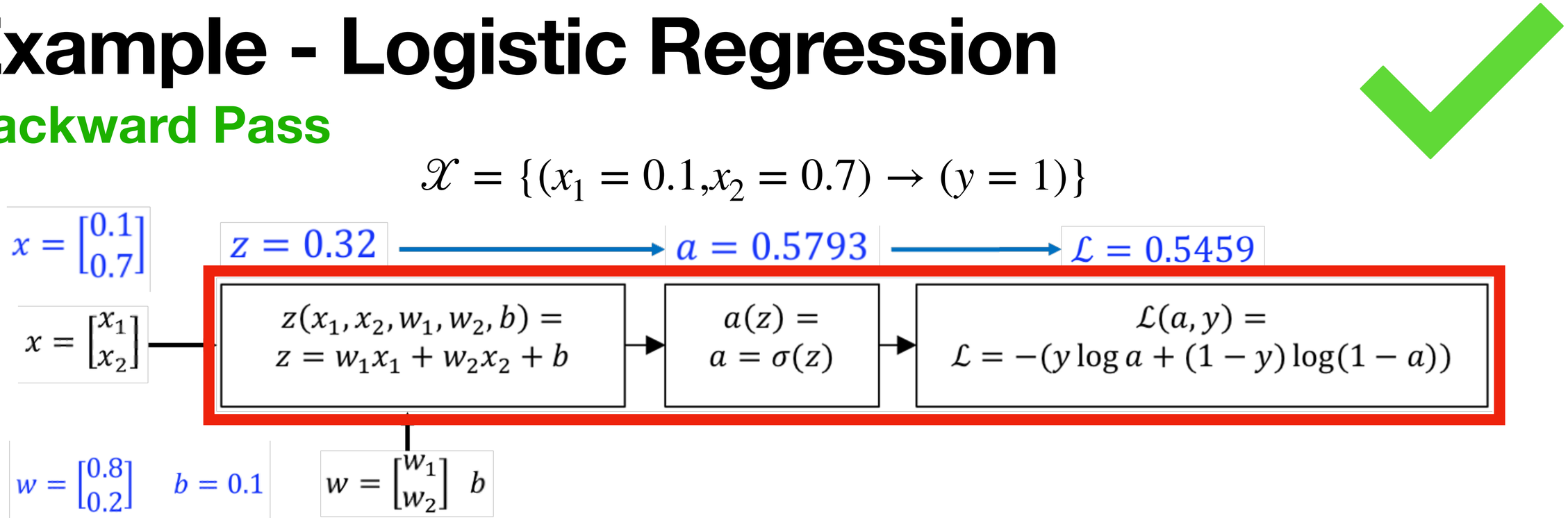
$$\frac{d\mathcal{L}}{dw_1} = x_1(a - y) = 0.1(0.5793 - 1) = -0.04207$$

$$\frac{d\mathcal{L}}{dw_2} = x_2(a - y) = 0.7(0.5793 - 1) = -0.29447$$

$$\frac{d\mathcal{L}}{db} = (a - y) = (0.5793 - 1) = -0.4207$$

Example - Logistic Regression

Backward Pass



$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$

$x = \begin{bmatrix} 0.1 \\ 0.7 \end{bmatrix}$ $z = 0.32$ $a = 0.5793$ $\mathcal{L} = 0.5459$

$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$z(x_1, x_2, w_1, w_2, b) = z = w_1 x_1 + w_2 x_2 + b$

$a(z) = a = \sigma(z)$

$\mathcal{L}(a, y) = \mathcal{L} = -(y \log a + (1 - y) \log(1 - a))$

$w = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}$ $b = 0.1$

$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$ b

$$\nabla w = \begin{bmatrix} -0.04207 \\ -0.29447 \end{bmatrix} \qquad \nabla b = -0.4207$$

Example - Logistic Regression

One Step of Gradient Descent

- Update model parameters in the **opposite direction** of the gradient

$$\Theta := \Theta - \alpha \nabla J(\Theta)$$

Example - Logistic Regression

One Step of Gradient Descent

- Update model parameters in the **opposite direction** of the gradient

$$\Theta := \Theta - \alpha \nabla J(\Theta)$$

$$\alpha = 0.5$$

Example - Logistic Regression

One Step of Gradient Descent

- Update model parameters in the **opposite direction** of the gradient

$$w_1 := w_1 - \alpha \frac{d\mathcal{L}}{dw_1} = 0.8 - 0.5(-0.04207) \quad = 0.82103$$

$$w_2 := w_2 - \alpha \frac{d\mathcal{L}}{dw_2} = 0.2 - 0.5(-0.29447) = 0.3472$$

$$b := b - \alpha \frac{d\mathcal{L}}{db} = 0.1 - 0.5(-0.4207) \quad = 0.31033$$

Example - Logistic Regression

One Step of Gradient Descent

- Update model parameters in the **opposite direction** of the gradient

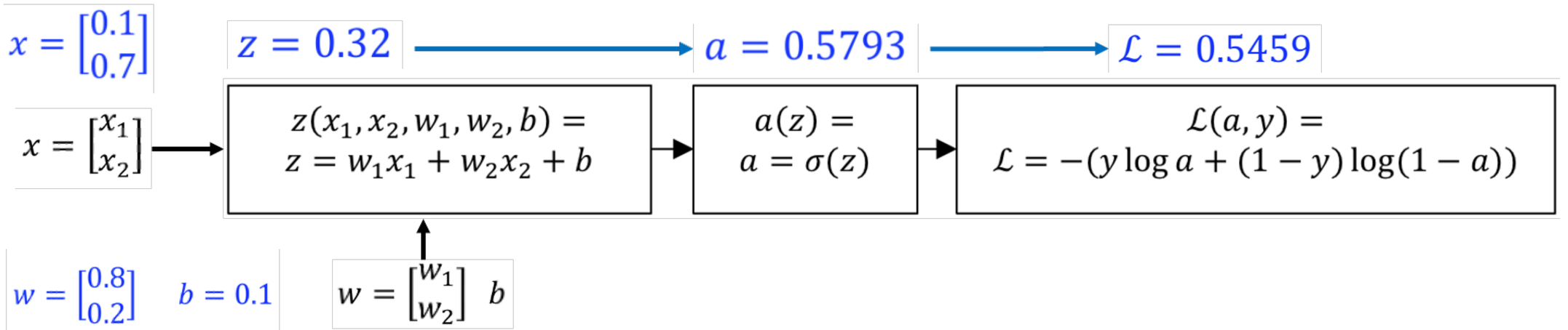
$$w = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix} \quad b = 0.1$$

$$w = \begin{bmatrix} 0.82103 \\ 0.3472 \end{bmatrix} \quad b = 0.31033$$

Example - Logistic Regression

Verification

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



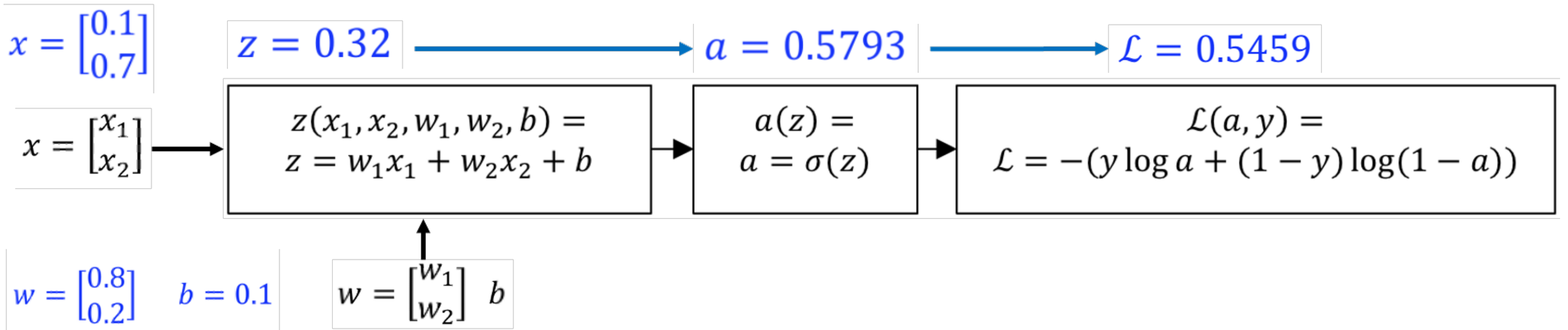
$$z = 0.6355$$

$$w = \begin{bmatrix} 0.82103 \\ 0.3472 \end{bmatrix} \quad b = 0.31033$$

Example - Logistic Regression

Verification

$$\mathcal{X} = \{(x_1 = 0.1, x_2 = 0.7) \rightarrow (y = 1)\}$$



$z = 0.6355$ → $a = 0.6537$

$w = \begin{bmatrix} 0.82103 \\ 0.3472 \end{bmatrix}$ $b = 0.31033$

Example - Logistic Regression

Verification