UNIVERSITY OF BATH

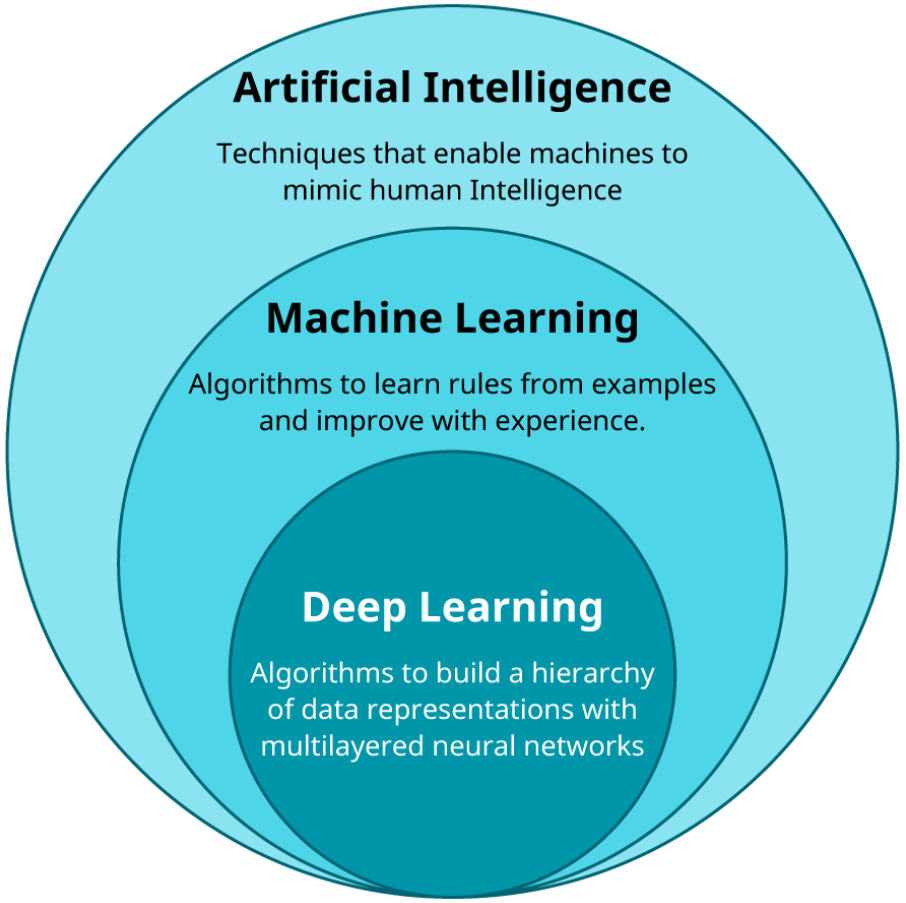
Week 02:
Multi Layer Perceptron (MLP)

CM50265 Machine Learning 2

UNIVERSITY OF BATH

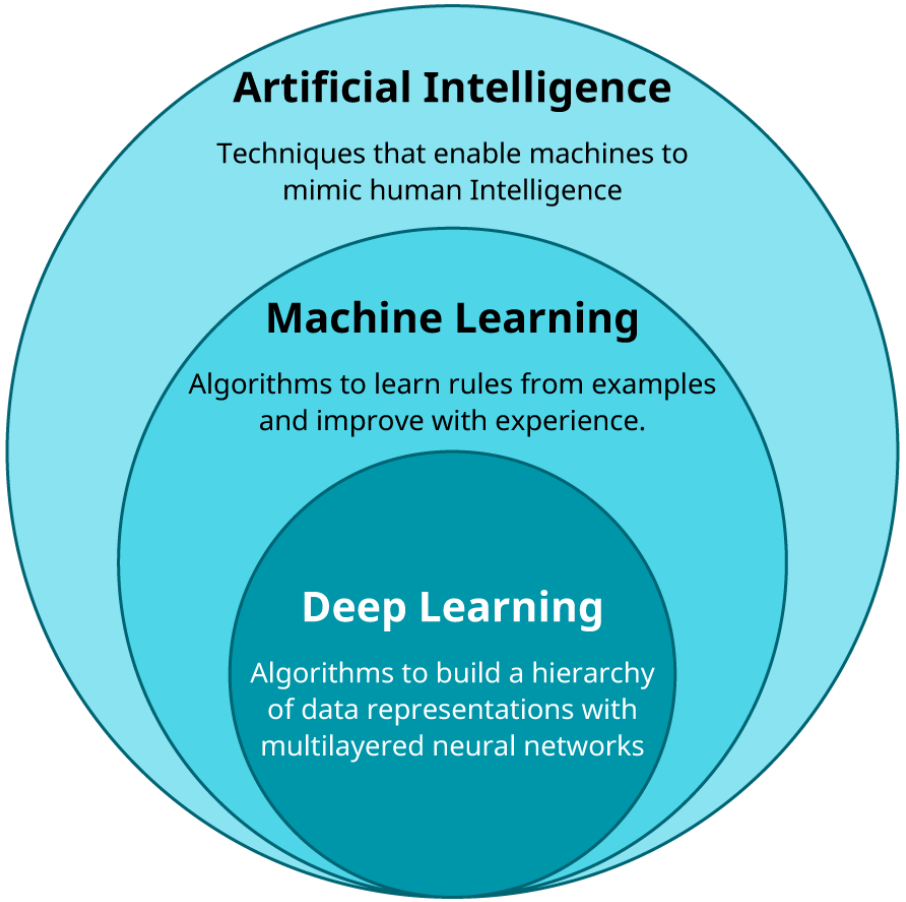
Topic 1:
Introduction to Deep Learning

AI vs Machine Learning vs Deep Learning



- From 1950s to late 1980s, the rule-based or expert system was the most popular solution to AI.
- However, it is intractable to figure out explicit rules for most real-life problems. E.g., how does a toddler start to know a giraffe is a giraffe?

AI vs Machine Learning vs Deep Learning



- **Deep learning**, a subfield of machine learning, refers to algorithms to build a hierarchy of data representations with multi-layered neural networks.

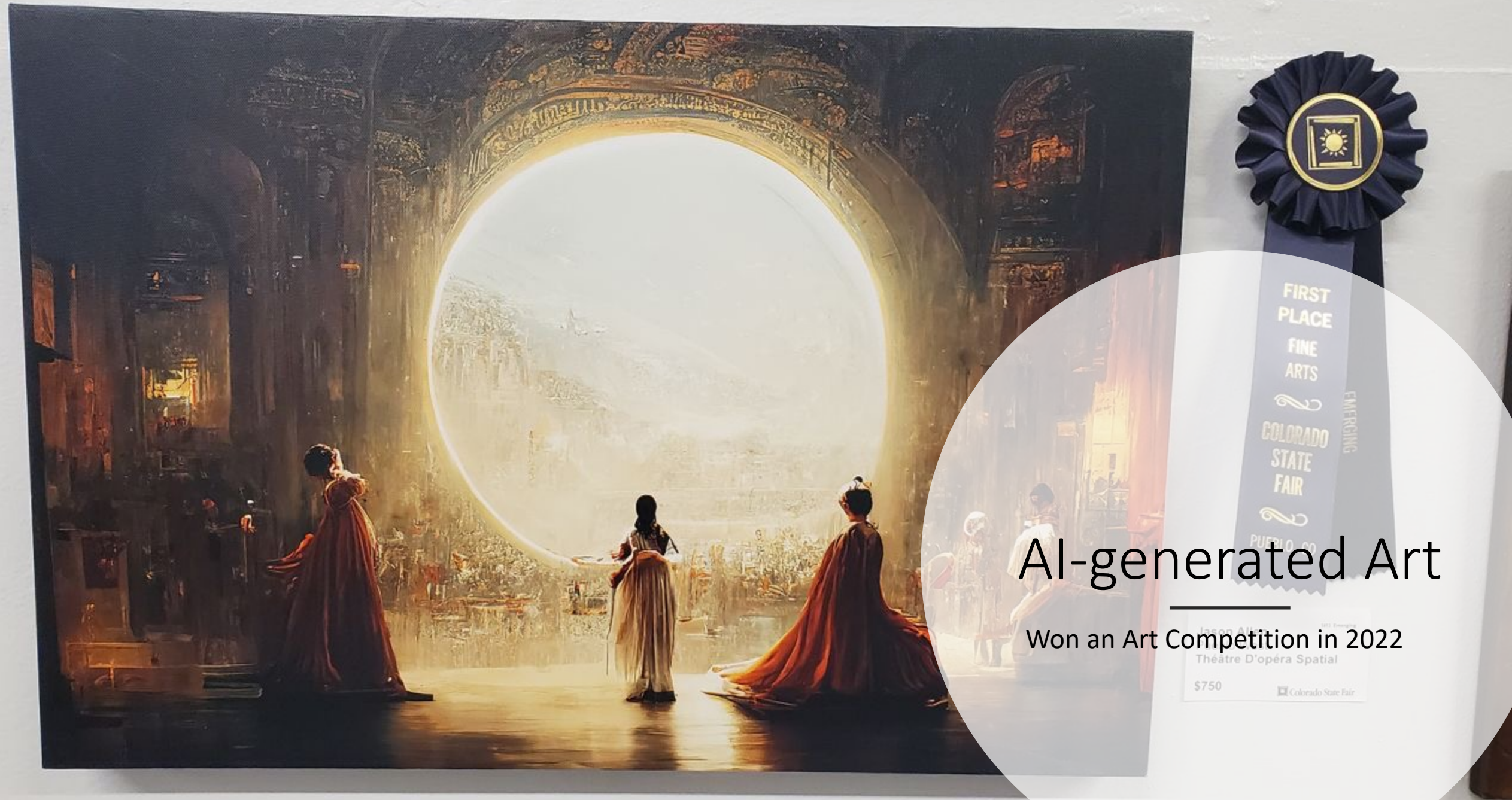
Shocking news

2016, AlphaGo Vs Lee Sedol

4 : 1

2017, AlphaGo Vs Ke Jie

3 : 0



AI-generated Art

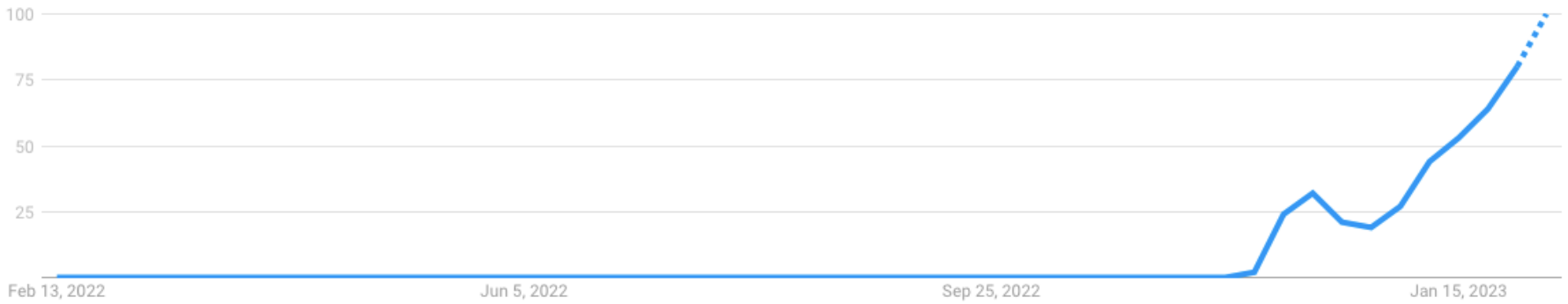
Won an Art Competition in 2022

ChatGPT

By OpenAI in Nov. 2022

How I feel when ChatGPT is at capacity and I go back to google

Img from: https://www.reddit.com/r/ChatGPT/comments/10vac7y/how_i_feel_when_chatgpt_is_at_capacity_and_i_go/



Google search interest in “ChatGPT”

ChatGPT

By OpenAI in Nov. 2022

Create a list of 3 startup ideas in enterprise B2B SaaS. The startup ideas should have a strong and compelling mission and also use AI in some way. Avoid cryptocurrency or blockchain. The startup ideas should have a cool and interesting name. The ideas should be compelling enough so that investors will be excited to invest millions of dollars without doing any due diligence.

1. "Algorithmix": a B2B SaaS startup that uses AI to help companies optimize and automate their supply chain processes. Algorithmix helps businesses make data-driven decisions, reduce costs, and improve efficiency in their supply chain operations.
2. "AI-Assist": a B2B SaaS startup that uses AI to help businesses improve their customer service operations. AI-Assist uses natural language processing and machine learning to provide real-time assistance to customer service agents, helping them resolve customer issues faster and more effectively.
3. "BrainBoost": a B2B SaaS startup that uses AI to help businesses improve the productivity and performance of their employees. BrainBoost uses machine learning to analyze employee data and provide personalized training and development recommendations, helping employees learn new skills and grow in their careers.

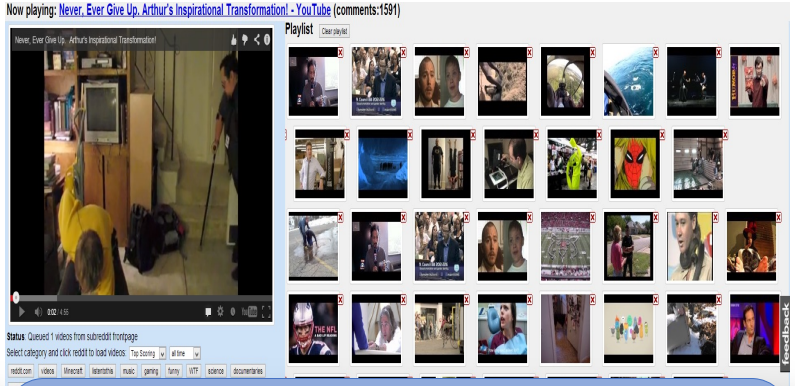
Say I go with the first idea, Algorithmix. Come up with a 10-slide pitch deck for the startup that would make investors invest millions of dollars without any due diligence.

1. Title slide: "Introducing Algorithmix: Optimizing and Automating the Supply Chain with AI"
2. Problem slide: The supply chain is a critical, but complex and often inefficient process for

From: https://sharegpt.com/c/bPdEf2F

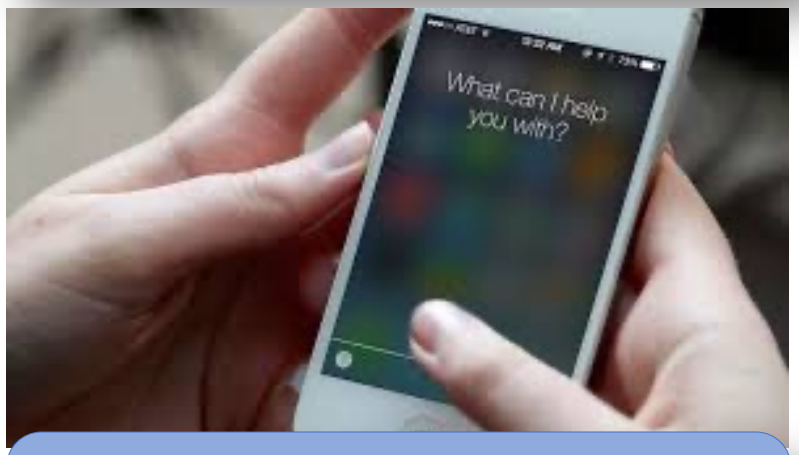
Deep learning applications

Recommendation

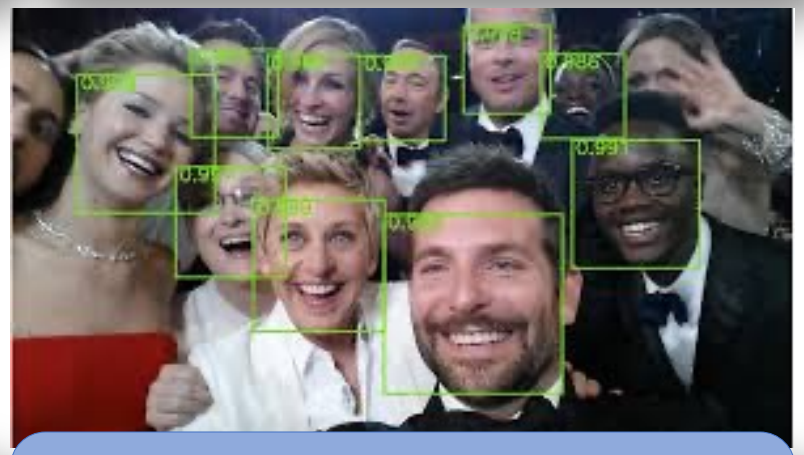


Video/image search

Self-driving car

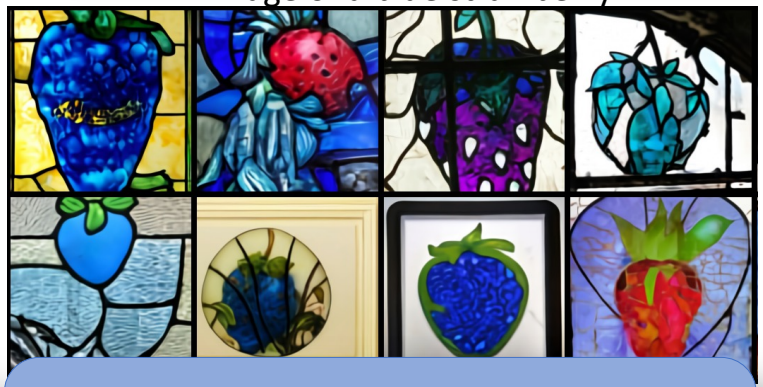


NLP



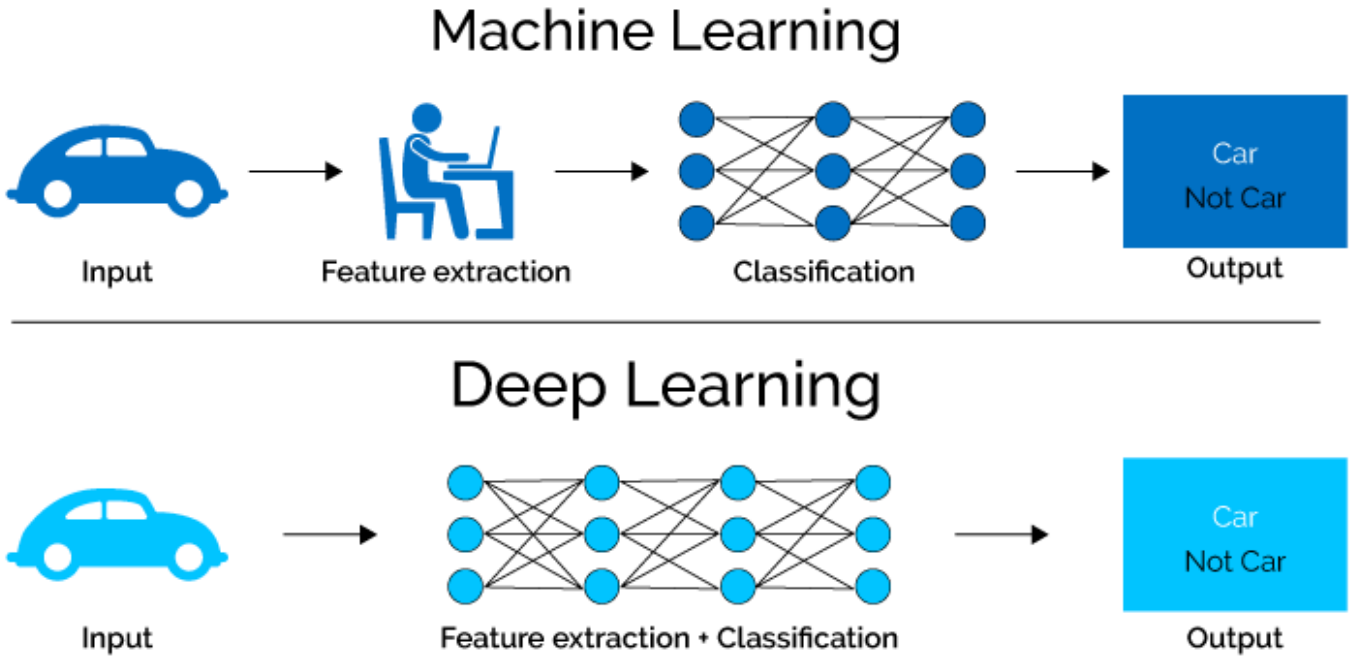
Object detection/recognition

Text: A stained glass window with an image of a blue strawberry



Text to Image

Why Deep Learning?



- Deep learning is an **end-to-end** process.
- It learns the **hierarchical features** from the raw data **automatically**.

Why Deep Learning?

- The hierarchical structure represents **different levels** of features.

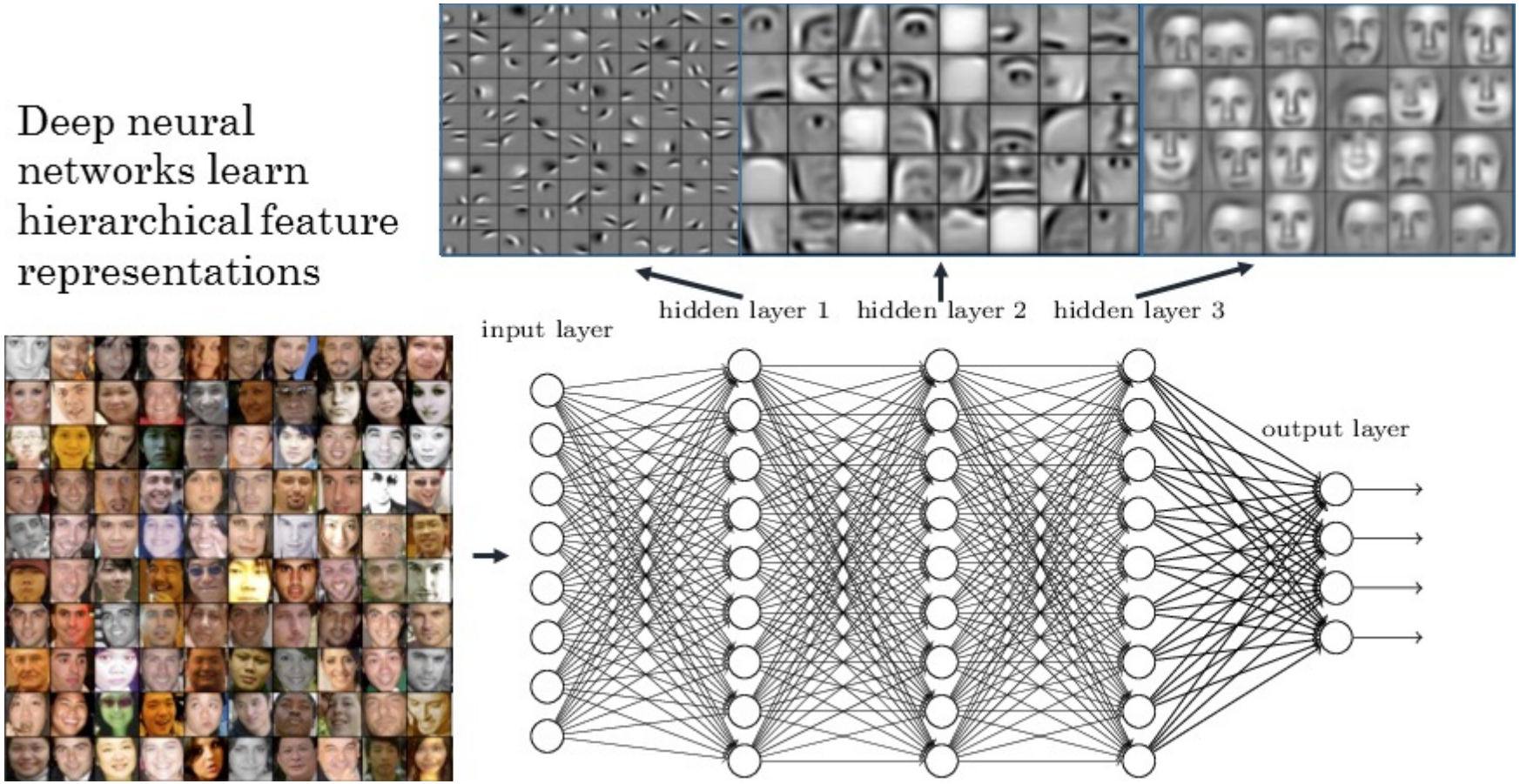


Image from: https://www.rsipvision.com/exploring-deep-learning/

- Key ingredients of DL have been around for a while!
  - Neural networks, 1940s
  - Stochastic Gradient Descent (SGD) for optimization, 1950s
  - Convolutional Neural Networks (CNN), 1989
  - Long short-term memory (LSTM), 1997
  - …

Why Now?

Why Now?

- Big Data
  - Thanks to the internet and the smartphone popularity, it is feasible to collect and store huge amount of data.

Milestone: ImageNet dataset (2010), 1.4 million images, 1000 categories

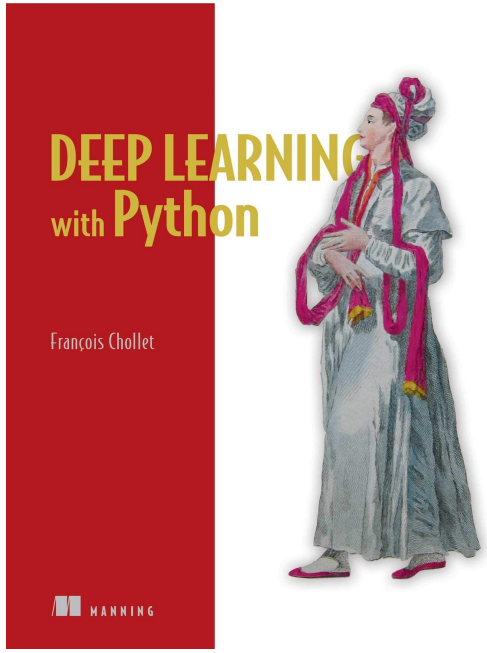
Why Now?

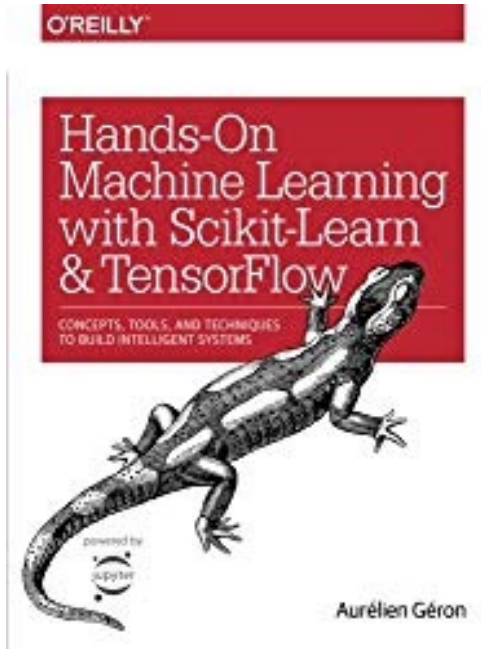
- Hardware
  - GPU, even TPU (2016)
  - Cloud services

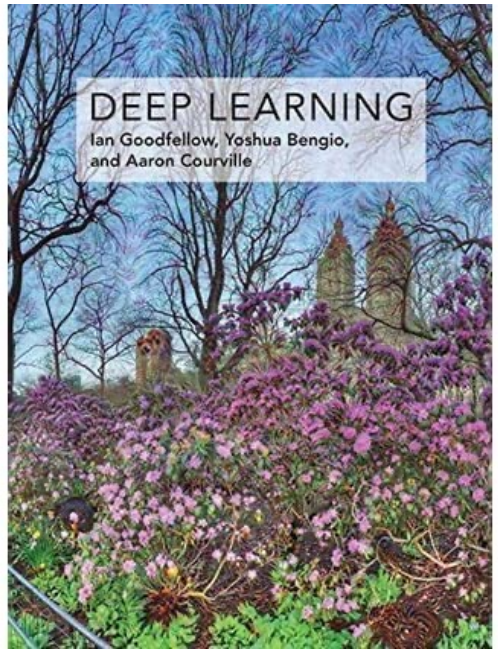
- Softwares and algorithms
  - Caffe, Theano, Tensorflow, Keras, …
  - Several simple but important algorithm improvements: activation functions, weight-initialization schemes, optimization schemes, etc.

Reading List

- Books



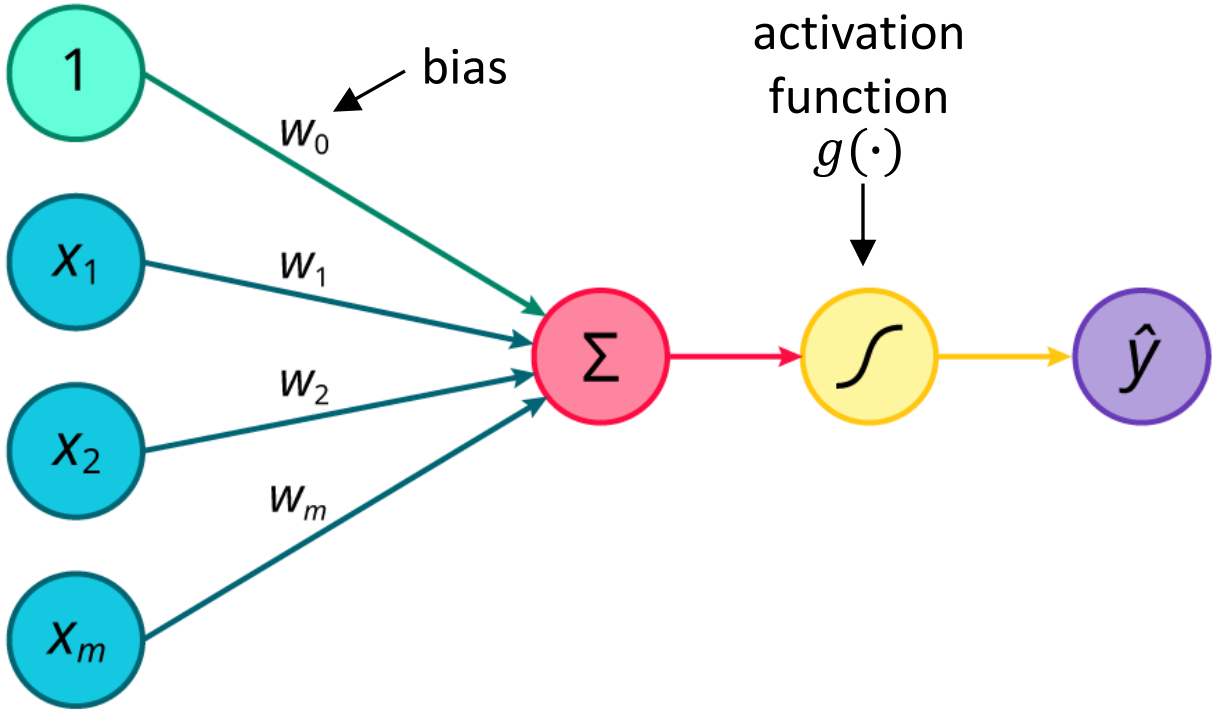




- Online tutorials
  - http://introtodeeplearning.com
  - http://cs231n.stanford.edu
  - …

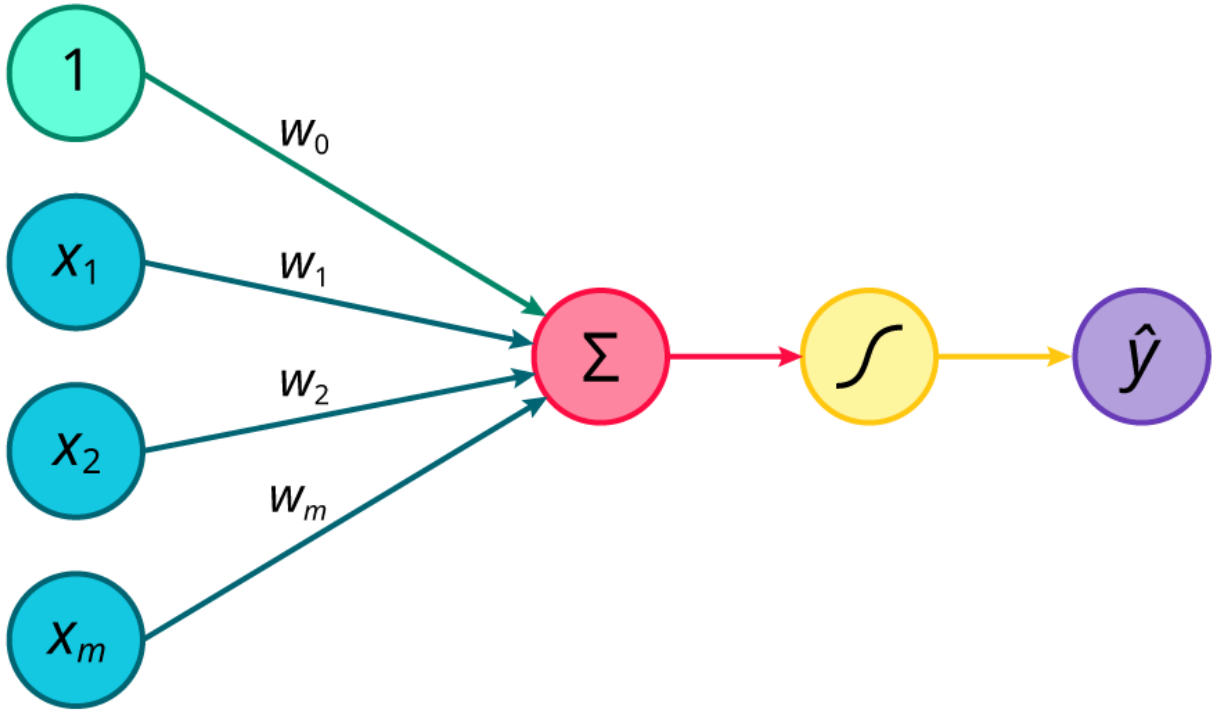
Topic 2: Perceptron

Perceptron



$$\hat{y} = g(w_0 + \sum_{i=1}^{m} x_i \cdot w_i)$$

Perceptron

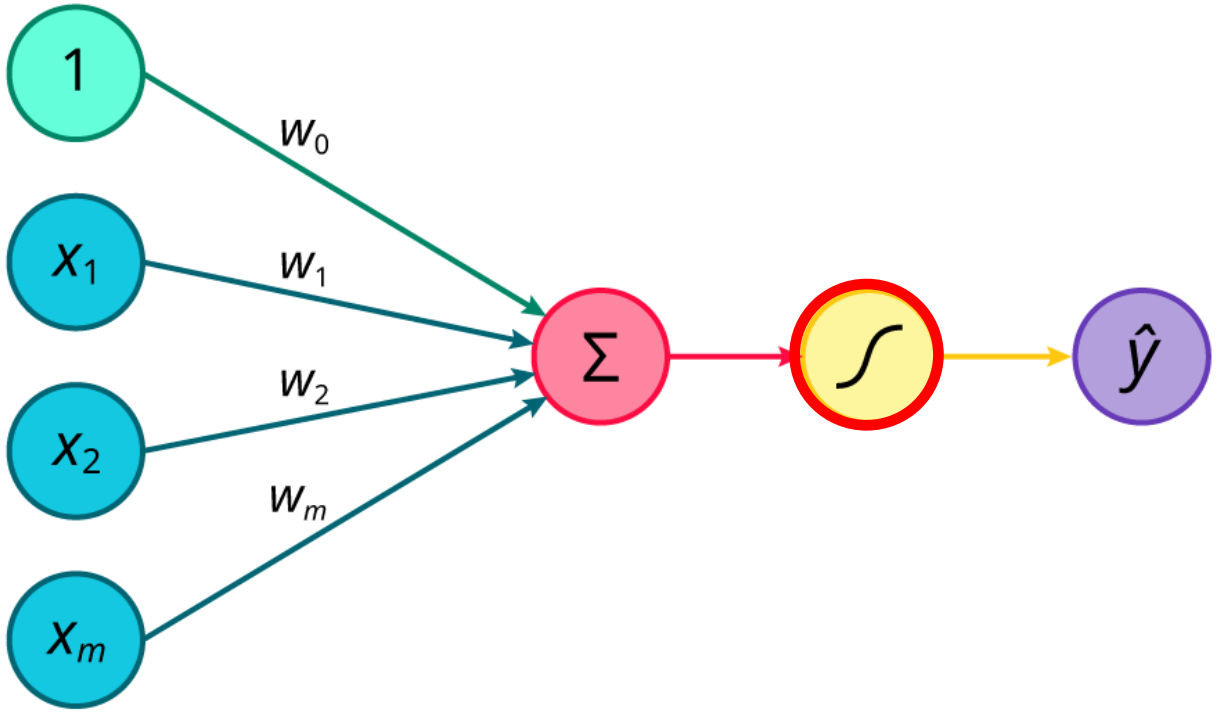


Re-write it with linear algebra:

$$\hat{y} = g(w_0 + X^T W)$$

where $X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ $W = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

Perceptron



Inputs Weights Sum Non-Linearity Output

Re-write it with linear algebra:

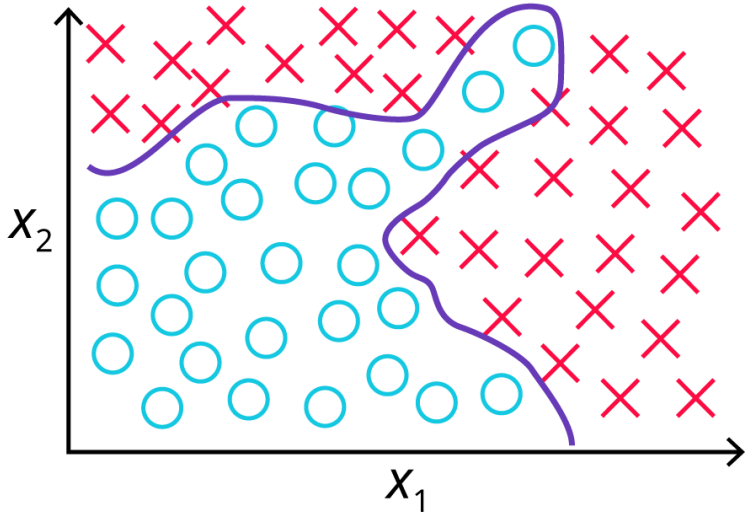
$$\hat{y} = g(w_0 + X^T W)$$

Activation function

where $X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$ $W = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$

Activation function

- It is attached to each neuron, determines whether it should be activated (or fired) or not.
- Activation functions are better to be non-linear. Why?
  - Because in real-life applications, the data are usually not separable with linear boundaries (or hyper-planes).
  - Activation function's key function is to **introduce non-linearities** into the network.

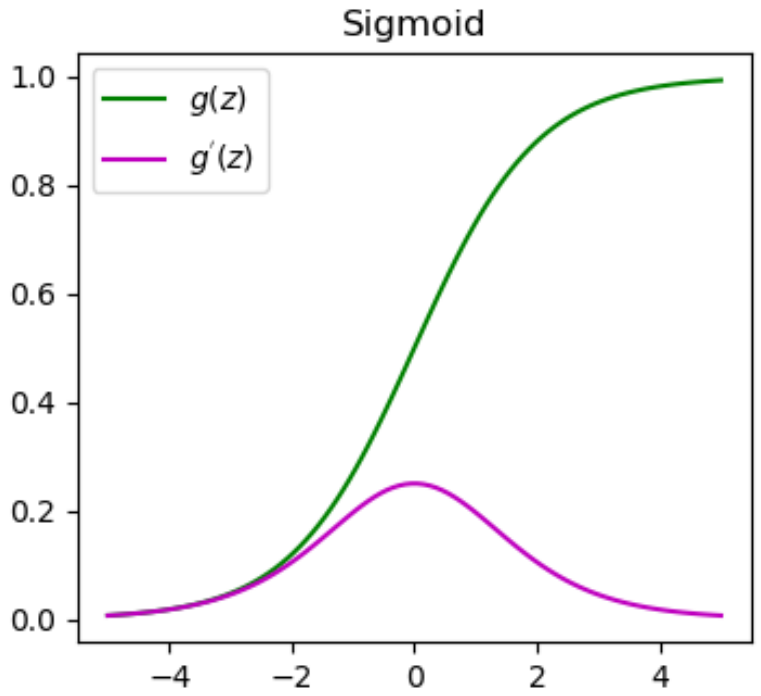


Commonly Used Activation Functions

Sigmoid/Logistic function

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

$$g'(z) = g(z)(1 - g(z))$$



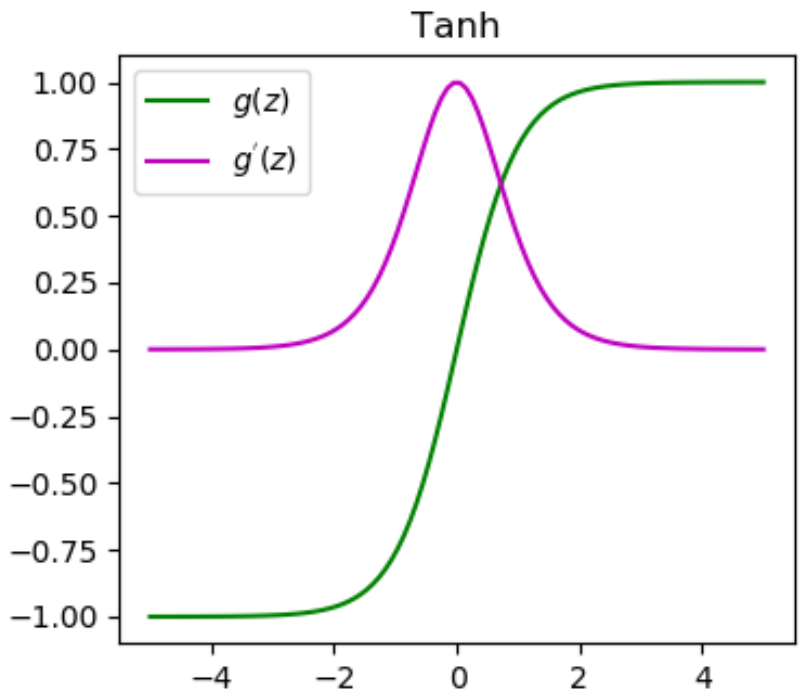
+ Smooth gradient
+ Output bound

- Outputs not zero-centered
- Computationally expensive
- Vanishing gradient problem

Hyperbolic Tangent (TanH) function

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

$$g'(z) = 1 - g^2(z)$$



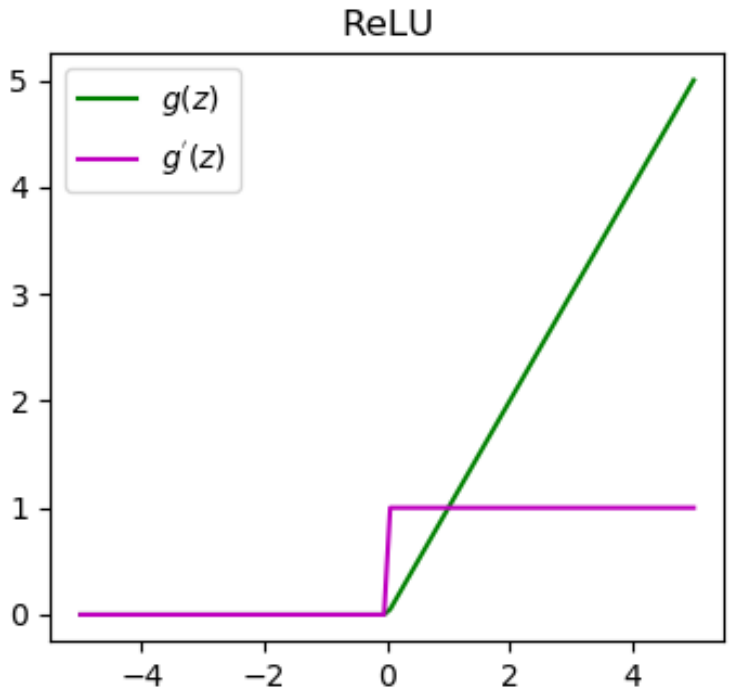
+ Smooth gradient
+ Output bound
+ Zero-centered

- Computationally expensive
- Vanishing gradient problem

ReLU (Rectified Linear Unit) function

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

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$

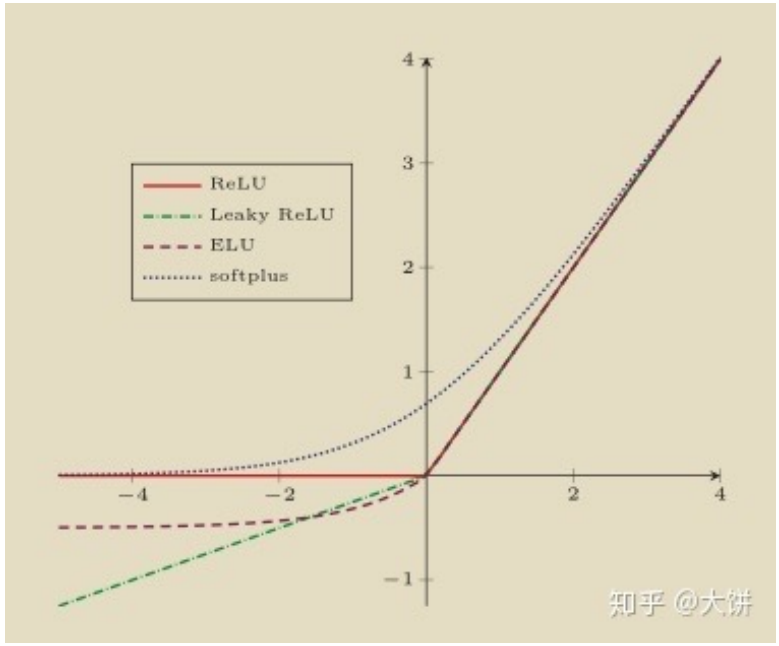


+ Allows neuron to express a *strong opinion*
+ Computationally efficient
+ Cause sparsity and efficiency

- Output not bound
- The dying problem
- Gradient discontinuous

More …

- ReLU variants:



- Softmax – for the output layer for multi-class classification

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

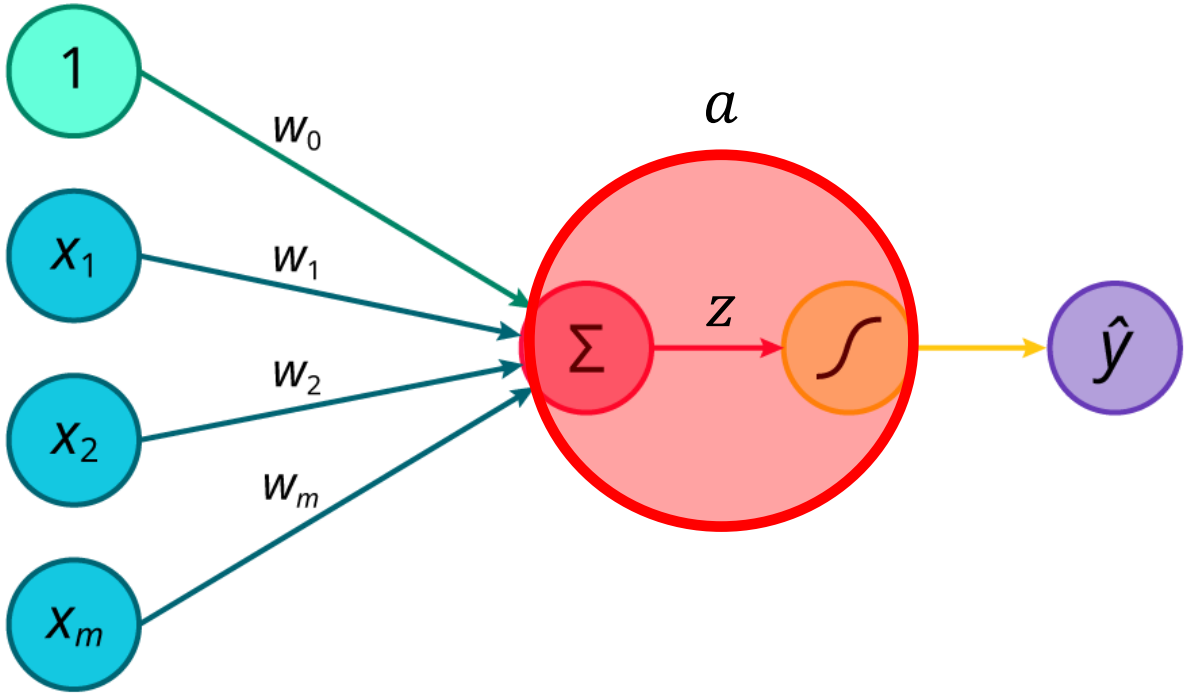
- …

Reference

- Video lecture by Alexander Amini: MIT course on deep learning, https://www.youtube.com/watch?v=njKP3FqW3Sk
- https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/
- https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/
- https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253
- https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

Topic 3:
Multi Layer Perceptron (MLP)

Perceptron



$$a = g(z)$$

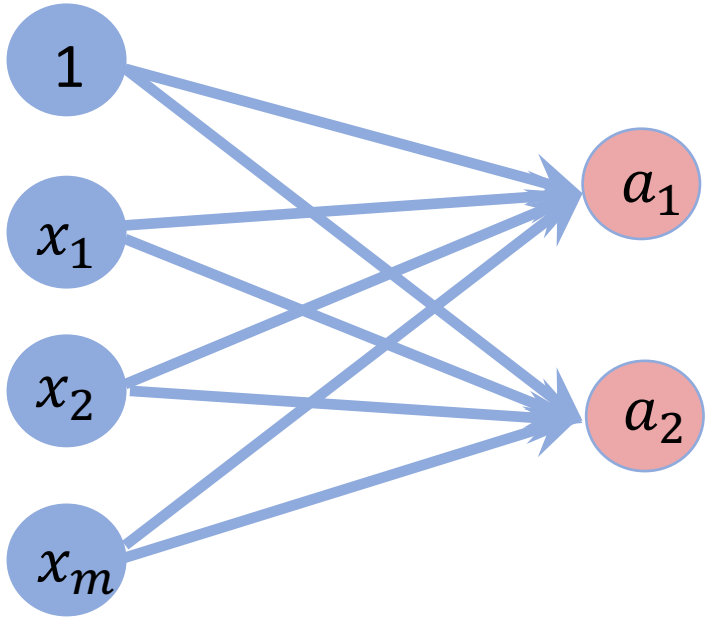
$$z = w_0 + \sum_{j=1}^{m} x_j w_j$$

How about adding more perceptrons?

$$a_1, a_2, \dots, a_i, \dots$$

$$z_1, z_2, \dots, z_i, \dots$$

More Perceptrons



$$a_i = g(z_i)$$

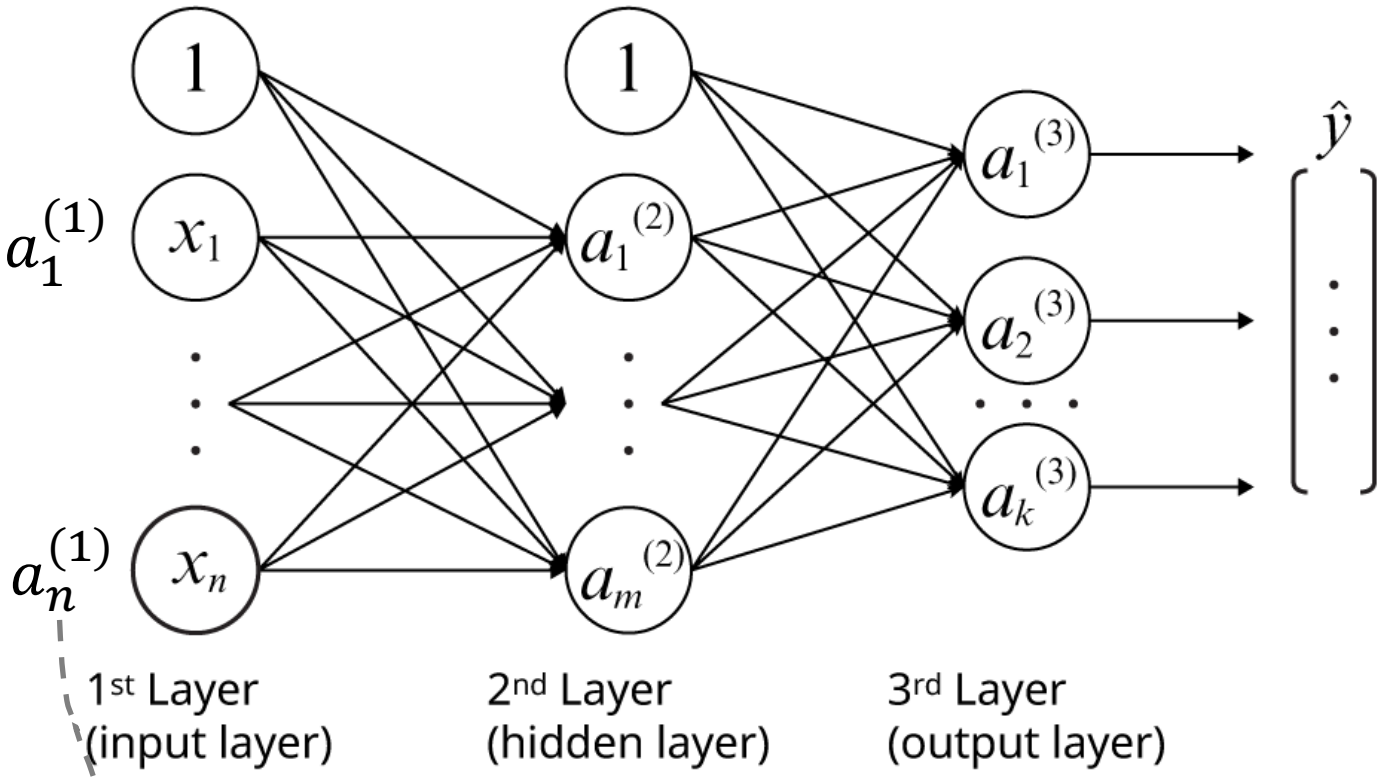
$$z_i = w_{0i} + \sum_{j=1}^{m} x_j w_{ji}$$

How about adding more layers?

$$a_1^{(l)}, a_2^{(l)}, \ldots, a_i^{(l)} \ \ldots$$

$$z_1^{(l)}, z_2^{(l)}, \ldots, z_i^{(l)} \ \ldots$$

Multi Layer Perceptron (MLP)



$$a_i^{(l)} = g(z_i^{(l)})$$

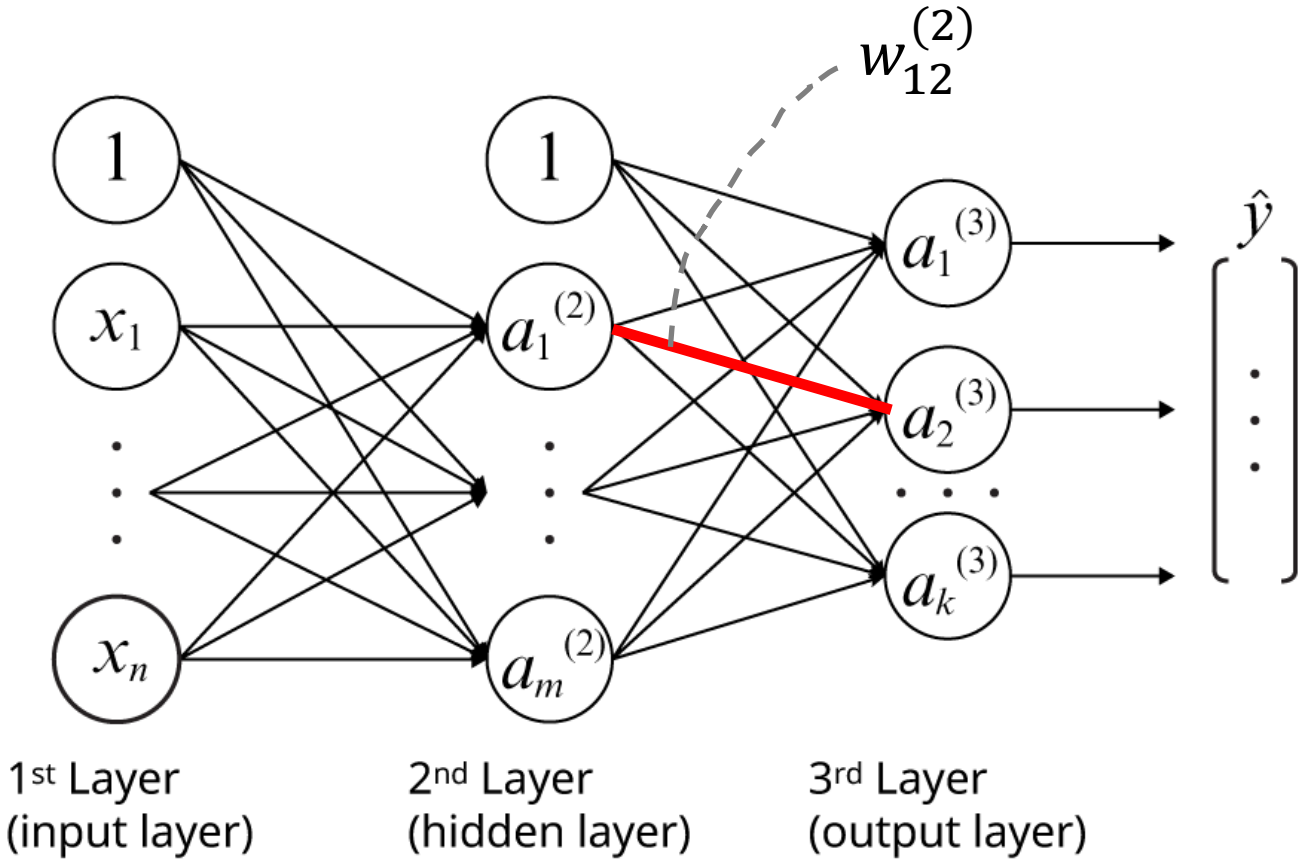
$$z_i^{(l)} = w_{0i}^{(l-1)} + \sum_{j=1}^{m} a_j^{(l-1)} w_{ji}^{(l-1)}$$

Where $l = 2, 3, \ldots$

The weight from the j-th neuron in the l-1-th layer to the i-th neuron in the l-th layer.

The input layer can be also written in the form of $a_i^{(1)}$

Multi Layer Perceptron (MLP)

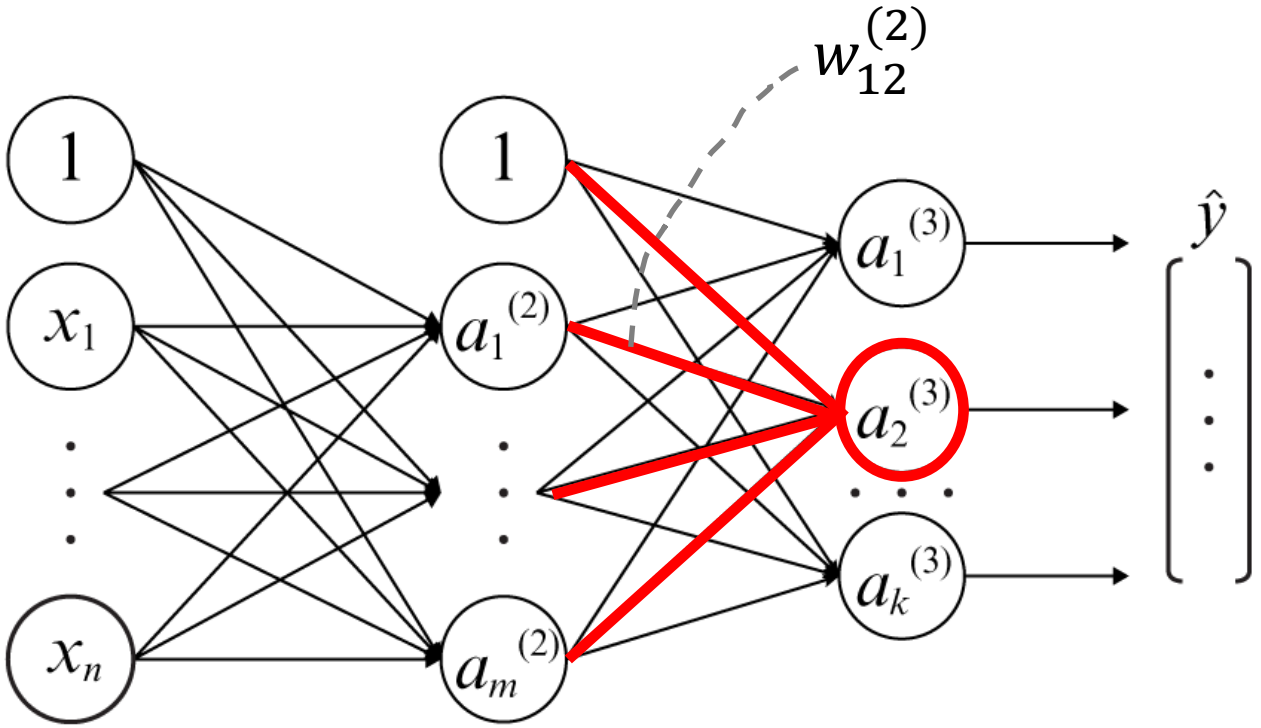


$$a_i^{(l)} = g(z_i^{(l)})$$

$$z_i^{(l)} = w_{0i}^{(l-1)} + \sum_{j=1}^{m} a_j^{(l-1)} w_{ji}^{(l-1)}$$

The weight from the j-th neuron in the l-1-th layer to the i-th neuron in the l-th layer.

Multi Layer Perceptron (MLP)



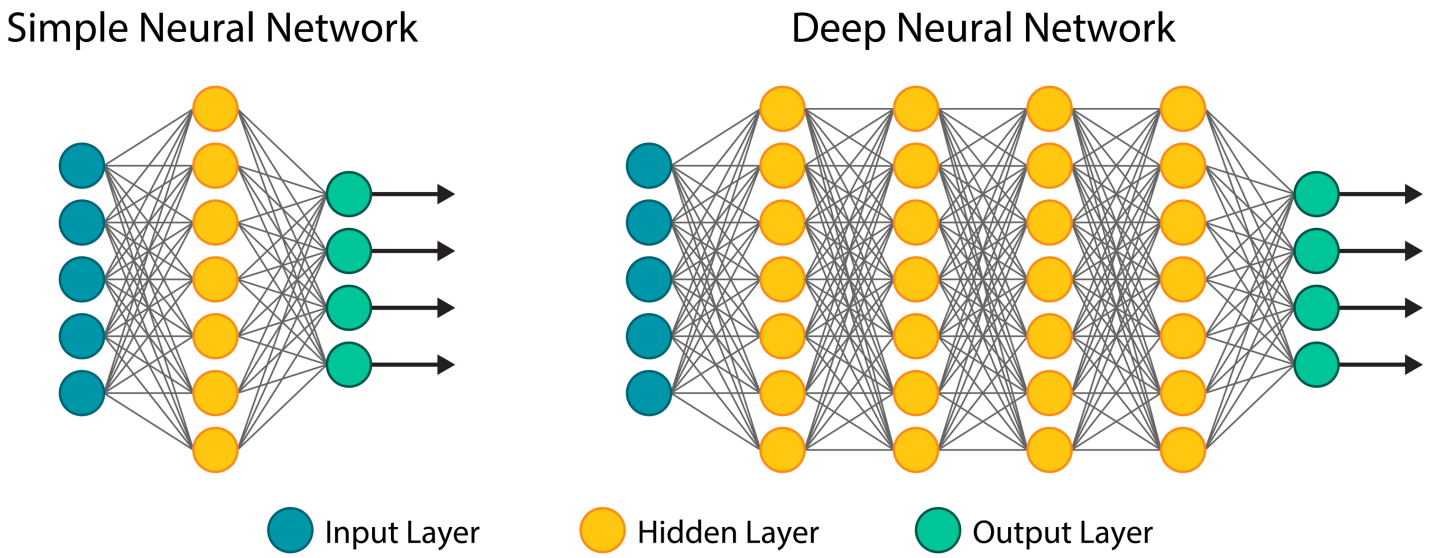
$$a_i^{(l)} = g(z_i^{(l)})$$

$$z_i^{(l)} = w_{0i}^{(l-1)} + \sum_{j=1}^{m} a_j^{(l-1)} w_{ji}^{(l-1)}$$

Let's see how to calculate $a_2^{(3)}$

$$a_2^{(3)} = g(w_{02}^{(2)} + a_1^{(2)} w_{12}^{(2)} + a_2^{(2)} w_{22}^{(2)} + \cdots + a_m^{(2)} w_{m2}^{(2)})$$

Summary for MLP



- Stacking even more layers to create a **deep neural network**.
- Multiple hidden layers are able to learn complex data with high levels of accuracy.
- MLP network consists of **fully connected layers** (or **dense layers**) - each neuron is connected to every other neuron in the adjacent layer

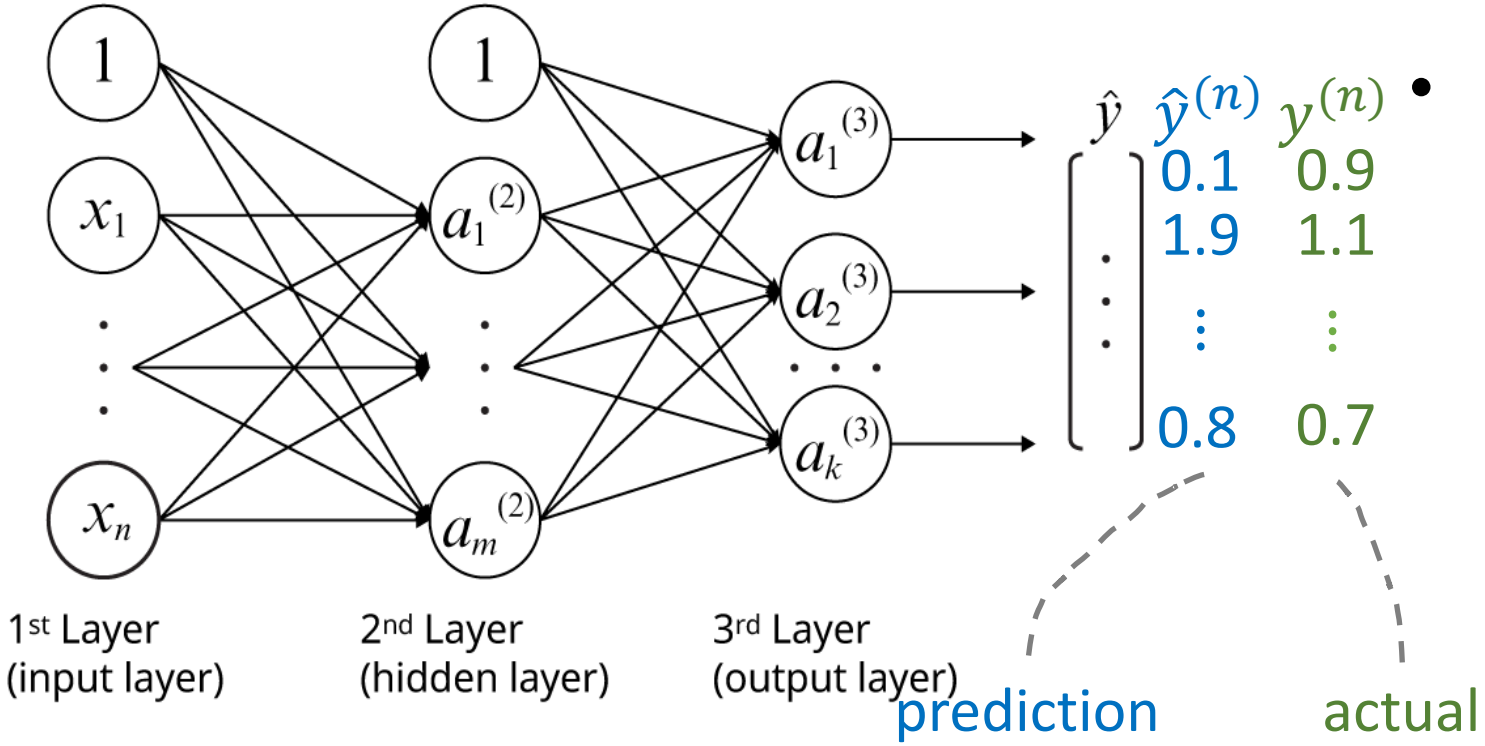
Reference

- Video lecture by Alexander Amini: MIT course on deep learning, https://www.youtube.com/watch?v=njKP3FqW3Sk
- Book: Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly. 2019.

UNIVERSITY OF BATH

Topic 4:
Loss Function

Loss function

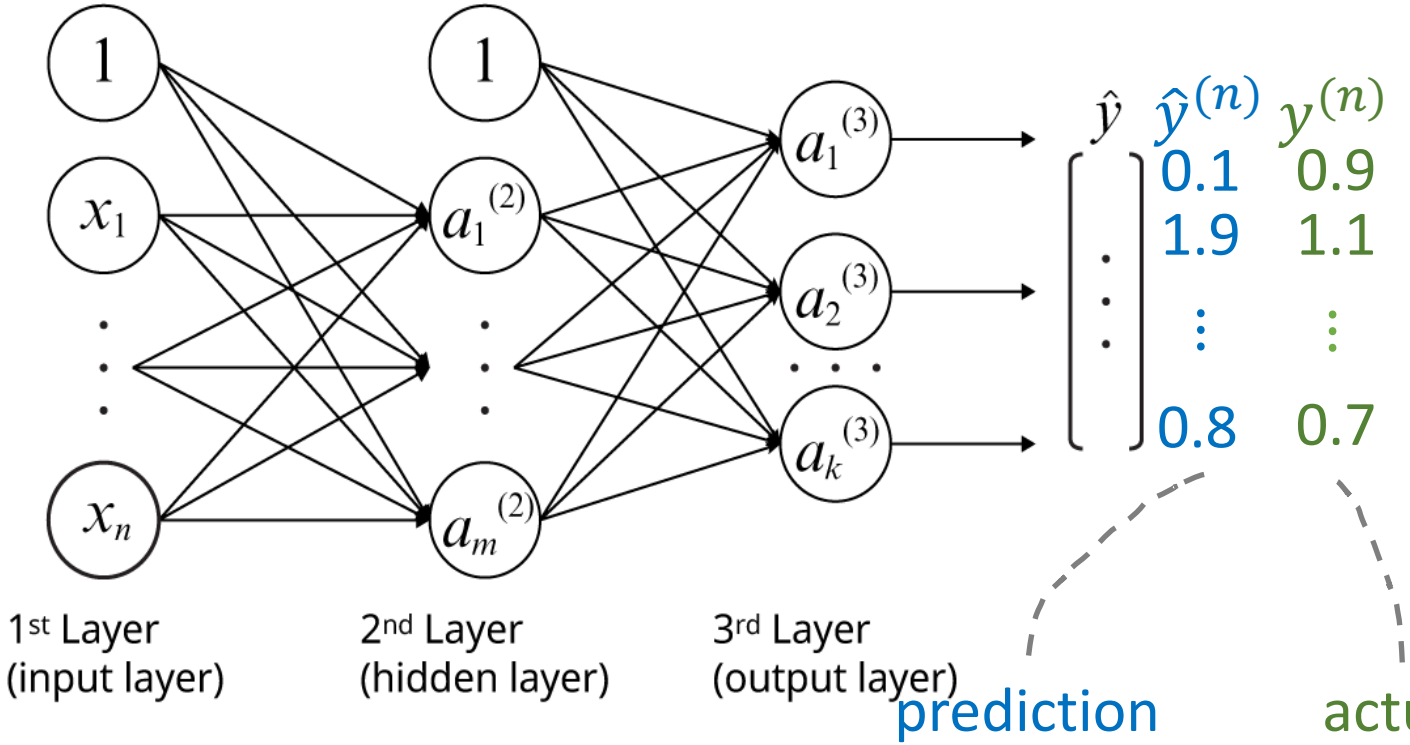


- We need to measure how good/bad the network prediction compared to the actual value(s).
- For a single sample $(x^{(n)}, y^{(n)})$, the prediction error is:

$$L(\hat{y}^{(n)}, y^{(n)})$$

$$= L(f(x^{(n)}; W), y^{(n)})$$

Loss function



- The **empirical loss** measures the average loss over the entire dataset:

$$J(W) = \frac{1}{N}\sum_{n=1}^{N} L(f(x^{(n)}; W), y^{(n)})$$

prediction actual

Binary Cross Entropy Loss

- **Binary cross-entropy loss** is used for binary classification, i.e., only two classes ($y^{(n)}$ is either 0 or 1).

$$J(W) = -\frac{1}{N}\sum_{n=1}^{N}(y^{(n)}\log f\left(x^{(n)};W\right) + (1-y^{(n)})\log(1-f\left(x^{(n)};W\right)))$$

actual
{0, 1}

prediction
[0,1]

Multi-class Cross Entropy Loss

- **Multi-class cross-entropy loss** is used for multi-class classification, i.e., each example belong to ONE of C classes, $C>2$

The network have C outputs, gathered into a vector

$f(x^{(n)}; W)$

$$\begin{bmatrix} 0.02 \\ \vdots \\ 0.81 \\ \vdots \\ 0.09 \end{bmatrix}$$

A **one-hot** vector: only the c-th element is 1, means this example belongs to class c.

$y^{(n)}$

$$\begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

$$J(W) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_c^{(n)} \log f_c\left(x^{(n)}; W\right)$$

actual
{0, 1}

prediction
[0,1]

Mean Squared Error (MSE) Loss

- **Mean Squared Error (MSE)** loss is the default loss to use for regression problems which output continuous real numbers.

$$J(W) = \frac{1}{N}\sum_{n=1}^{N}(y^{(n)} - f(x^{(n)}; W))^2$$

actual

prediction

Reference

- Video lecture by Alexander Amini: MIT course on deep learning, https://www.youtube.com/watch?v=njKP3FqW3Sk
- https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

UNIVERSITY OF BATH

Topic 5:
Training Neural Networks

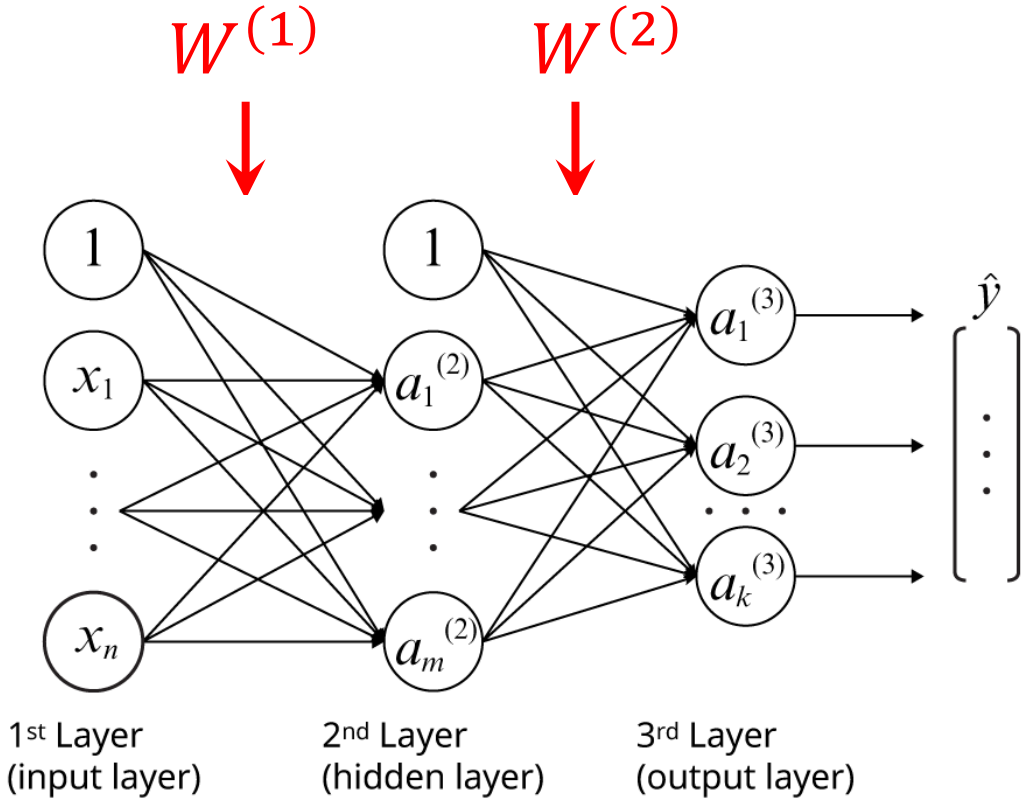
Optimization

- *Objective*: find the weights W^* produce the lowest loss:

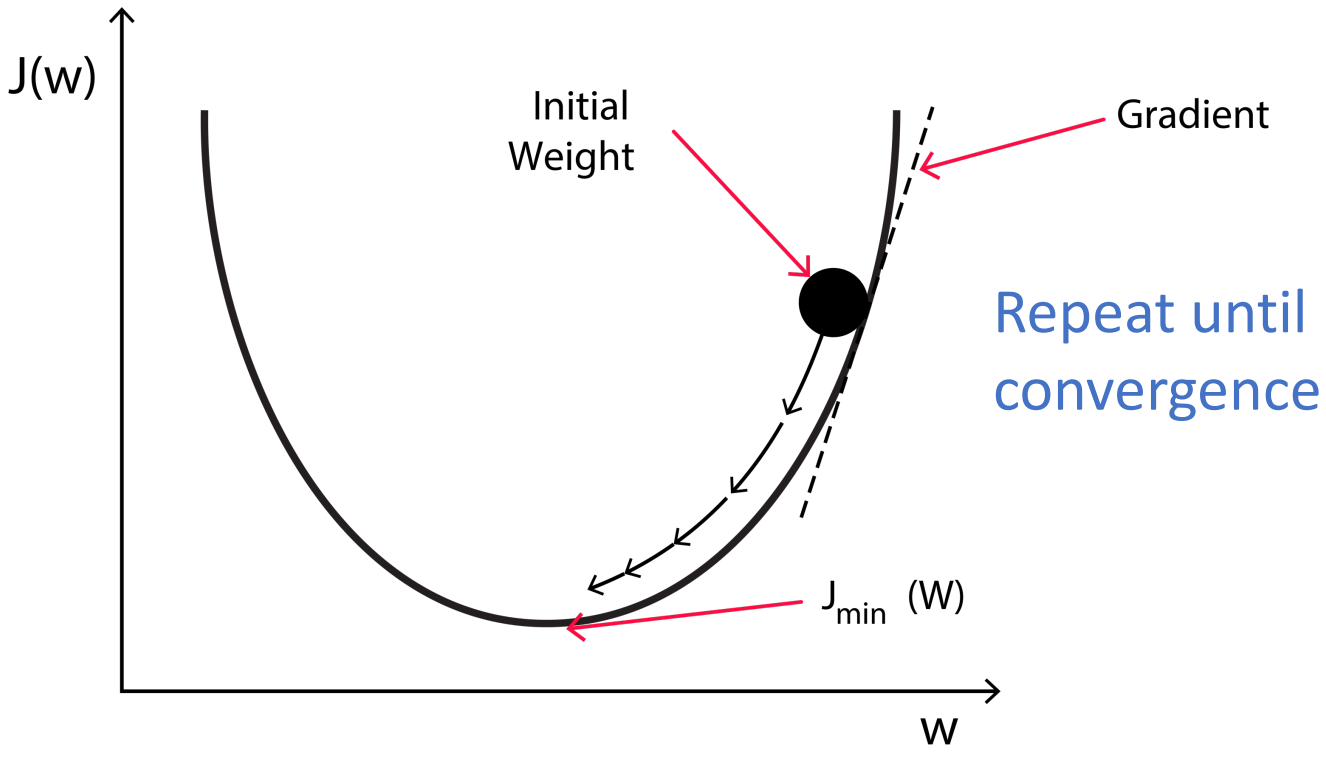
$$W^* = \underset{W}{\operatorname{argmin}} J(W)$$

$$= \underset{W}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} L(f(x^{(n)}; W), y^{(n)})$$

Where $W = \{W^{(1)}, W^{(2)}, \ldots W^{(L-1)}\}$



Gradient Descent



Repeat until convergence

Randomly pick one initial weight W .

Calculate its gradient $\frac{\partial J(W)}{\partial W}$.

Take small steps in its opposite direction $-\eta \frac{\partial J(W)}{\partial W}$.

Update the weight

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$$

Gradient Descent

learning rate

$$W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$$

For each weight:

$$w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} - \eta \frac{\partial J(W)}{\partial w_{ji}^{(l)}}$$

Q: How to compute the gradient with respect to each weight?

A: **Backpropagation**

Chain Rule

$$[f(g(x))]' = f'\big(g(x)\big) \cdot g'(x)$$

OR:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

- Example: $y = (5x + 1)^2$

$$y = u^2 \quad u = 5x + 1$$

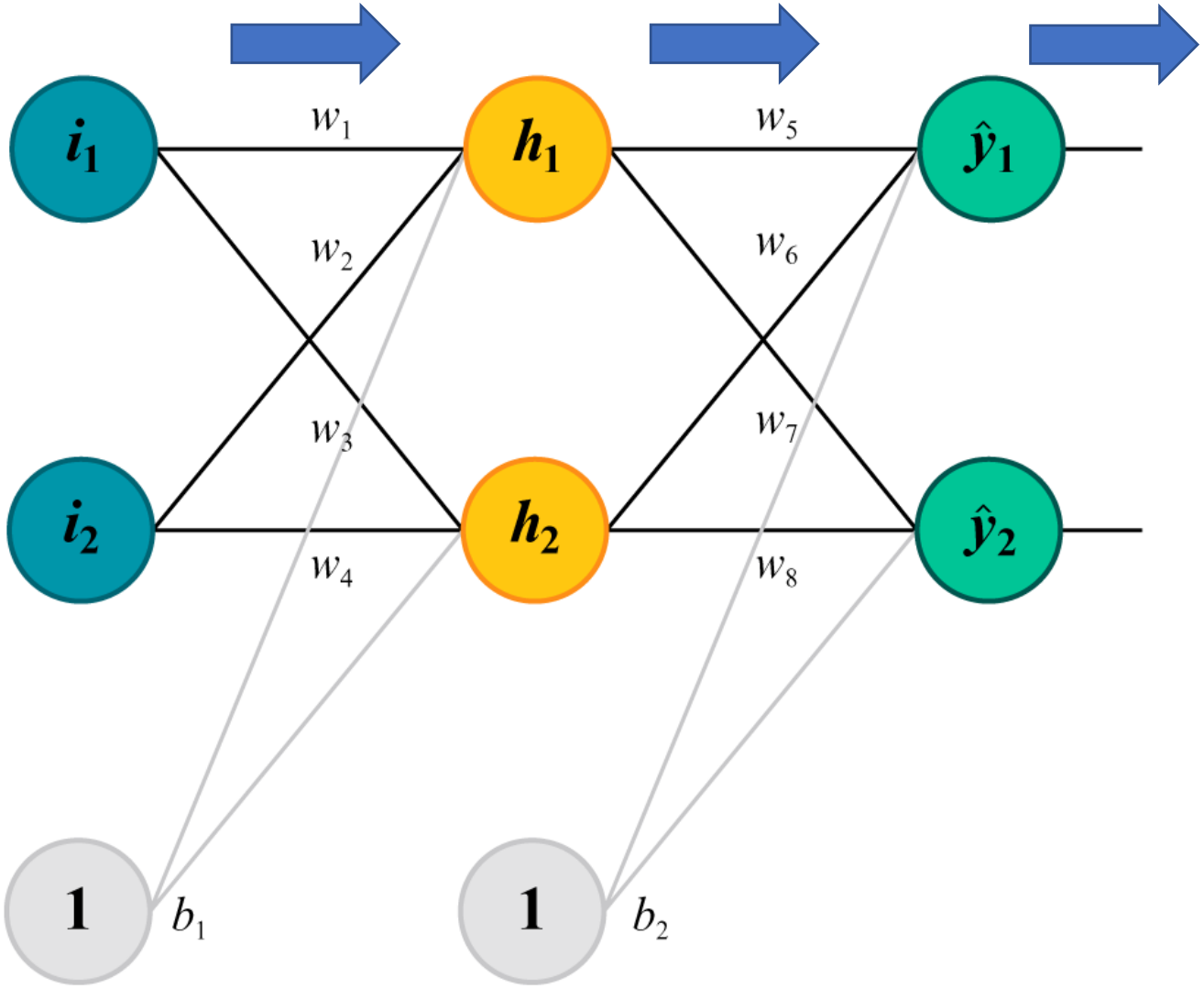
$$y' = \frac{dy}{du} \cdot \frac{du}{dx} = 2u \cdot 5 = 10(5x + 1)$$

Reference

- Video lecture by Alexander Amini: MIT course on deep learning, https://www.youtube.com/watch?v=njKP3FqW3Sk
- Book: Aurelien Geron. Hands-On Machine Learning with Scikit-Learn and TensorFlow. O'Reilly. 2019.

Topic 6: Backpropagation

Backpropagation



Forward Pass

- The 2nd-layer:

$$h_1 = g(b_1 + i_1 \cdot w_1 + i_2 \cdot w_2)$$
$$h_2 = g(b_1 + i_1 \cdot w_3 + i_2 \cdot w_4)$$

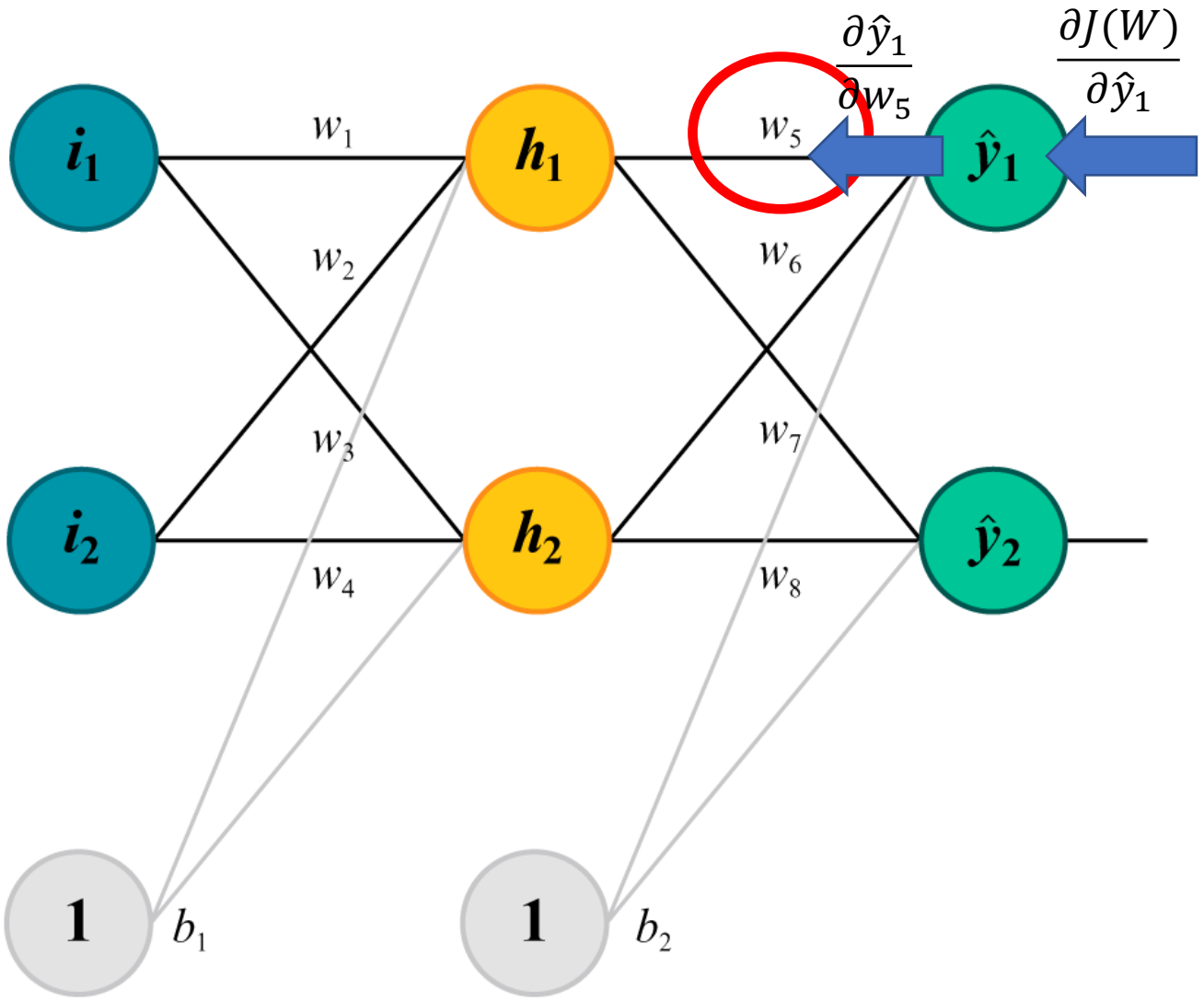
- The 3rd-layer:

$$\hat{y}_1 = g(b_2 + h_1 \cdot w_5 + h_2 \cdot w_6)$$
$$\hat{y}_2 = g(b_2 + h_1 \cdot w_7 + h_2 \cdot w_8)$$

- The loss (MSE):

$$J(W) = \frac{1}{2}[(\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2]$$

Backpropagation



Backward Pass

$$\frac{\partial J(W)}{\partial w_5}$$

- Apply chain rule:

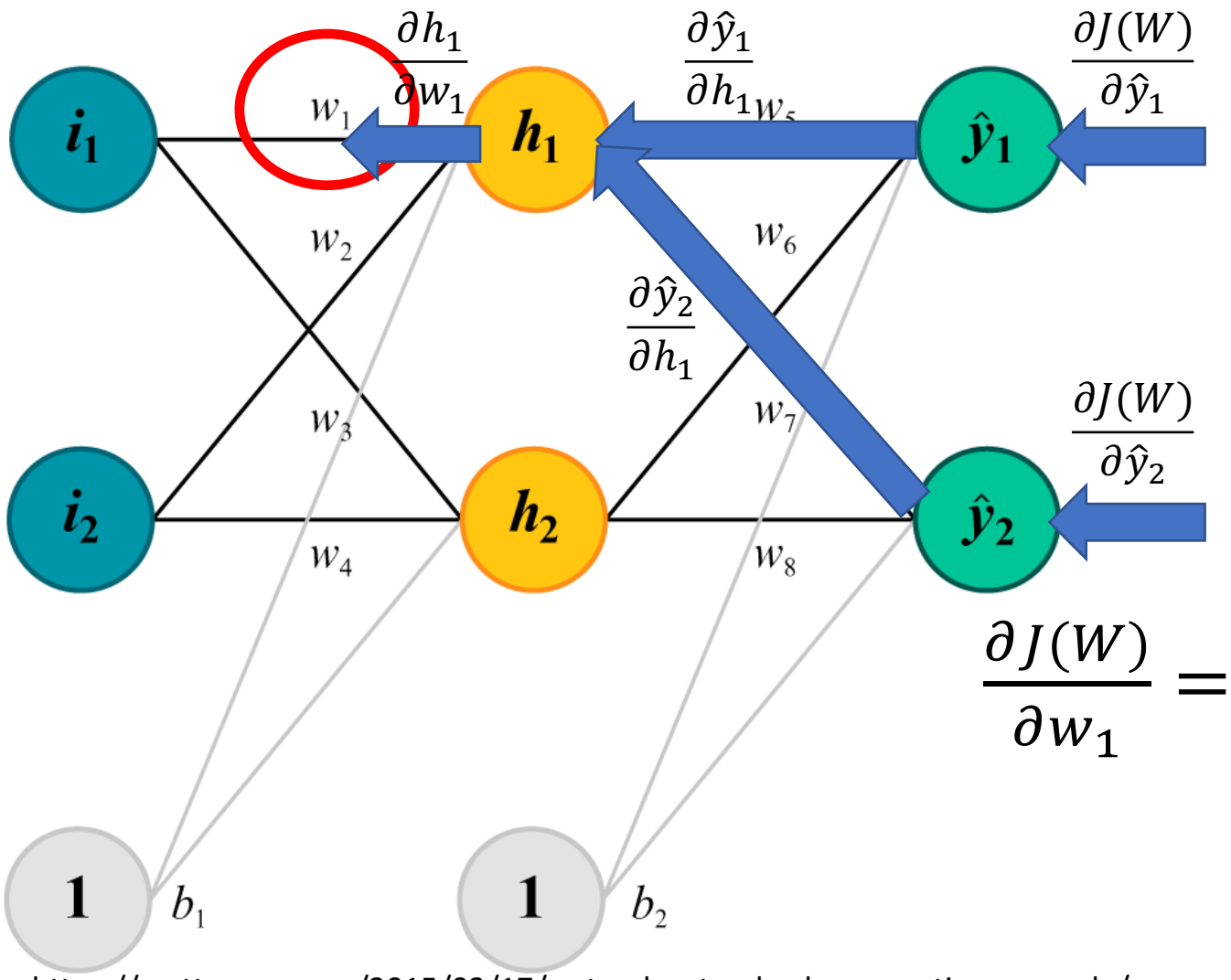
$$\frac{\partial J(W)}{\partial w_5} = \frac{\partial J(W)}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial w_5}$$

- Update the weight:

$$w_5 \leftarrow w_5 - \eta \frac{\partial J(W)}{\partial w_5}$$

- The same to w_6, w_7, w_8, b_2

Backpropagation



From: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Backward Pass

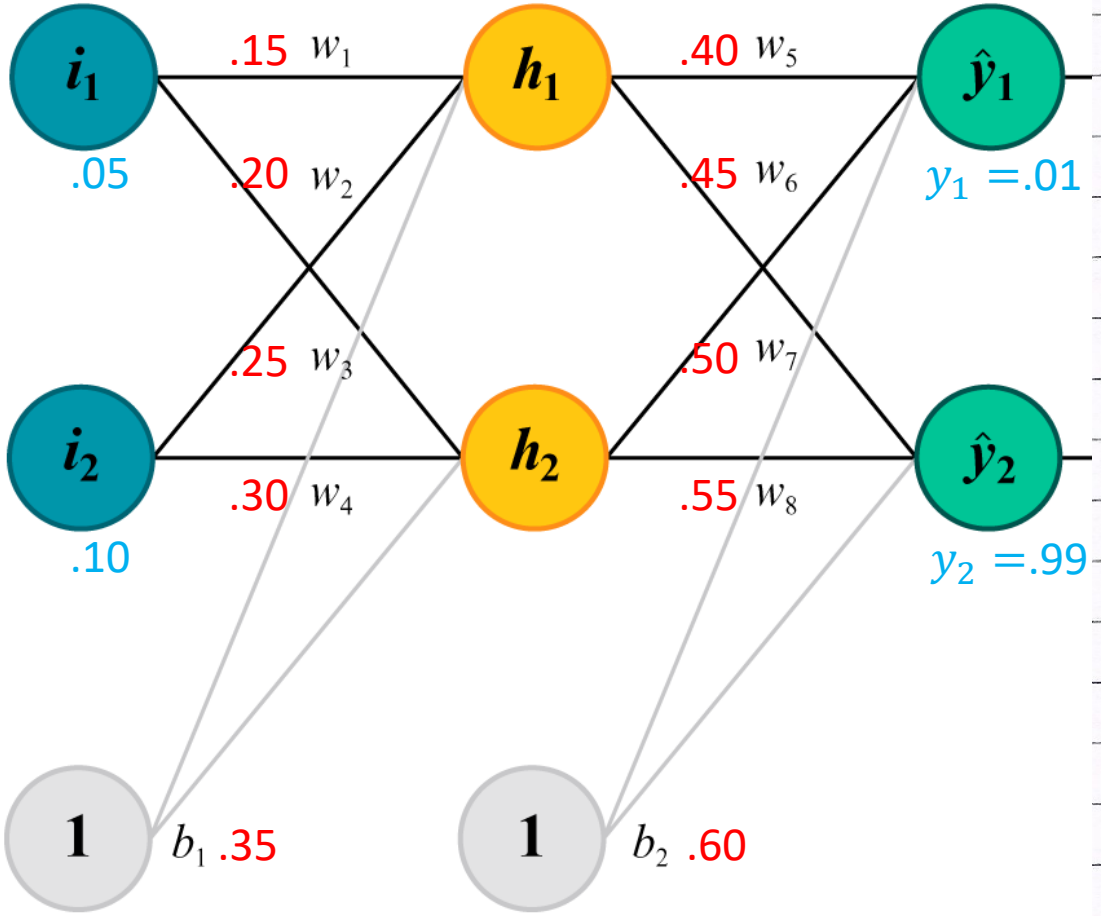
$$\frac{\partial J(W)}{\partial w_1}$$

- Apply chain rule:

$$\frac{\partial J(W)}{\partial w_1} = \left(\frac{\partial J(W)}{\partial \hat{y}_1} \cdot \frac{\partial \hat{y}_1}{\partial h_1} + \frac{\partial J(W)}{\partial \hat{y}_2} \cdot \frac{\partial \hat{y}_2}{\partial h_1}\right) \cdot \frac{\partial h_1}{\partial w_1}$$

- Update the weight w_1, w_2, w_3, w_4, b_1

Exercise: Write a simple Python code to do the backpropagation for the following example.



Given

- Input: $x = \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} = \begin{pmatrix} 0.05 \\ 0.10 \end{pmatrix}$
- Actual output: $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0.01 \\ 0.99 \end{pmatrix}$
- Weights, biases: $w_1, \dots, w_8, b_1, b_2 = \dots$

Suppose all activation functions are sigmoid function:

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

$$g'(z) = g(z)(1-g(z))$$

Calculate:

Forward Pass

- All hidden neurons: $h_1 = ?$, $h_2 = ?$
- All predictions: $\hat{y}_1 = ?$, $\hat{y}_2 = ?$
- The loss on this single example (x, y) using MSE Loss:

$$J(W) = ?$$

Backward Pass

- All partial derivatives:

$$\frac{\partial J(W)}{\partial w_1} = ?, \dots, \frac{\partial J(W)}{\partial w_8} = ?, \frac{\partial J(W)}{\partial b_1} = ?, \frac{\partial J(W)}{\partial b_2} = ?$$

- Update weights, biases (with $\eta = 0.5$):

$$w_1^{new} = ?, \dots, w_8^{new} = ?, b_1^{new} = ?, b_2^{new} = ?$$

Reference

- Blog by Matt Mazur: A Step by Step Backpropagation Example. https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
- Blog by Michael Nielsen: Chapter 2 How the backpropagation algorithm works. http://neuralnetworksanddeeplearning.com/chap2.html