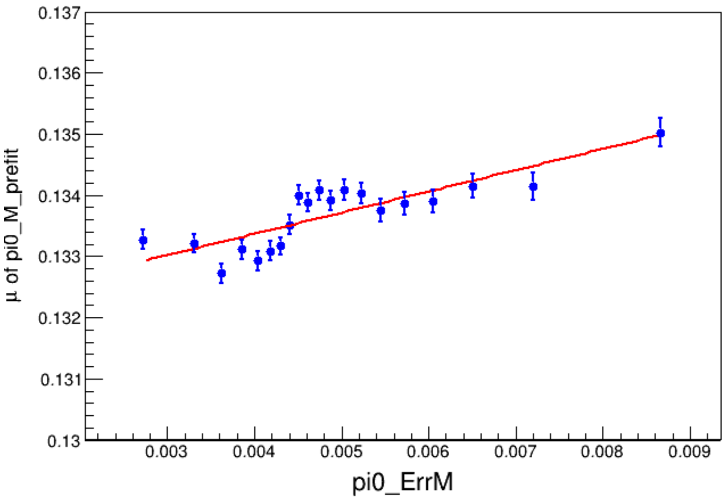
Deep Learning

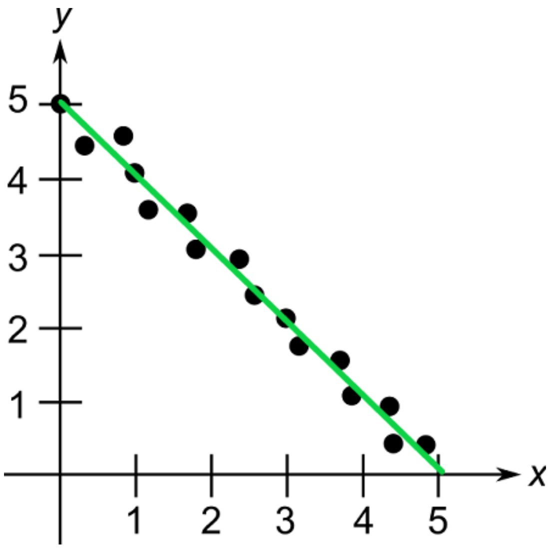
Prepared by Dr. Saw Shier Nee

Outline

- Introduction
  - Deep learning basics
  - Training a neural network
- Hands-on coding
  - Google Colab Introduction
  - Classification using Breast Cancer Data
  - Classification using MNIST
- Summary
- Quiz

Linear Regression



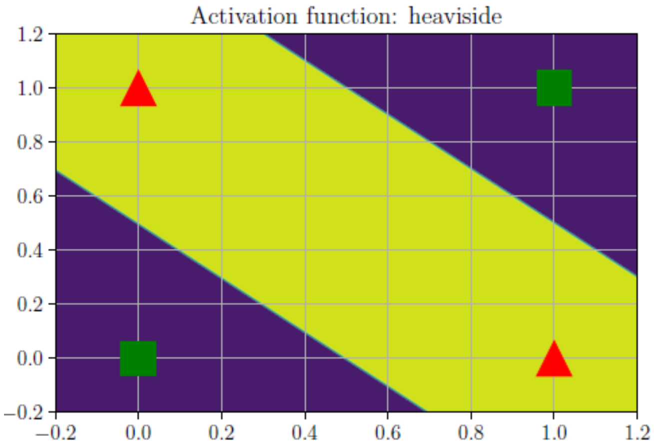


$$f(x) = w_0 x + b$$

Non-linear problem

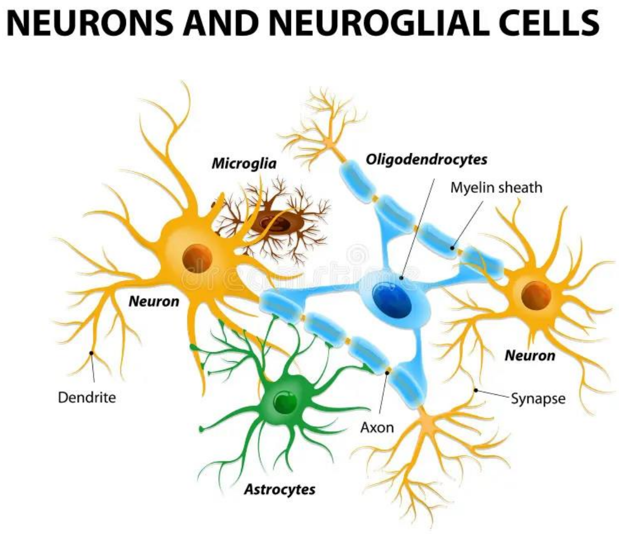
XOR Problem – not linearly separable.

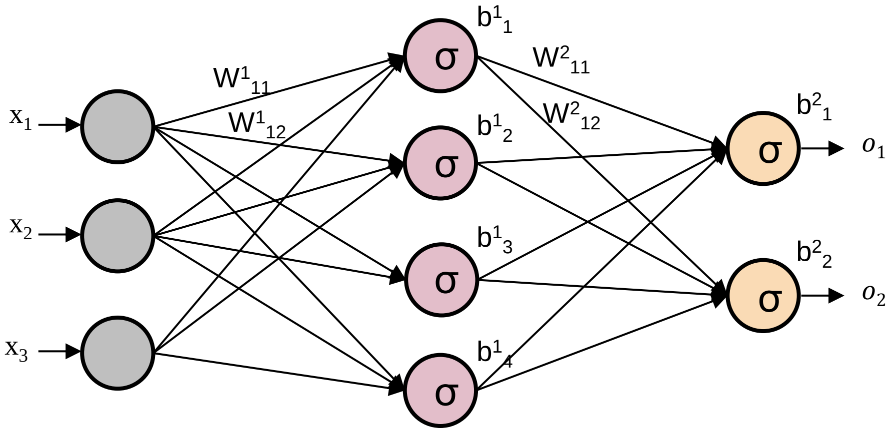
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



Solve this by *stacking linear regression* on top of each other, we call this multilayer perceptron.

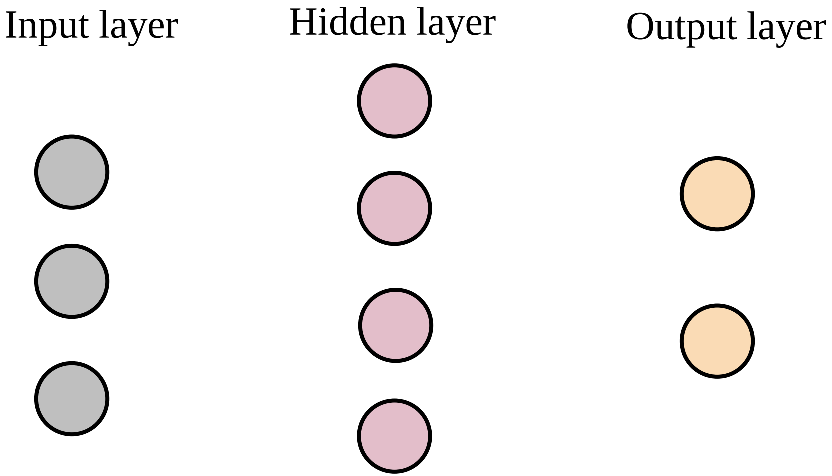
Deep Learning - Connection to Biology





Deep Learning Basics – Multilayer Perceptron

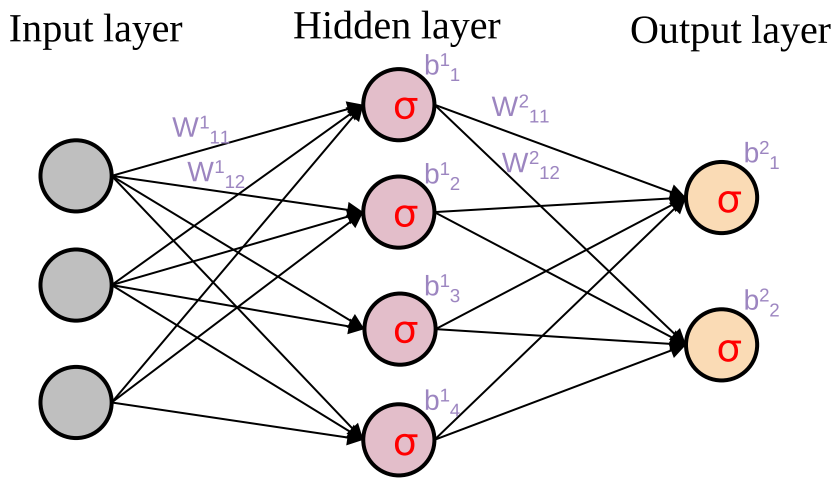
- Made up of <u>multiple layers of nodes</u>



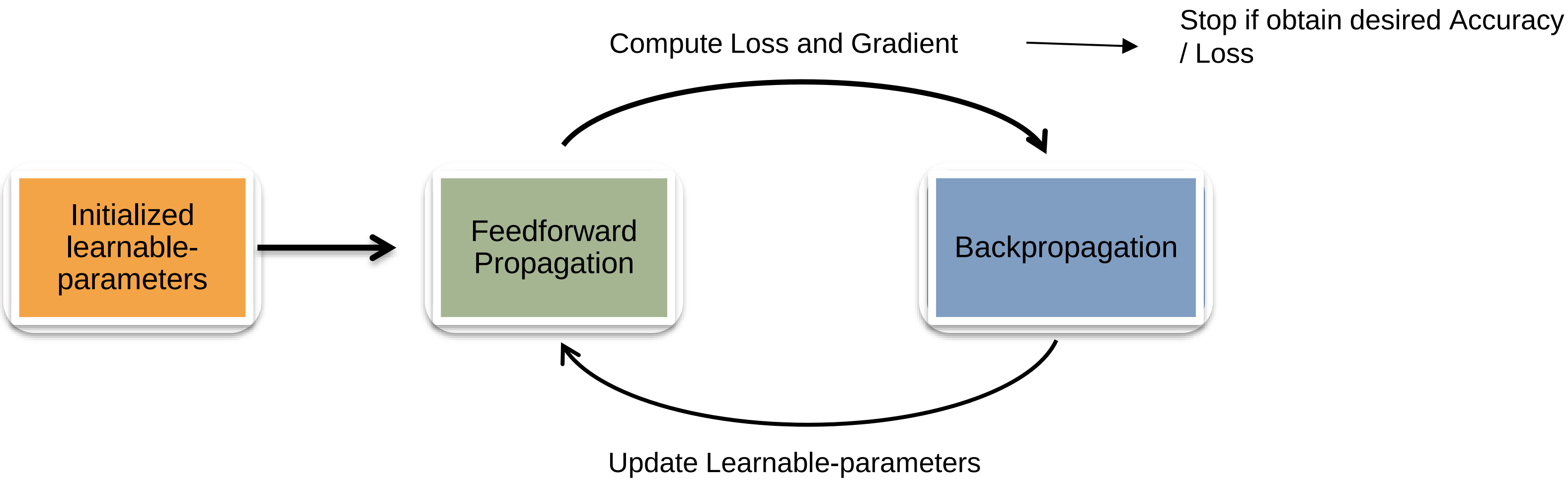
MLP Basics

- Made up of multiple layers of nodes.
- Each layer make simple decisions using different weights, w, bias, b and activation function, σ.

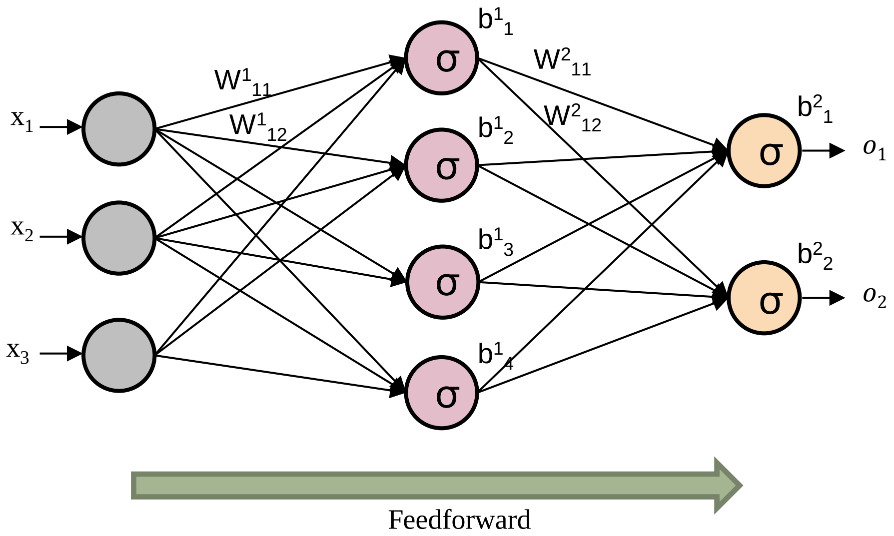
Learnable parameters



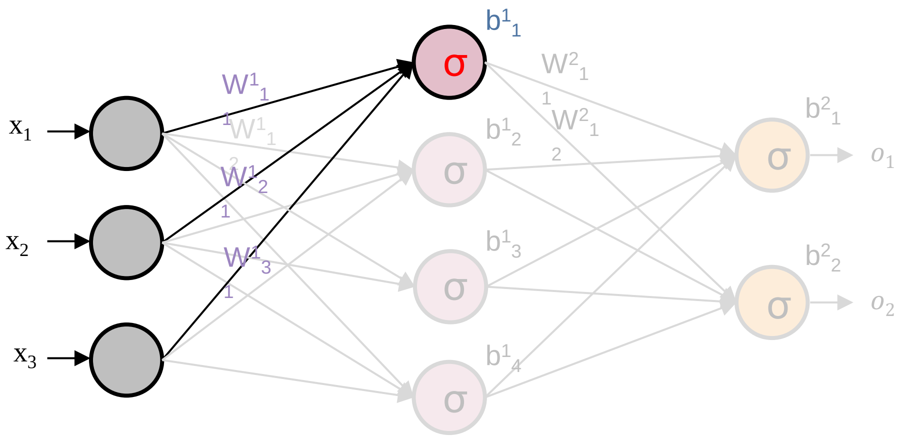
MLP Training Process



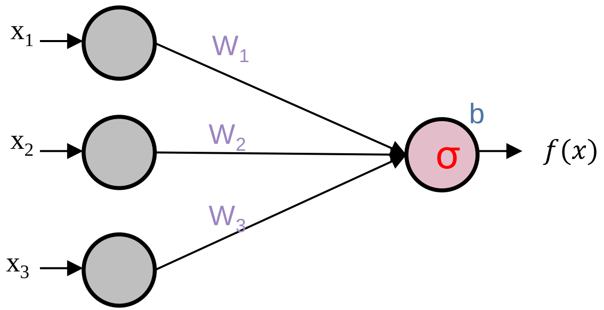
Feedforward Propagation



Feedforward Propagation



Feedforward Propagation (One Perceptron)

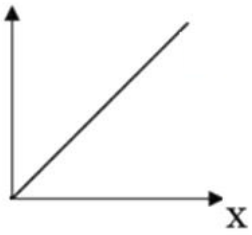


Input x Weight + Bias	$(x_1 w_1 + x_2 w_2 + x_3 w_3 + \mathrm{b})$
Apply Activation Function	$\sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + \mathrm{b})$
Obtain output	$f(x) = \sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + \mathrm{b})$

Activation Function

- A mathematics function that determines the output of each perceptron in the neural network

Linear



$\sigma(x) = x, for\ all\ x$

$$f(x) = \sigma(x_1 w_1 + x_2 w_2 + x_3 w_3 + b)$$

$$= \sigma(20 \cdot (-3) + 23 \cdot 2 + 4 \cdot 9 + 8)$$

$$= \sigma(-60 + 46 + 36 + 8)$$

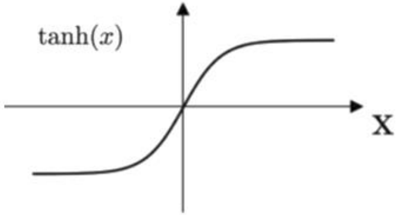
$$= \sigma(30)$$

$$= 30$$

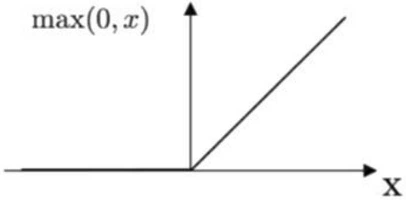
Activation Function

- Common activation functions:

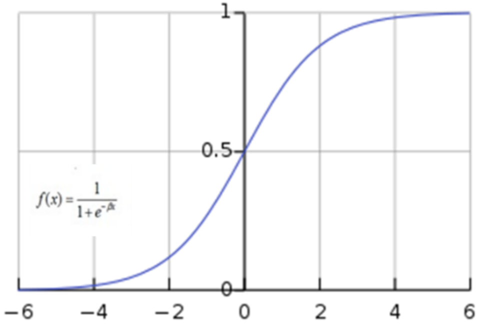
Tanh



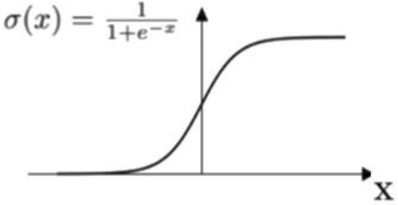
ReLU



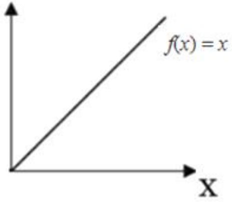
Softmax



Sigmoid



Linear

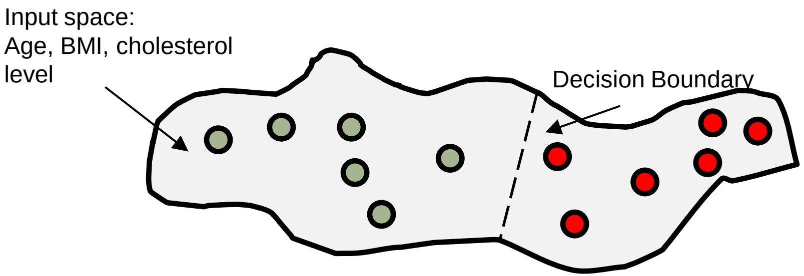


Why different activation functions?

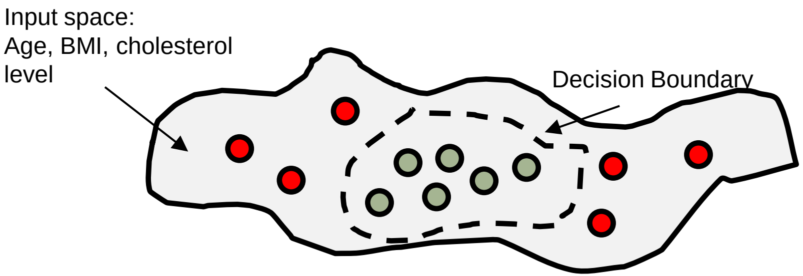
Normal Blood Pressure

High Blood Pressure

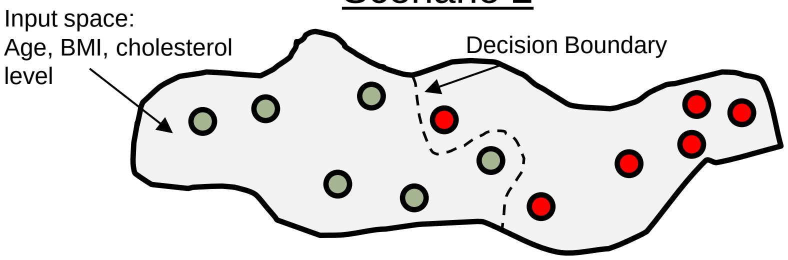
Scenario 1



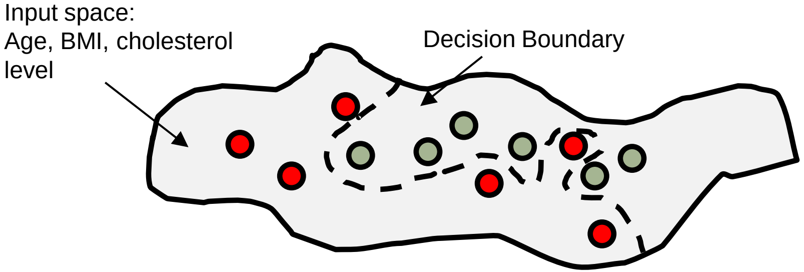
Scenario 3



Scenario 2

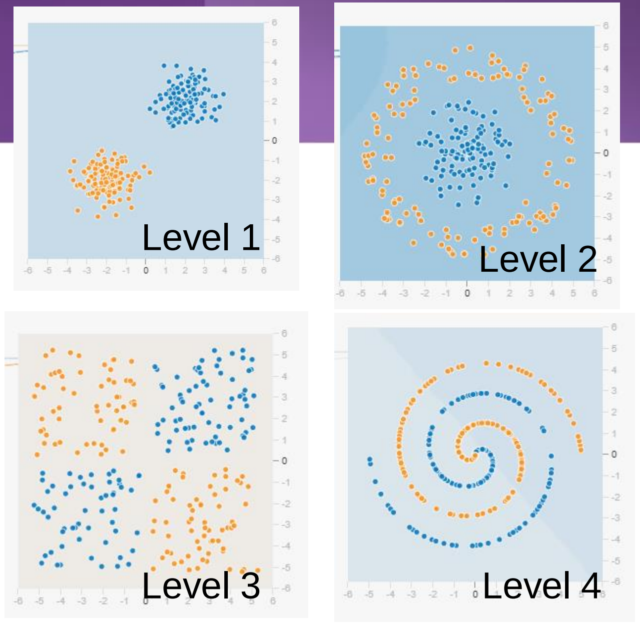


Scenario 4



Activity (30min)

- 4 persons per group
- Go to [Playground tensorflow]
- 4 datasets
- Starts with the following configuration
  - Features – x1, x2,
  - Zero hidden layer
  - Linear activation



- Try with different hyperparameters – add features / add hidden layers/ nodes / activation functions
- Share with us what you have observed

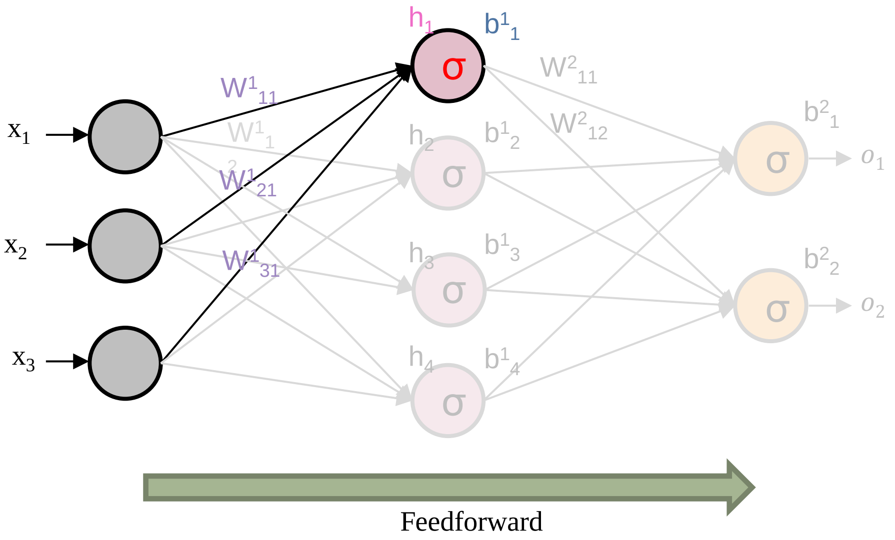
What have we learnt?

- ???
- ???
- ??

Let’s see what happen inside the MLP

Feedforward All Perceptron

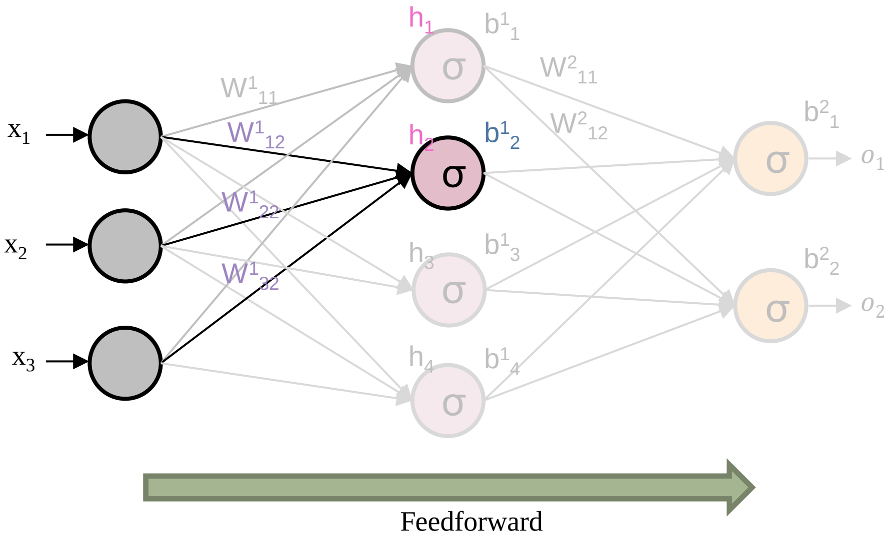
Feedforward Neural Network



$$h_1 = \underset{\text{Activation function}}{\sigma}\overbrace{(x_1 + x_2 w_{21}^{1} + x_3 w_{31}^{1} + b_1^1)}^{\textit{Input x Weight + Bias}}$$

w_{ij}^{n} ; n = nth layer; i = ith node is previous layer, j = jth node in next layer

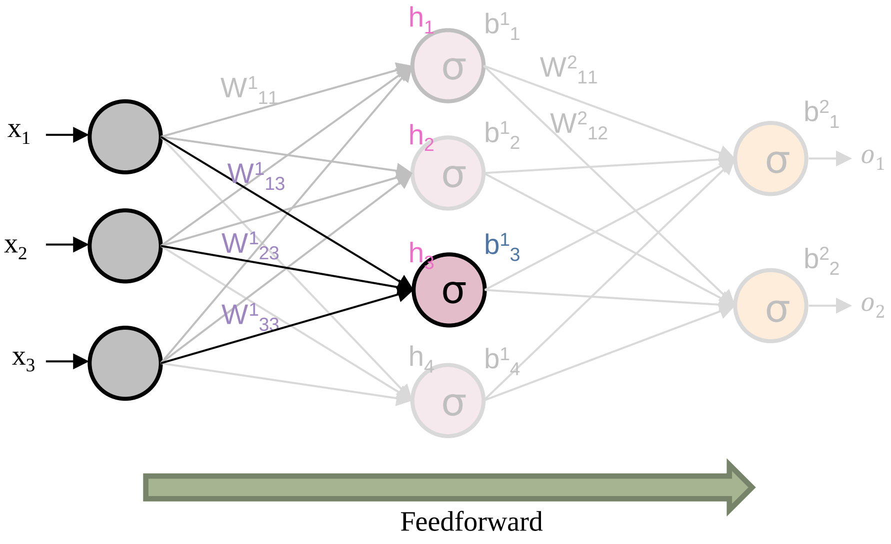
Feedforward All Perceptron



$$h_1 = \sigma(x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1)$$

$$h_2 = \sigma(x_1 w_{12}^1 + x_2 w_{22}^1 + x_3 w_{32}^1 + b_2^1)$$

Feedforward All Perceptron

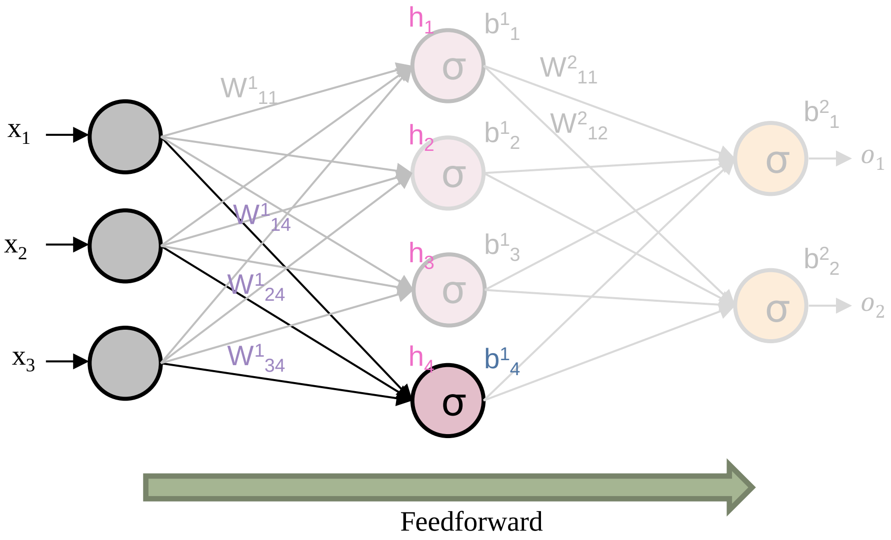


$$h_1 = \sigma(x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1)$$

$$h_2 = \sigma(x_1 w_{12}^1 + x_2 w_{22}^1 + x_3 w_{32}^1 + b_2^1)$$

$$h_3 = \sigma(x_1 w_{13}^1 + x_2 w_{23}^1 + x_3 w_{33}^1 + b_3^1)$$

Feedforward All Perceptron



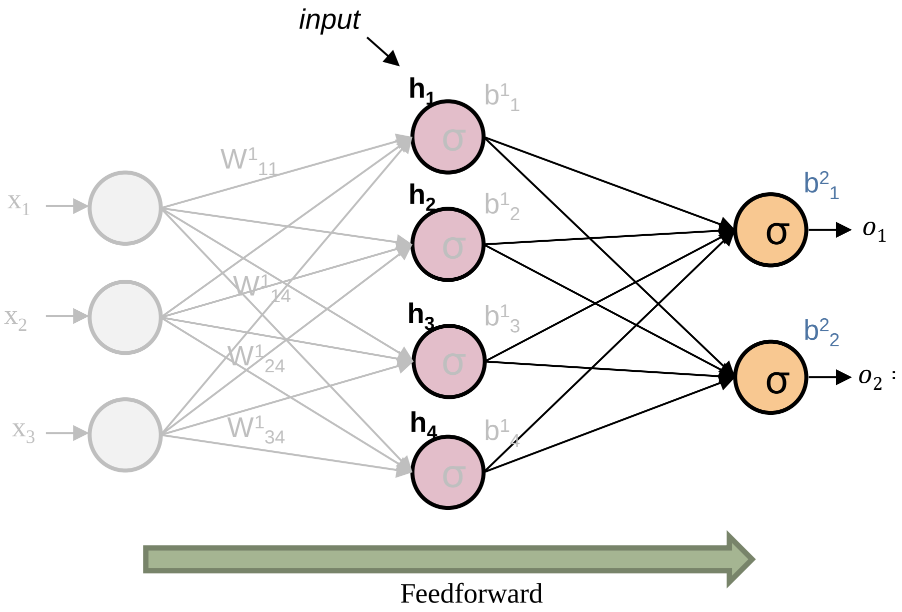
$$h_1 = \sigma(x_1 w_{11}^1 + x_2 w_{21}^1 + x_3 w_{31}^1 + b_1^1)$$

$$h_2 = \sigma(x_1 w_{12}^1 + x_2 w_{22}^1 + x_3 w_{32}^1 + b_2^1)$$

$$h_3 = \sigma(x_1 w_{13}^1 + x_2 w_{23}^1 + x_3 w_{33}^1 + b_3^1)$$

$$h_4 = \sigma(x_1 w_{14}^1 + x_2 w_{24}^1 + x_3 w_{34}^1 + b_4^1)$$

Feedforward All Perceptron



$$o_1 = \sigma(h_1 w_{11}^2 + h_2 w_{21}^2 + h_3 w_{31}^2 + h_4 w_{41}^2 + b_1^2)$$

$$o_2 = \sigma(h_1 w_{12}^2 + h_2 w_{22}^2 + h_3 w_{32}^2 + h_4 w_{42}^2 + b_2^2)$$

Loss – Negative Log Likelihood (NLL)

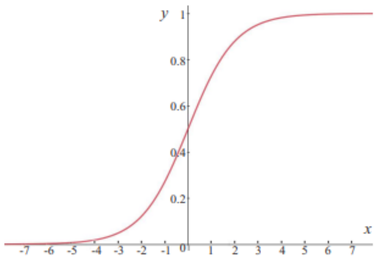
$o_1 = 0.4$

$o_2 = 0.2$

→ Sigmoid function → Output Probability → <u>Prediction, $\hat{y}$</u>

> 0.5 → 1

< 0.5 → 0



<u>Ground Truth</u>

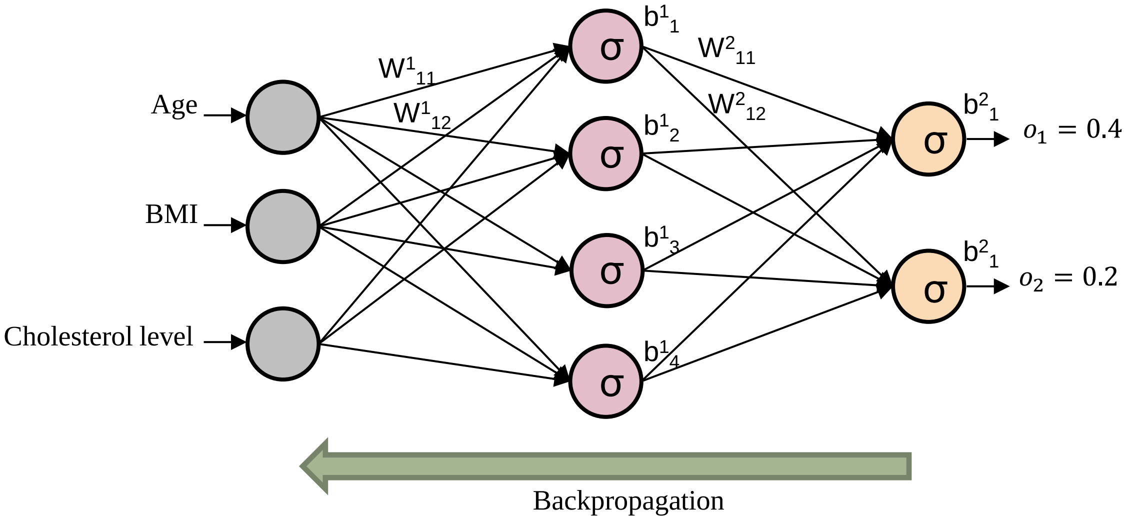
Normal blood pressure, y = 0

High blood pressure, y=1

$$\mathcal{L}(\theta) = -\log p(\mathcal{D}|\theta) = -\sum_{n=1}^{N} \log p(y_n|x_n;\theta)$$

Backpropagations to update weight

- Compute Gradients, i.e: $\frac{\partial L}{\partial w_{ij}^{n}}, \frac{\partial L}{\partial b_{i}^{n}}$
- Backpropagate the gradient to updates the weights



Gradient Descent

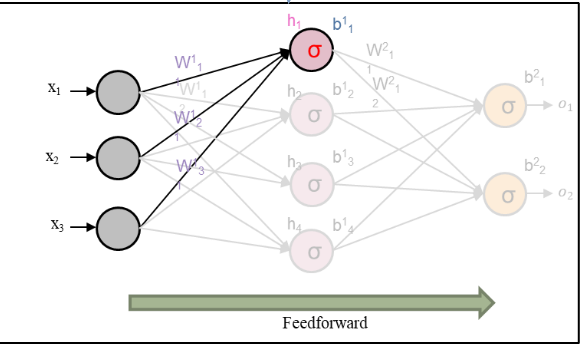
$$\theta_{t+1} = \theta_t + \rho_t d_t$$

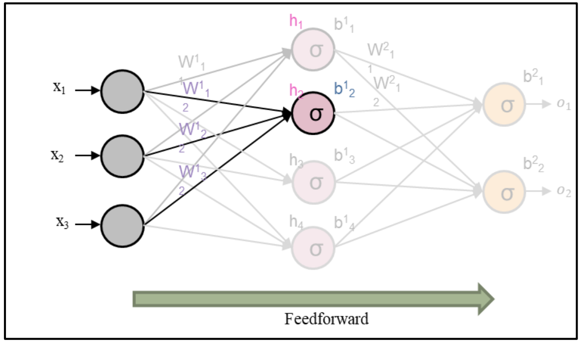
Updated parameter (weight, bias)

Step size

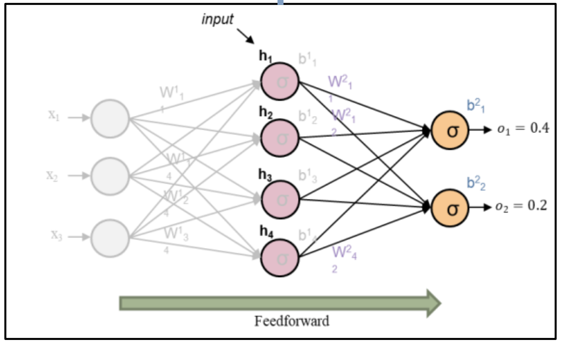
Descent direction (Gradient: $\frac{\partial L}{\partial w_{ij}^n}$, $\frac{\partial L}{\partial b_i^n}$)

Compute Gradients, i.e: $\frac{\partial L}{\partial w_{ij}^n}$, $\frac{\partial L}{\partial b_i^n}$

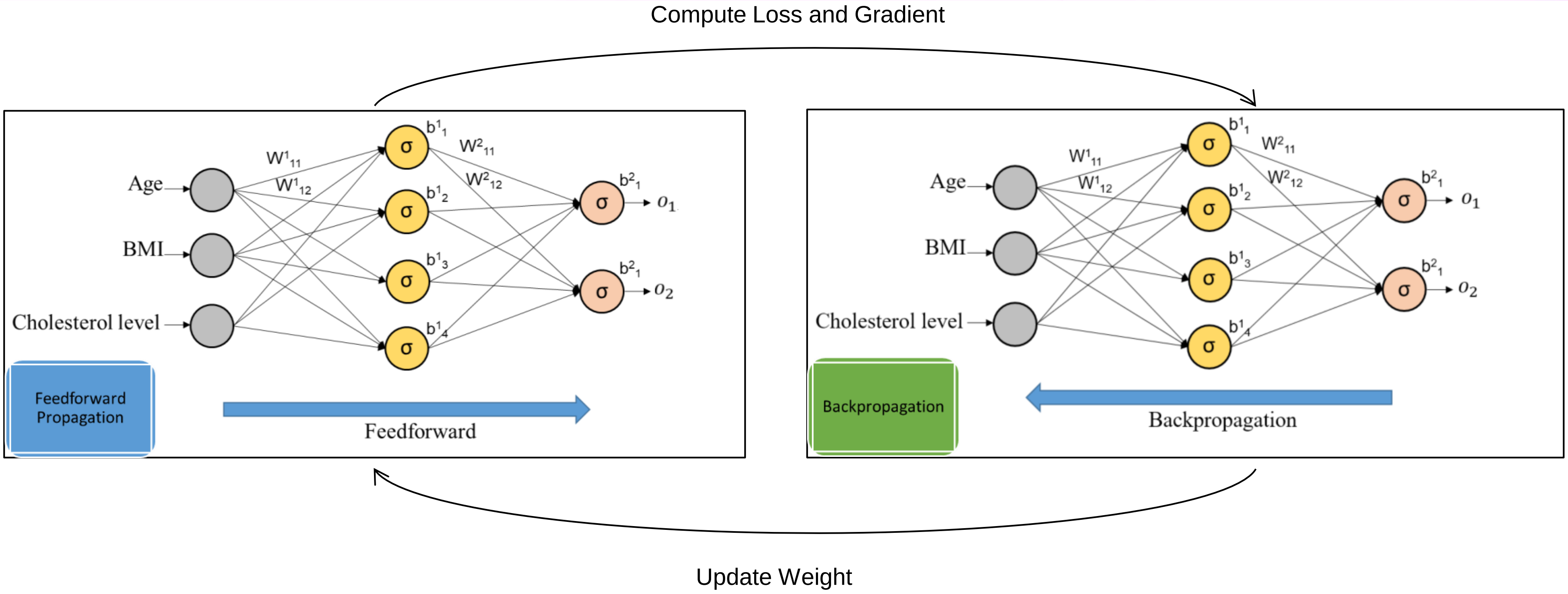




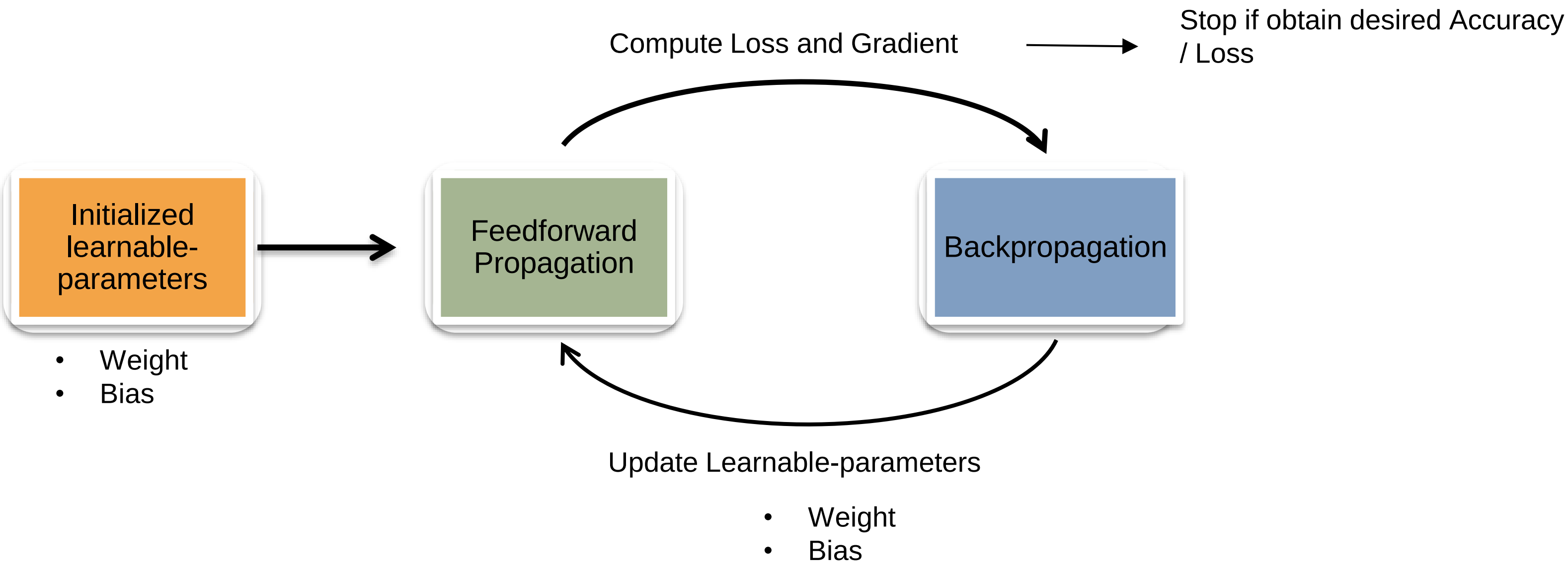
…



Training Iteratively until Loss ~ 0

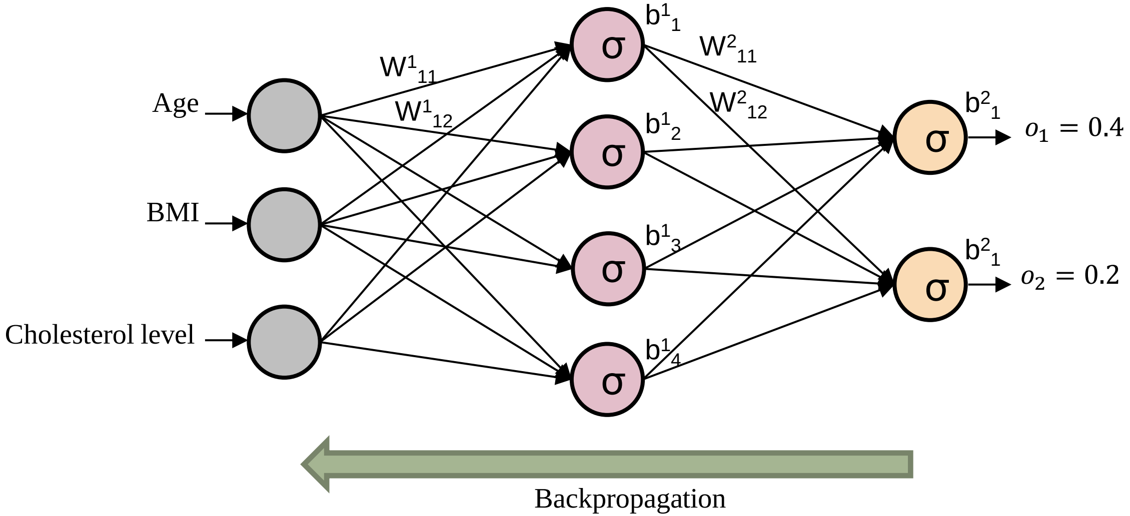


Neural Networks Training Process



Model Training - Backpropagation

- Compute Gradients, i.e: $\frac{\partial L}{\partial w_{ij}^{n}}, \frac{\partial L}{\partial b_{i}^{n}}$
- Backpropagate the gradient to updates the weights
- How to do that in computer?



Model Training - Backpropagation

Consider a mapping of the form $\boldsymbol{o} = \boldsymbol{f}(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{o} \in \mathbb{R}^m$. We assume that $\boldsymbol{f}$ is defined as a composition of functions:

$$\boldsymbol{f} = \boldsymbol{f}_4 \circ \boldsymbol{f}_3 \circ \boldsymbol{f}_2 \circ \boldsymbol{f}_1 \tag{13.22}$$

where $\boldsymbol{f}_1 : \mathbb{R}^n \to \mathbb{R}^{m_1}$, $\boldsymbol{f}_2 : \mathbb{R}^{m_1} \to \mathbb{R}^{m_2}$, $\boldsymbol{f}_3 : \mathbb{R}^{m_2} \to \mathbb{R}^{m_3}$, and $\boldsymbol{f}_4 : \mathbb{R}^{m_3} \to \mathbb{R}^m$. The intermediate steps needed to compute $\boldsymbol{o} = \boldsymbol{f}(\boldsymbol{x})$ are $\boldsymbol{x}_2 = \boldsymbol{f}_1(\boldsymbol{x})$, $\boldsymbol{x}_3 = \boldsymbol{f}_2(\boldsymbol{x}_2)$, $\boldsymbol{x}_4 = \boldsymbol{f}_3(\boldsymbol{x}_3)$, and $\boldsymbol{o} = \boldsymbol{f}_4(\boldsymbol{x}_4)$.

$$\boldsymbol{x}_2 = \boldsymbol{f}_1(\boldsymbol{x}),$$

$$\boldsymbol{x}_3 = \boldsymbol{f}_2(\boldsymbol{x}_2)$$

$$\boldsymbol{x}_4 = \boldsymbol{f}_3(\boldsymbol{x}_3)$$

$$\boldsymbol{o} = \boldsymbol{f}_4(\boldsymbol{x}_4).$$

Model Training - Backpropagation

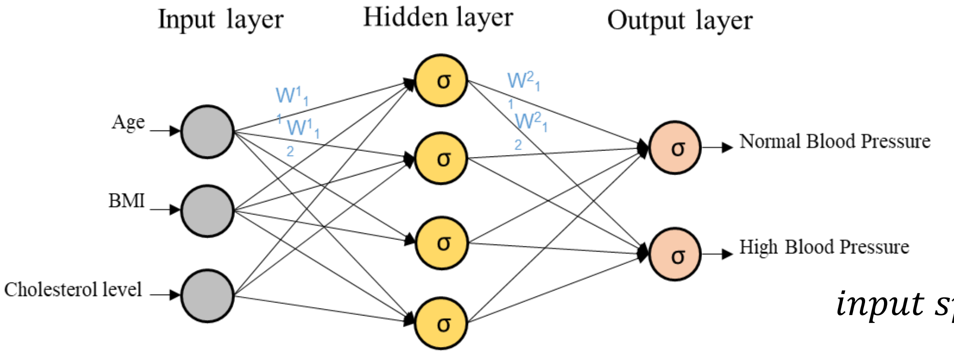
We can compute the Jacobian $\mathbf{J}_{\boldsymbol{f}}(x) = \frac{\partial \boldsymbol{o}}{\partial \boldsymbol{x}} \in \mathbb{R}^{m \times n}$ using the chain rule:

$$\frac{\partial \boldsymbol{o}}{\partial x} = \frac{\partial \boldsymbol{o}}{\partial x_4}\frac{\partial x_4}{\partial x_3}\frac{\partial x_3}{\partial x_2}\frac{\partial x_2}{\partial x} = \frac{\partial \boldsymbol{f}_4(x_4)}{\partial x_4}\frac{\partial \boldsymbol{f}_3(x_3)}{\partial x_3}\frac{\partial \boldsymbol{f}_2(x_2)}{\partial x_2}\frac{\partial \boldsymbol{f}_1(x)}{\partial x}$$

$$= \mathbf{J}_{\boldsymbol{f}_4}(x_4)\mathbf{J}_{\boldsymbol{f}_3}(x_3)\mathbf{J}_{\boldsymbol{f}_2}(x_2)\mathbf{J}_{\boldsymbol{f}_1}(x)$$

Example – Tabular Data

- Data:
  - Input (Age, BMI, Cholesterol level) - 1 dimension
  - Output (Normal / High Blood Pressure)



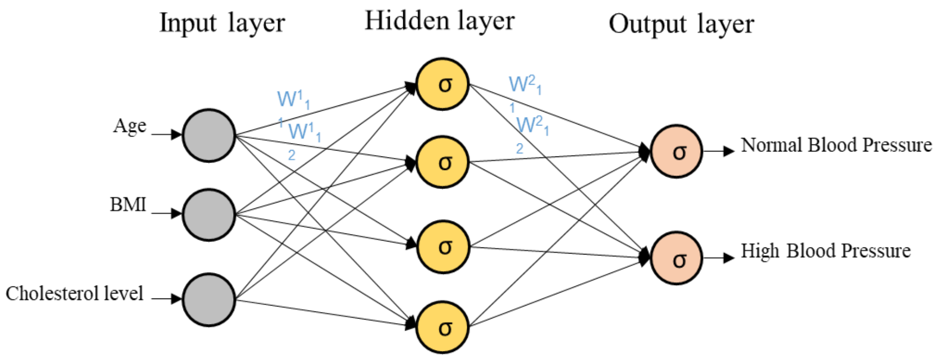
$$input\ space, \mathcal{X} = \ set\ of\ instances$$

$$\mathcal{X} = \mathbb{R}^D, where\ D = 1$$

$$f: \mathcal{X} \rightarrow \mathcal{Y}$$

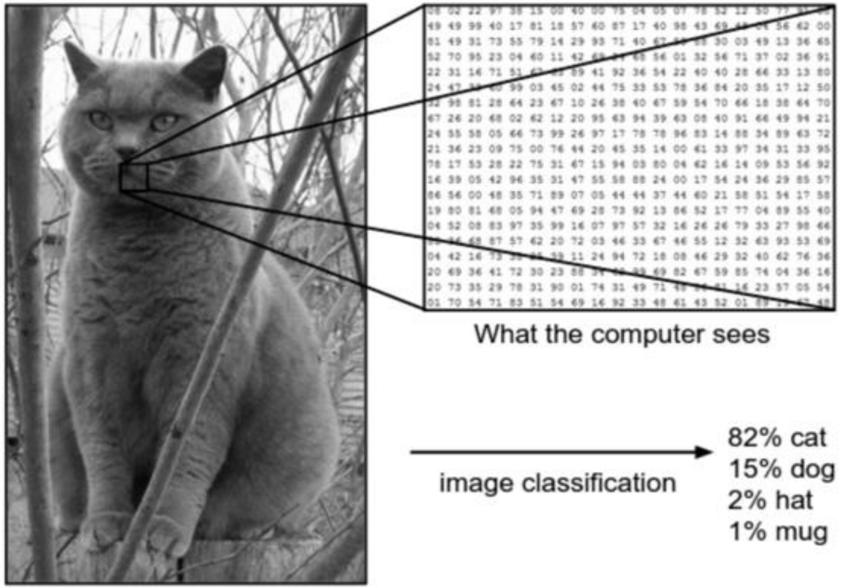
Example – Tabular Data

- We call such data “structured data” or “tabular data”
- Since the data is often stored in an N x D matrix where N = number of samples, D = number of features



Convolutional Neural Network

- When your input is a Grey Scaled Image

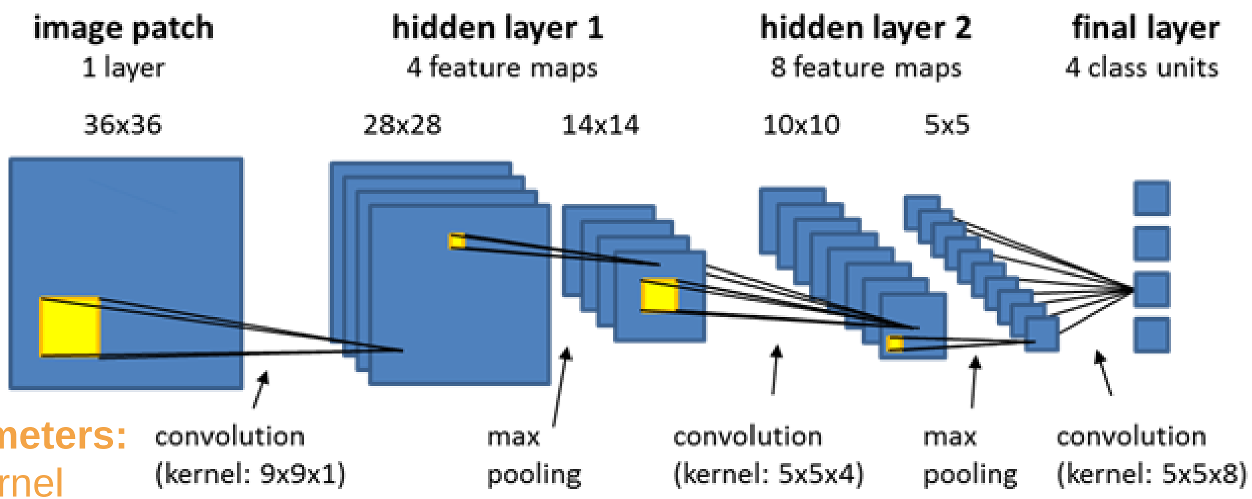


$$input\ space, \mathcal{X} = \ \ set\ of\ images$$

$$\mathcal{X} = \ \mathbb{R}^{D}, where\ D = 2$$

$$f: \mathcal{X} \rightarrow \mathcal{Y}$$

Convolutional Neural Network



Trainable parameters:
Weight in the kernel

Summary

- Deep Learning Foundations
- Multilayer Perceptron (MLP) Basics
- Training Process
- Applications and Examples
- Quiz

Quiz (30min)

- Go to this link to test your understanding
- https://docs.google.com/forms/d/1joN4LojoS1foc-Xbnj0Spa5E-8uhGeb-onEnZjZP1ts/preview