THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

CREATE CHANGE

Machine Leaning

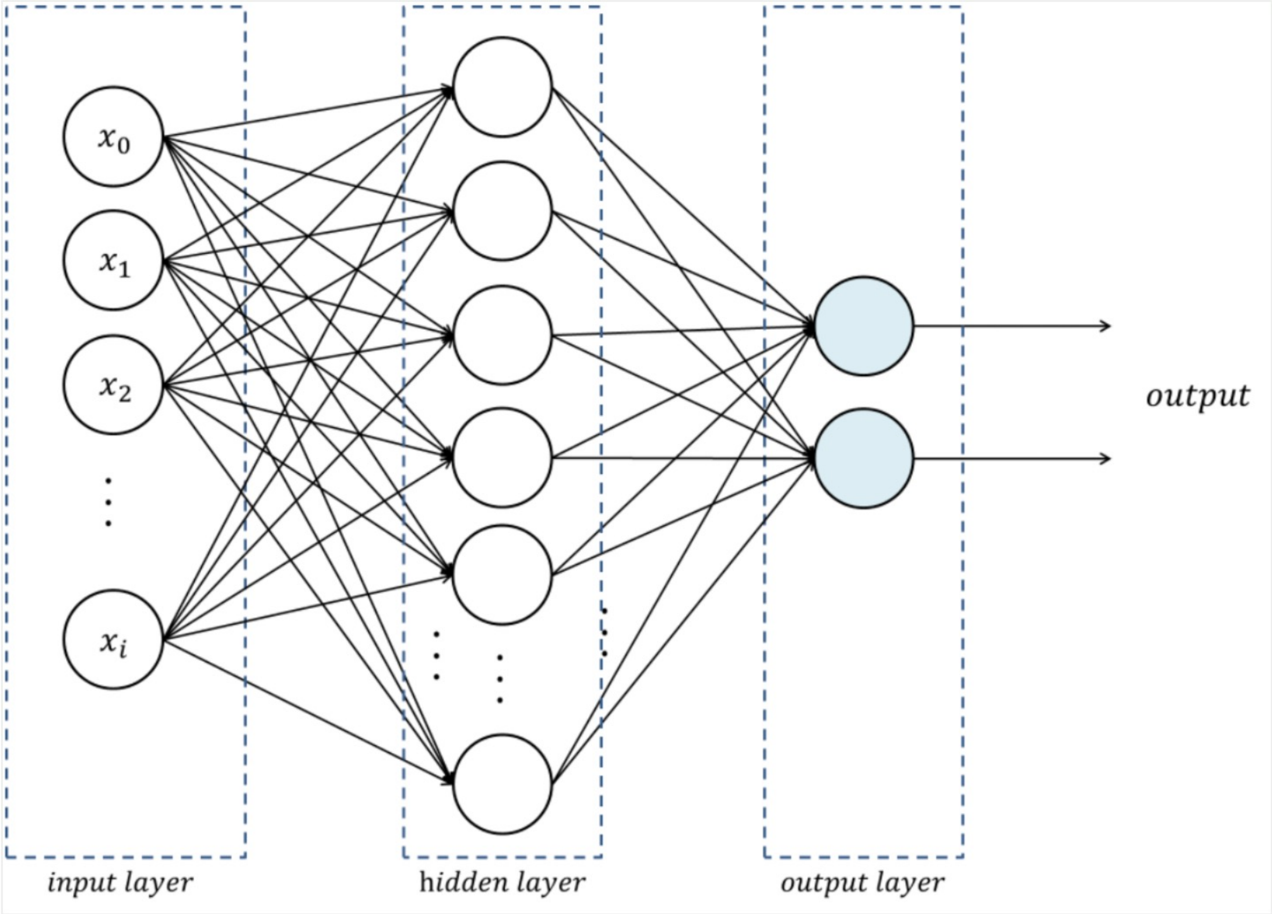
COMP4702/COMP7703

Prac 7

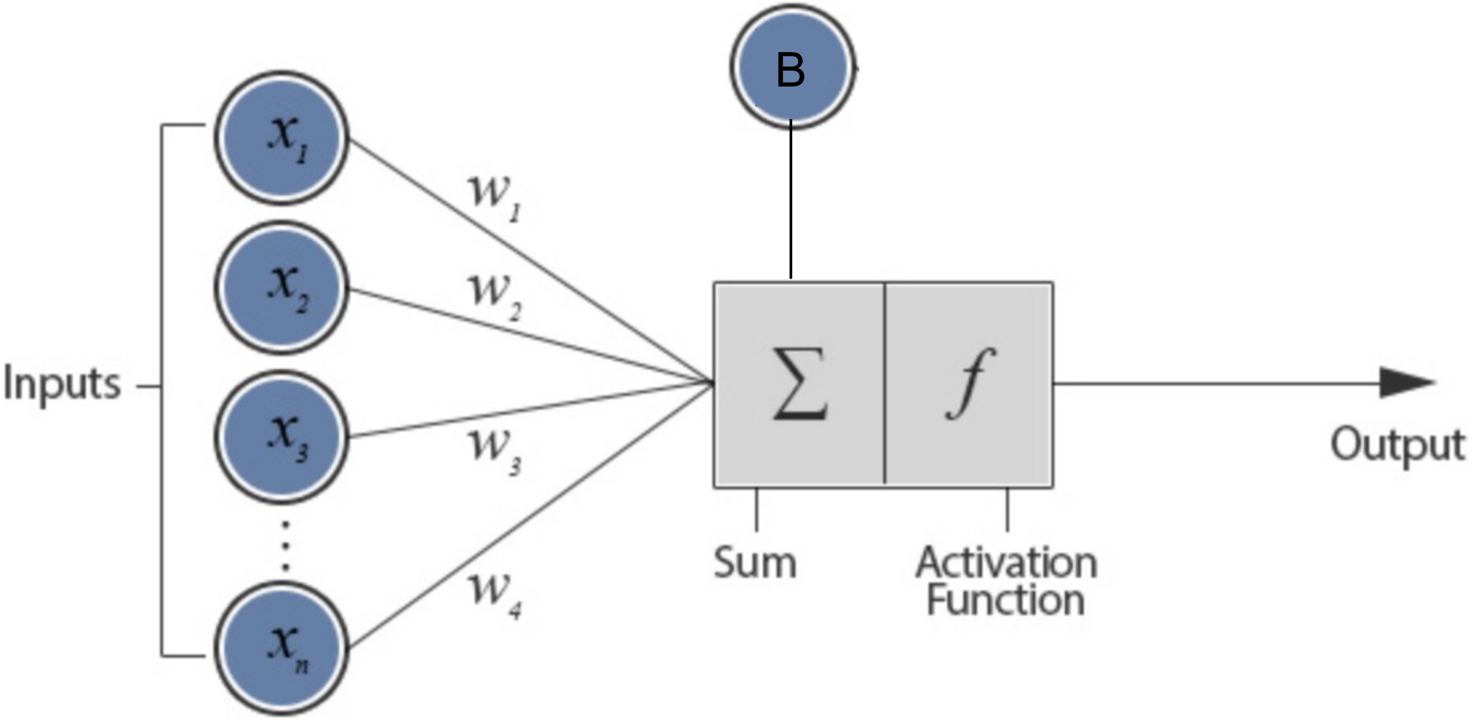
Amelia Qiu

https://github.com/Amelia-Tong/MachineLearning_COMP4702/blob/main

Model Architecture (Multi-later Perceptron)



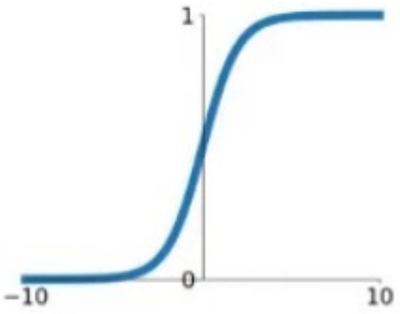
Model Architecture (Perceptron/Neuron)



Activation Functions

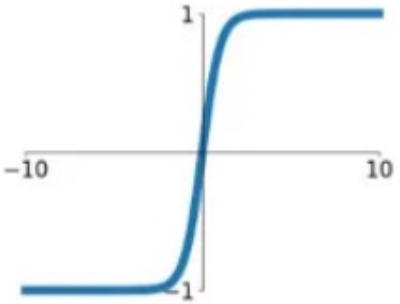
Sigmoid

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



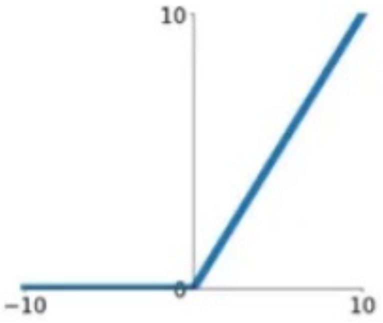
tanh

$\tanh(x)$



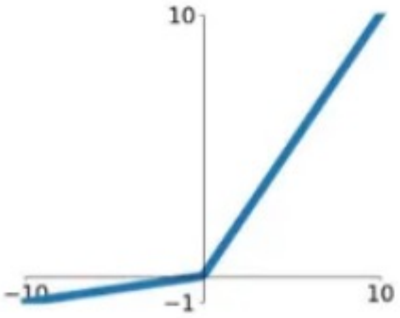
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$



Learning Mechanism

MLPs performs a forward pass and a backward pass through the neural net:

- the forward pass propagates information from the input neurons to the output neurons to compute the outputs of all neurons.

- the backward pass propagates information from the output neurons to the input neurons to compute derivatives and adjust the weights.
    - Loss functions: to compare the predictions and the true values
    - Optimizers: to minimize the loss

Loss Function

- Mean Squared Error:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

- Cross-Entropy/
  Binary Cross-Entropy:

$$\mathrm{CE} = -\sum_{i=1}^{n} Y_i \cdot \log(\hat{Y}_i)$$

$$\mathrm{BCE} = -\frac{1}{n}\sum_{i=1}^{n}[Y_i \cdot \log(\hat{Y}_i) + (1 - Y_i) \cdot \log(1 - \hat{Y}_i)]$$

- Dice Loss:

$$\text{Dice Loss} = 1 - \frac{2\times\sum_{i=1}^{n} Y_i\hat{Y}_i}{\sum_{i=1}^{n} Y_i + \sum_{i=1}^{n} \hat{Y}_i}$$

Optimizer

- **Gradient Descent**: The simplest optimizer that updates weights in the network by taking a step proportional to the negative of the gradient of the loss function.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t)$$

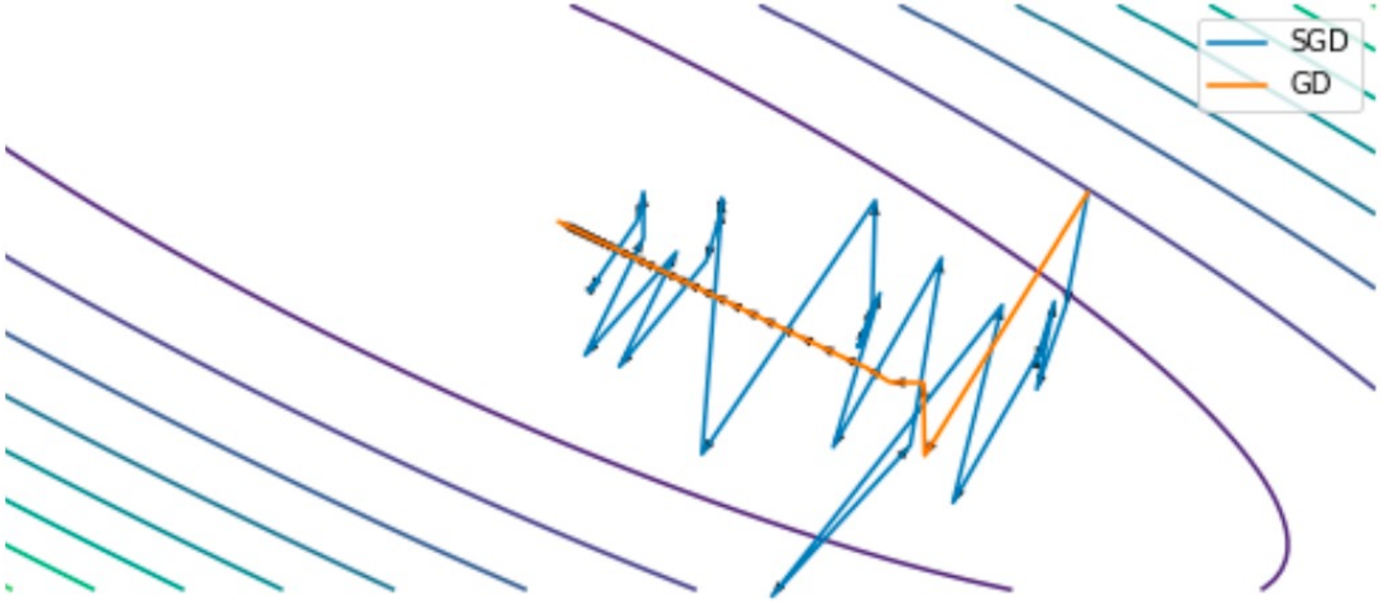
$$f(\mathbf{w}) = \frac{1}{n} \sum_i f_i(\mathbf{w})$$

- **Stochastic Gradient Descent (SGD)**: A variant of gradient descent, it updates weights using only a single sample or a batch of samples which makes the computation much faster.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \tilde{g}_t,$$

$$\tilde{g}_t = \frac{1}{|S|} \sum_{i \in S} \nabla f_i(\mathbf{w})$$

Optimizer



- Although SGD can jump around in the solution space, it is often computationally attractive when the dataset is very large.

Optimizer

- **Gradient Descent**: The simplest optimizer that updates weights in the network by taking a step proportional to the negative of the gradient of the loss function.
- **Stochastic Gradient Descent (SGD)**: A variant of gradient descent, it updates weights using only a single sample or a batch of samples which makes the computation much faster.
- **Adam (Adaptive Moment Estimation)**: Combines momentum (like RMSProp) and adaptive learning rate (like AdaGrad), which often leads to faster and more stable convergence.