MACHINE LEARNING CDS503

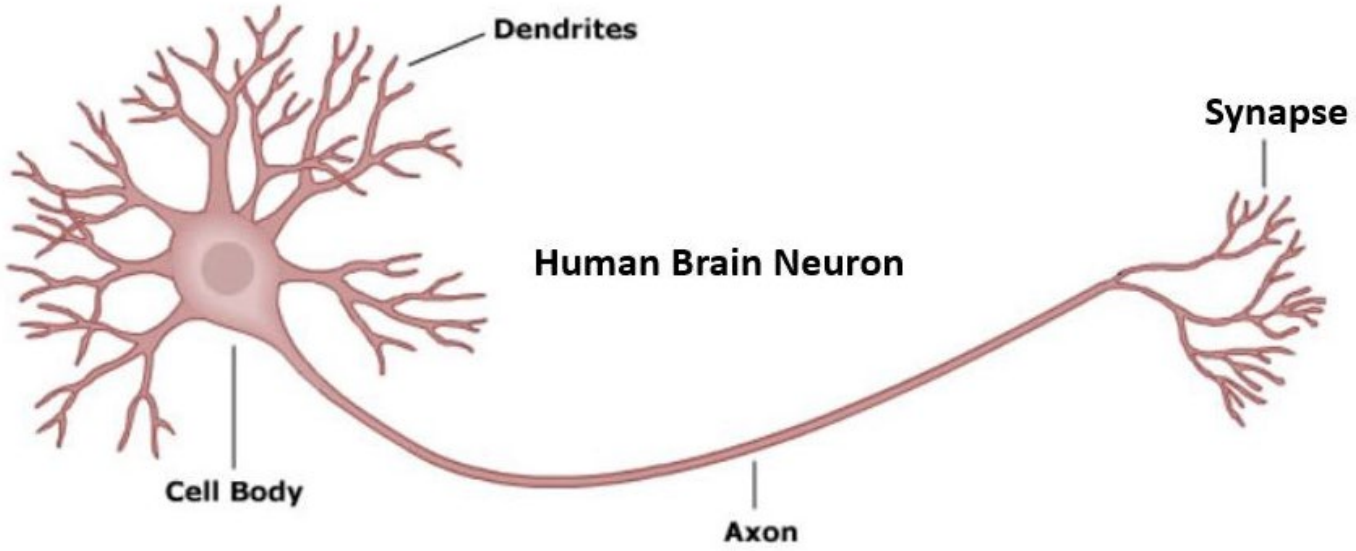
Topic 12: Neural Networks

Mohd Halim Mohd Noor, PhD

Outline

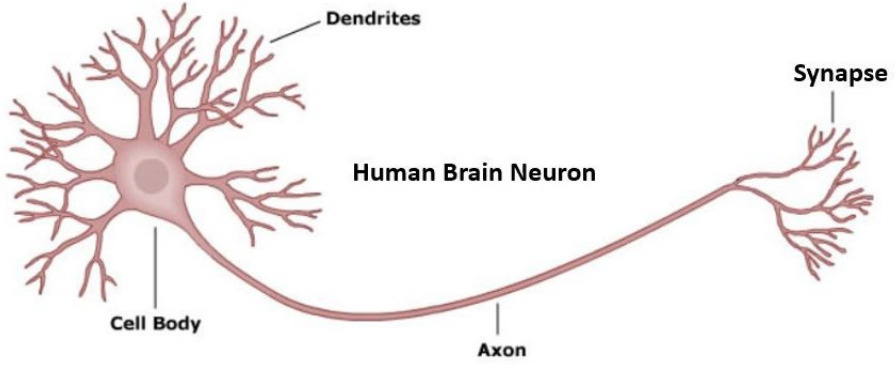
- Introduction
- Building Neural Networks
- Training Neural Networks
- Regularization
  - Early Stopping
  - Dropout

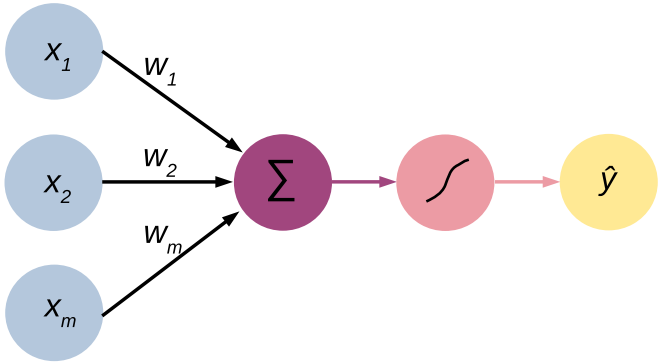
Motivation for Neural Nets



Use biology as inspiration for mathematical model

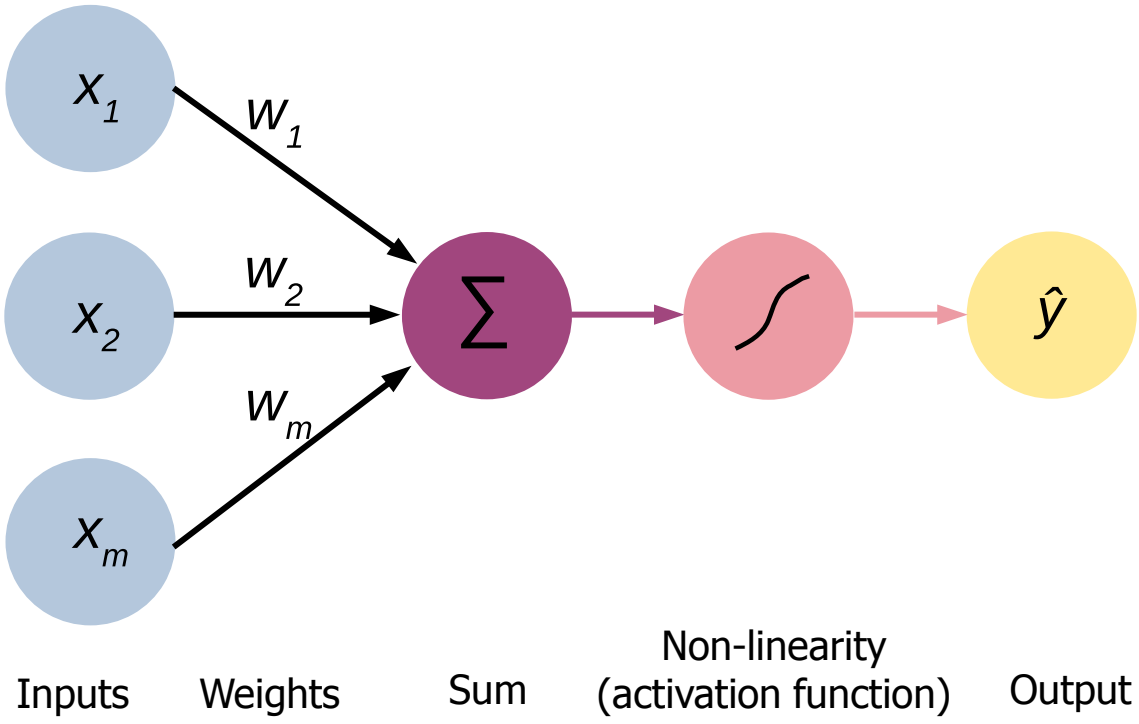
Motivation for Neural Nets



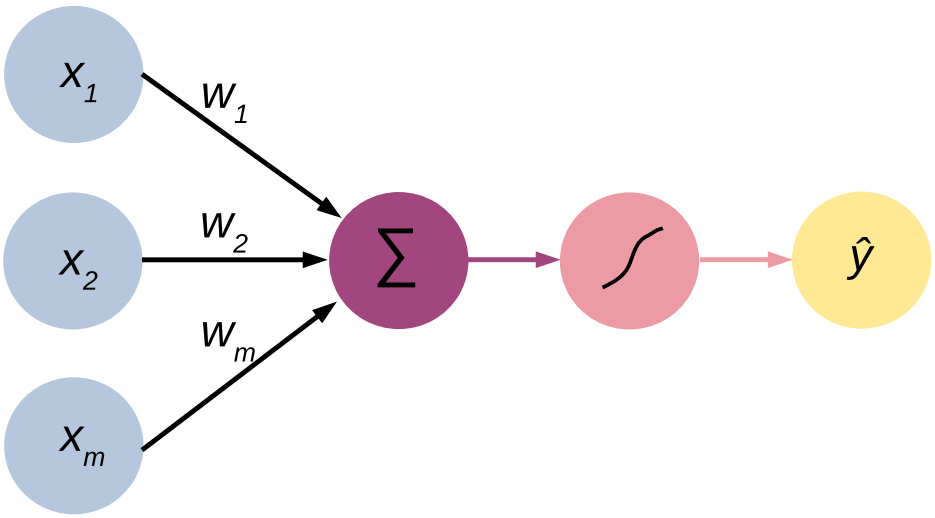


Perceptron

The Perceptron (Neuron)



The Perceptron: Forward Propagation



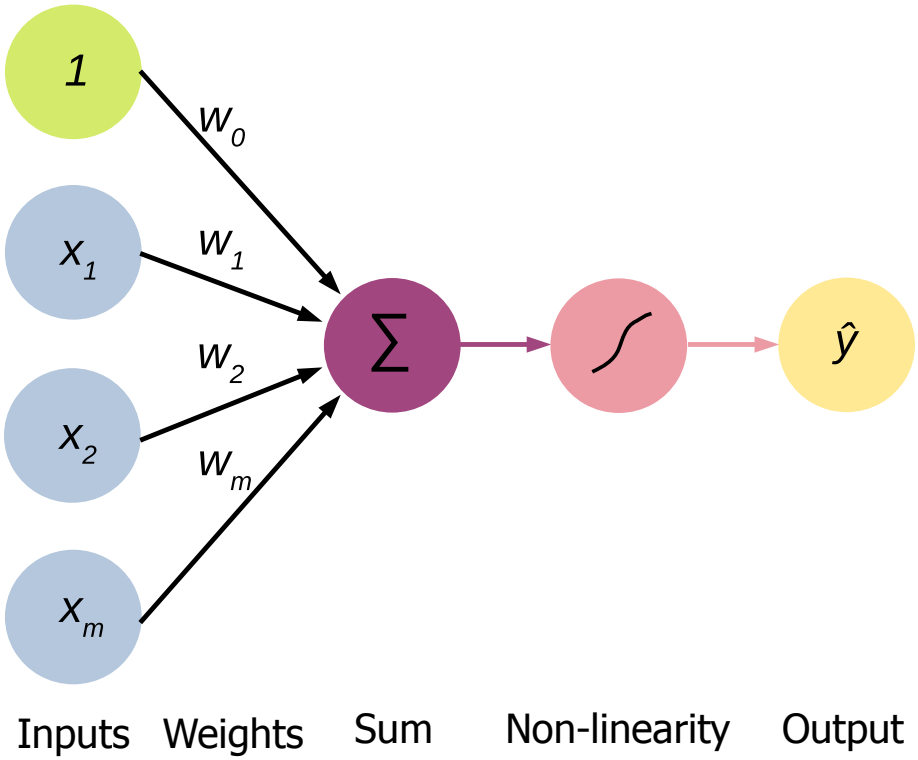
Inputs Weights Sum Non-linearity Output

Linear combination of inputs

$$\hat{y} = g\left(\sum_{i=1}^{m} w_i x_i\right)$$

Non-linearity

The Perceptron: Forward Propagation



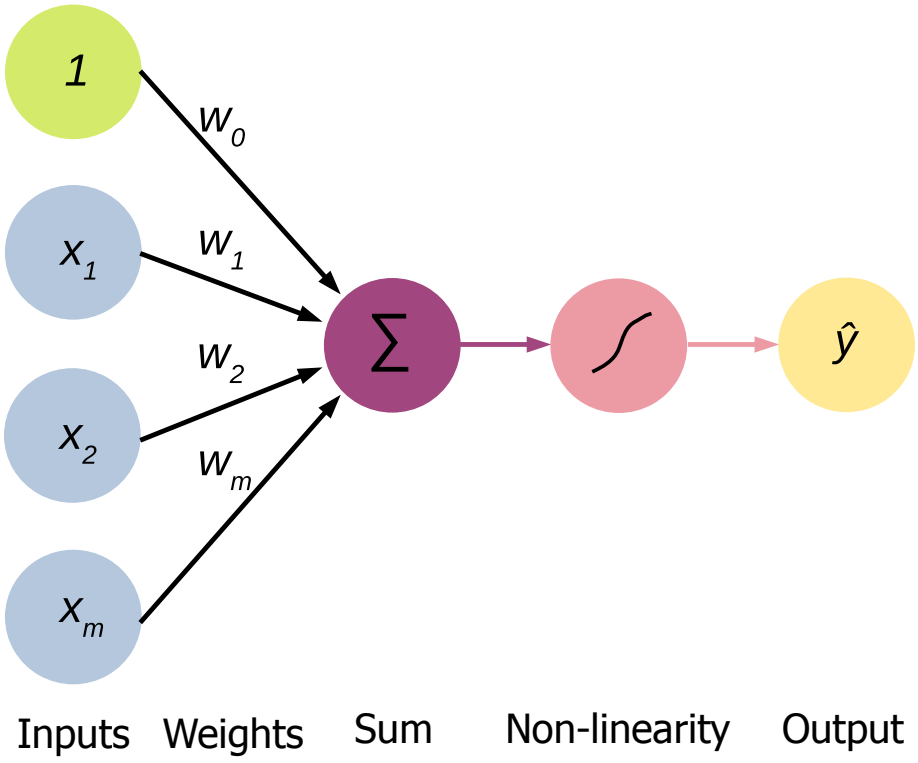
Linear combination of inputs

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

Non-linearity activation function

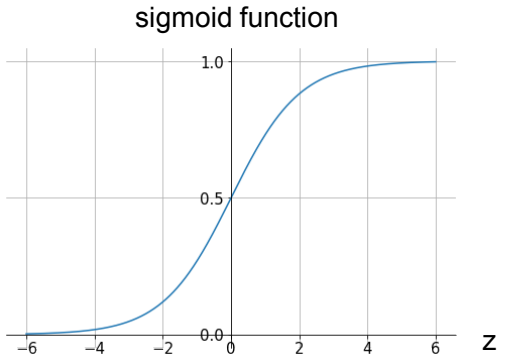
bias

The Perceptron: Forward Propagation



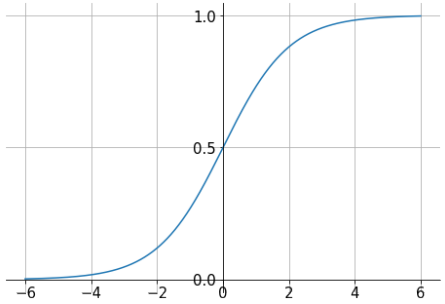
$$\hat{y} = g\left(w_0 + \sum_{i=0}^{m} w_i x_i\right)$$

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



Activation Functions

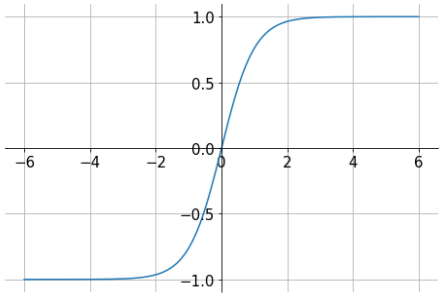
Sigmoid



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

tf.nn.sigmoid(z)

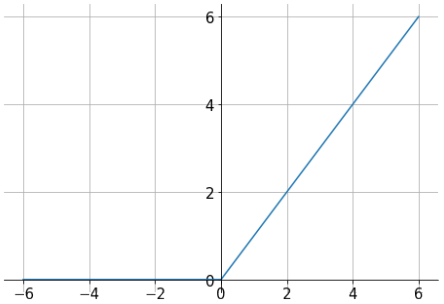
Hyperbolic tangent



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

tf.nn.tanh(z)

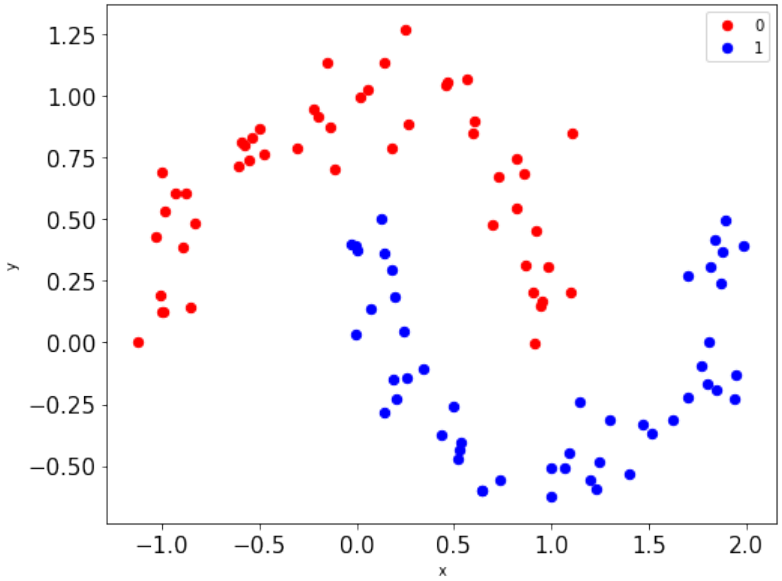
Rectified Linear Unit (ReLU)



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

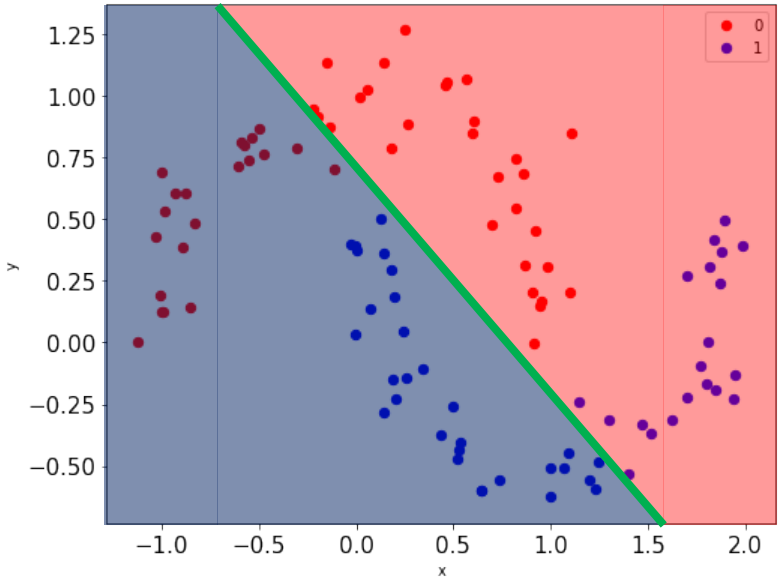
tf.nn.relu(z)

Importance of Activation Functions



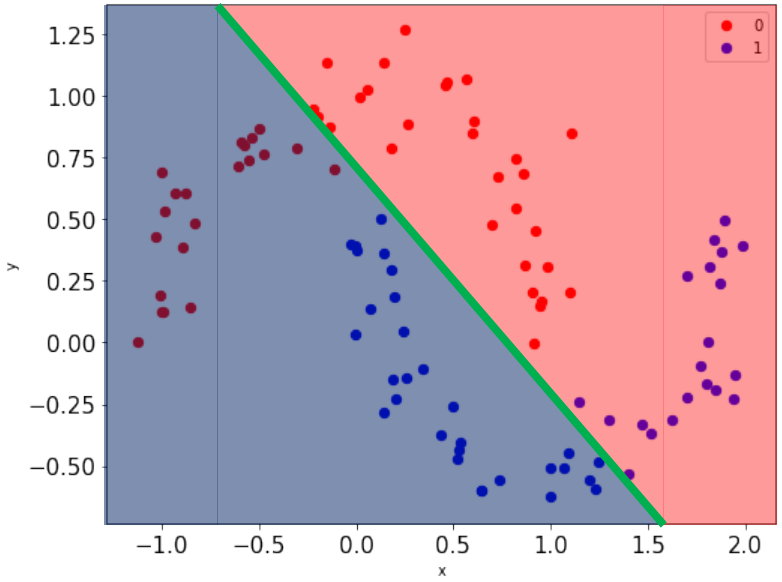
What if we want to build a Neural Network to distinguish between blue and red data points?

Importance of Activation Functions

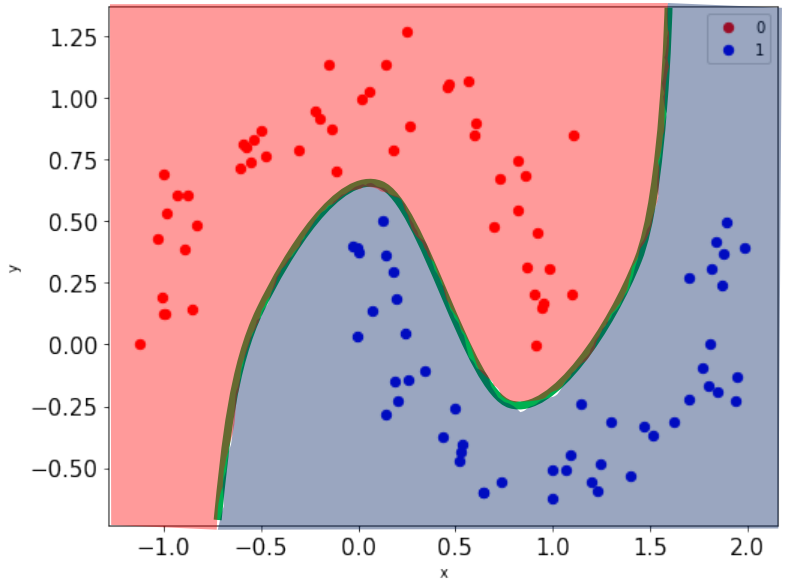


Linear activation functions produce linear decision boundary

Importance of Activation Functions



Linear activation functions produce linear decision boundary

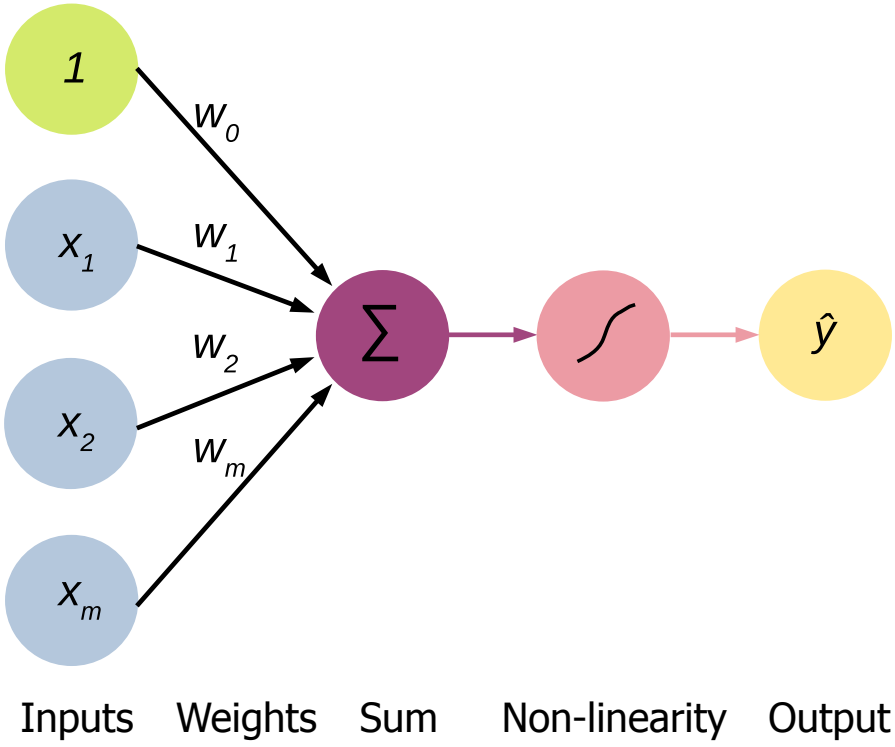


Non-linear activation functions allow us to approximate arbitrarily complex functions

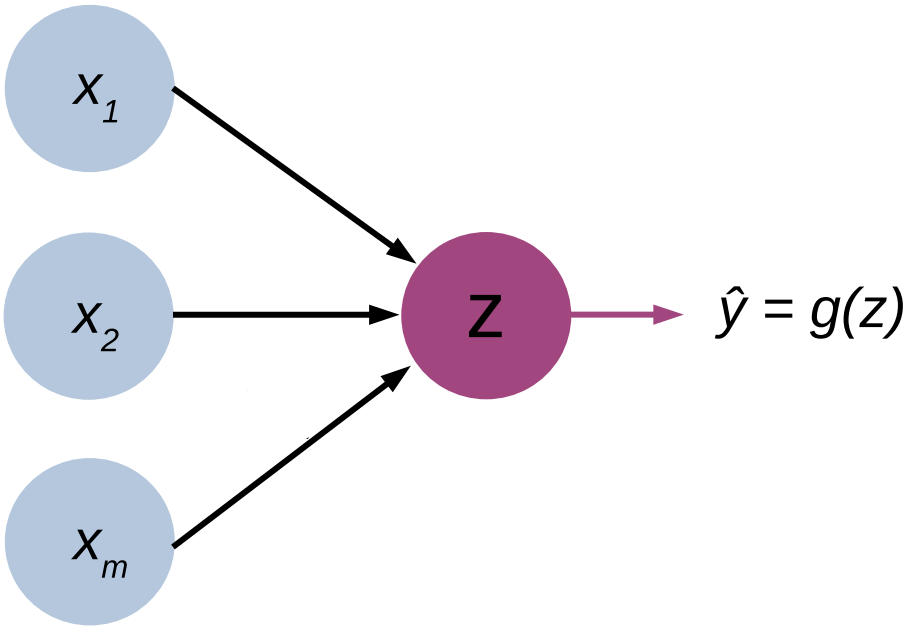
UNIVERSITI SAINS MALAYSIA

Building Neural Networks

Perceptron

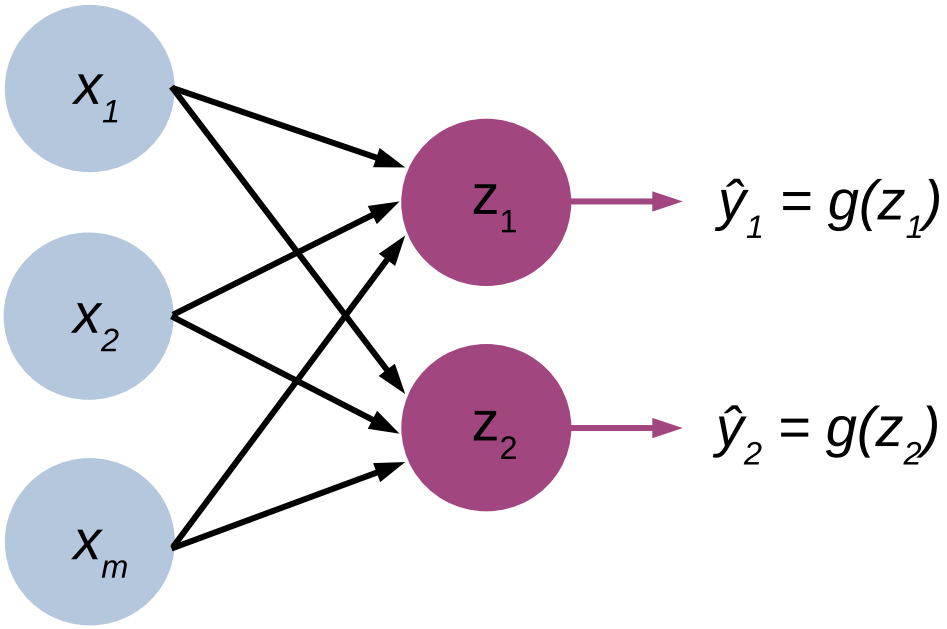

1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
$\sum$
$\int$
$\hat{y}$
Inputs
Weights
Sum
Non-linearity
Output

Perceptron: Simplified



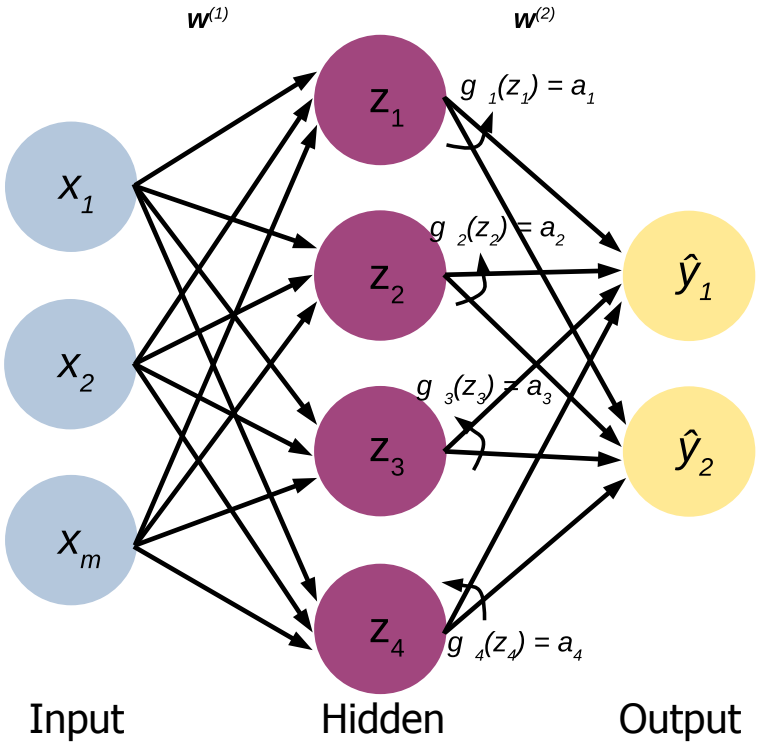
$$z = w_0 + \sum_{i=0}^{m} w_i x_i$$

Multi Output Perceptron



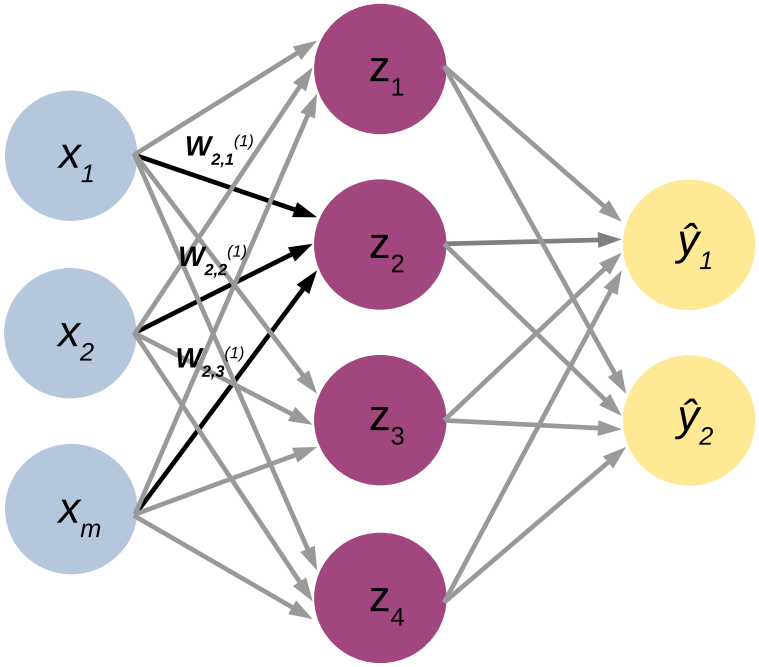
$$z_i = w_{0,i} + \sum_{j=0}^{m} w_{ij} x_j$$

Single Layer Neural Network

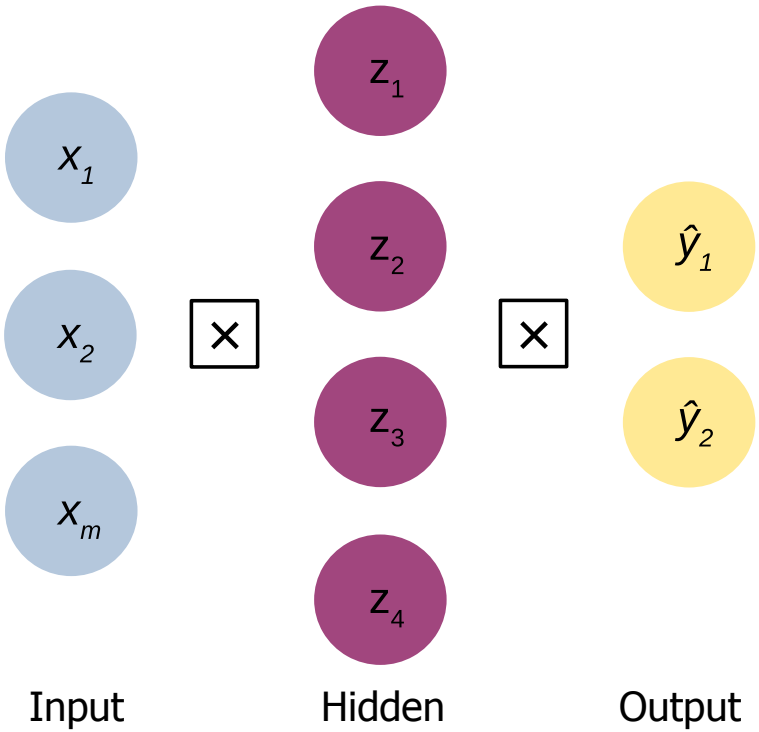


$$z_i^{(2)} = w_{0,i}^{(1)} + \sum_{j=1}^{m} w_{ij}^{(1)} x_j \quad a_i^{(2)} = g(z_i) \qquad z_i^{(3)} = w_{0,i}^{(2)} + \sum_{j=1}^{m} w_{ij}^{(2)} a_j^{(2)} \quad \hat{y}_i = g\left(z_i^{(3)}\right)$$

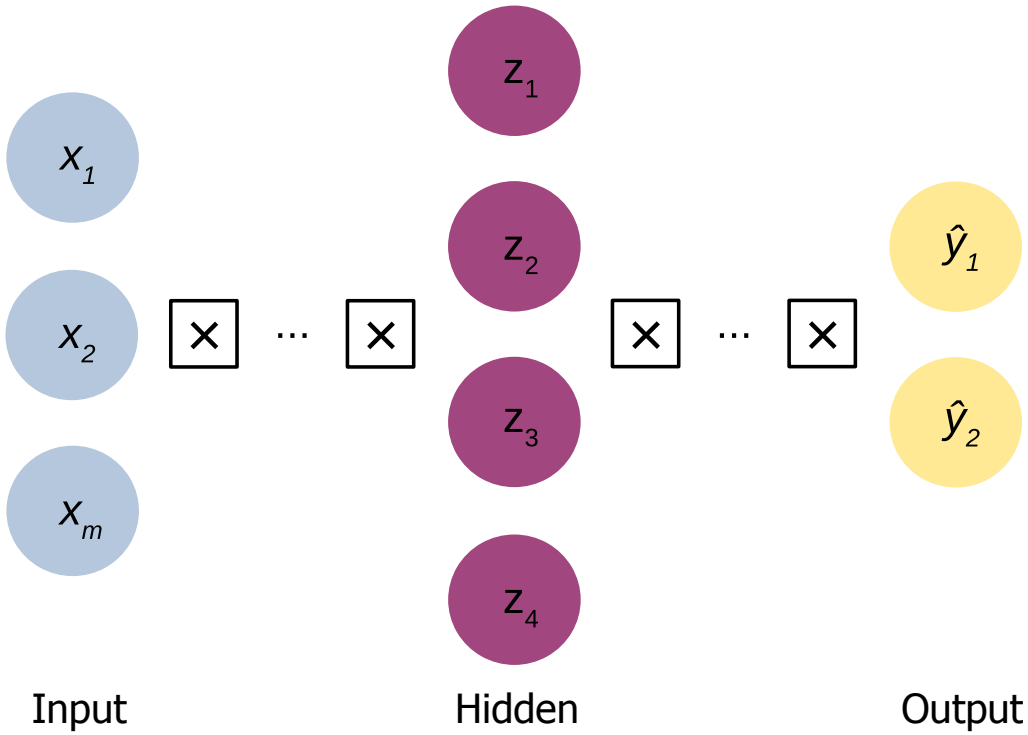
Single Layer Neural Network



Neural Network: Fully Connected Layer



Deep Neural Network



Perceptron – linear combination of the inputs, the bias and non-linearity

UNIVERSITI SAINS MALAYSIA

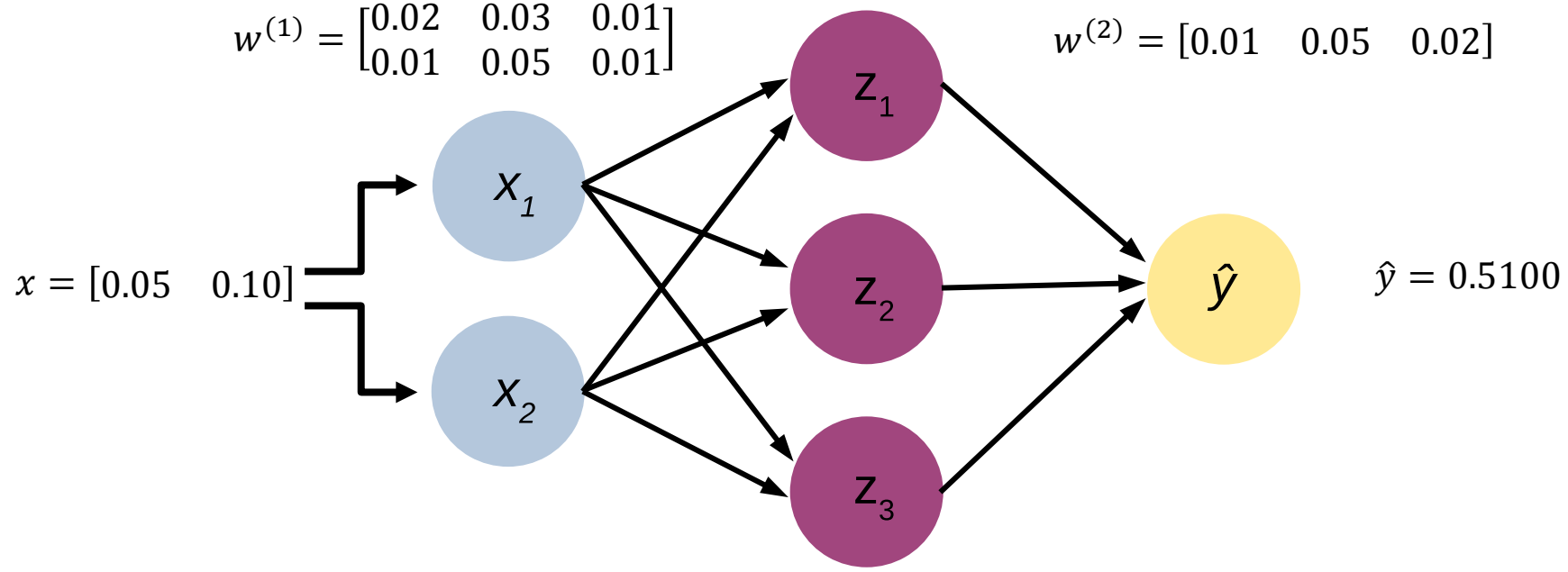
Training Neural Networks

Training Neural Networks

$$a^{(2)} = [0.5005 \quad 0.5016 \quad 0.5003]$$

$$w^{(1)} = \begin{bmatrix} 0.02 & 0.03 & 0.01 \\ 0.01 & 0.05 & 0.01 \end{bmatrix}$$

$$w^{(2)} = [0.01 \quad 0.05 \quad 0.02]$$



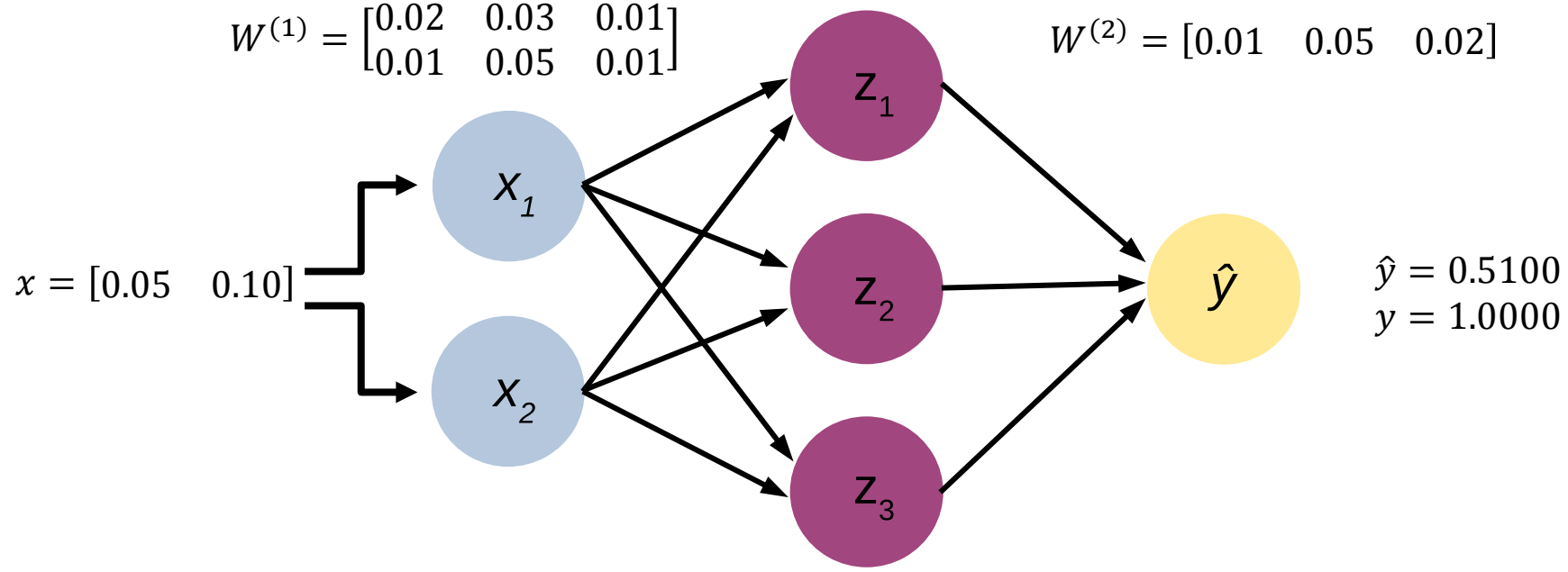
$$\hat{y} = 0.5100$$

Training Neural Networks

$$a^{(2)} = [0.5005 \quad 0.5016 \quad 0.5003]$$

$$W^{(1)} = \begin{bmatrix} 0.02 & 0.03 & 0.01 \\ 0.01 & 0.05 & 0.01 \end{bmatrix}$$

$$W^{(2)} = [0.01 \quad 0.05 \quad 0.02]$$



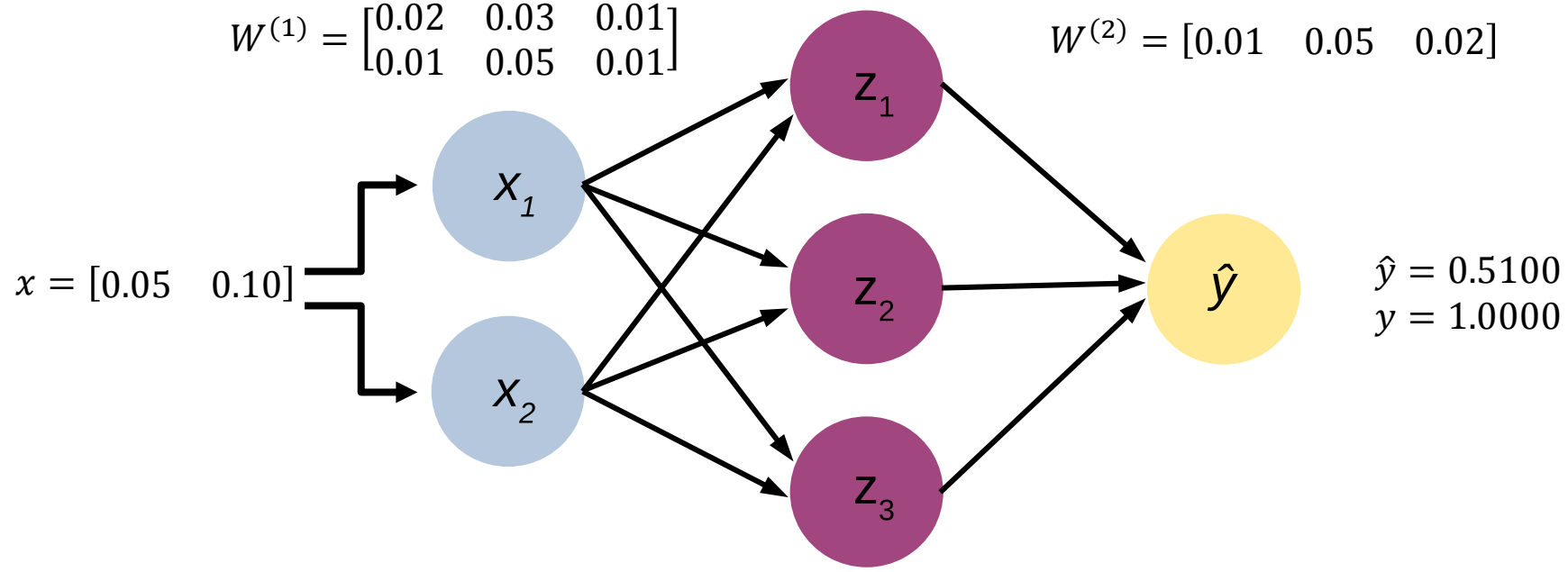
$$\hat{y} = 0.5100$$
$$y = 1.0000$$

Training Neural Networks

$$a^{(2)} = [0.5005 \quad 0.5016 \quad 0.5003]$$

$$W^{(1)} = \begin{bmatrix} 0.02 & 0.03 & 0.01 \\ 0.01 & 0.05 & 0.01 \end{bmatrix}$$

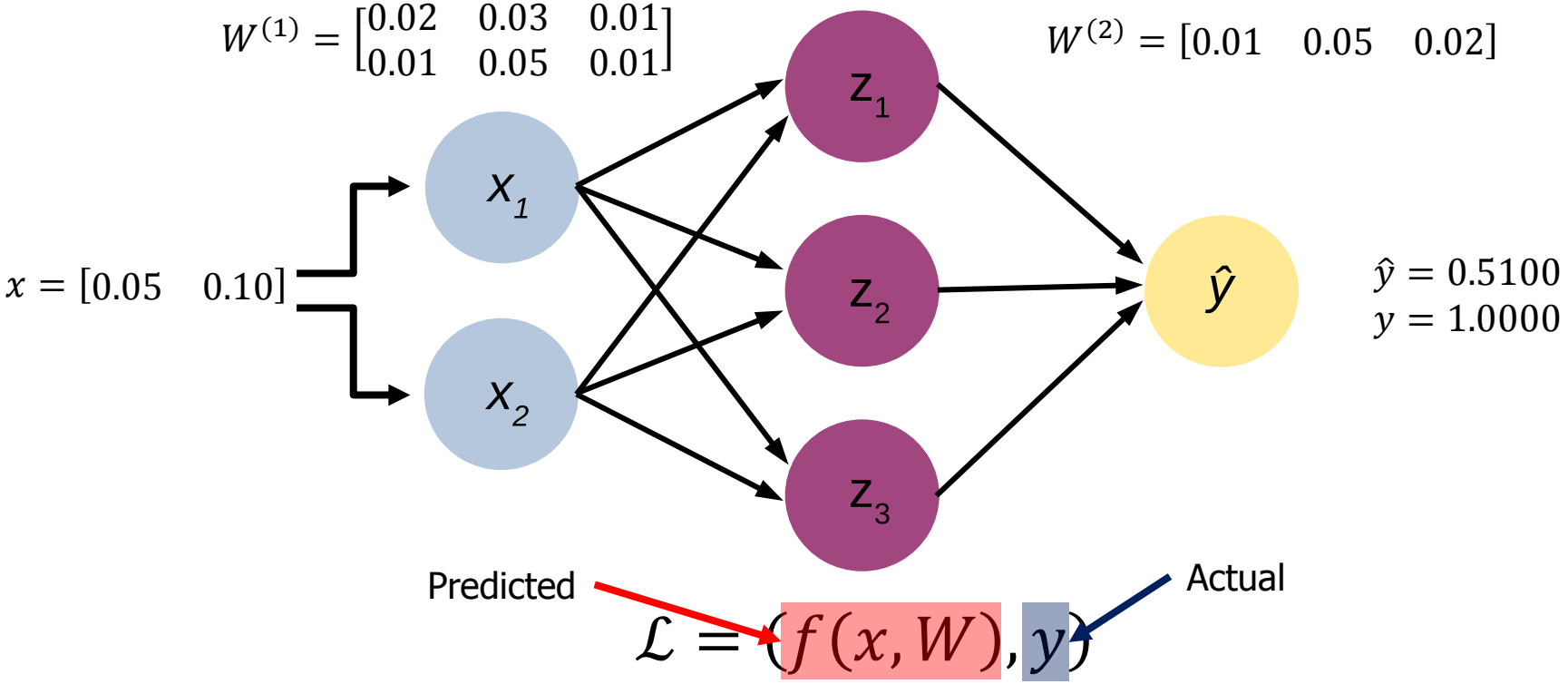
$$W^{(2)} = [0.01 \quad 0.05 \quad 0.02]$$



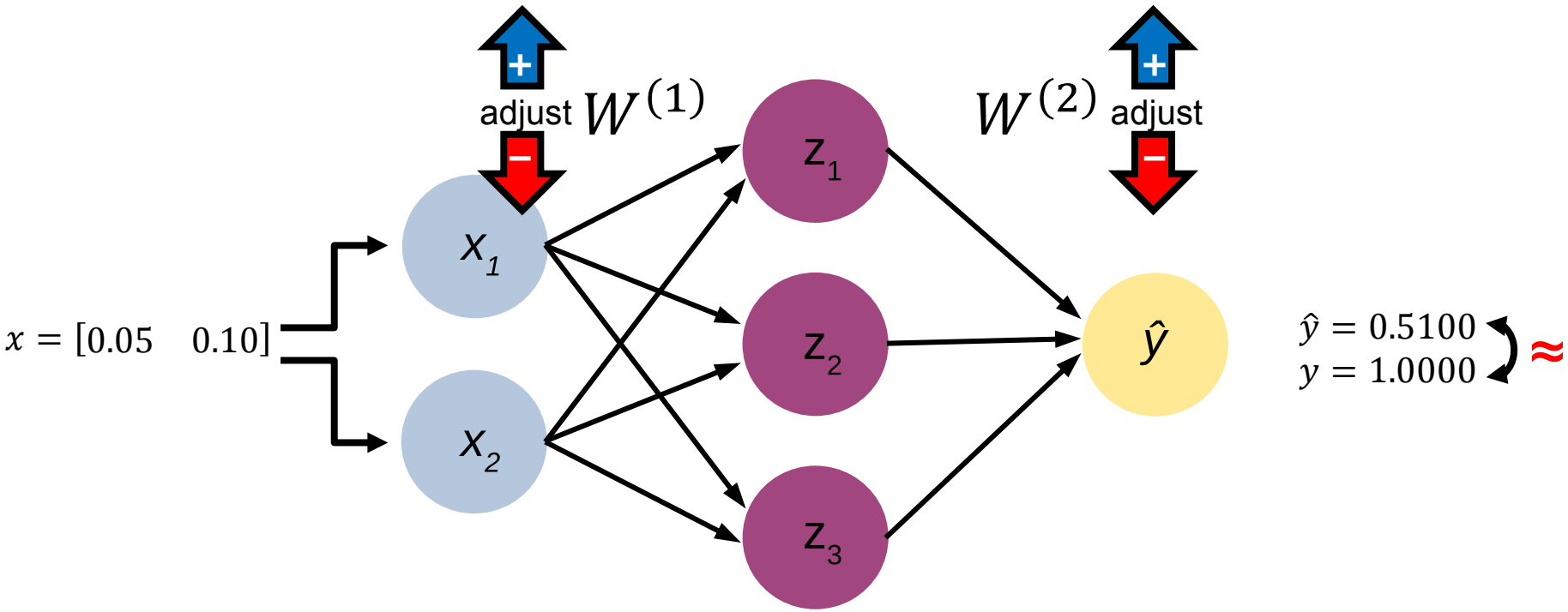
$$\hat{y} = 0.5100$$
$$y = 1.0000$$

Training Neural Networks: Loss

$$a^{(2)} = [0.5005 \quad 0.5016 \quad 0.5003]$$

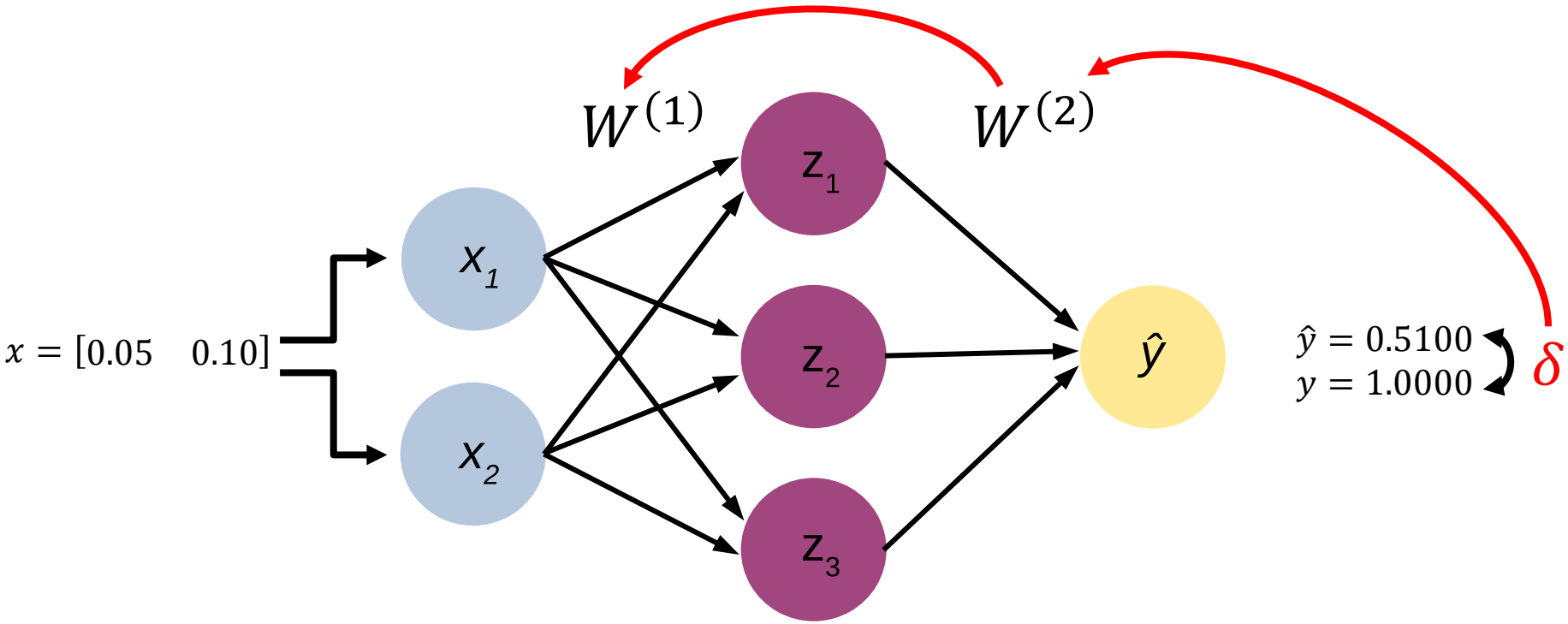


Training Neural Networks: Loss



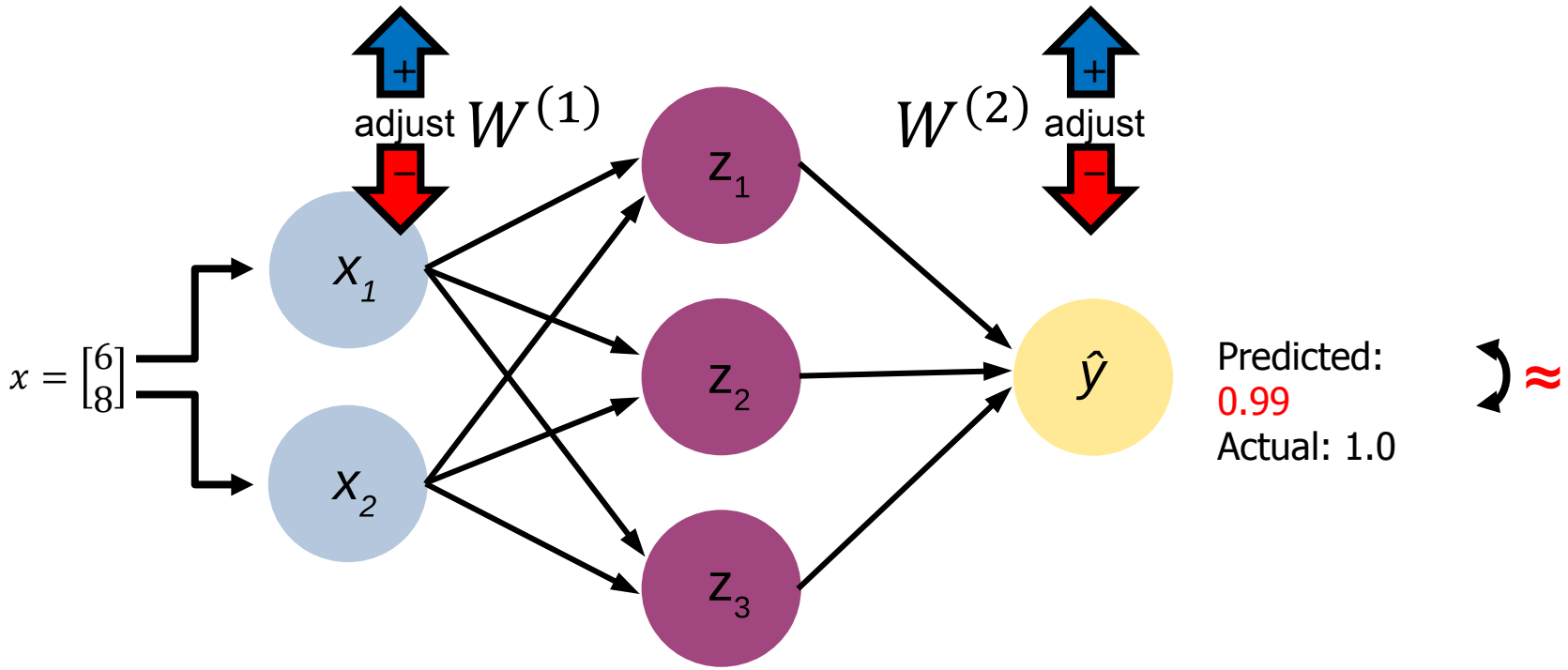
$$\mathcal{L} = (f(x, W), y)$$

Training Neural Networks: Backpropagation



$$\mathcal{L} = (f(x, W), y)$$

Training the Model: Loss



$$\mathcal{L} = (f(x, W), y)$$

Backpropagation

- Let $\mathcal{L} = \frac{1}{2}(\hat{y} - y)^2$ (squared error)
- Compute the partial derivatives (gradients) with respect to each weight
- To compute partial derivatives with respect to $W^{(2)}$
- $\frac{\partial \mathcal{L}}{\partial W^{(2)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial W^{(2)}}$
- To calculate partial derivatives with respect to $W^{(2)}$
- $\frac{\partial \mathcal{L}}{\partial W^{(1)}} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(3)}} \cdot \frac{\partial z^{(3)}}{\partial W^{(1)}}$
- $\frac{\partial z^{(3)}}{\partial W^{(1)}} = \frac{\partial z^{(3)}}{\partial a^{(2)}} \cdot \frac{\partial a^{(2)}}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial W^{(1)}}$

Regularization

Regularization

- Techniques to make the predictive models perform well on new inputs
  - The model generalize well outside the training data
- Trading increased bias for reduced variance
- Variance is reduced to make the model not sensitive to small fluctuation in training data
- Predictive models generalize well outside the training data

Early Stopping

- A technique that stops the training algorithm before overfitting occurs
- Make use of a set of data (validation set) to determine if the model is overfitting or not
- Store a copy of the model parameters every time the error on the validation set is reduced
- When the training algorithm terminates, return the parameters (with the lowest validation set error) rather than the latest parameters

Let n be the number of steps

Let p be the "patience", the number of times to observe worsening validation set error before giving up

Let θ_o be the initial parameters

$\theta = \theta_o$, $i = 0$, $j = 0$, $v = \infty$

while $j < p$ **do**

update θ by running the training algorithm for n steps

$i \leftarrow i + n$

$v' \leftarrow \text{ValidationSetError}(\theta)$

if $v' < v$ **then**

$j \leftarrow 0$

$\theta^* \leftarrow \theta$

$i^* \leftarrow i$

$v \leftarrow v'$

else

$j \leftarrow j + 1$

end if

end while

Early Stopping

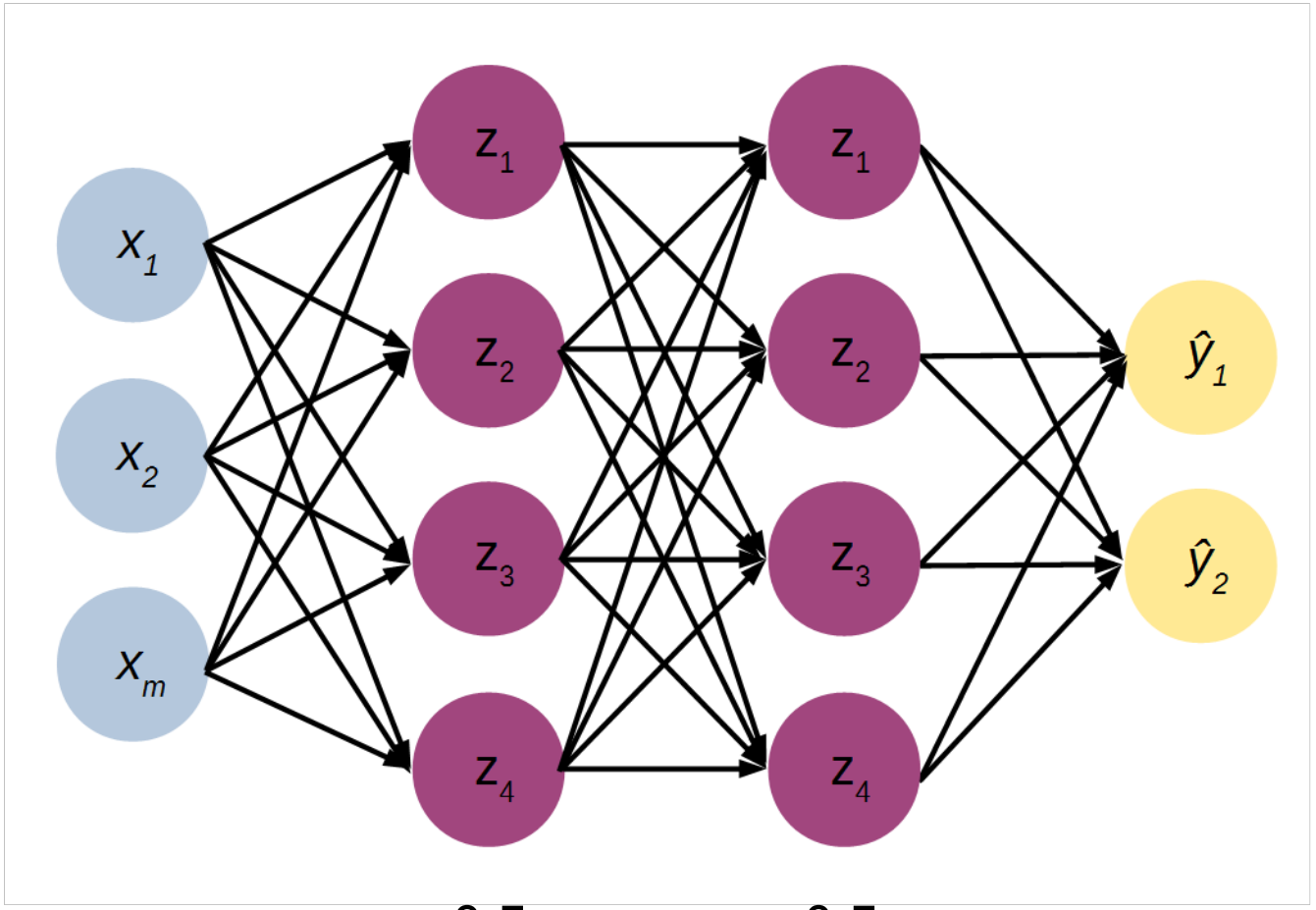
- Validation set evaluation is done periodically during training
- Need to maintain a copy of the best parameters

- Create a small validation set
- Evaluate the validation set error less frequently
- Storing the parameters has little effect on the total training time
  - Parameters are not frequently stored and never read during training
  - Only the best parameters are stored in disk drive

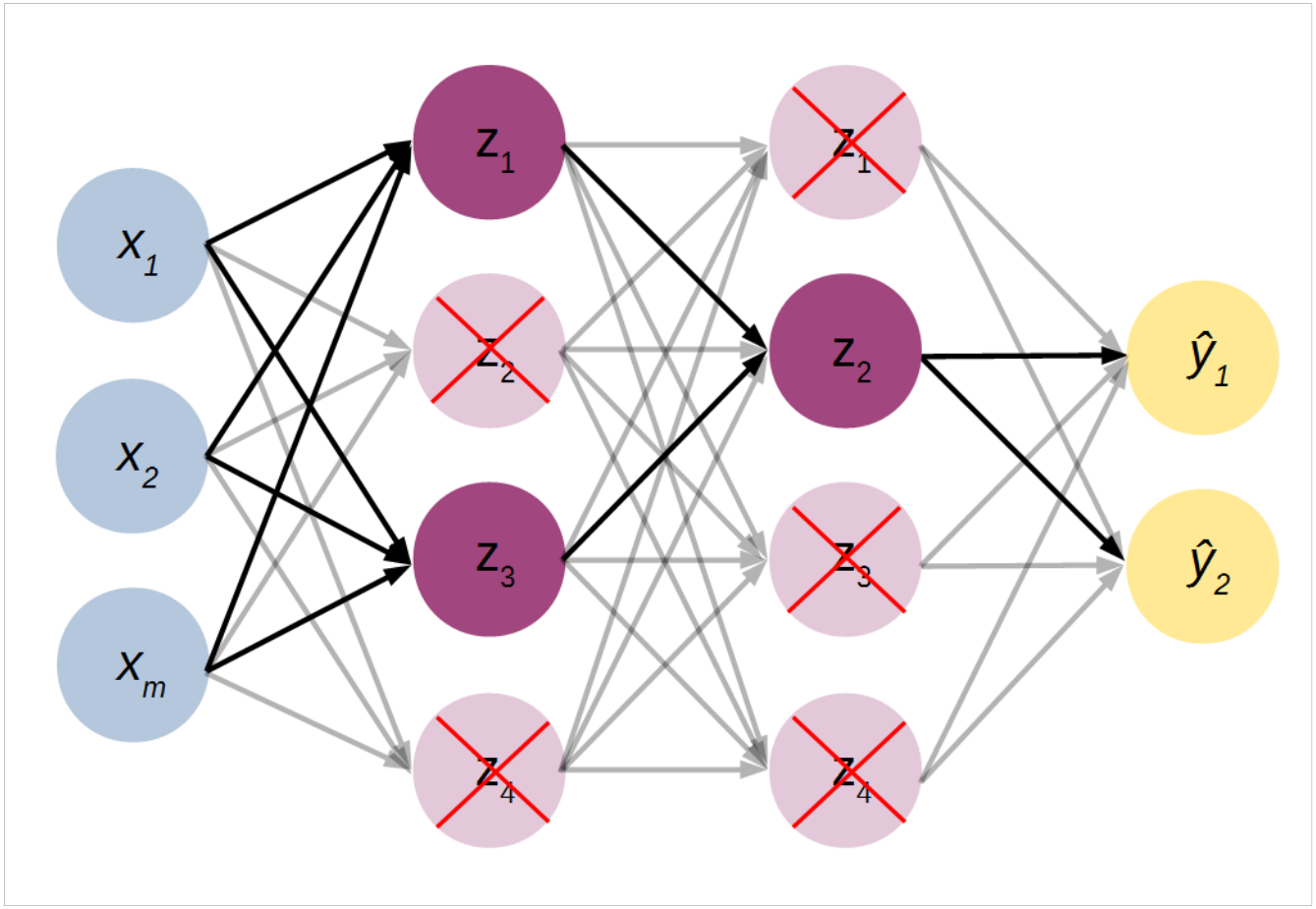
Dropout Regularization

- A regularization technique that dropping out neurons randomly in a neural network
- Each hidden layer is given a probability to remove some of its neurons
- Neurons of the input layer can also be dropped
- All the ingoing and outgoing links of the neurons are removed as well
- Perform training on the diminished network

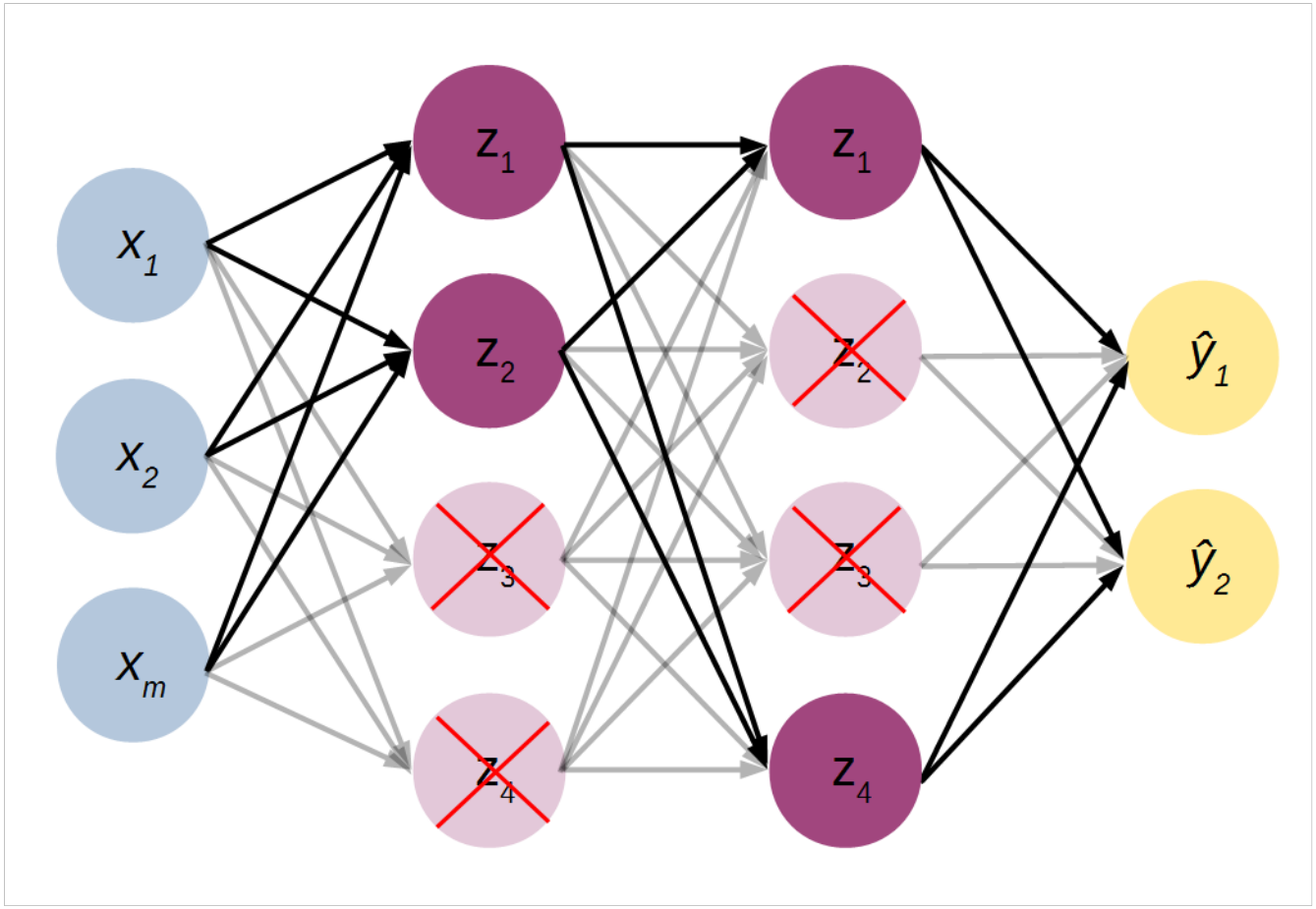
Dropout Regularization



Dropout Regularization



Dropout Regularization



Dropout Regularization

- The neurons cannot co-adapt to other neurons because they cannot expect all the neurons will always be present since neurons are stochastically removed
- The neurons will be forced to learn good representation of the data while not relying on other neurons

Dropout Regularization

- The neurons cannot co-adapt to other neurons because they cannot expect all the neurons will always be present since neurons are stochastically removed
- The neurons will be forced to learn good representation of the data while not relying on other neurons
- Dropout regularization has the effect of producing an ensemble of neural networks
- An ensemble model will always perform at least as well as any of its members thus reducing generalization errors

UNIVERSITI SAINS MALAYSIA

End