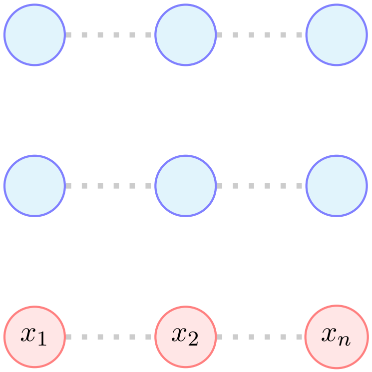
# Module 4.1: Feedforward Neural Networks (a.k.a. multilayered network of neurons)

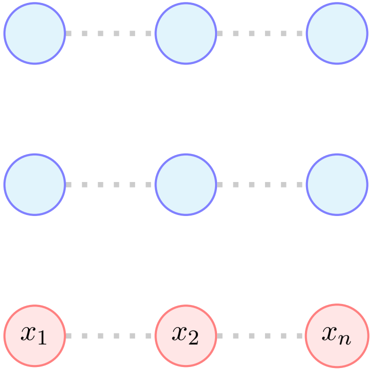
- The input to the network is an **n**-dimensional vector

- The input to the network is an **n**-dimensional vector

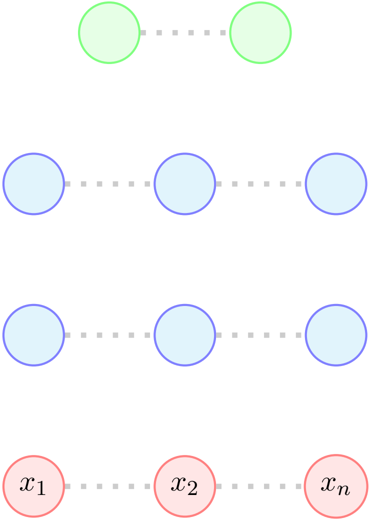
- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each

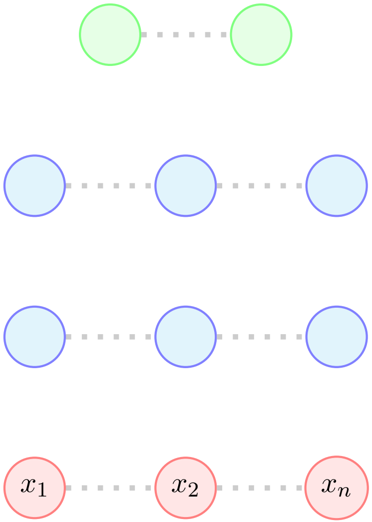




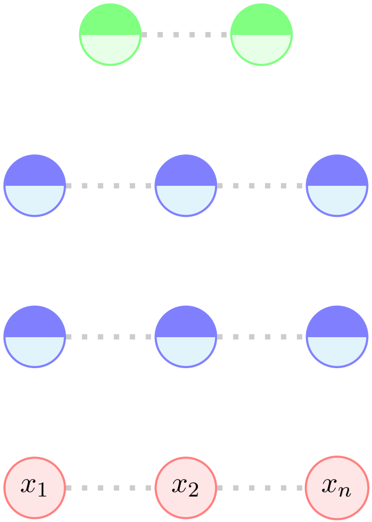

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)



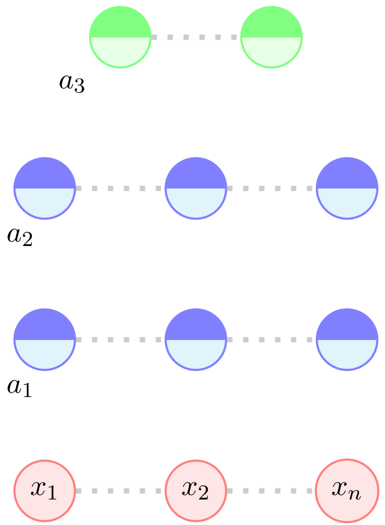

- The input to the network is an $\mathbf{n}$-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having $\mathbf{n}$ neurons each
- Finally, there is one output layer containing $\mathbf{k}$ neurons (say, corresponding to $\mathbf{k}$ classes)



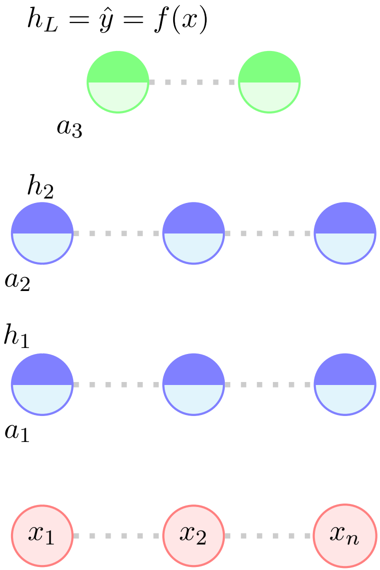

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts :



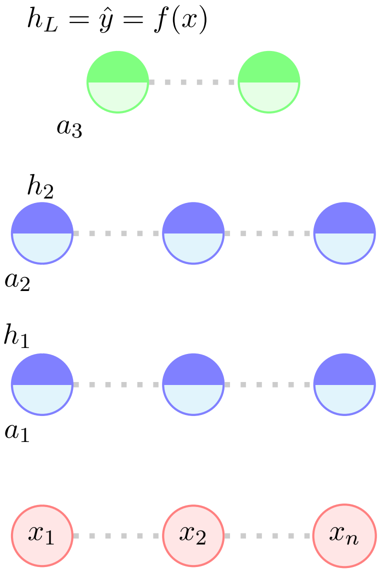

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts :



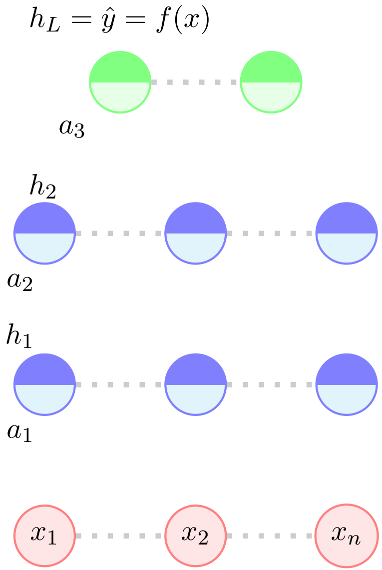

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation



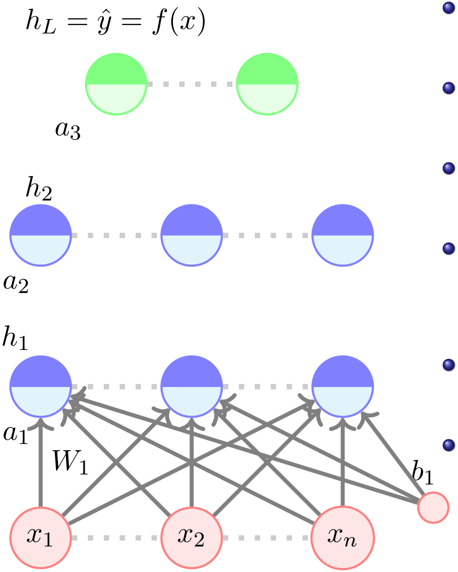

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation



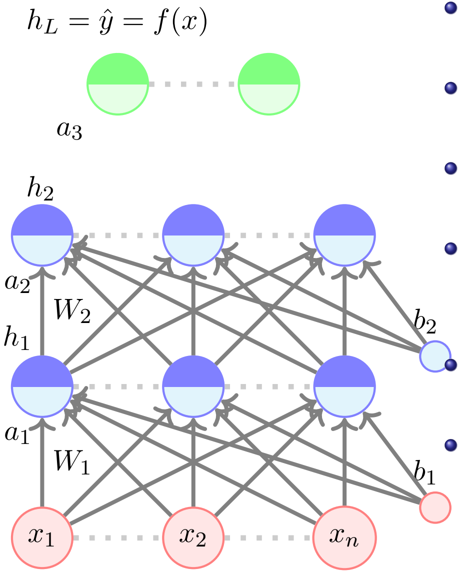

- The input to the network is an **n**-dimensional vector
- The network contains **L − 1** hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation ($a_i$ and $h_i$ are vectors)



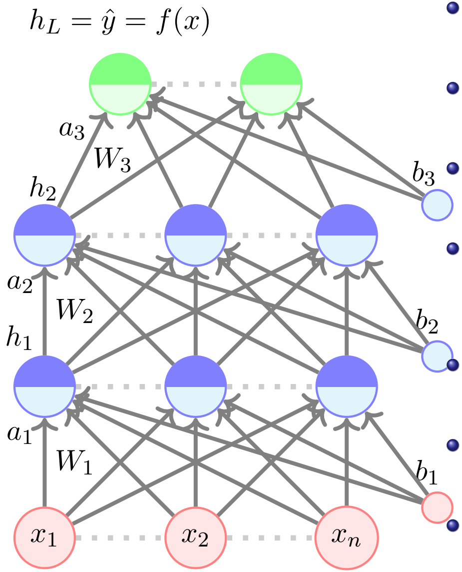

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation ($a_i$ and $h_i$ are vectors)
- The input layer can be called the 0-th layer and the output layer can be called the $(L)$-th layer



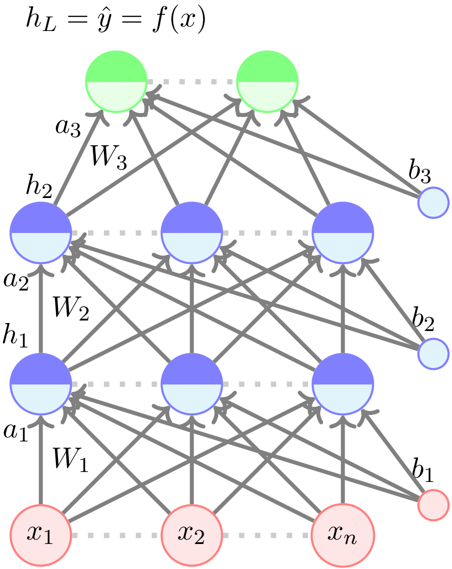

- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation ($a_i$ and $h_i$ are vectors)
- The input layer can be called the 0-th layer and the output layer can be called the $(L)$-th layer
- $W_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$ are the weight and bias between layers $i-1$ and $i$ $(0 < i < L)$




- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation ($a_i$ and $h_i$ are vectors)
- The input layer can be called the 0-th layer and the output layer can be called the $(L)$-th layer
- $W_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$ are the weight and bias between layers $i-1$ and $i$ $(0 < i < L)$

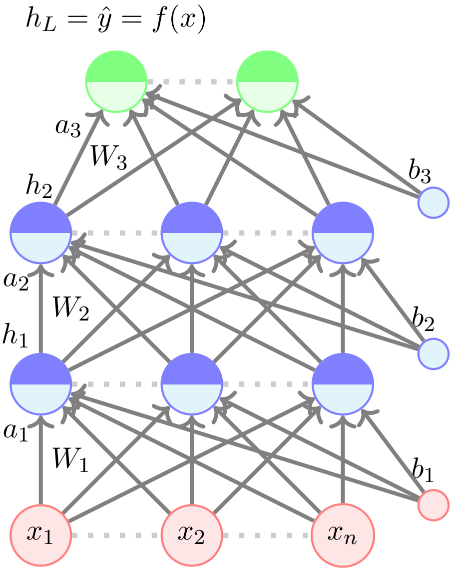



- The input to the network is an **n**-dimensional vector
- The network contains $\mathbf{L-1}$ hidden layers (2, in this case) having **n** neurons each
- Finally, there is one output layer containing **k** neurons (say, corresponding to **k** classes)
- Each neuron in the hidden layer and output layer can be split into two parts : pre-activation and activation ($a_i$ and $h_i$ are vectors)
- The input layer can be called the 0-th layer and the output layer can be called the $(L)$-th layer
- $W_i \in \mathbb{R}^{n \times n}$ and $b_i \in \mathbb{R}^n$ are the weight and bias between layers $i-1$ and $i$ $(0 < i < L)$
- $W_L \in \mathbb{R}^{n \times k}$ and $b_L \in \mathbb{R}^k$ are the weight and bias between the last hidden layer and the output layer ($L = 3$ in this case)




- The pre-activation at layer $i$ is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

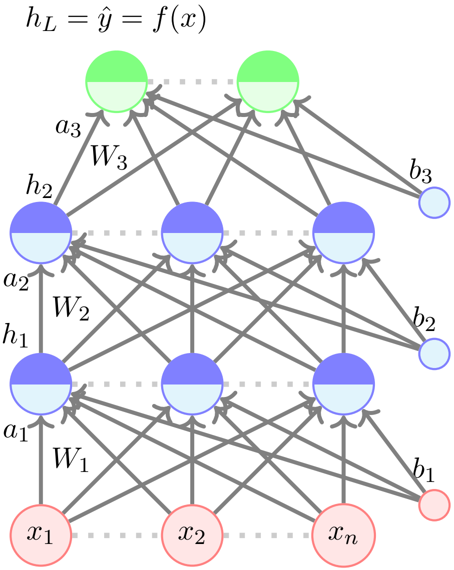



- The pre-activation at layer $i$ is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

- The activation at layer $i$ is given by

$$h_i(x) = g(a_i(x))$$



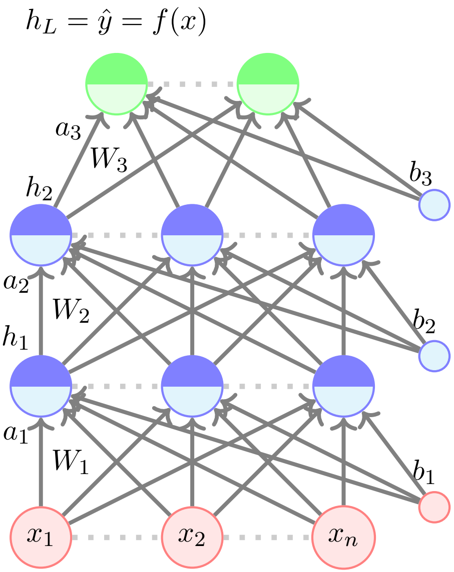

- The pre-activation at layer $i$ is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

- The activation at layer $i$ is given by

$$h_i(x) = g(a_i(x))$$

where $g$ is called the activation function (for example, logistic, tanh, linear, *etc.*)




- The pre-activation at layer $i$ is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

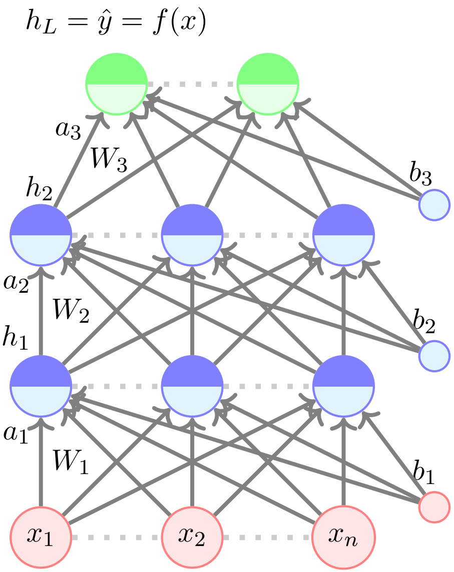
- The activation at layer $i$ is given by

$$h_i(x) = g(a_i(x))$$

where $g$ is called the activation function (for example, logistic, tanh, linear, *etc.*)

- The output at layer $i$ is given by

$$f(x) = h_{L+1}(x) = O(a_{L+1}(x))$$




- The pre-activation at layer $i$ is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

- The activation at layer $i$ is given by

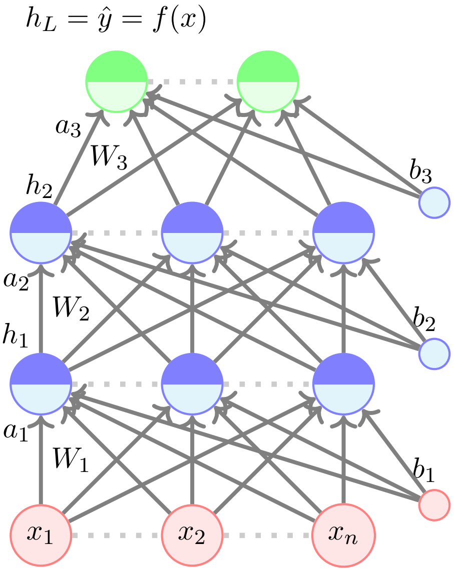
$$h_i(x) = g(a_i(x))$$

where $g$ is called the activation function (for example, logistic, tanh, linear, *etc.*)

- The output at layer $i$ is given by

$$f(x) = h_{L+1}(x) = O(a_{L+1}(x))$$

where $O$ is the output activation function (for example, softmax, linear, *etc.*)




- The pre-activation at layer $i$ is given by

$$a_i(x) = b_i + W_i h_{i-1}(x)$$

- The activation at layer $i$ is given by

$$h_i(x) = g(a_i(x))$$

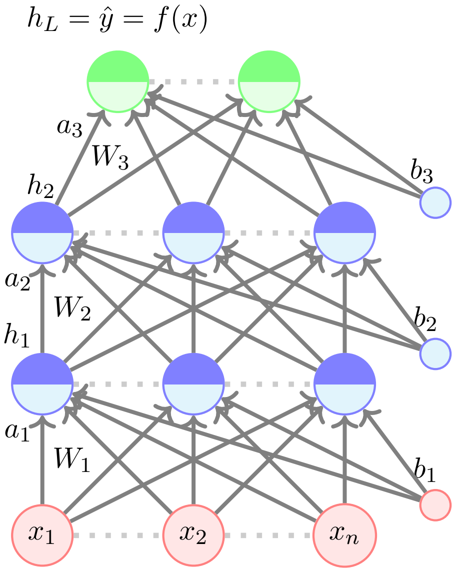
where $g$ is called the activation function (for example, logistic, tanh, linear, *etc.*)

- The output at layer $i$ is given by

$$f(x) = h_{L+1}(x) = O(a_{L+1}(x))$$

where $O$ is the output activation function (for example, softmax, linear, *etc.*)

- To simplify notation we will refer to $a_i(x)$ as $a_i$ and $h_i(x)$ as $h_i$




- The pre-activation at layer $i$ is given by

$$a_i = b_i + W_i h_{i-1}$$

- The activation at layer $i$ is given by

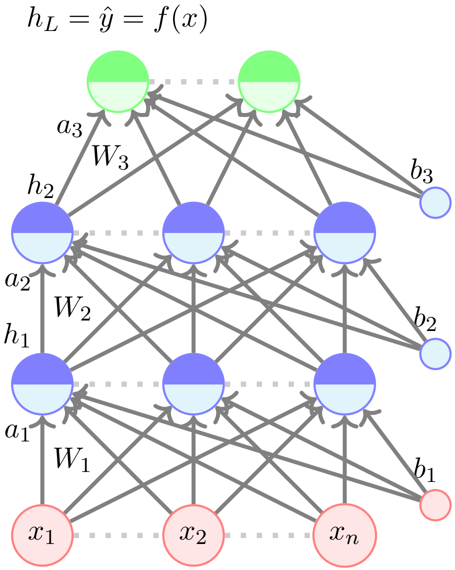
$$h_i = g(a_i)$$

where $g$ is called the activation function (for example, logistic, tanh, linear, *etc.*)

- The activation at layer $i$ is given by

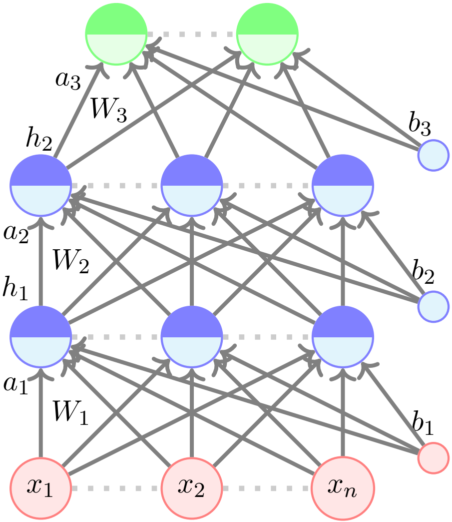
$$f(x) = h_{L+1} = O(a_{L+1})$$

where $O$ is the output activation function (for example, softmax, linear, *etc.*)



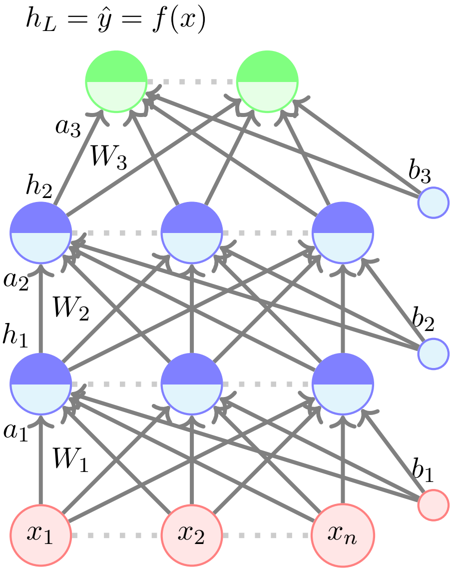

- **Data:** $\{x_i, y_i\}_{i=1}^N$

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




- **Data:** $\{x_i, y_i\}_{i=1}^N$
- **Model:**

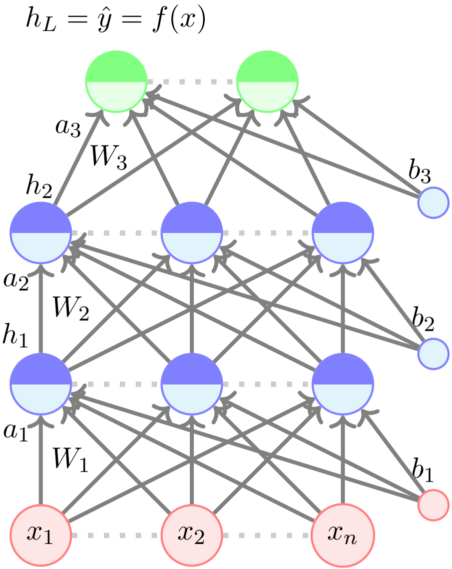
$$\hat{y}_i = f(x_i) = O(W^3 g(W^2 g(W^1 x + b_1) + b_2) + b_3)$$




- **Data:** $\{x_i, y_i\}_{i=1}^{N}$
- **Model:**

$$\hat{y}_i = f(x_i) = O(W^3 g(W^2 g(W^1 x + b_1) + b_2) + b_3)$$

- **Parameters:**
  $\theta = W_1, .., W_L, b_1, b_2, ..., b_L (L = 3)$

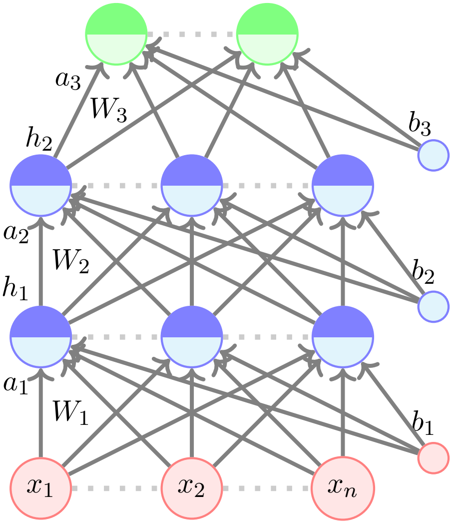



- **Data:** $\{x_i, y_i\}_{i=1}^{N}$
- **Model:**

$$\hat{y}_i = f(x_i) = O(W^3 g(W^2 g(W^1 x + b_1) + b_2) + b_3)$$

- **Parameters:**
  $\theta = W_1, .., W_L, b_1, b_2, ..., b_L (L = 3)$
- **Algorithm:** Backpropagation

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




- **Data:** $\{x_i, y_i\}_{i=1}^N$
- **Model:**

$$\hat{y}_i = f(x_i) = O(W^3 g(W^2 g(W^1 x + b_1) + b_2) + b_3)$$

- **Parameters:**
  $\theta = W_1, .., W_L, b_1, b_2, ..., b_L (L = 3)$
- **Algorithm:** Backpropagation
- **Objective/Loss/Error function:** Say,

$$min \ \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

$$\textit{In general, } min \ \mathscr{L}(\theta)$$

where $\mathscr{L}(\theta)$ is some function of the parameters