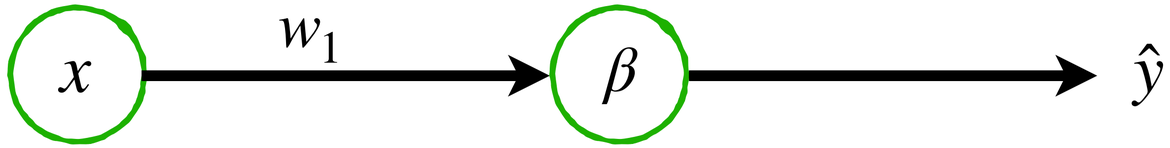
Neural Networks

Forward & Back Propagation

Rahul Singh
rsingh@arrsingh.com

A Simple Neural Network

$$\hat{y} = \beta + w_1 x$$

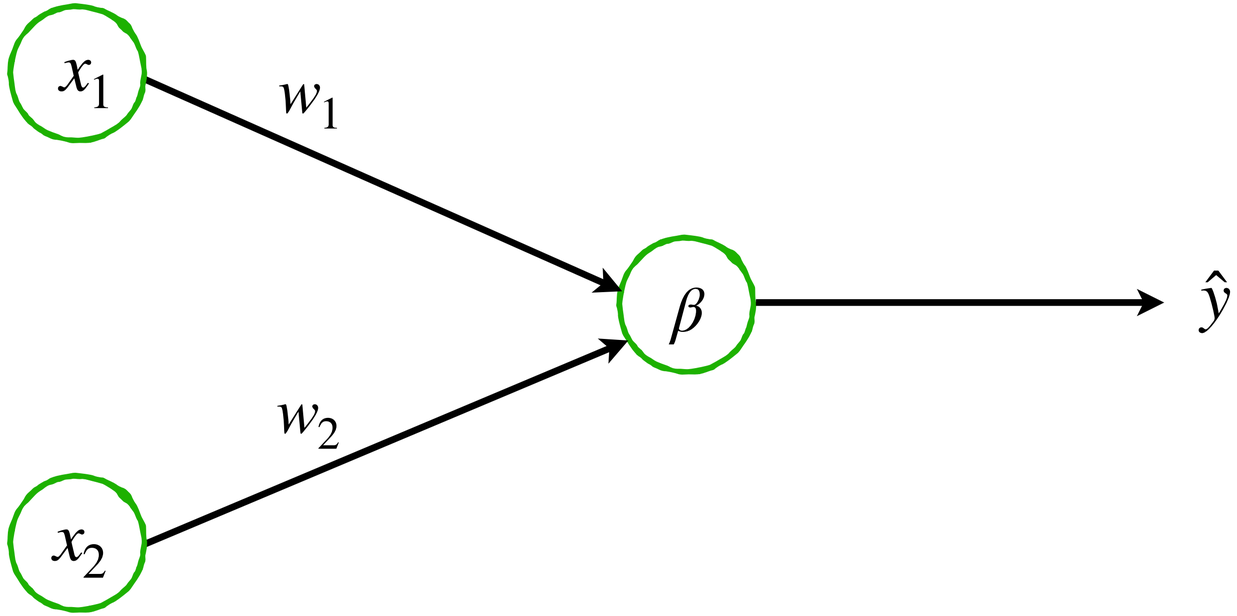


This is the simplest possible neural network

Multiply the input (x) with the Weight (w_1) and add the Bias (β) to compute the output ($\hat{y}$)

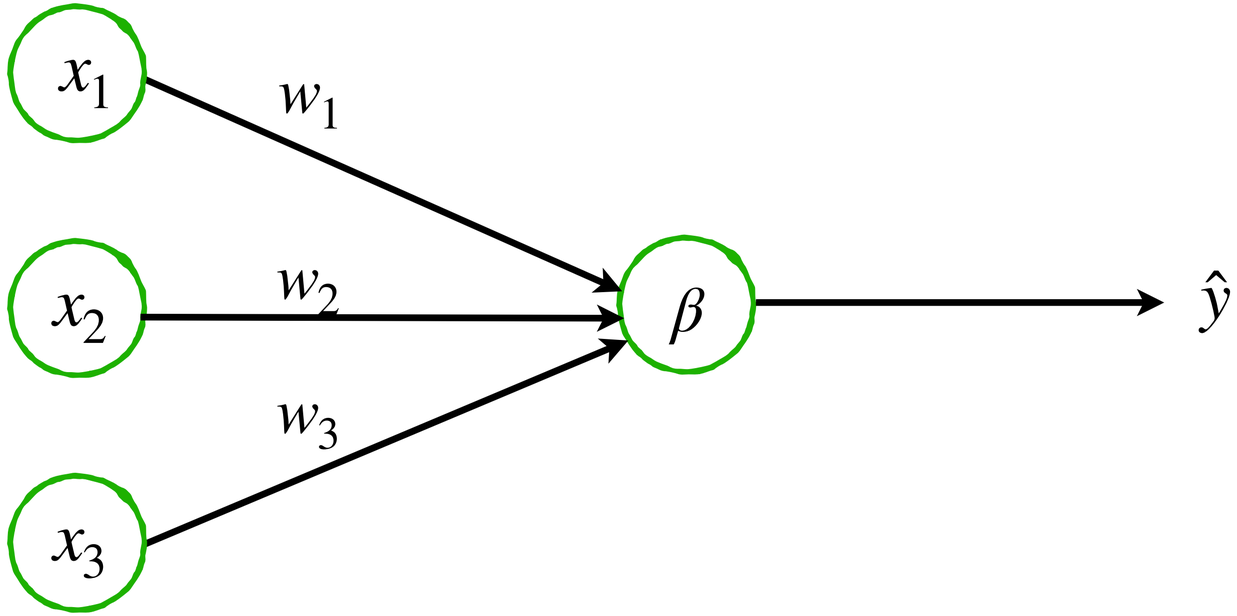
A Simple Neural Network

$$\hat{y} = \beta + w_1 x_1 + w_2 x_2$$

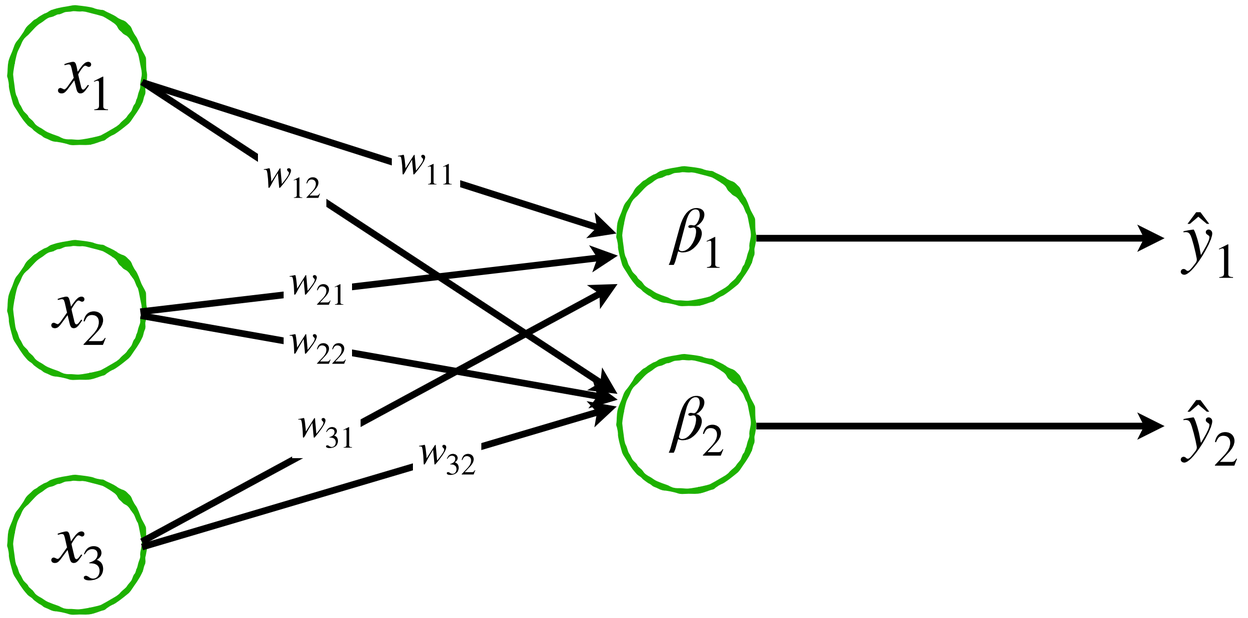


A Simple Neural Network

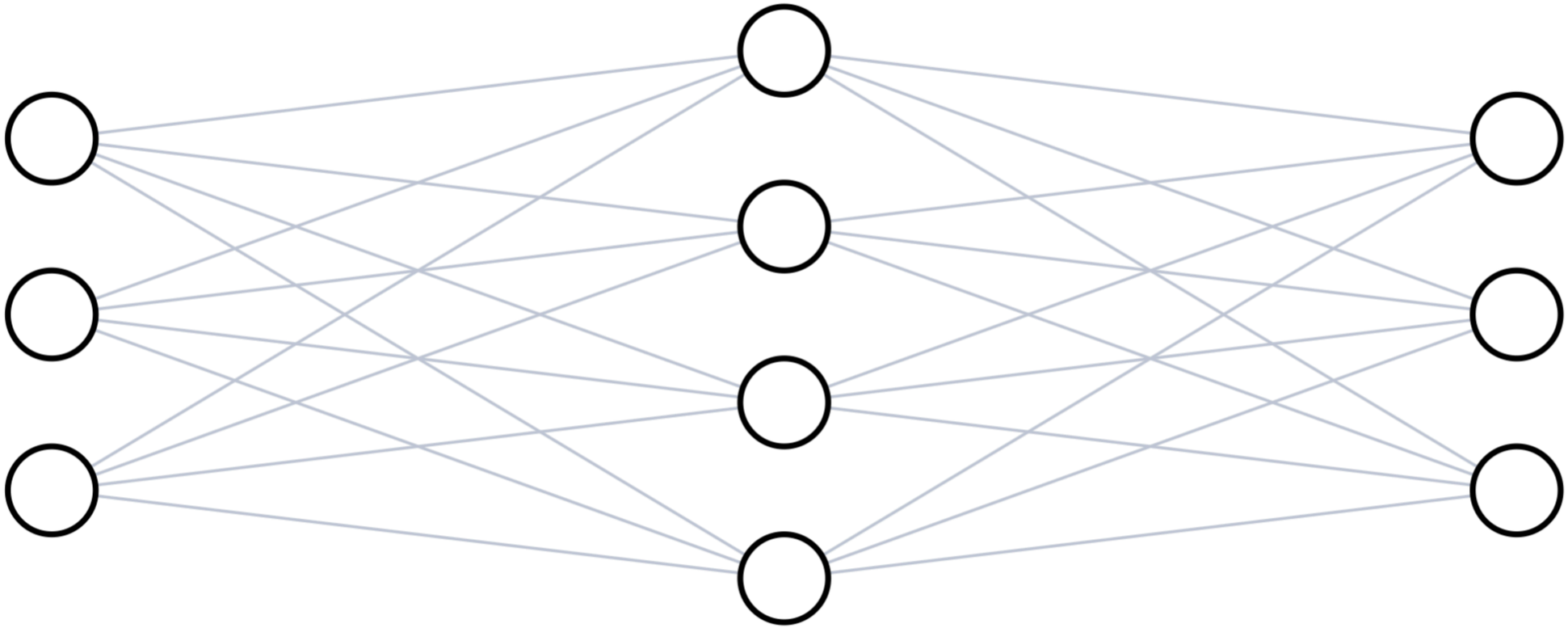
$$\hat{y} = \beta + w_1 x_1 + w_2 x_2 + w_3 x_3$$



A Simple Neural Network



Neural Networks

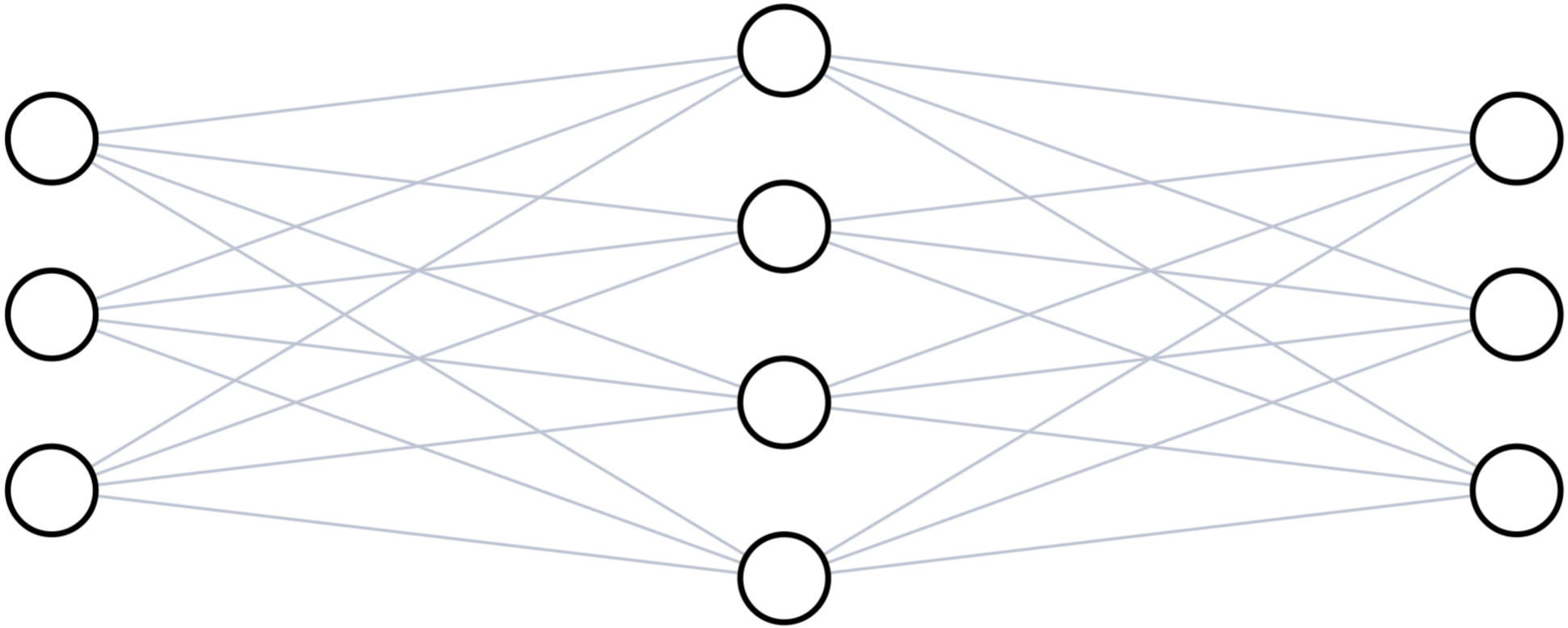


Input Layer (L_0)

Hidden Layer (L_1)

Output Layer (L_2)

Neural Networks



n_0
Neurons

n_1
Neurons

n_2
Neurons

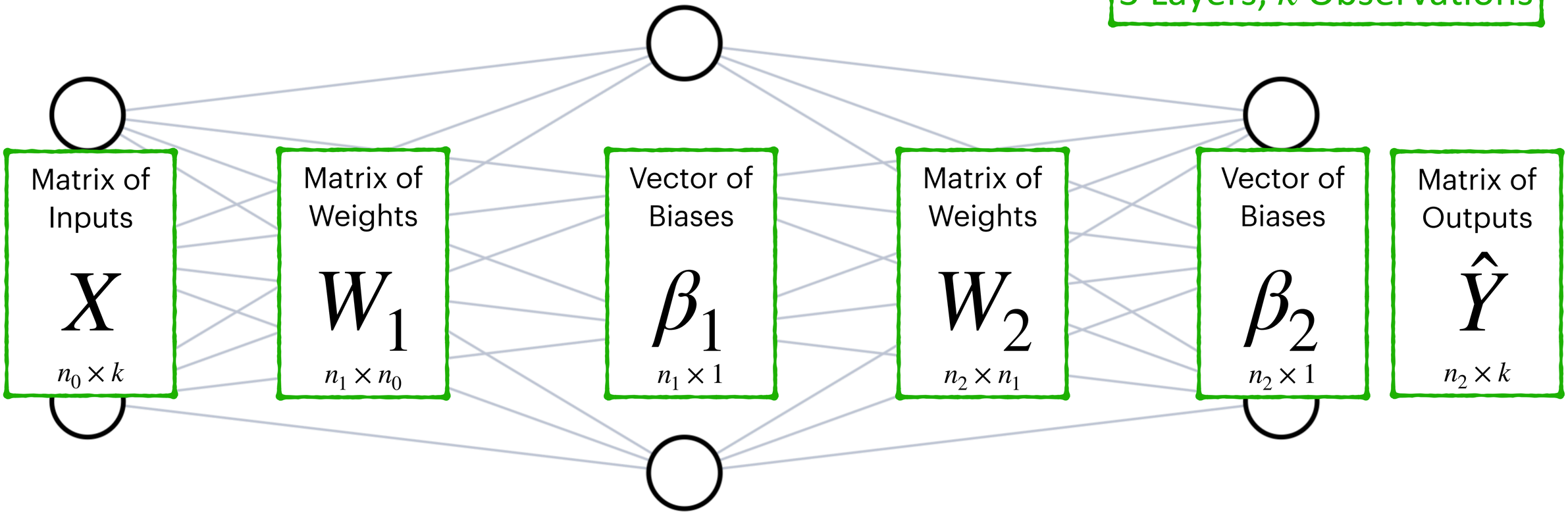
Input Layer (L_0)

Hidden Layer (L_1)

Output Layer (L_2)

Neural Networks

3 Layers, k Observations



n_0 Neurons

Input Layer (L_1)

n_1 Neurons

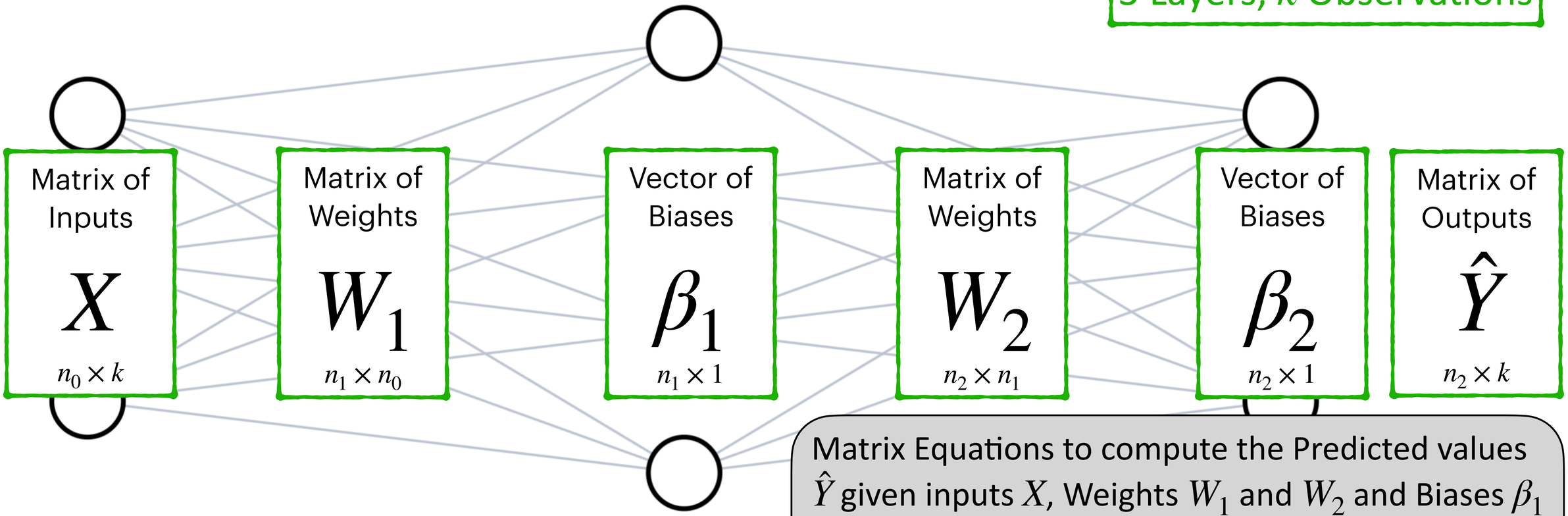
Hidden Layer (L_2)

n_2 Neurons

Output Layer (L_3)

Neural Networks

3 Layers, k Observations



n_0 Neurons

Input Layer (L_1)

n_1 Neurons

Hidden Layer

Matrix Equations to compute the Predicted values $\hat{Y}$ given inputs X, Weights W_1 and W_2 and Biases β_1 and β_2

$$Z_1 = f_1(W_1 X + \beta_1)$$

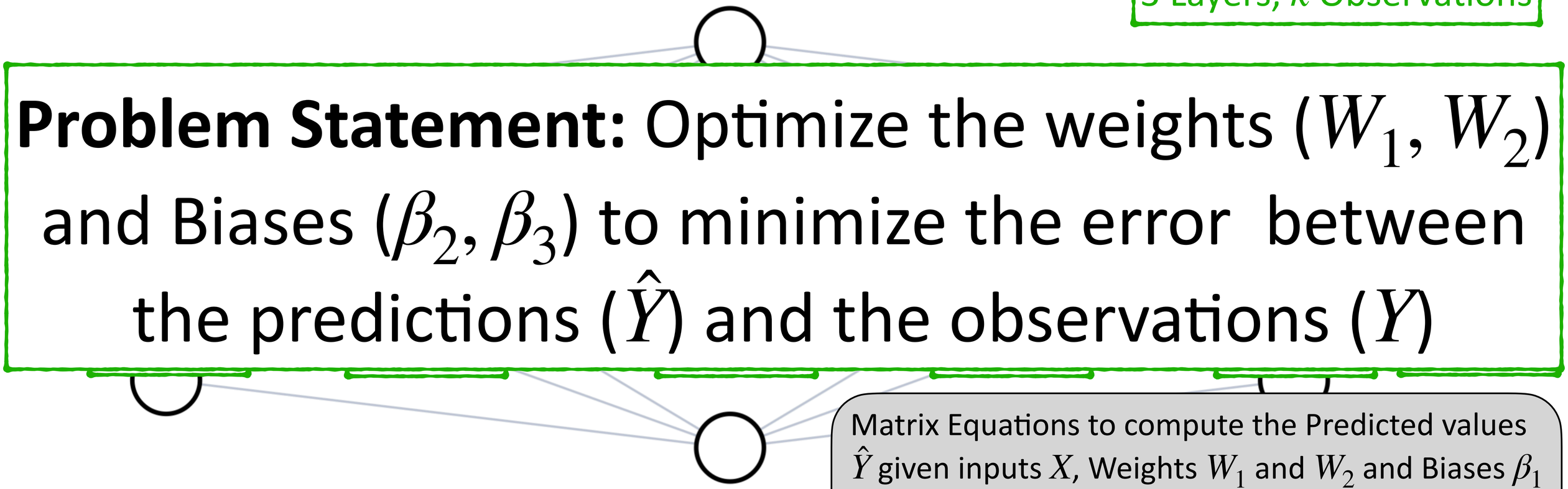
$$\hat{Y} = f_2(W_2 Z_1 + \beta_2)$$

$f_1(g)$ and $f_2(g)$ are the activation functions in Layers L_1 and L_2

Neural Networks

3 Layers, k Observations

Problem Statement: Optimize the weights (W_1, W_2) and Biases (β_2, β_3) to minimize the error between the predictions ($\hat{Y}$) and the observations (Y)



n_0 Neurons

Input Layer (L_1)

n_1 Neurons

Hidden Layer

Matrix Equations to compute the Predicted values $\hat{Y}$ given inputs X, Weights W_1 and W_2 and Biases β_1 and β_2

$$Z_1 = f_1(W_1 X + \beta_1)$$

$$\hat{Y} = f_2(W_2 Z_1 + \beta_2)$$

$f_1(g)$ and $f_2(g)$ are the activation functions in Layers L_1 and L_2

Neural Networks

Question: How do we train a Neural Network

(Hint: We'll use Gradient Descent)

Lets start with a simple example…

Neural Networks

A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

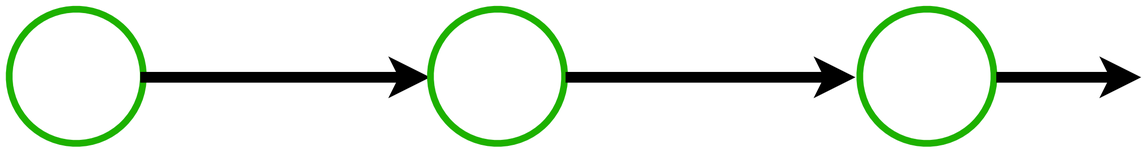
Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

We'll start with a simple model trained on only 1 observation and then build on it as we go…

Neural Networks

Lets start with a simple example…



A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

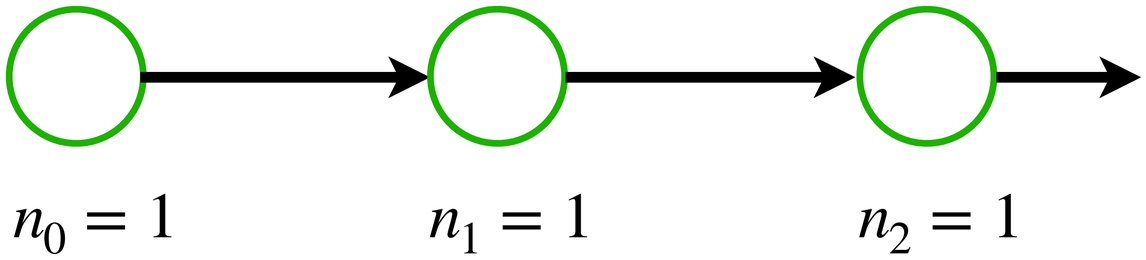
Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

We'll start with a simple 3 layer network…

Neural Networks

Lets start with a simple example…



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

We'll start with a simple 3 layer network…

A simple Neural Network…

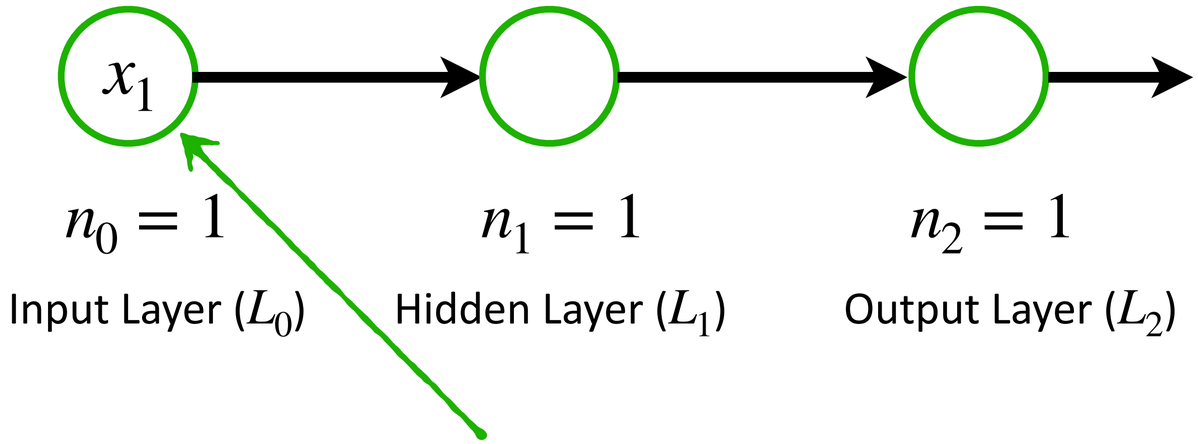
Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

Neural Networks

Lets start with a simple example…



x_1 represents the first neuron in the input layer

A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

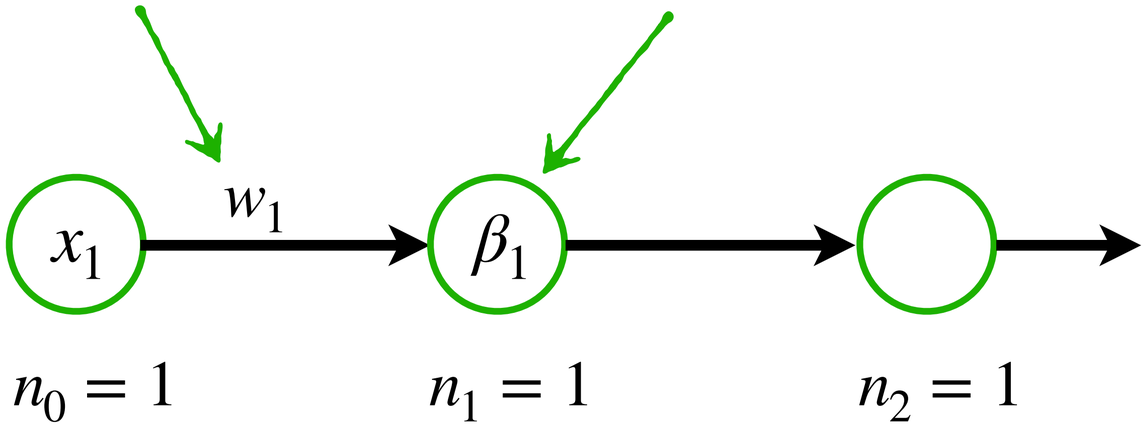
x_{11} represents the single observation for input neuron x_1

Neural Networks

Lets start with a simple example…

w_1 is the weight from L_0 to L_1

β_1 is the Bias on Neuron 1 of L_1



A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

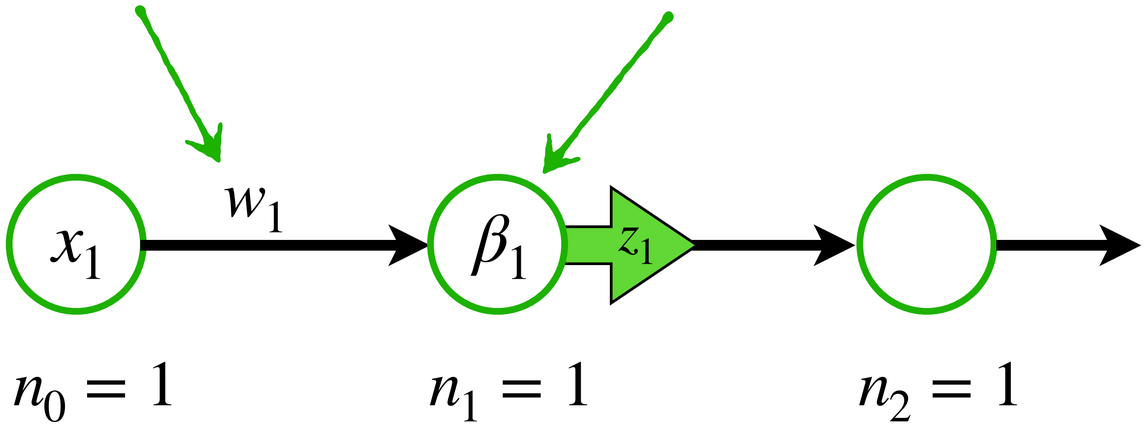
i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

Neural Networks

Lets start with a simple example…

w_1 is the weight from L_0 to L_1

β_1 is the Bias on Neuron 1 of L_1



A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

$$z_1 = f(w_1 x_{11} + \beta_1)$$

f is the activation function. To keep things simple we'll use the identity function $f(x) = x$ as the activation function

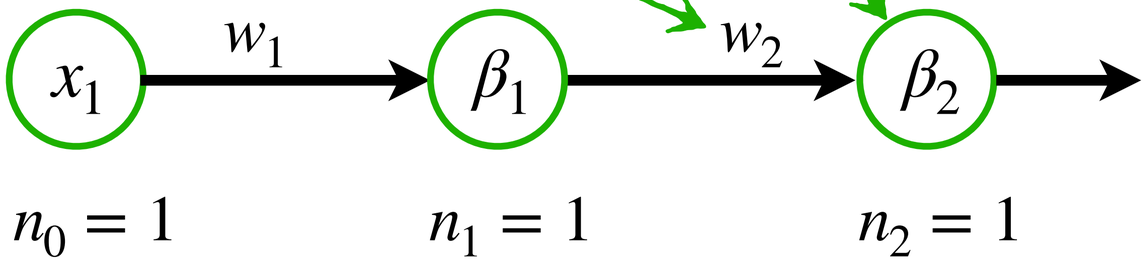
Multiply the weight with the input and add the bias…
… then pass it to the activation function

Neural Networks

Lets start with a simple example…

w_2 is the weight from L_1 to L_2

β_2 is the Bias on Neuron 1 of L_2



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

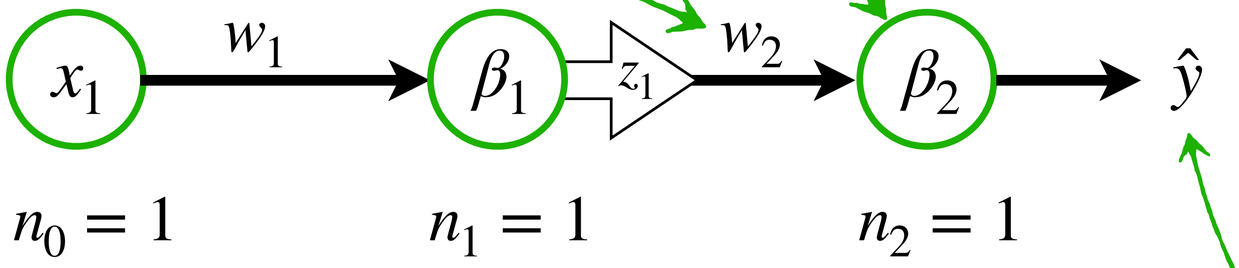
$f(x) = x$ is the activation function

Neural Networks

Lets start with a simple example…

w_2 is the weight from L_1 to L_2

β_2 is the Bias on Neuron 1 of L_2



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

Multiply the weight with the input and add the bias…
… then pass it to the activation function

A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

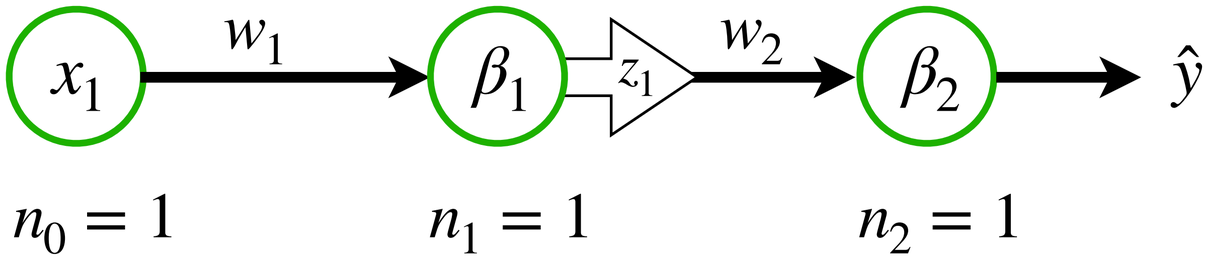
i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

$\hat{y}$ is the predicted value

$f(x) = x$ is the activation function

Neural Networks

Lets start with a simple example…



$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

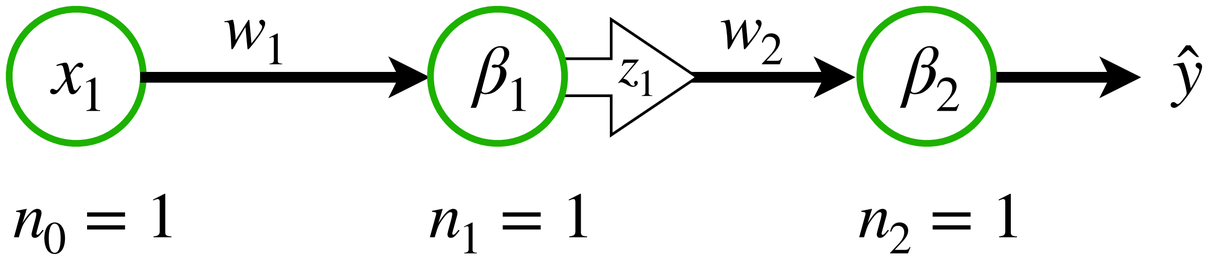
Cost Function

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

$f(x) = x$ is the activation function

Neural Networks

Lets start with a simple example…



$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

A simple Neural Network…

Simple Linear Regression Problem:
Predict the Sale Price ($\hat{y}$) of a House
Based on Number of Bedrooms (x)

Simple Training data (1 observation)

i	Bedrooms (*x*)	Sale Price (*y*)
1	3	$450,000

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

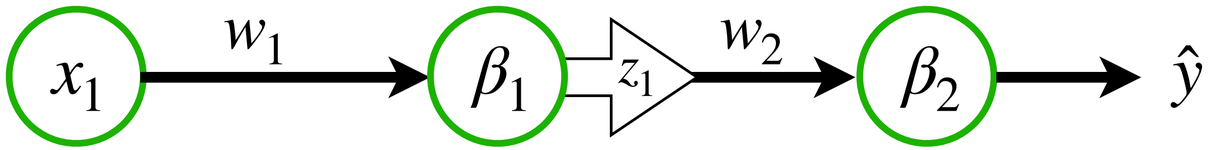
Since there is only one observation $cost = \frac{1}{2}(\hat{y} - y)^2$

$f(x) = x$ is the activation function

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

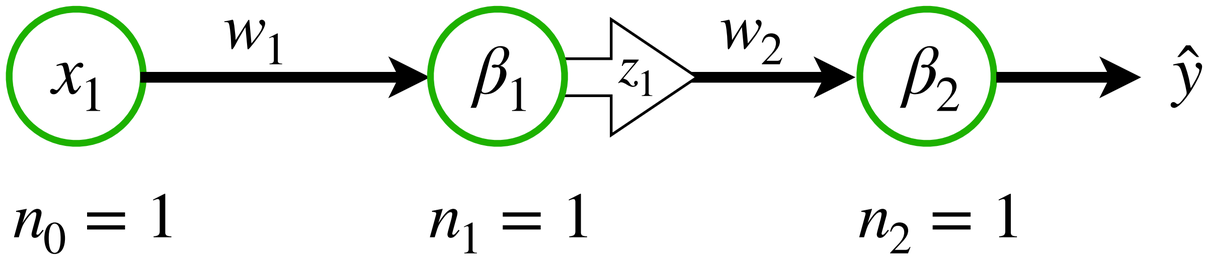
Neural Networks

Lets start with a simple example…

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

Problem Statement: Find the values of weights w_1, w_2 and biases β_1, β_2 to minimize the error between the predicted values ($\hat{y}$) and the observations (y)

Let's use Gradient Descent



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

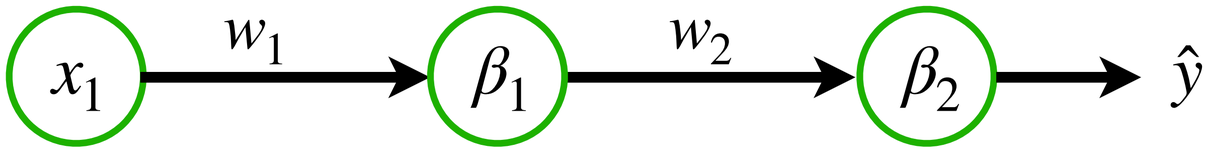
$f(x) = x$ is the activation function

Lets start with a simple example…

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

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

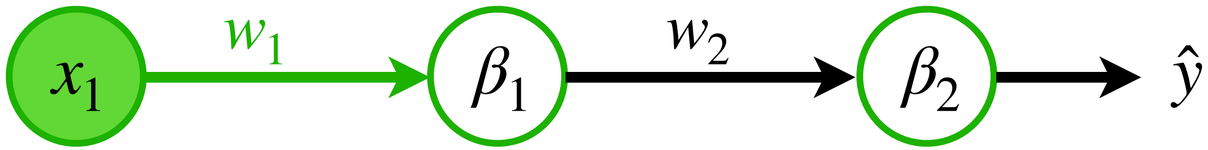
$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

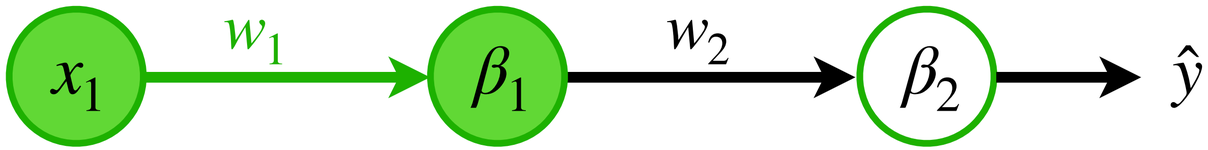
Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $w_1 x_{11}$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

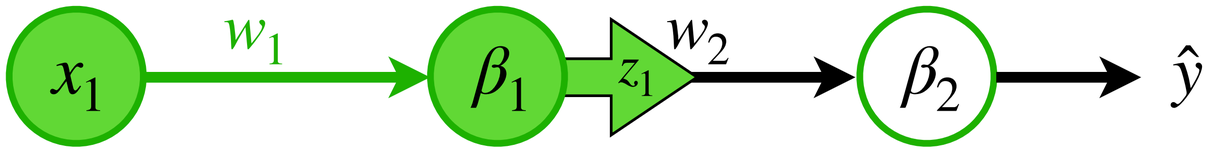
Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $w_1 x_{11} + \beta_1$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

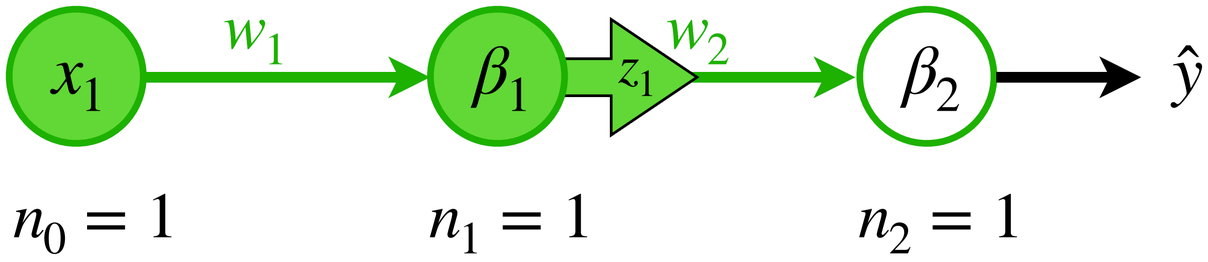
Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

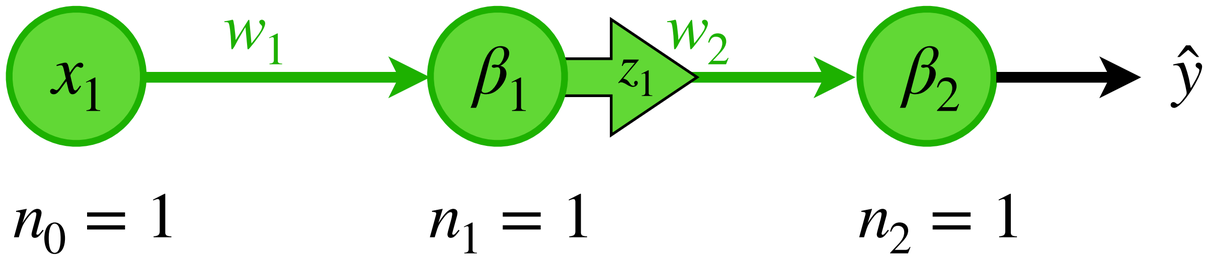
Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $w_2 z_1$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

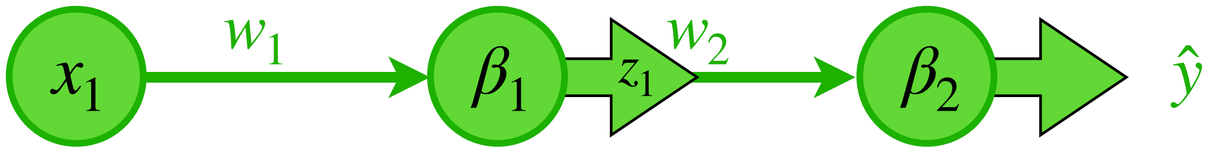
Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $w_2 z_1 + \beta_2$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

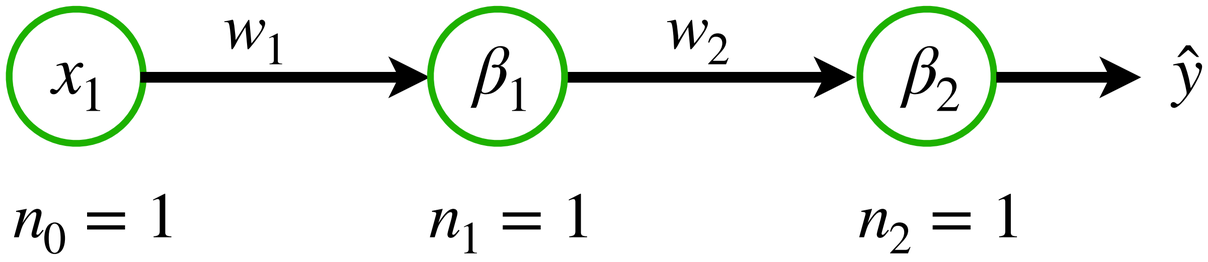
Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

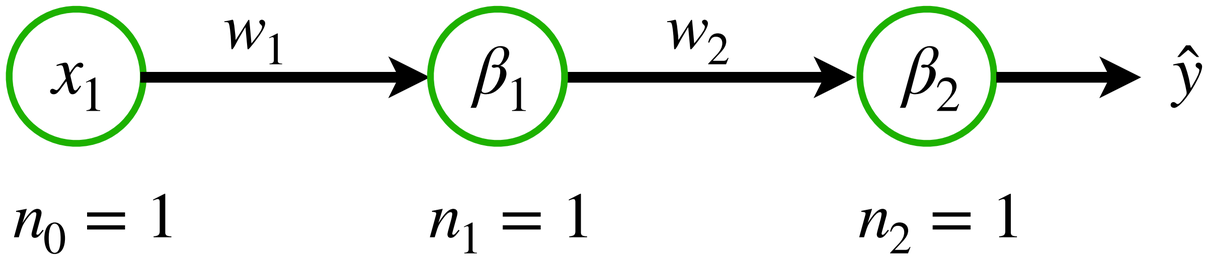
Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 … $\frac{\partial}{\partial \beta_2} cost$ $\frac{\partial}{\partial w_2} cost$

Neural Networks

Lets start with a simple example…

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



$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 … $\frac{\partial}{\partial \beta_2} cost$ $\frac{\partial}{\partial w_2} cost$

Step 3b: Compute step size…

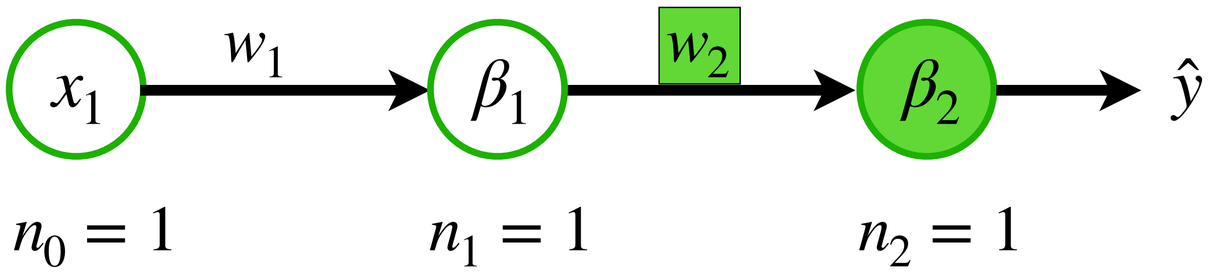
$$step\_size_{\beta_2} = \frac{\partial}{\partial \beta_2} cost \times learning\_rate$$

$$step\_size_{w_2} = \frac{\partial}{\partial w_2} cost \times learning\_rate$$

Neural Networks

Lets start with a simple example…

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



$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 … $\frac{\partial}{\partial \beta_2} cost$ $\frac{\partial}{\partial w_2} cost$

Step 3b: Compute step size…

$$step\_size_{\beta_2} = \frac{\partial}{\partial \beta_2} cost \times learning\_rate$$

$$step\_size_{w_2} = \frac{\partial}{\partial w_2} cost \times learning\_rate$$

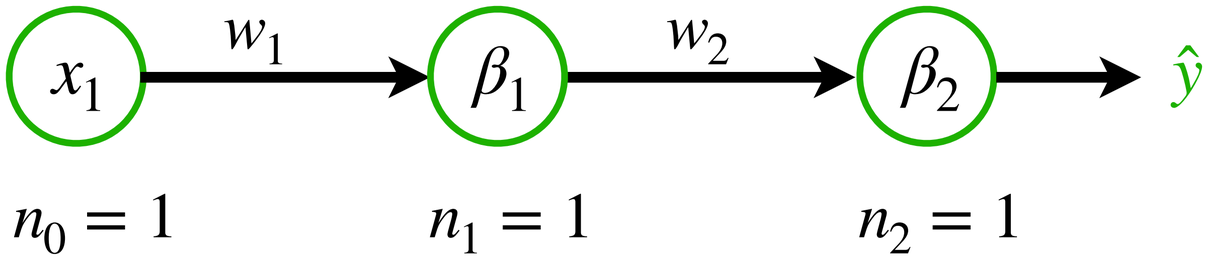
Step 3c: Compute new values for β_2 and w_2

$$\beta_2 = \beta_2 - step\_size_{\beta_2} \quad w_2 = w_2 - step\_size_{w_2}$$

Neural Networks

Lets start with a simple example…

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



$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

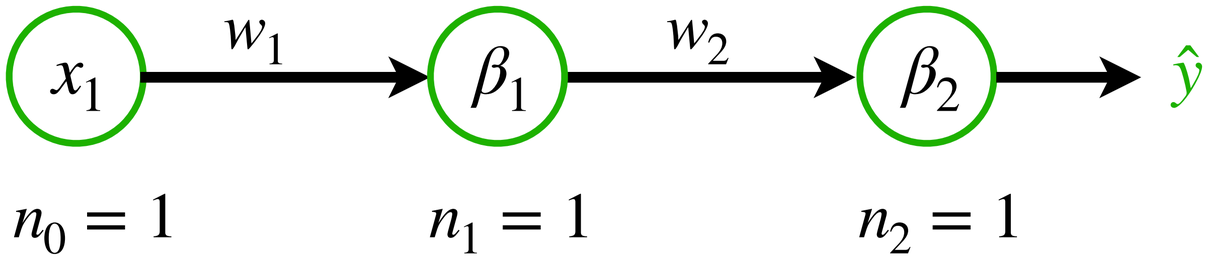
Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Neural Networks

Lets start with a simple example…

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



$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

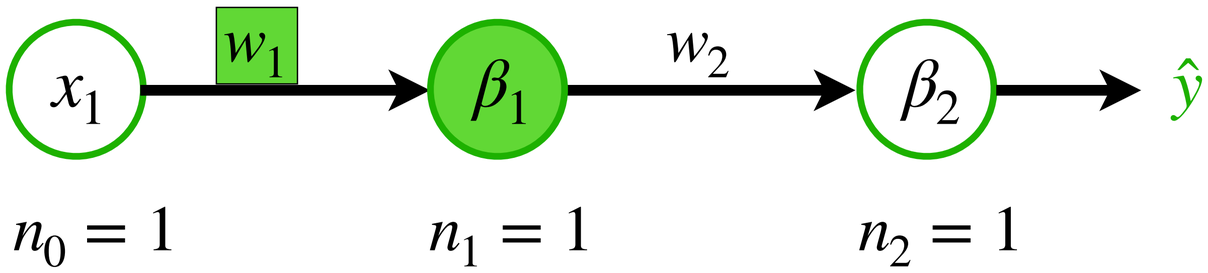
$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

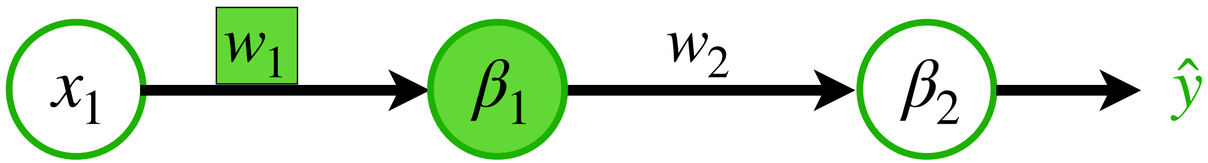
Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Neural Networks

Lets start with a simple example…

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



$n_0 = 1$ $n_1 = 1$ $n_2 = 1$

Input Layer (L_0) Hidden Layer (L_1) Output Layer (L_2)

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

Lets start with a simple example…

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

Gradient Descent

Step 1: Start with initial values for $w_1, w_2, \beta_1, \beta_2$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Gradient descent continues iteratively until either…

a) the gradient magnitude falls below a specified threshold,

b) the change in loss between iterations becomes negligible,

c) a maximum number of iterations is reached

In

$$z_1 = f(w_1 x_{11} + \beta_1)$$

$$\hat{y} = f(w_2 z_1 + \beta_2)$$

$f(x) = x$ is the activation function

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

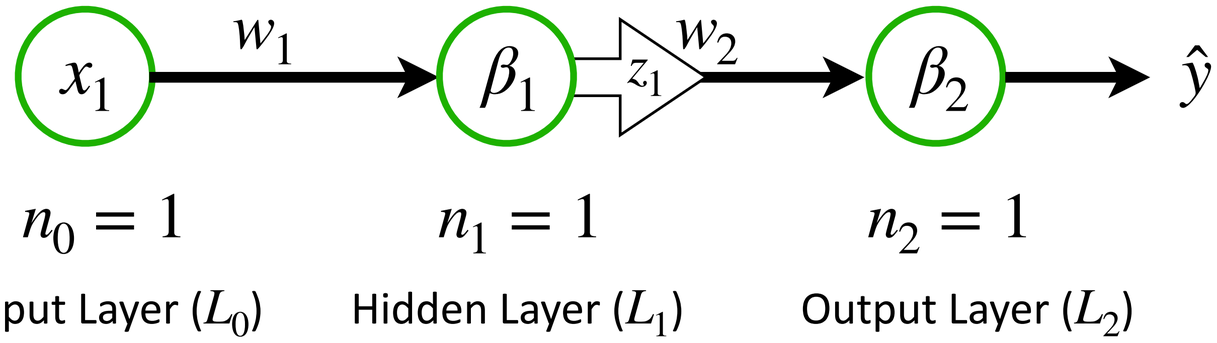
$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks



A neural network is trained using Gradient Descent. There are two passes in each iteration of Gradient Descent

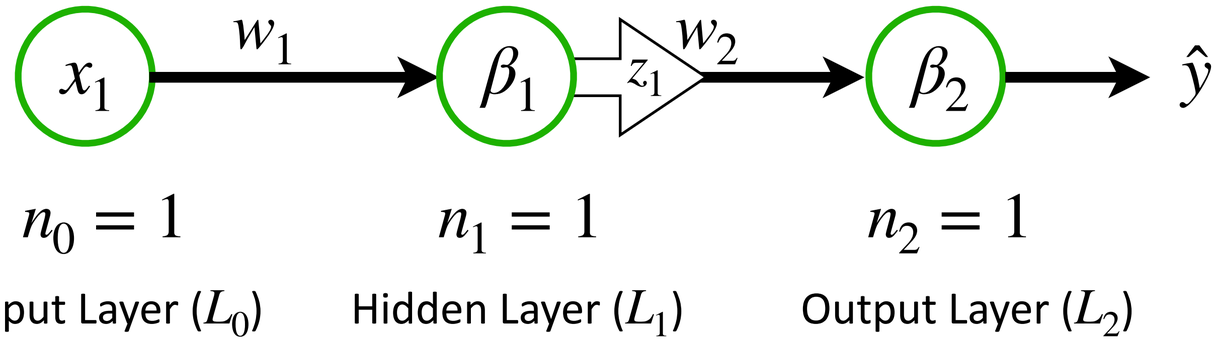
A Forward Pass (Forward Propagation)

Forward pass computes the predicted values given the parameters

A Backward Pass (Back Propagation)

The backward pass computes new values of the parameters (using the partial derivative of the cost function)

Neural Networks



A neural network is trained using Gradient Descent. There are two passes in each iteration of Gradient Descent

A Forward Pass (Forward Propagation)

Forward pass computes the predicted values given the parameters

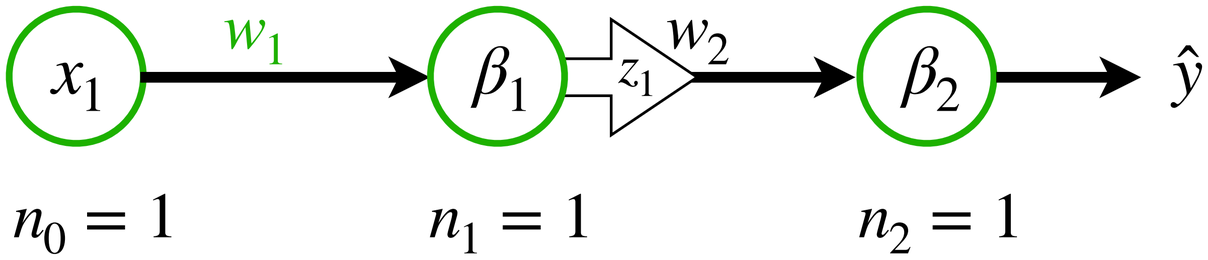
A Backward Pass (Back Propagation)

The backward pass computes new values of the parameters (using the partial derivative of the cost function)

Lets see it in action…

Neural Networks

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

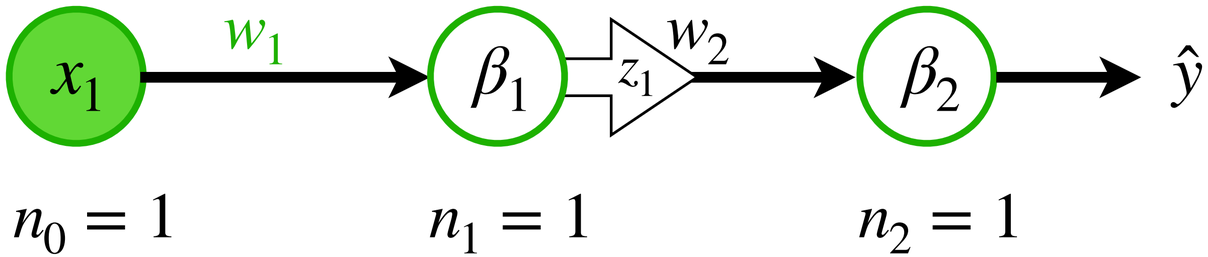
Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

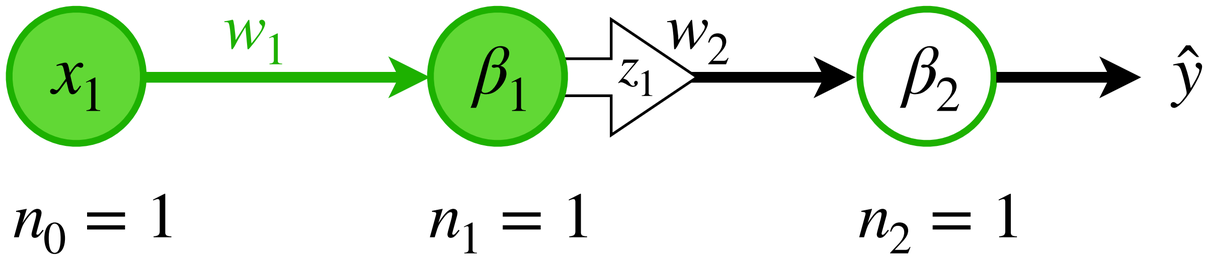
Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

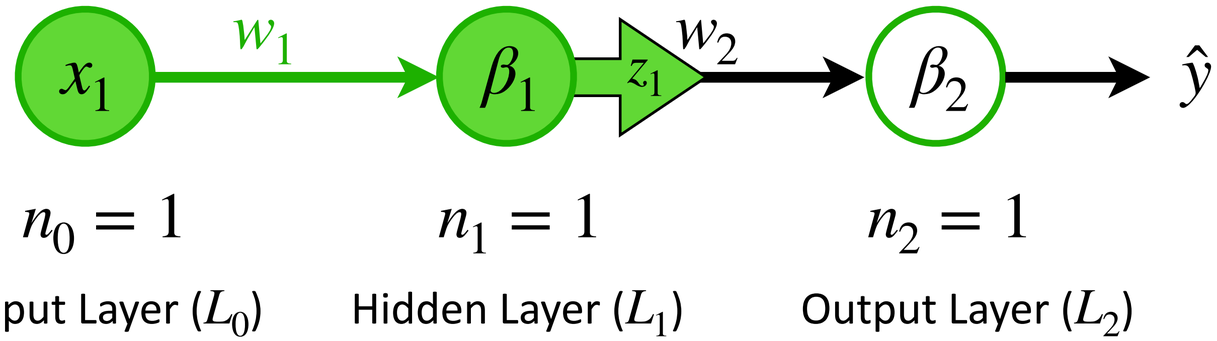
$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

Gradient Descent

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

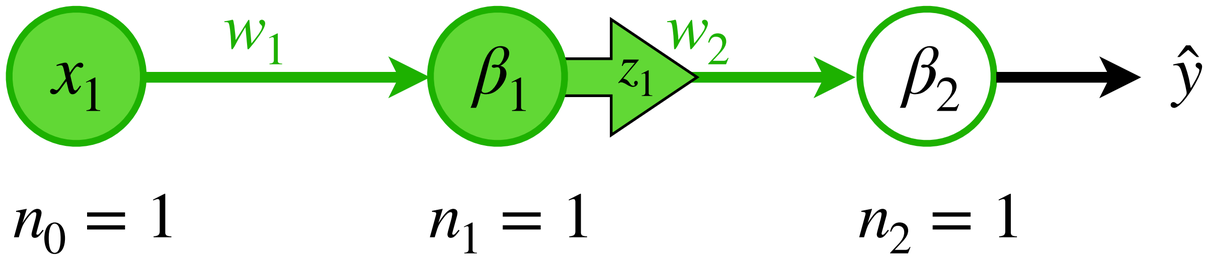
Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

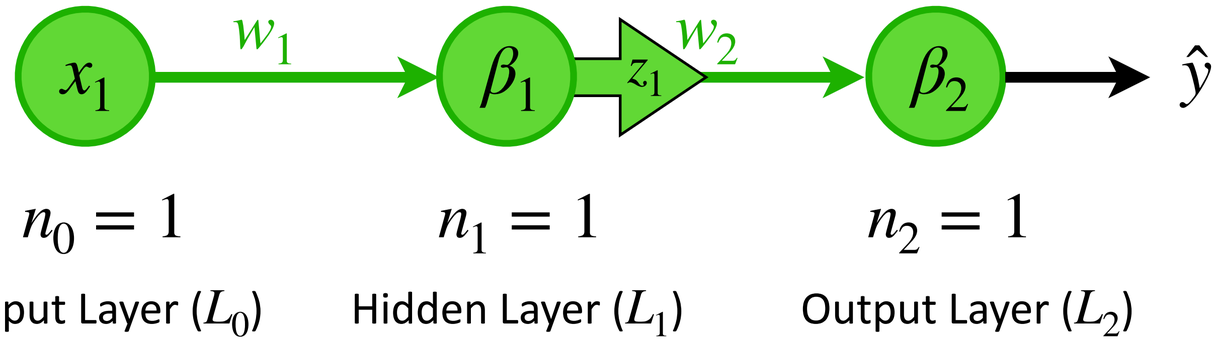
Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

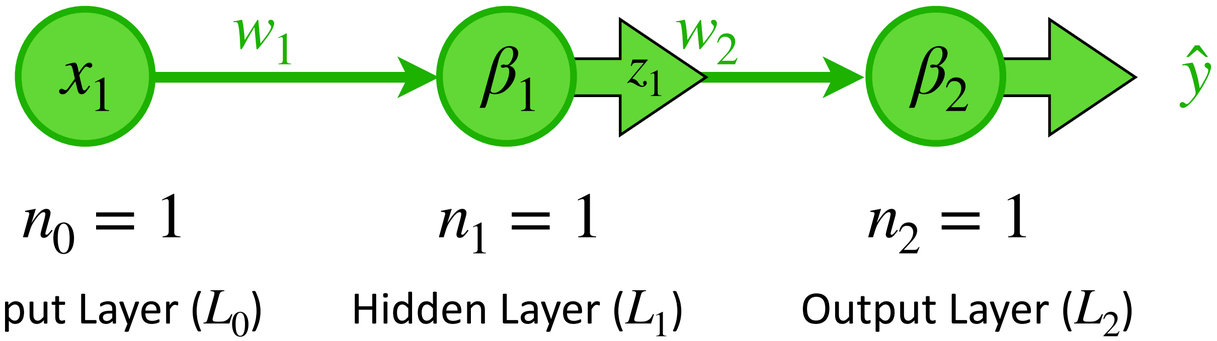
$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

Gradient Descent

In the Forward Pass, inputs move through the Neural Network to compute predictions



Forward Propagation

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

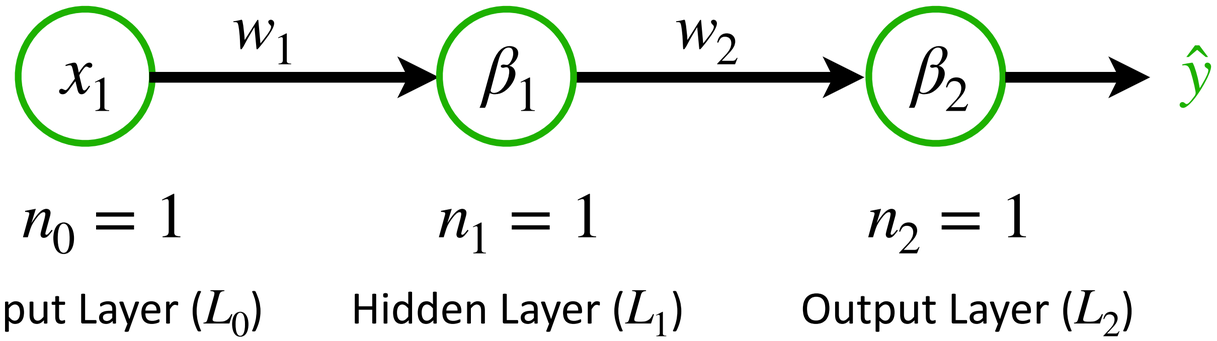
$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

Gradient Descent

In the Backward Pass, weights and Biases are updated through the Neural Network



Back Propagation

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

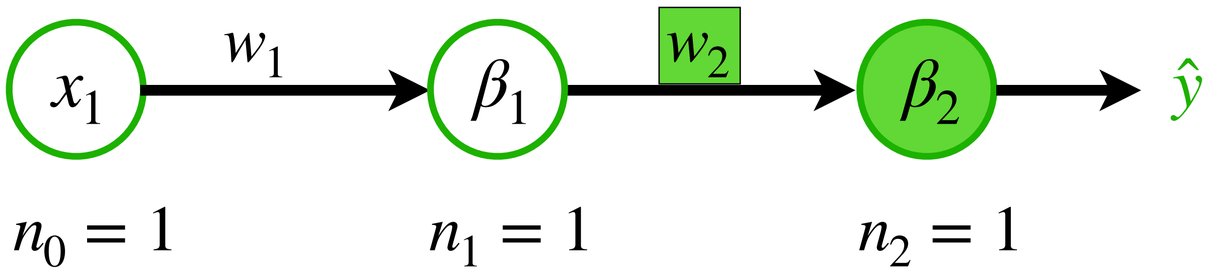
Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

In the Backward Pass, weights and Biases are updated through the Neural Network



Back Propagation

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

Gradient Descent

Step 1: Start with random values for $w_1, w_2, \beta_2, \beta_3$

Step 2a: Compute… $z_1 = f(w_1 x_{11} + \beta_1)$

Step 2b: Compute… $\hat{y} = f(w_2 z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and w_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and w_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and w_1 … $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial w_1} cost$

Step 4b: Compute step size…

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

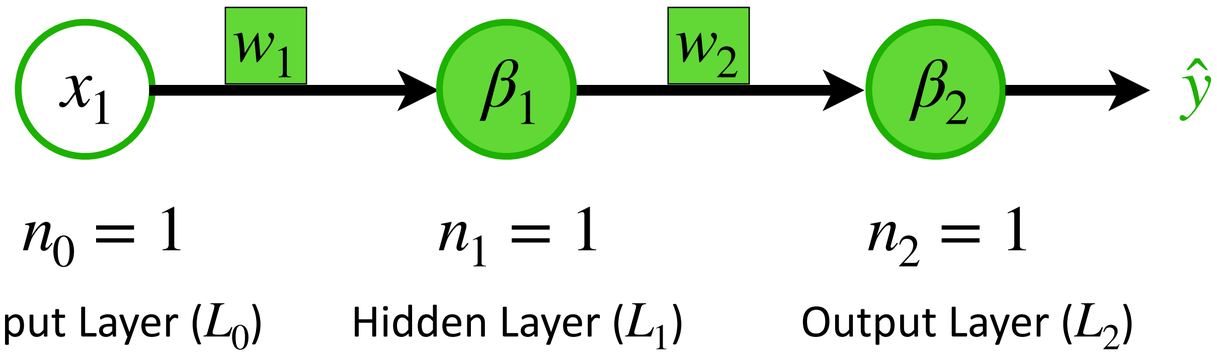
$$step\_size_{w_1} = \frac{\partial}{\partial w_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and w_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad w_1 = w_1 - step\_size_{w_1}$$

Step 5: Go to step 2 and repeat

In the Backward Pass, weights and Biases are updated through the Neural Network



Back Propagation

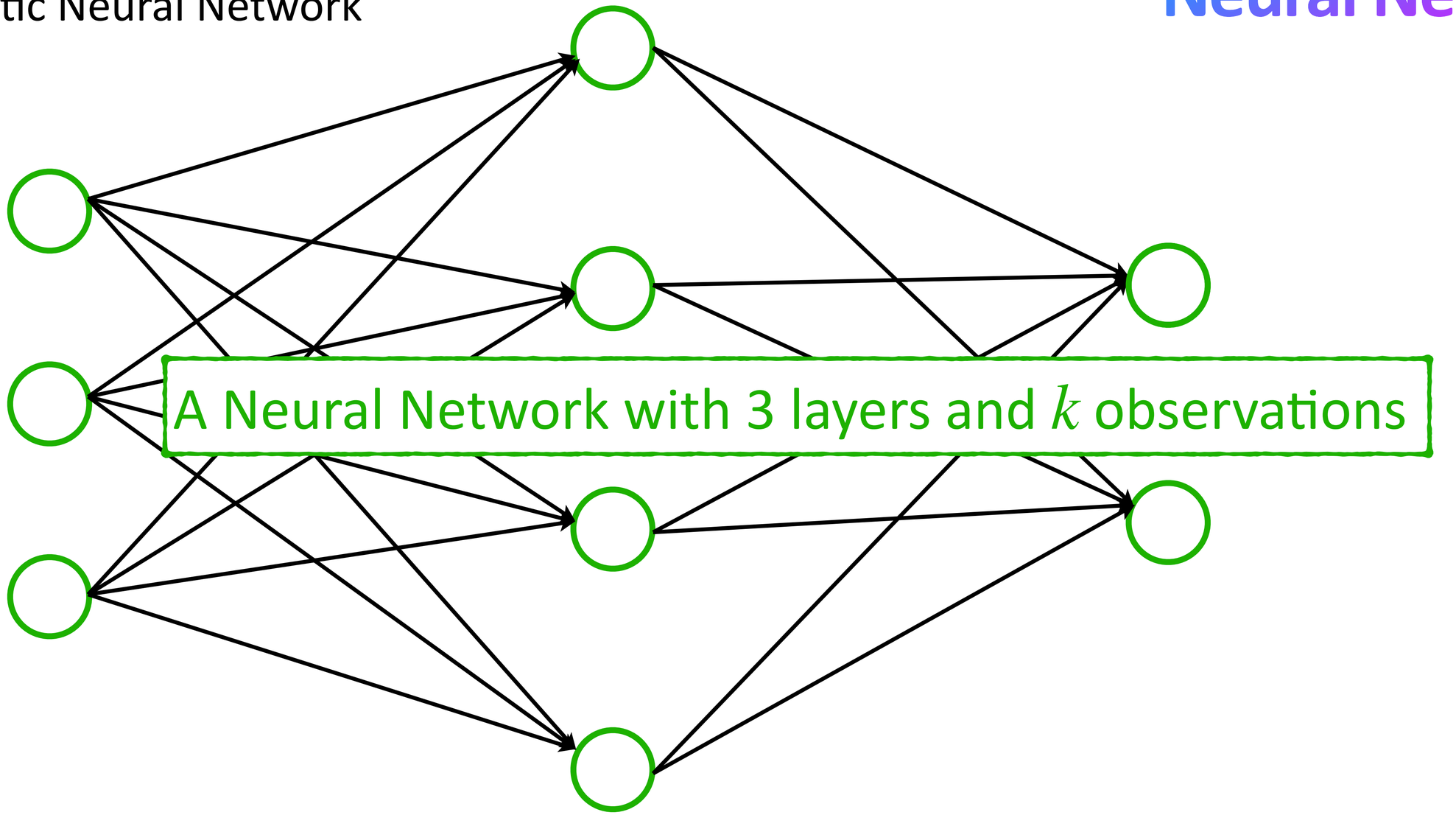
Neural Networks

Let’s take a more realistic example…

And we’ll use matrices this time

Neural Networks

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

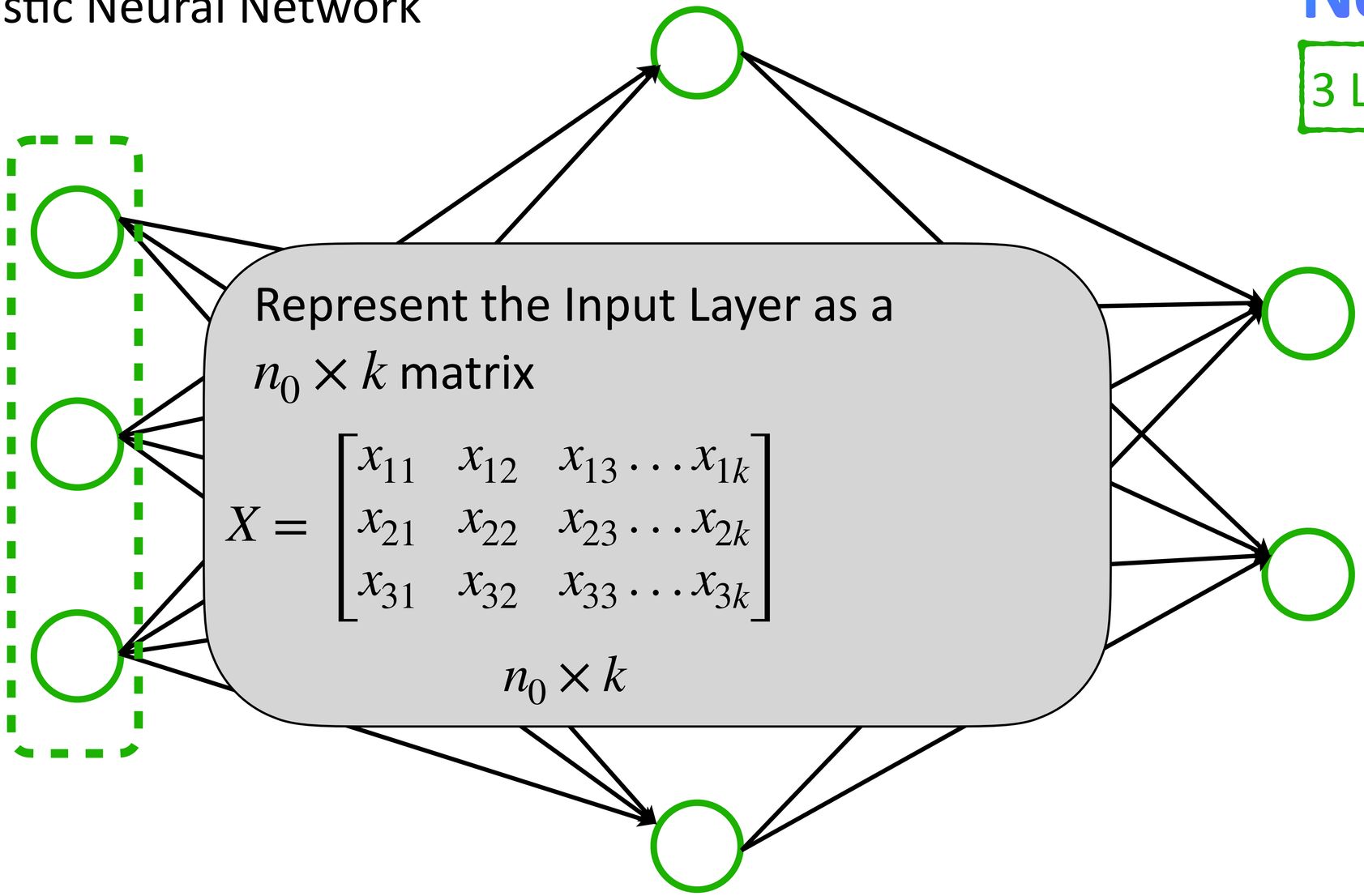
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

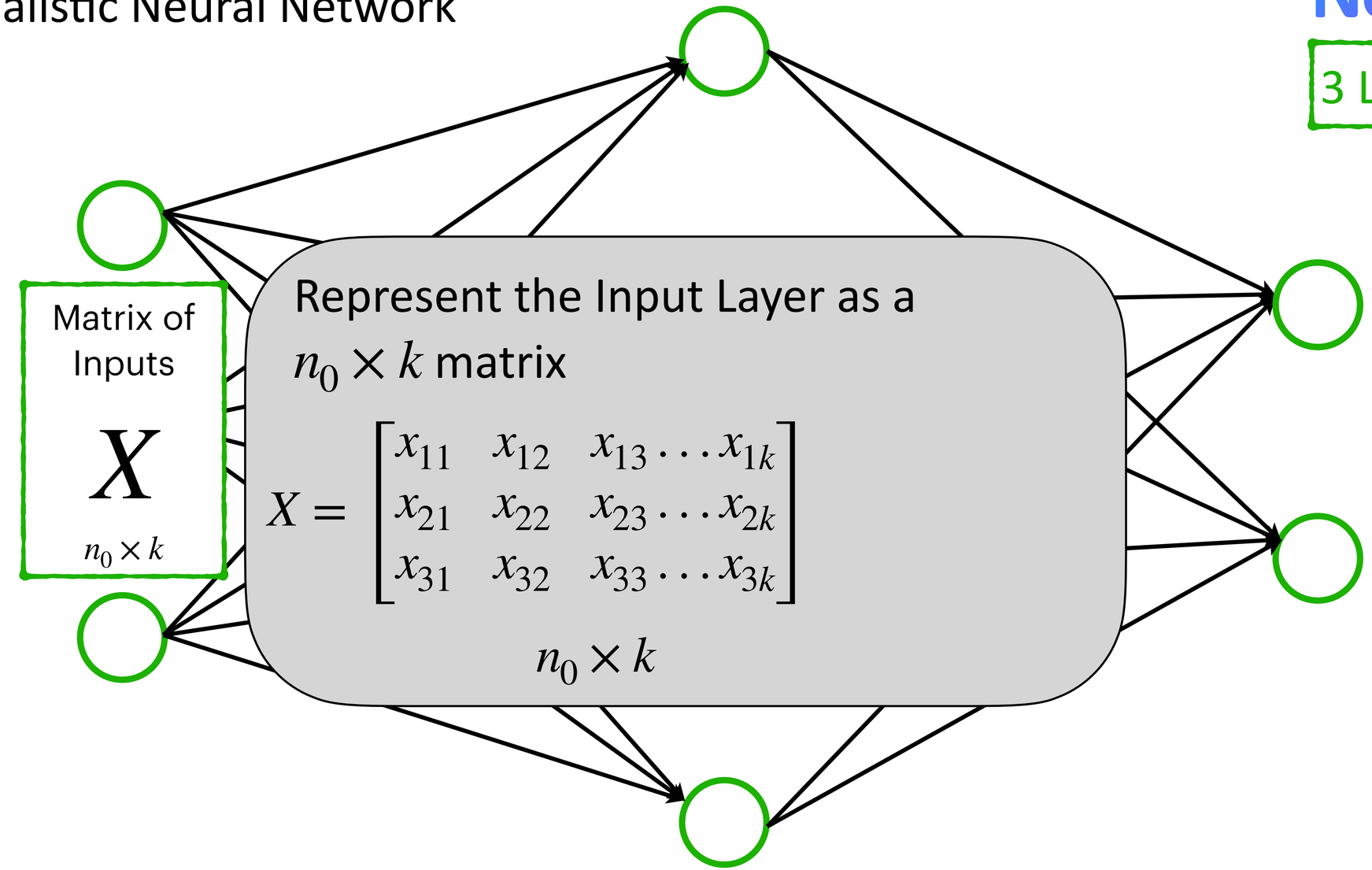
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

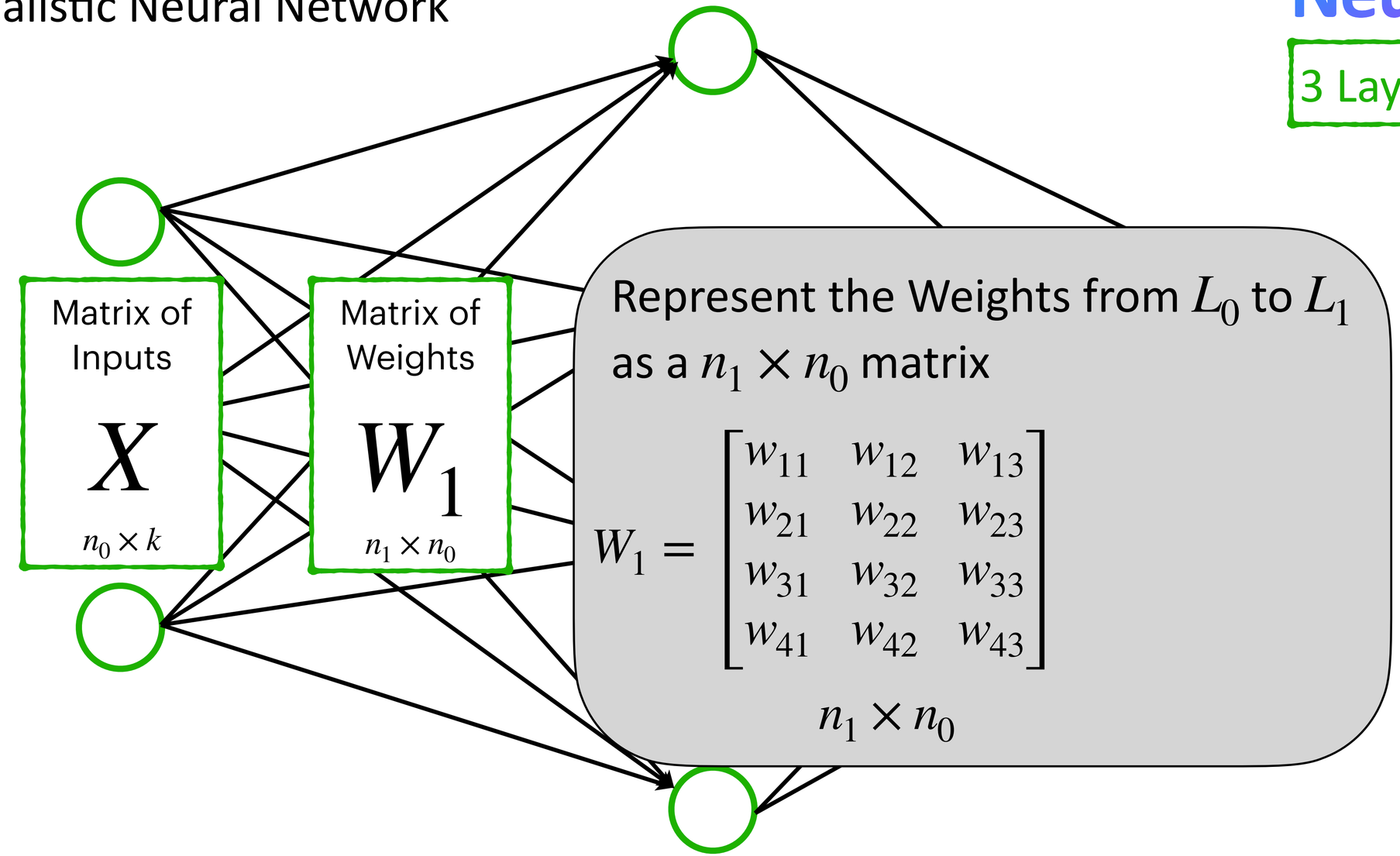
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

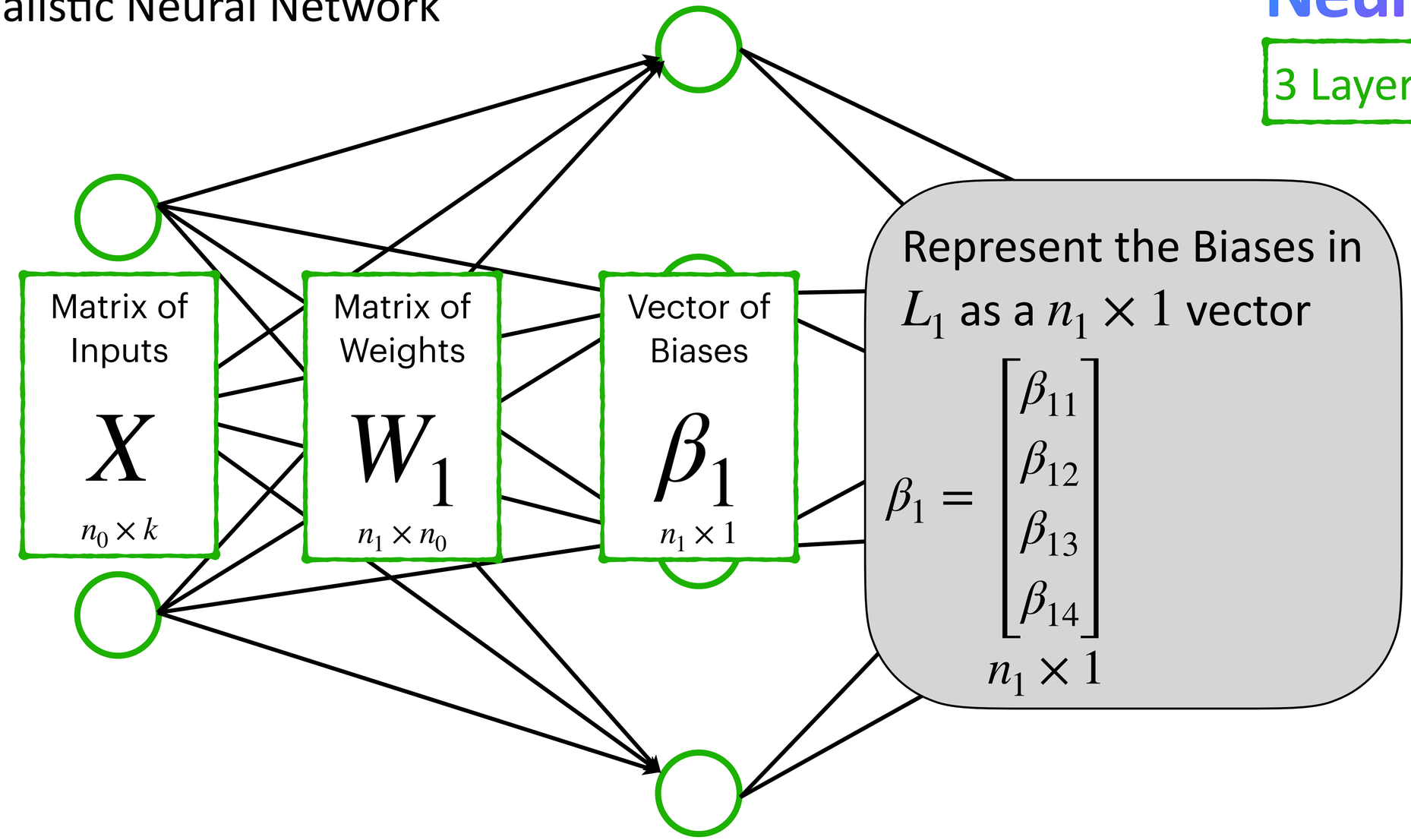
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

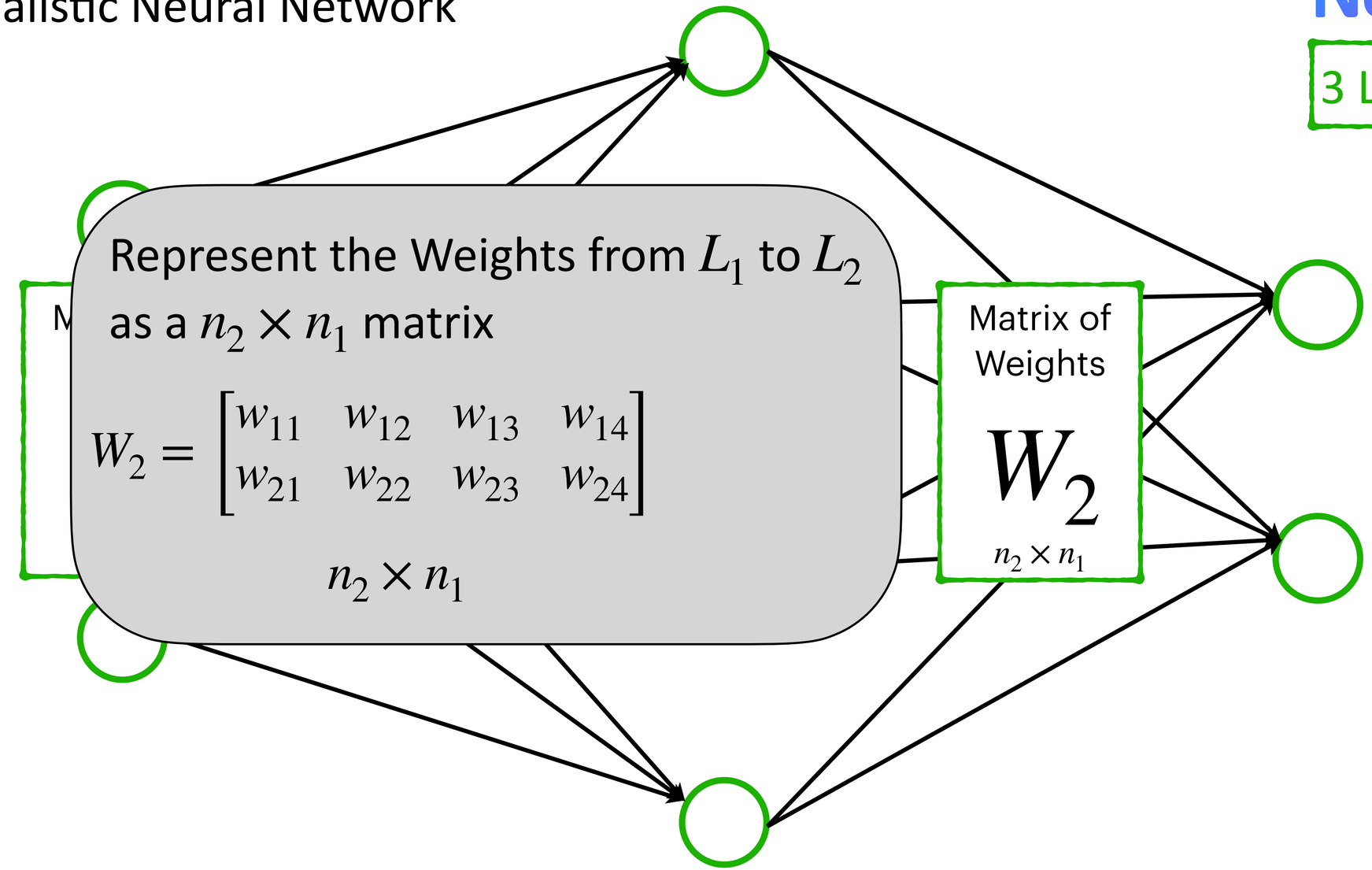
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

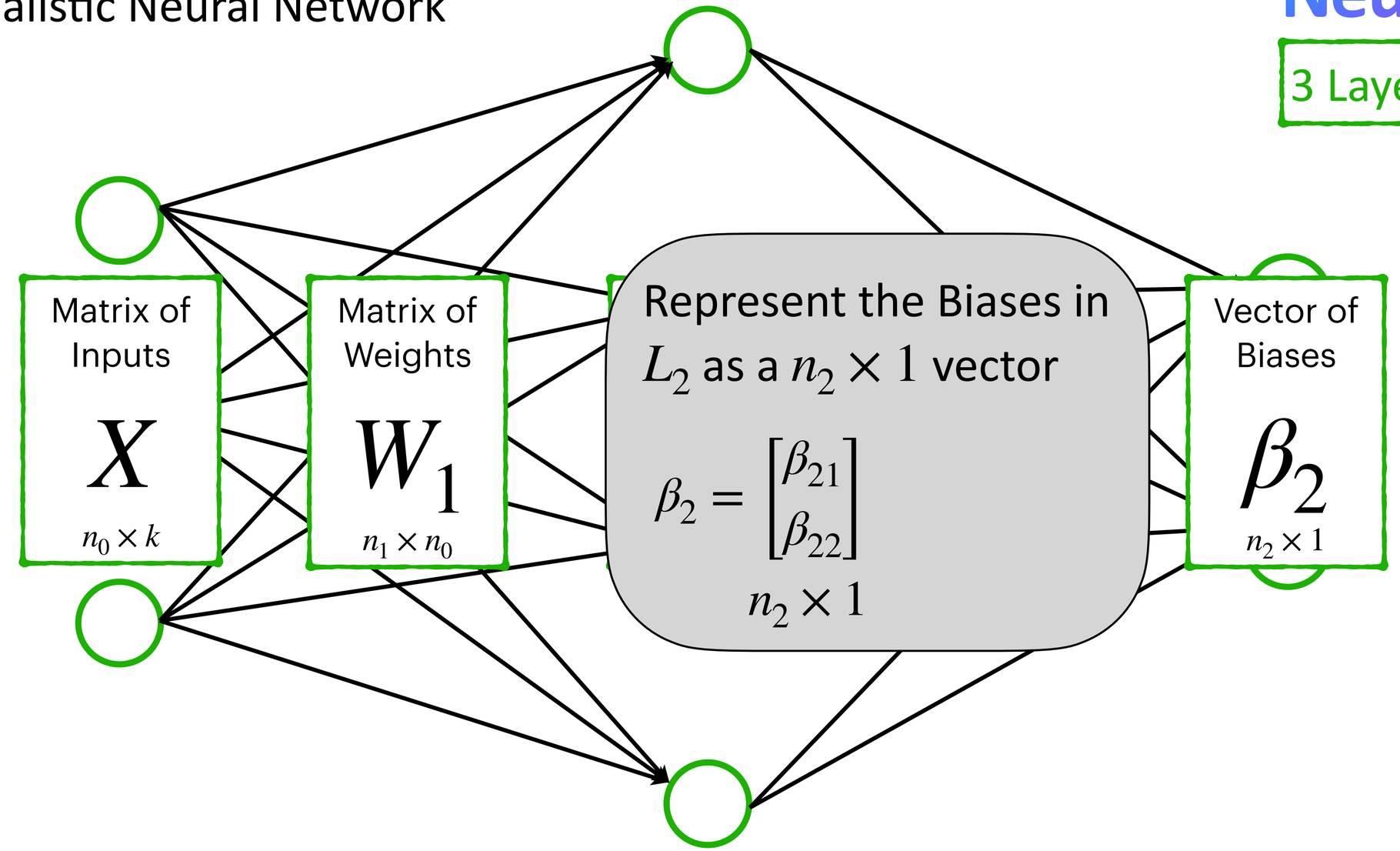
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

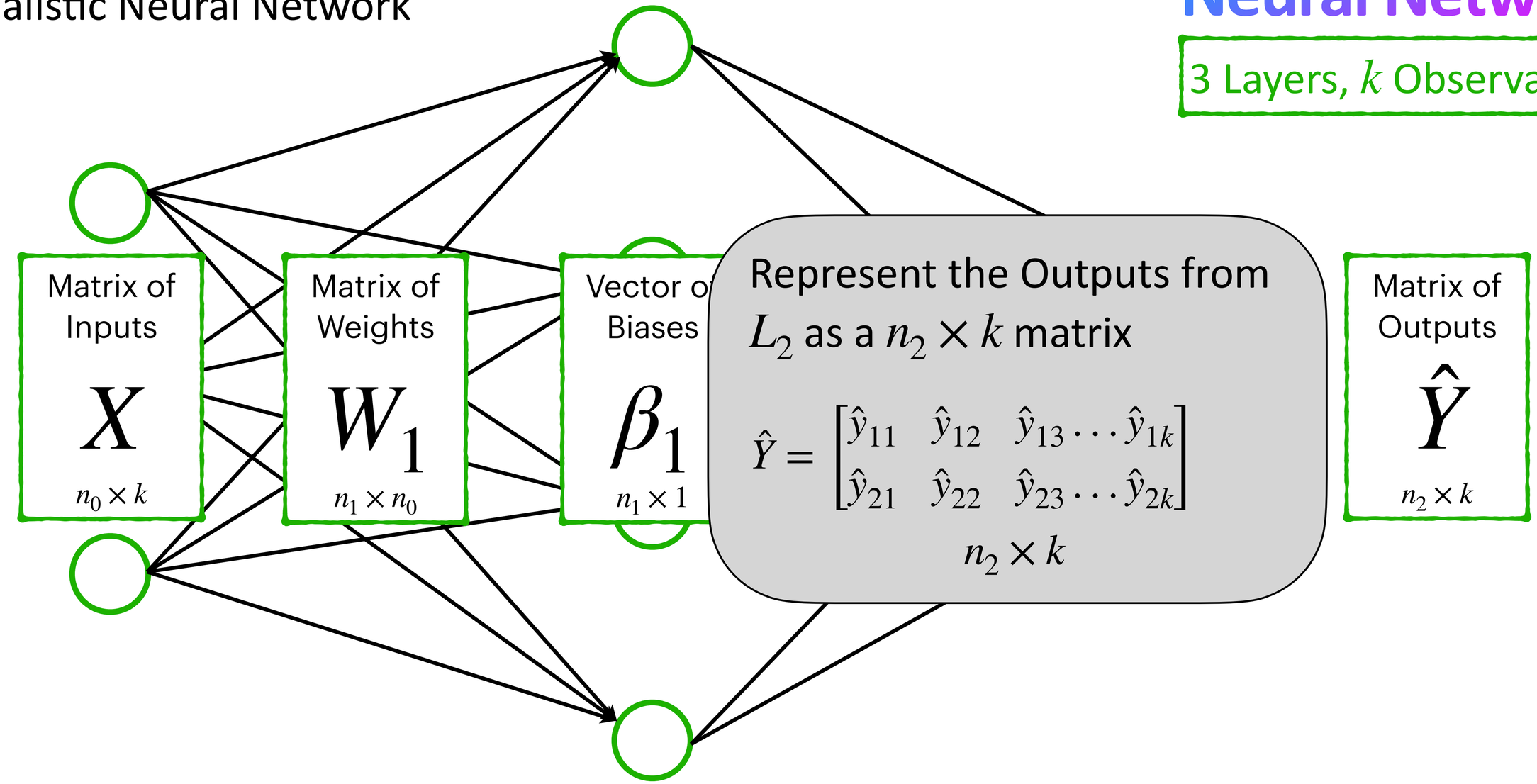
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

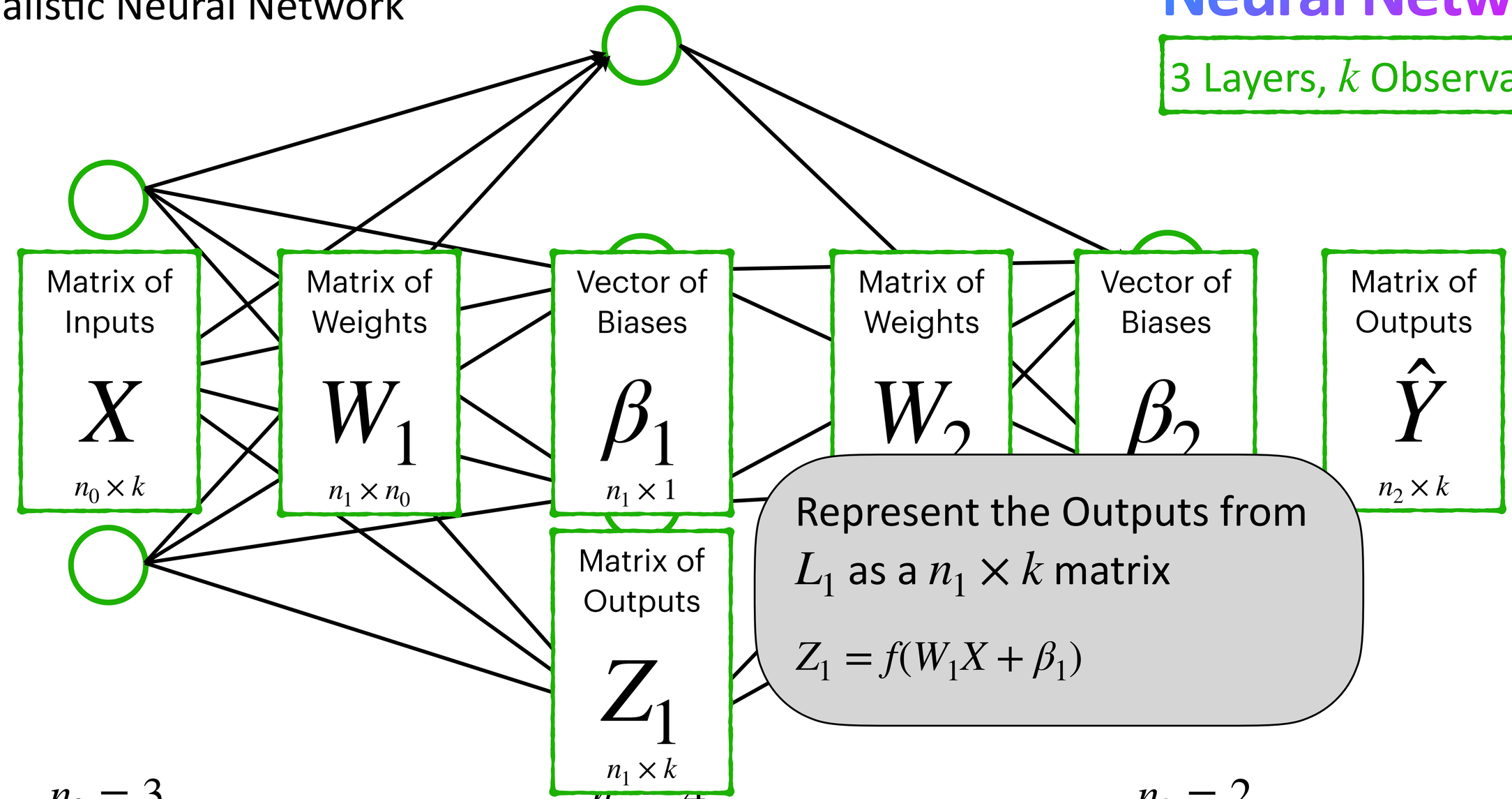
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



Represent the Outputs from L_1 as a $n_1 \times k$ matrix

$$Z_1 = f(W_1 X + \beta_1)$$

$n_0 = 3$
Input Layer (L_0)

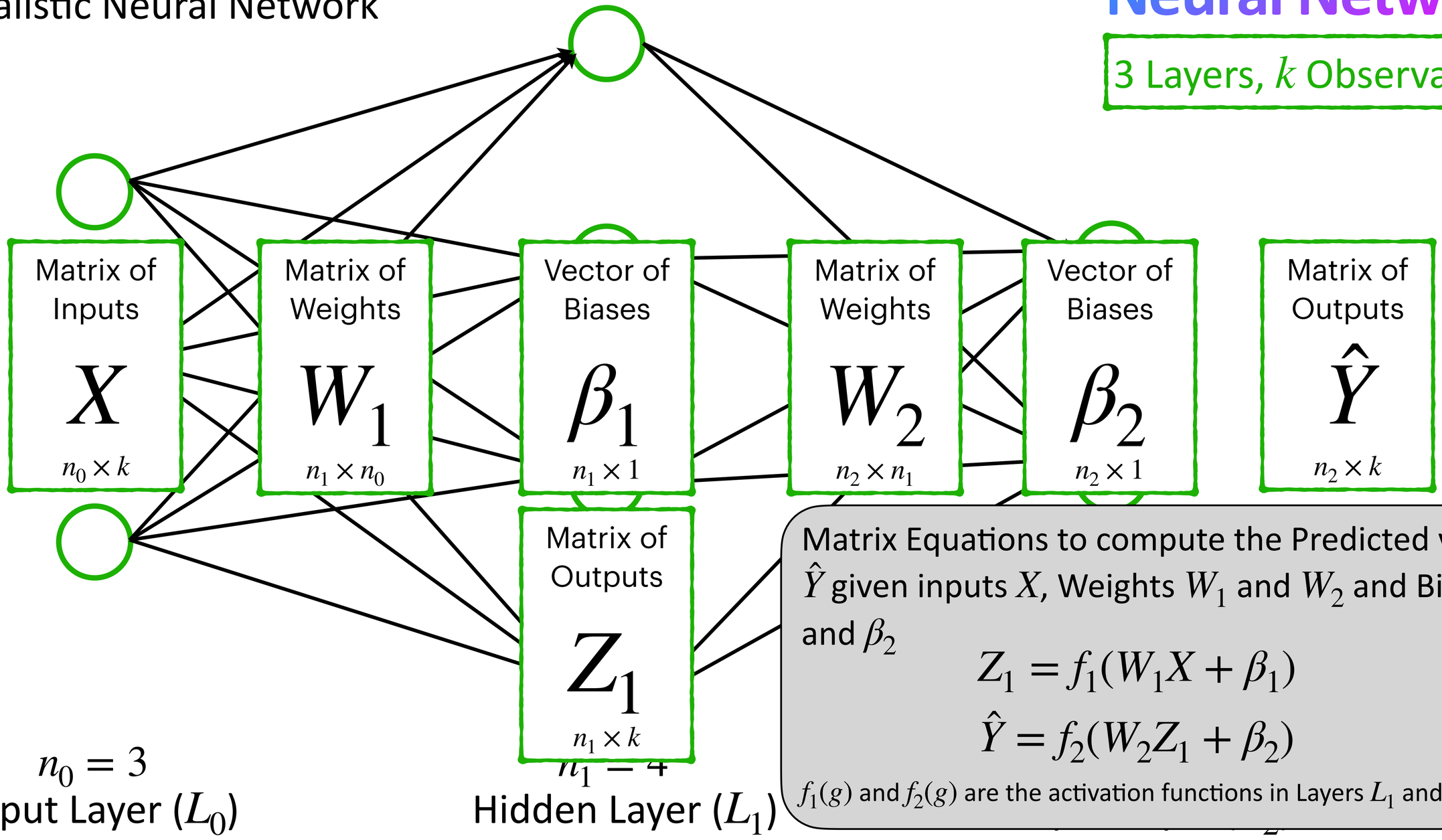
$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

3 Layers, k Observations

A Realistic Neural Network



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

Matrix Equations to compute the Predicted values $\hat{Y}$ given inputs X, Weights W_1 and W_2 and Biases β_1 and β_2

$$Z_1 = f_1(W_1 X + \beta_1)$$

$$\hat{Y} = f_2(W_2 Z_1 + \beta_2)$$

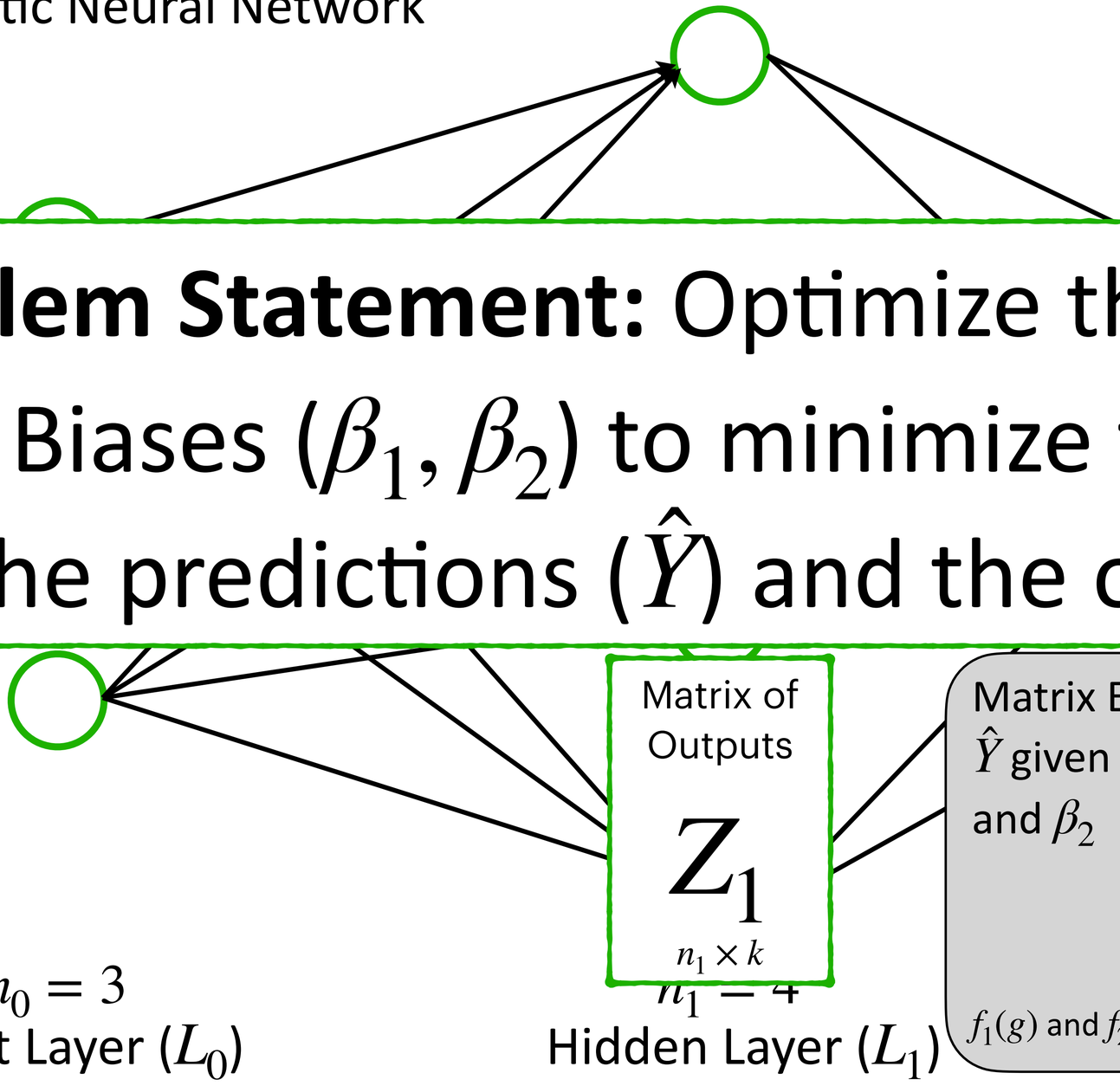
$f_1(g)$ and $f_2(g)$ are the activation functions in Layers L_1 and L_2

A Realistic Neural Network

Neural Networks

3 Layers, k Observations

Problem Statement: Optimize the weights (W_1, W_2) and Biases (β_1, β_2) to minimize the error between the predictions $(\hat{Y})$ and the observations (X)



Matrix of Outputs

$$Z_1$$

$n_1 \times k$

$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

Matrix Equations to compute the Predicted values $\hat{Y}$ given inputs X, Weights W_1 and W_2 and Biases β_1 and β_2

$$Z_1 = f_1(W_1 X + \beta_1)$$

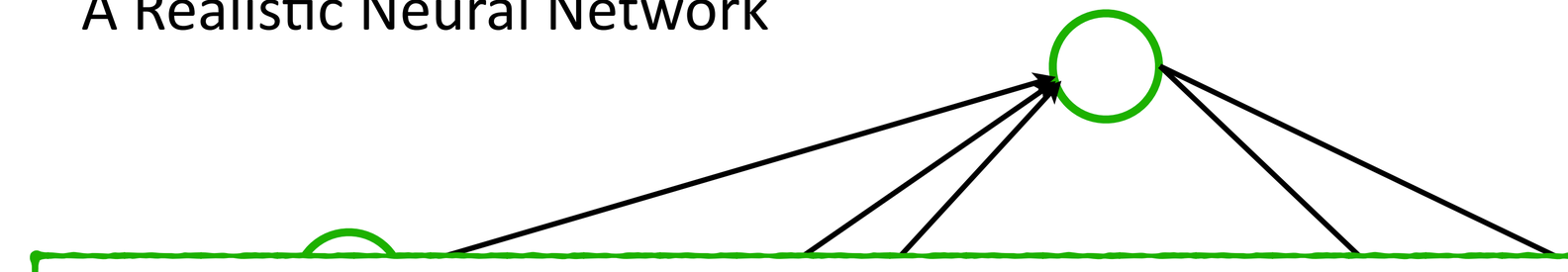
$$\hat{Y} = f_2(W_2 Z_1 + \beta_2)$$

$f_1(g)$ and $f_2(g)$ are the activation functions in Layers L_1 and L_2

A Realistic Neural Network

Neural Networks

3 Layers, k Observations



Problem Statement: Optimize the weights (W_1, W_2) and Biases (β_1, β_2) to minimize the error between the predictions $(\hat{Y})$ and the observations (X)

We'll use Gradient Descent to train the Neural Network

redicted values
W_2 and Biases β_1

$$\hat{Y} = f_2(W_2 Z_1 + \beta_2)$$

$f_1(g)$ and $f_2(g)$ are the activation functions in Layers L_1 and L_2

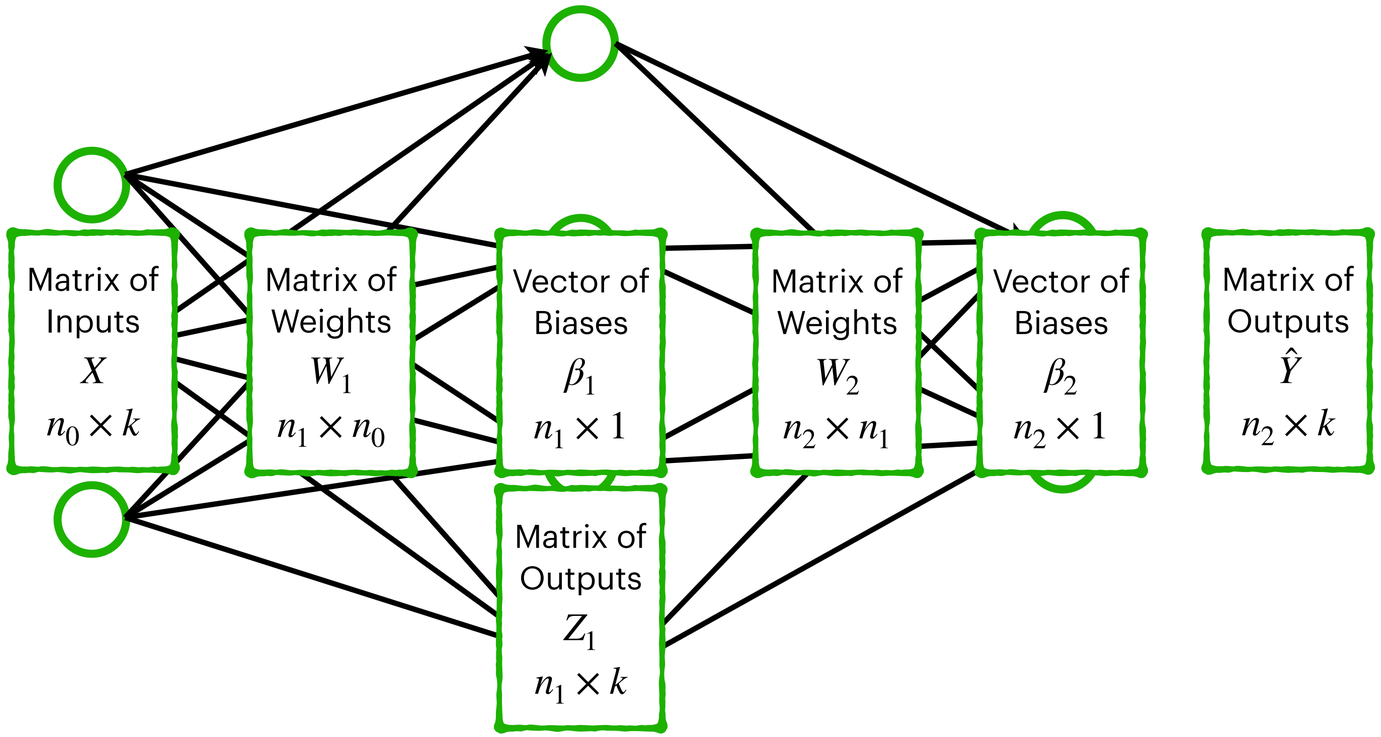
$n_1 \times k$

$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

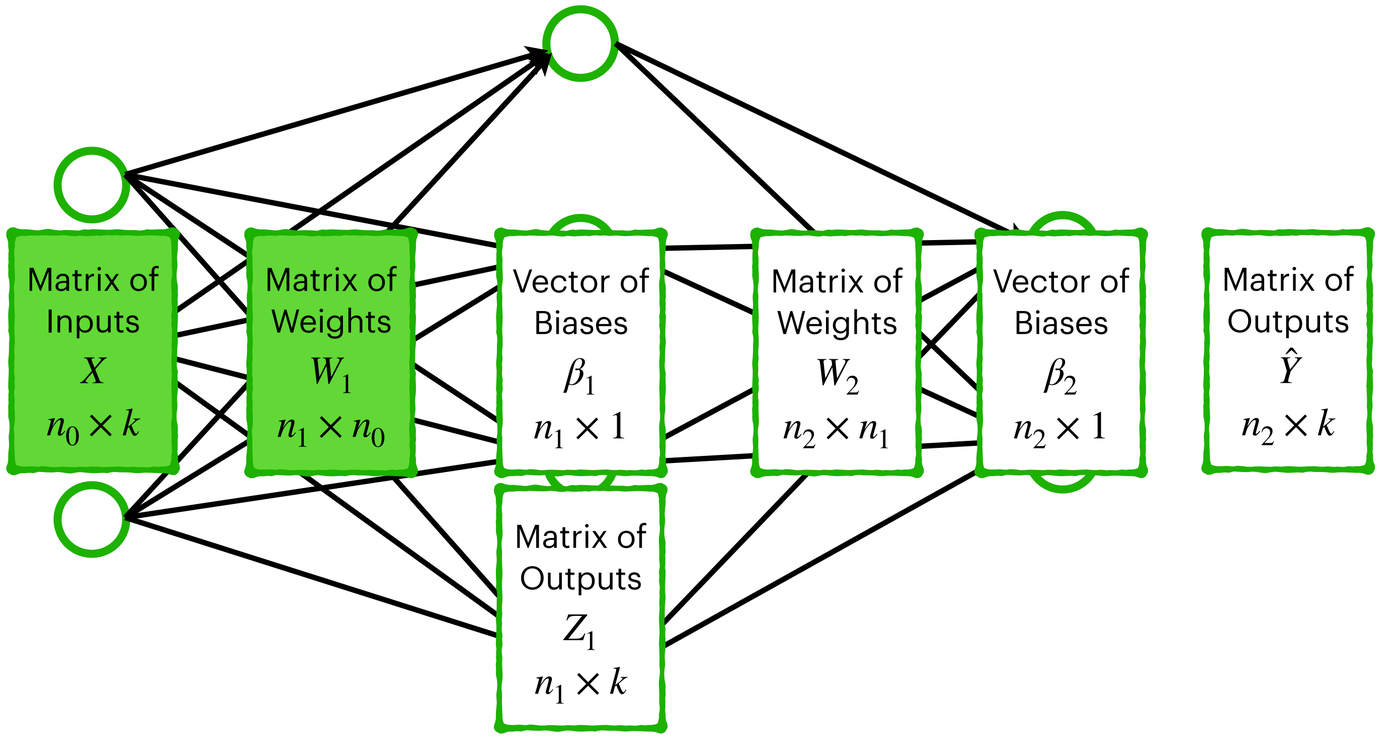
$n_2 = 2$
Output Layer (L_2)

Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

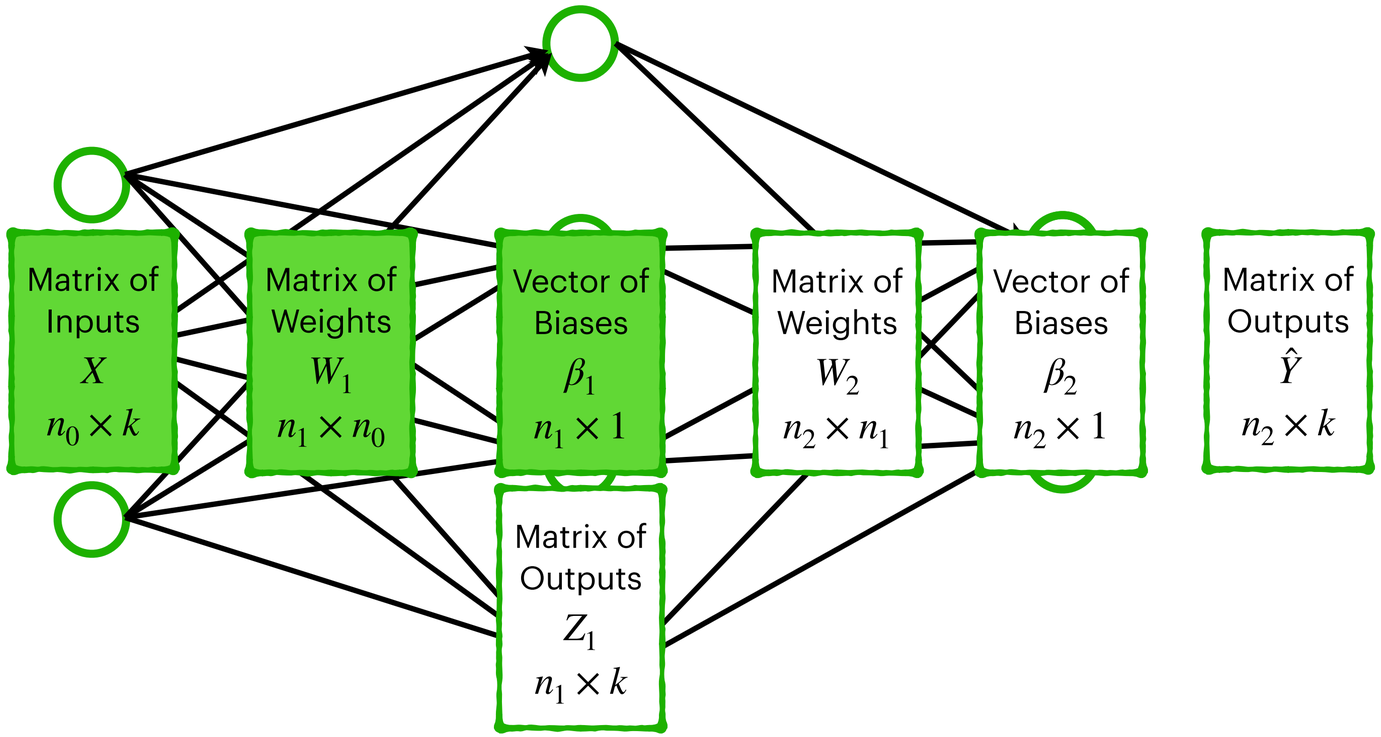
Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $W_1 X$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

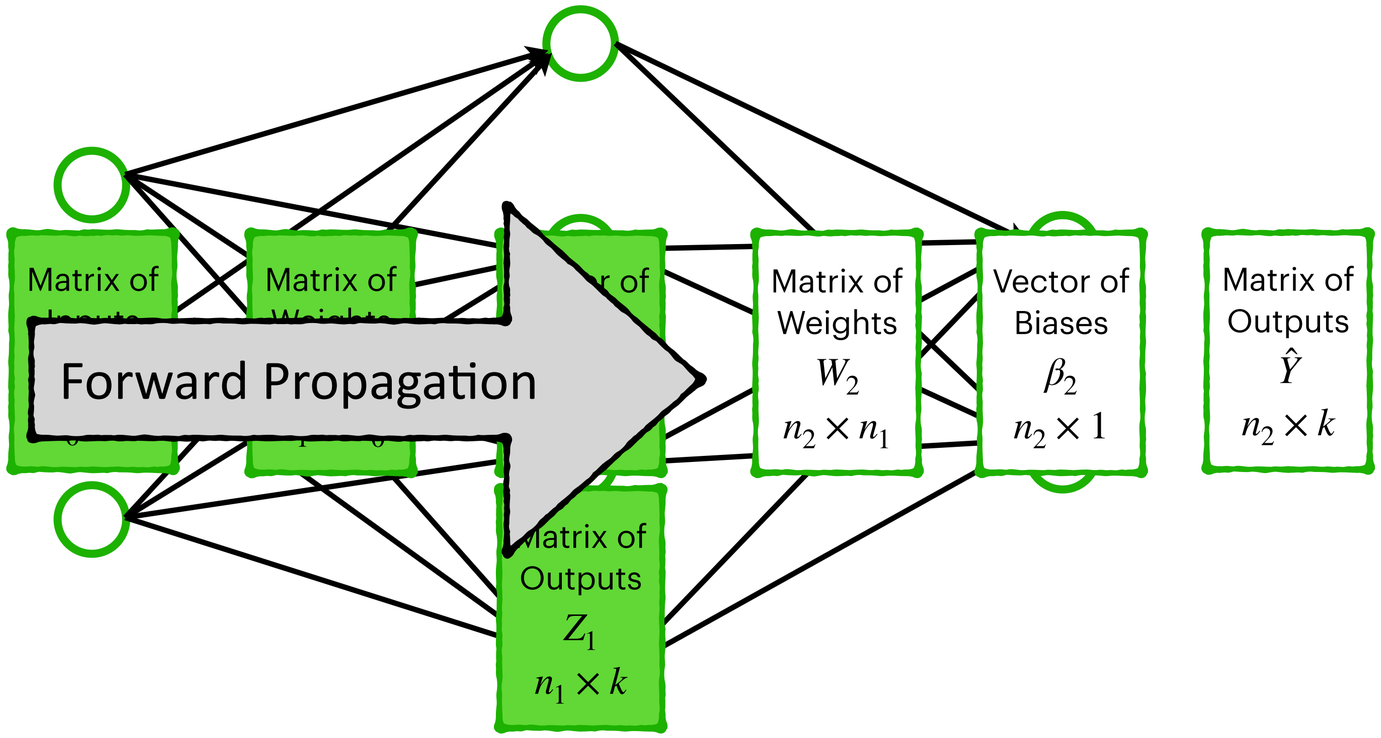
Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $W_1 X + \beta_1$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

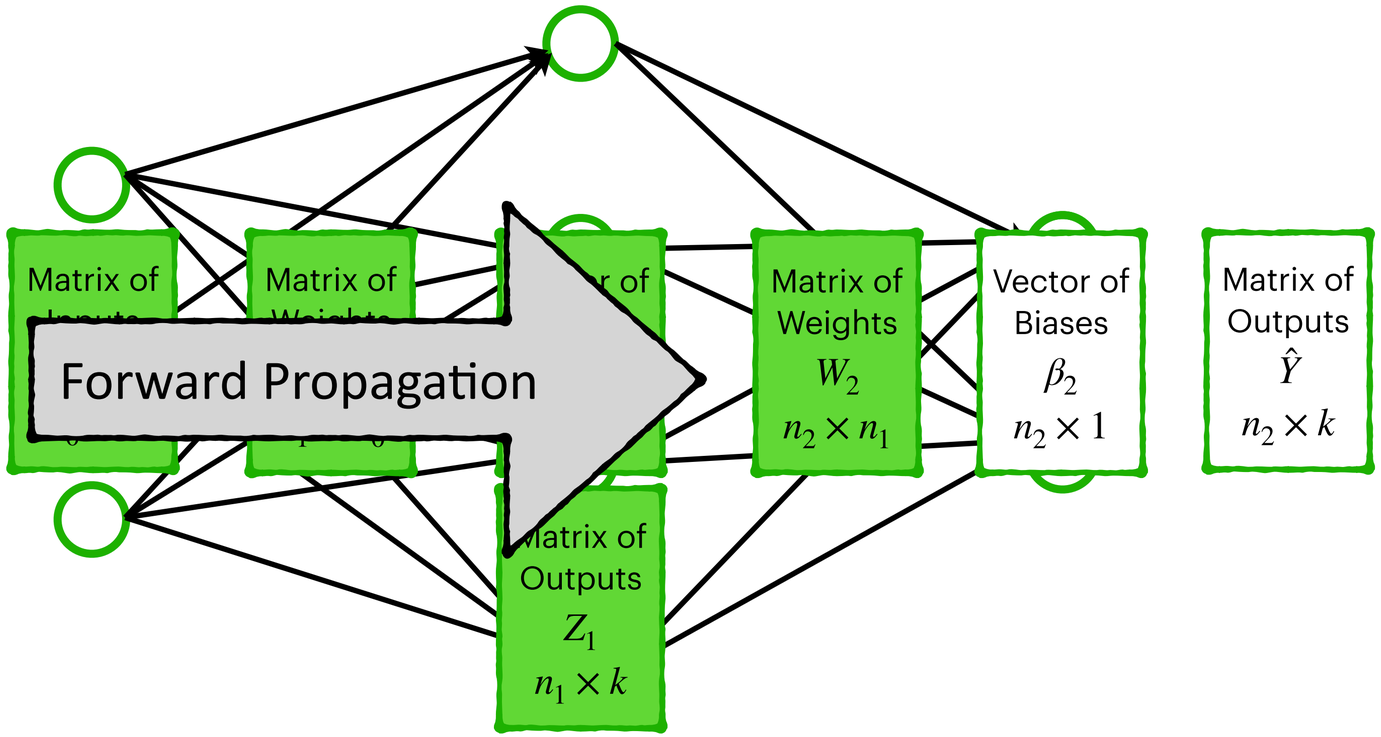
Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

$f(x)$ Is the activation function

We'll cover activation functions in more detail in later tutorials

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

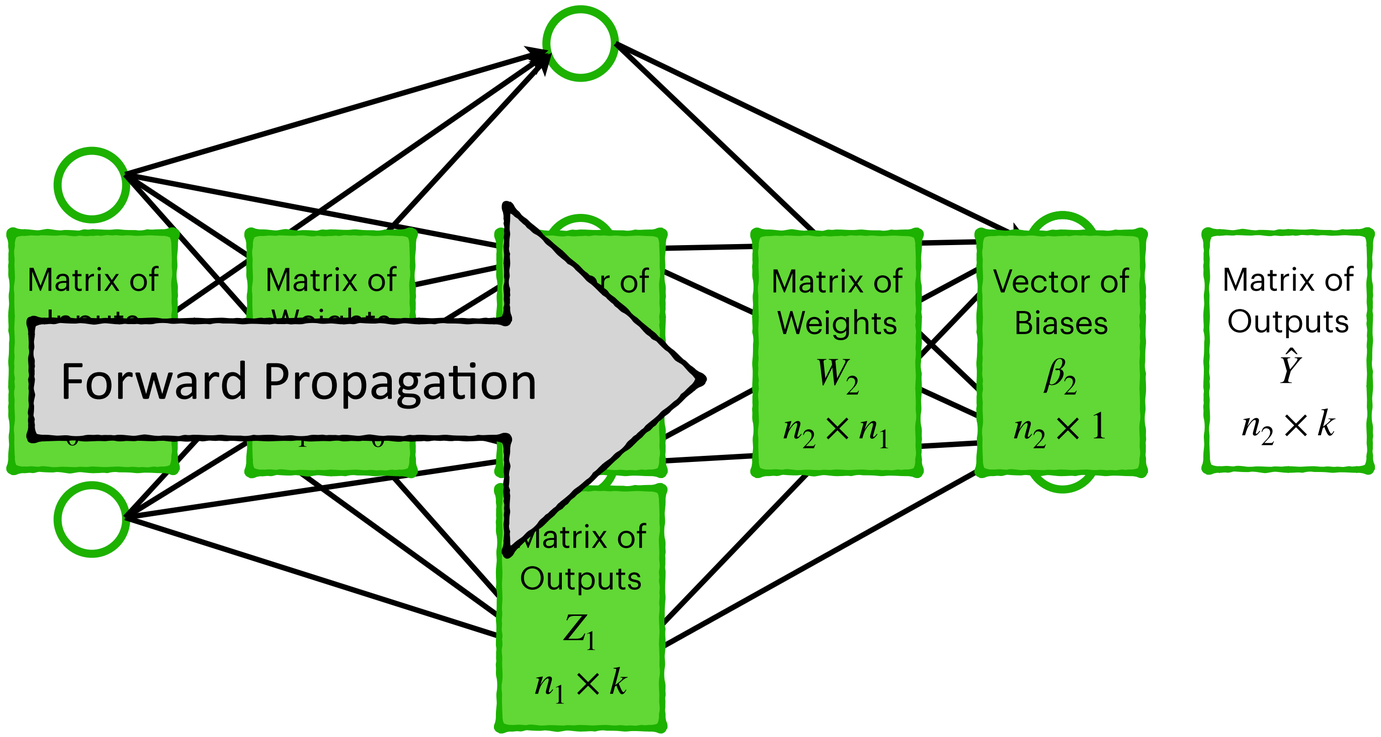
Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $W_2 Z_1$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

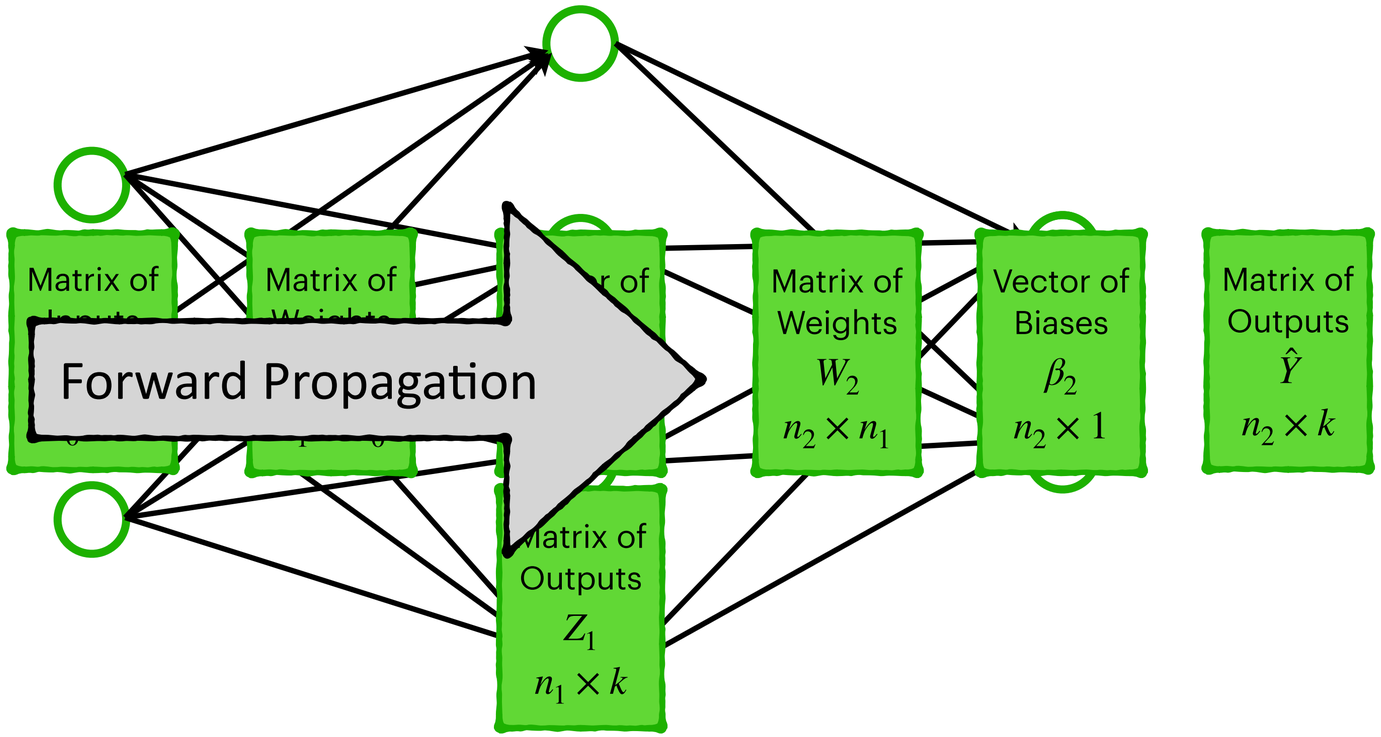
Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $W_2 Z_1 + \beta_2$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

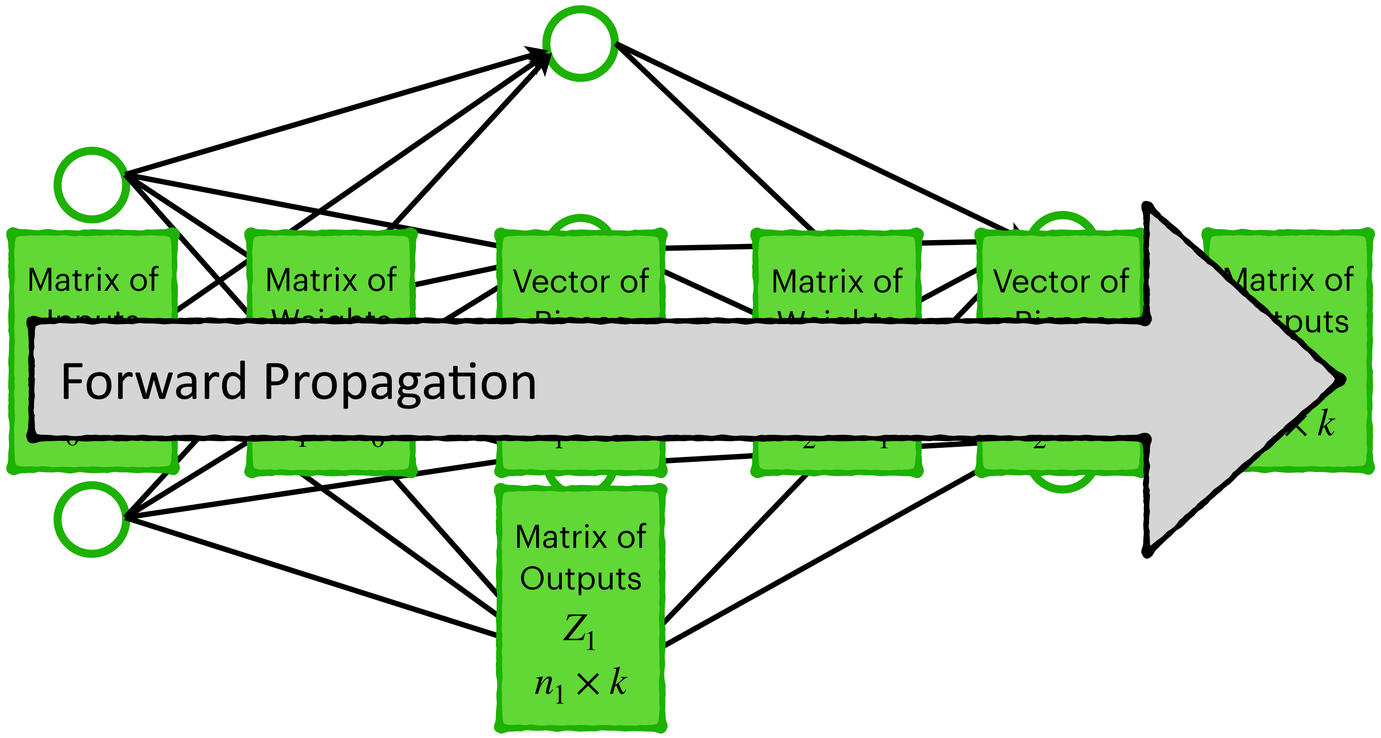
Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

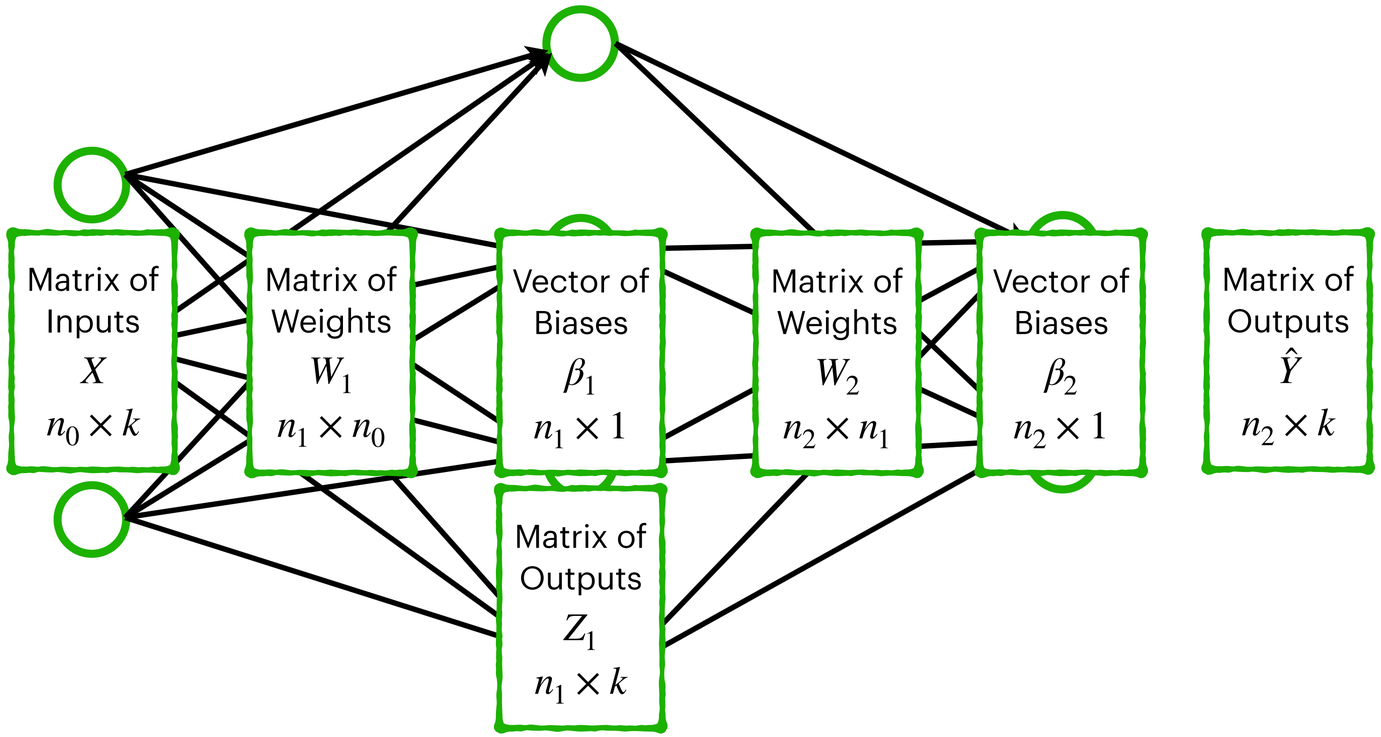
Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

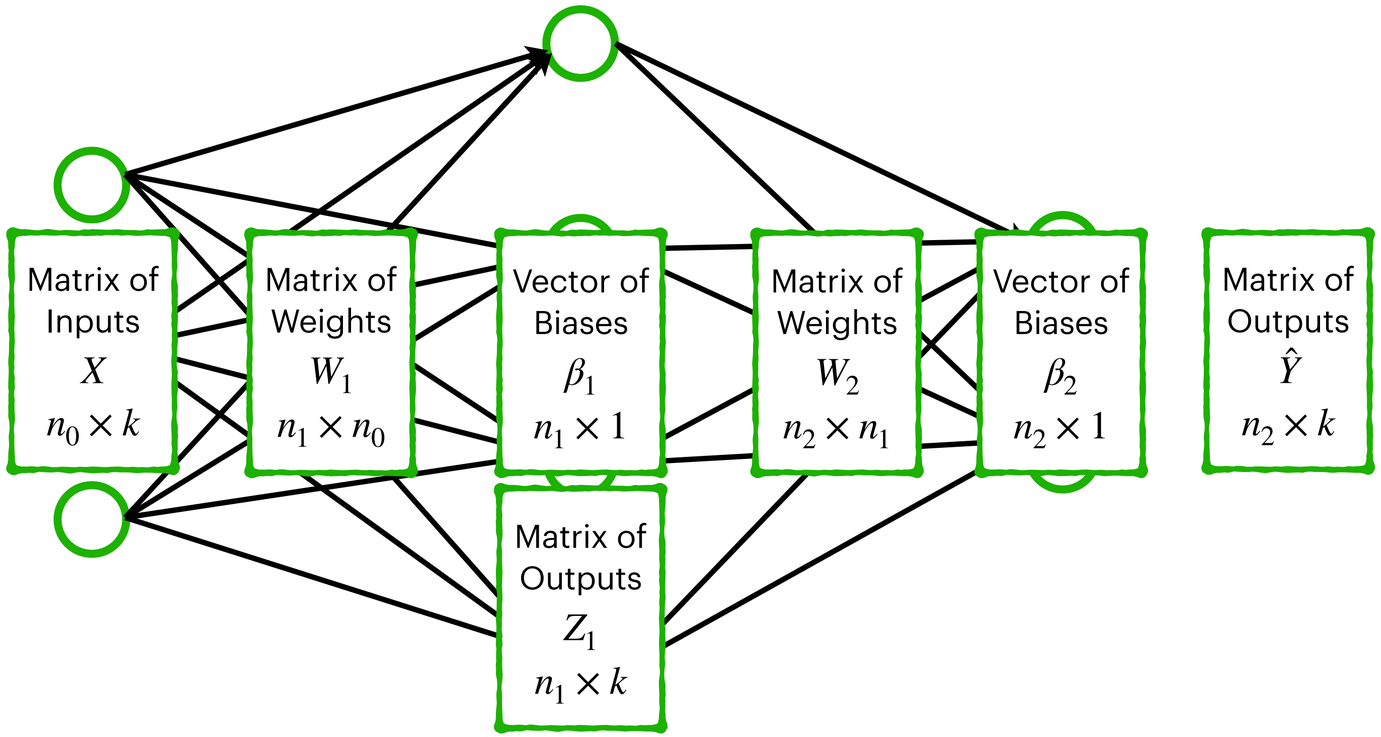
Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

$$\frac{\partial}{\partial \beta_2} cost \qquad \frac{\partial}{\partial W_2} cost$$

We'll cover cost functions in more details in later tutorials

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

$$\frac{\partial}{\partial \beta_2} cost \qquad \frac{\partial}{\partial W_2} cost$$

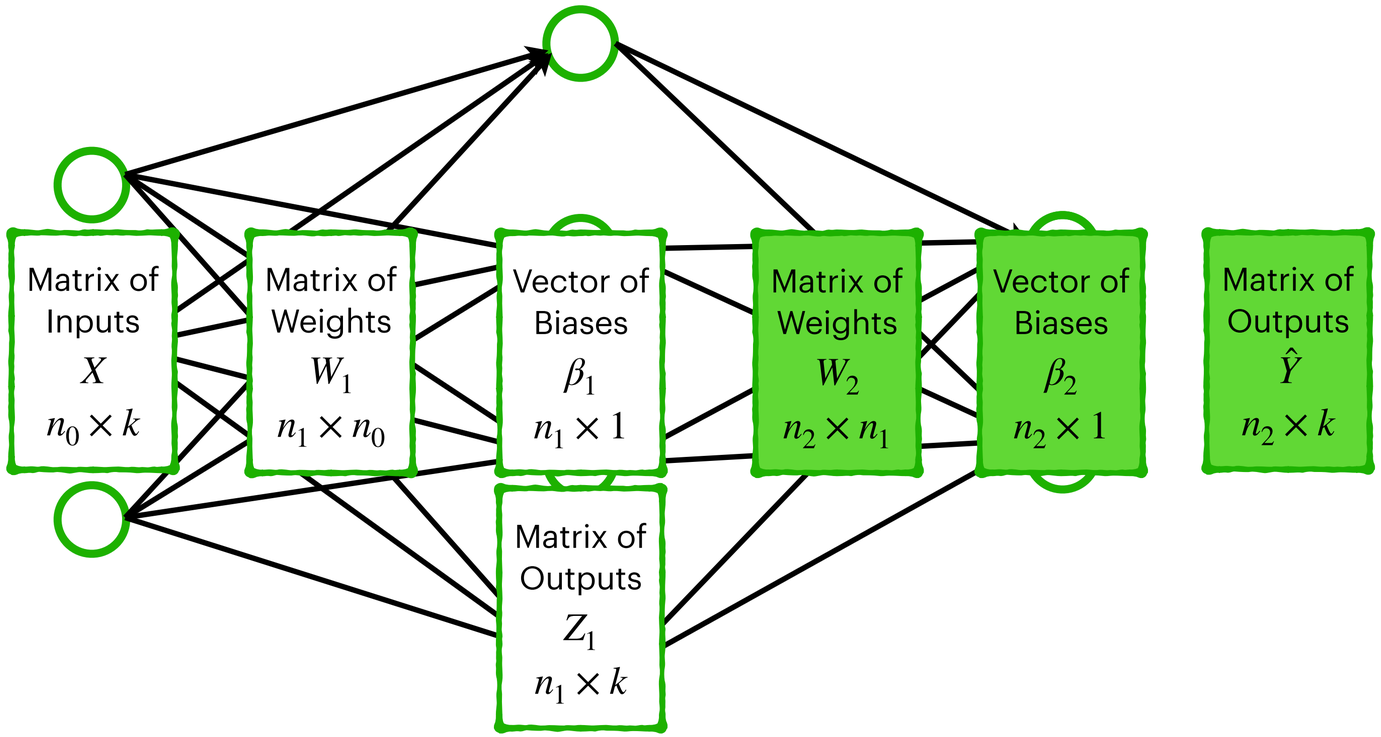
Step 3b: Compute step size…

$$step\_size_{\beta_2} = \frac{\partial}{\partial \beta_2} cost \times learning\_rate$$

$$step\_size_{W_2} = \frac{\partial}{\partial W_2} cost \times learning\_rate$$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

$$\frac{\partial}{\partial \beta_2} cost \qquad \frac{\partial}{\partial W_2} cost$$

Step 3b: Compute step size…

$$step\_size_{\beta_2} = \frac{\partial}{\partial \beta_2} cost \times learning\_rate$$

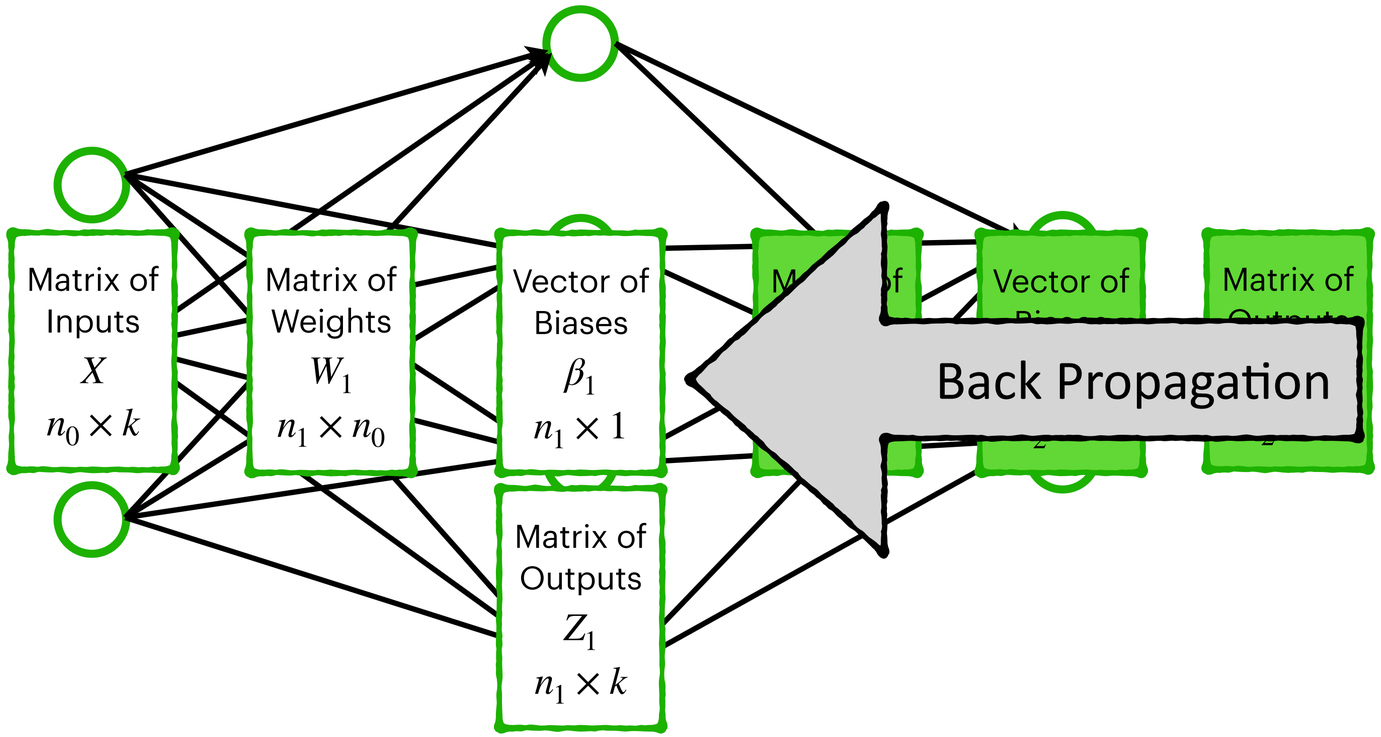
$$step\_size_{W_2} = \frac{\partial}{\partial W_2} cost \times learning\_rate$$

Step 3c: Compute new values for β_2 and W_2

$$\beta_2 = \beta_2 - step\_size_{\beta_2} \qquad W_2 = W_2 - step\_size_{W_2}$$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

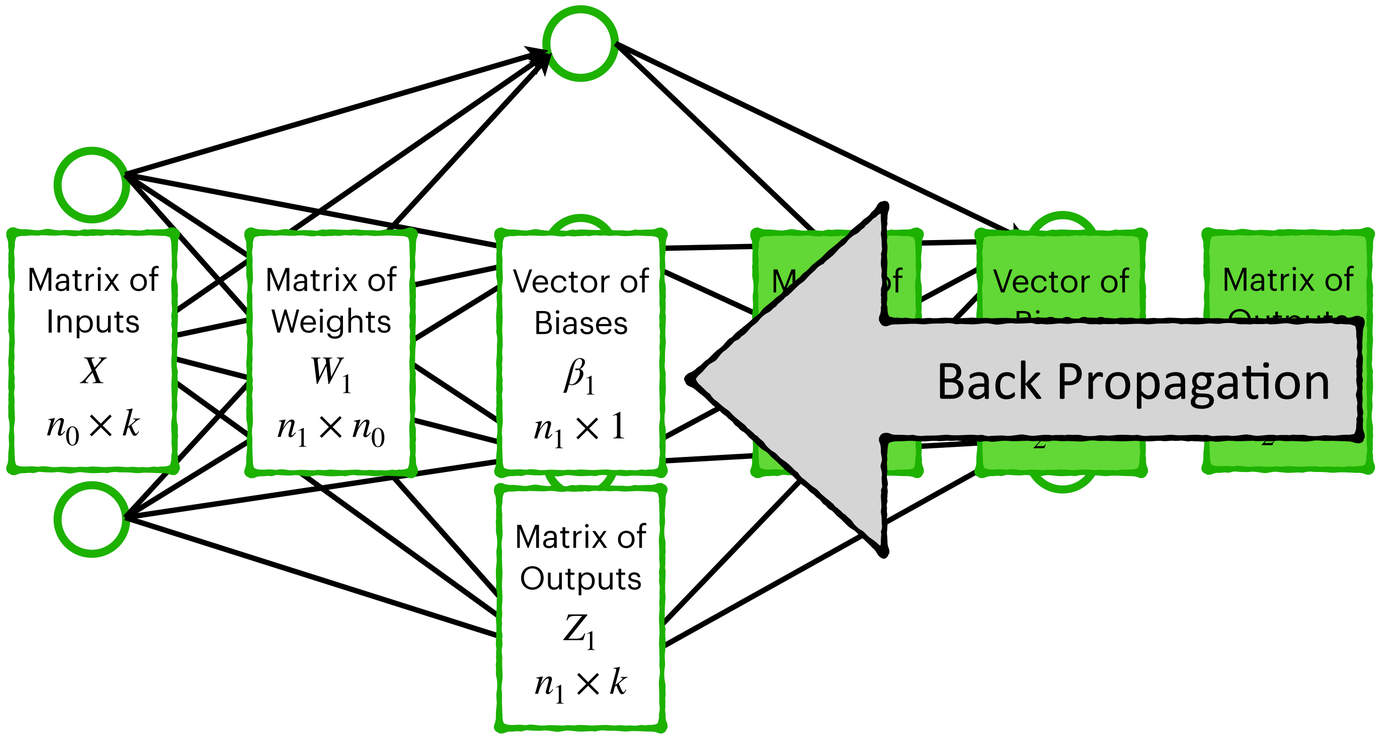
Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and W_2

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

Step 3b: Compute step size…

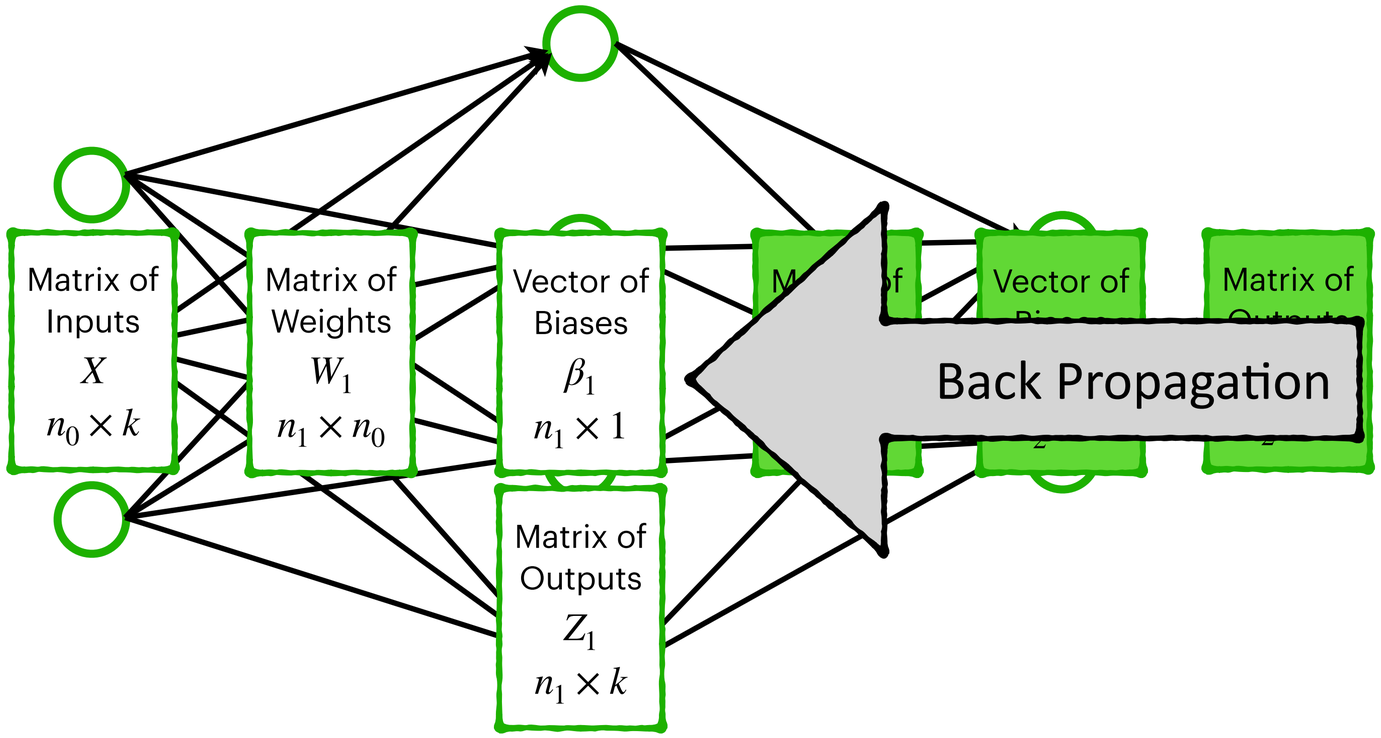
Step 3c: Compute new values for β_2 and W_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and W_1 …

$$\frac{\partial}{\partial \beta_1} cost \qquad \frac{\partial}{\partial W_1} cost$$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and W_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and W_1 …

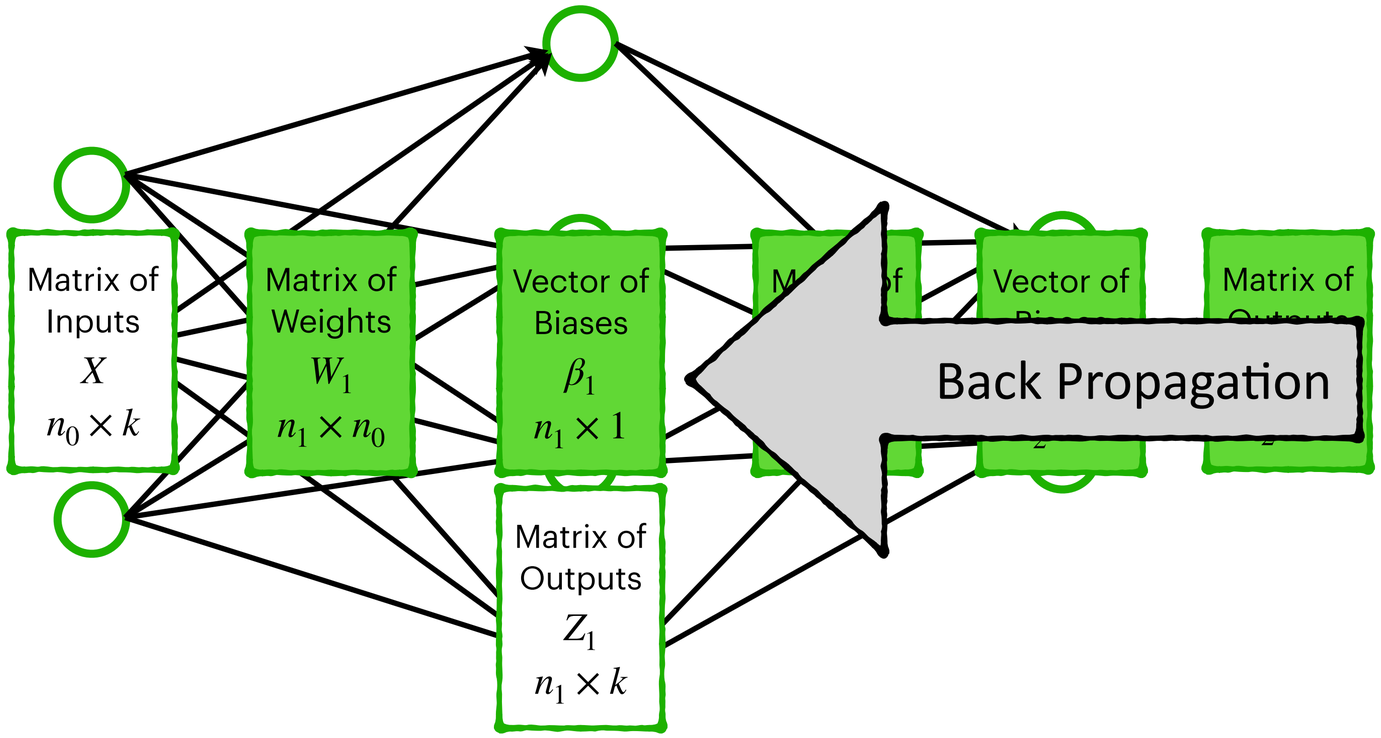
Step 4b: Compute step size… $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial W_1} cost$

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{W_1} = \frac{\partial}{\partial W_1} cost \times learning\_rate$$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and W_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and W_1 …

Step 4b: Compute step size… $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial W_1} cost$

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

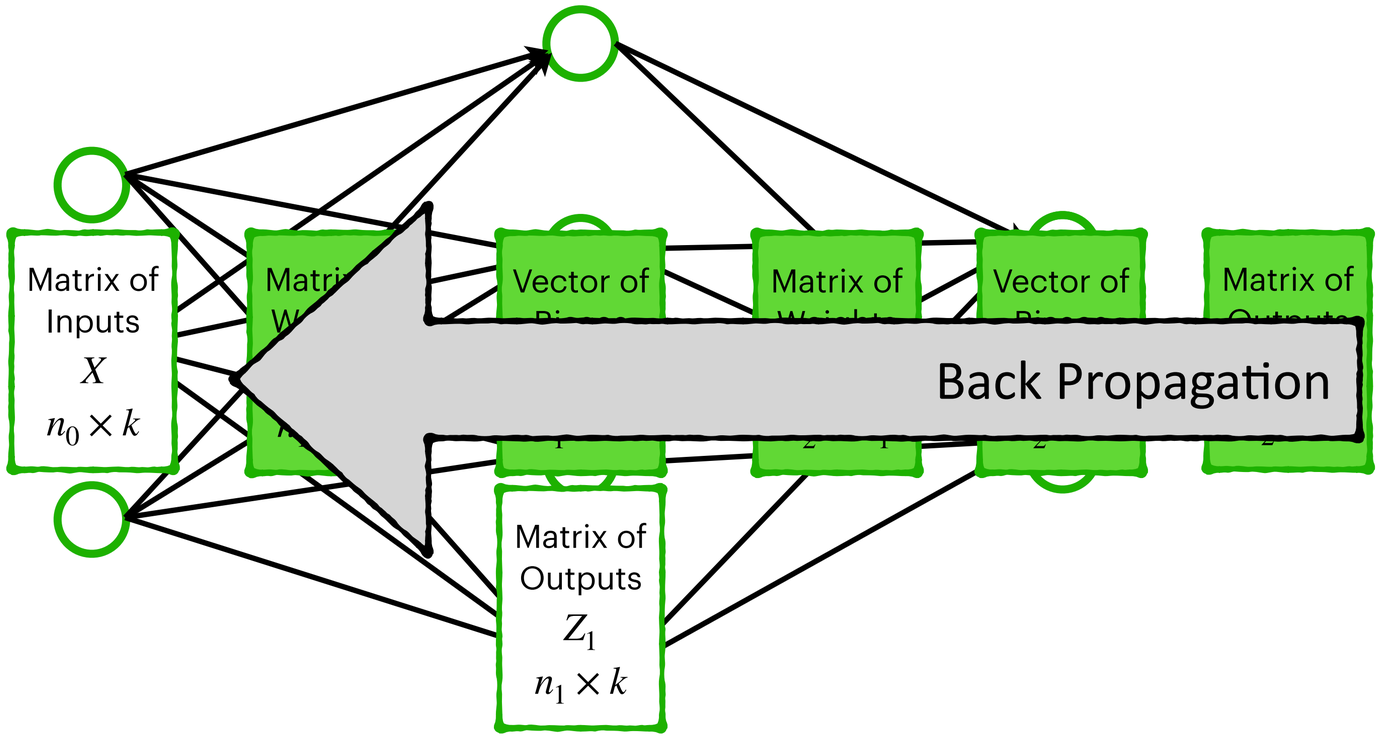
$$step\_size_{W_1} = \frac{\partial}{\partial W_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and W_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \quad W_1 = W_1 - step\_size_{W_1}$$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and W_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and W_1 …

Step 4b: Compute step size… $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial W_1} cost$

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

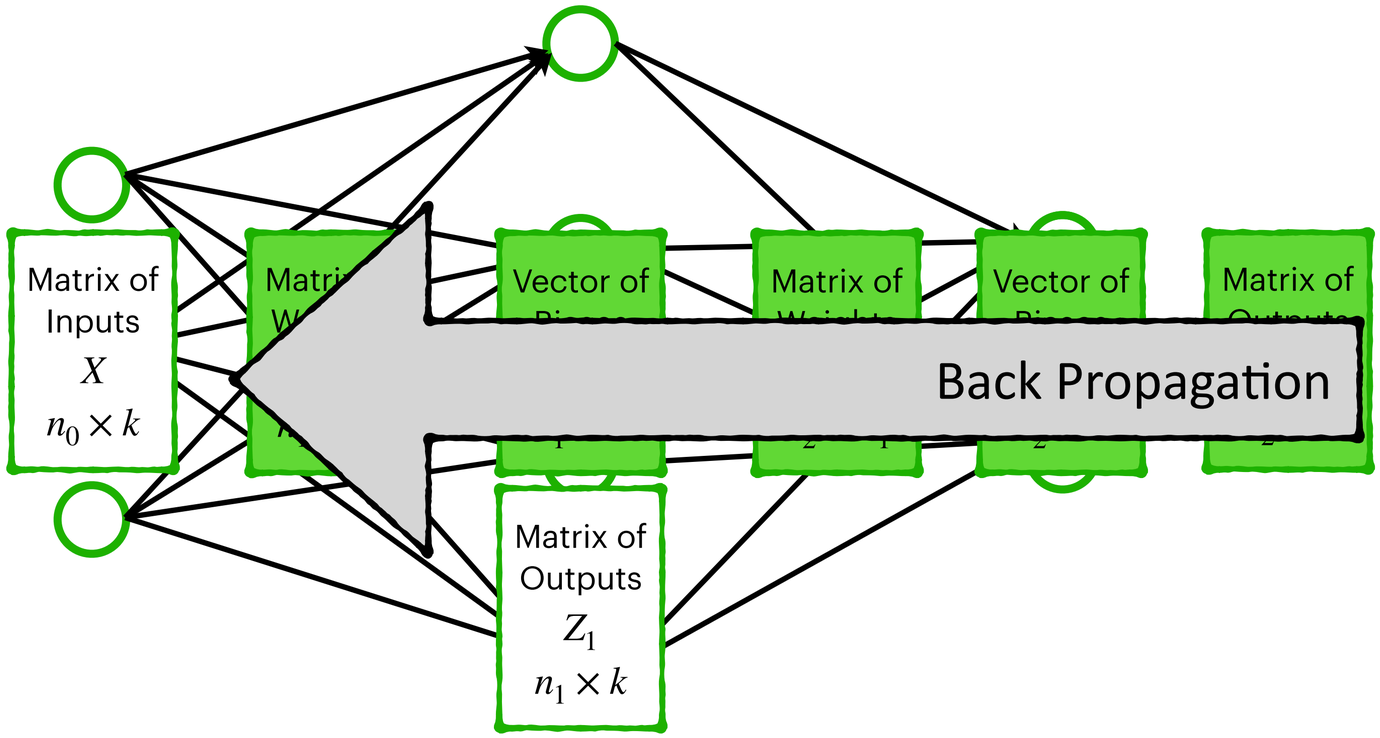
$$step\_size_{W_1} = \frac{\partial}{\partial W_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and W_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \quad W_1 = W_1 - step\_size_{W_1}$$

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$



Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Step 3a: Compute the partial derivative of the cost function w.r.t the parameters β_2 and W_2 …

Step 3b: Compute step size…

Step 3c: Compute new values for β_2 and W_2

Step 4a: Compute the partial derivative of the cost function w.r.t the parameters β_1 and W_1 …

Step 4b: Compute step size… $\frac{\partial}{\partial \beta_1} cost$ $\frac{\partial}{\partial W_1} cost$

$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{W_1} = \frac{\partial}{\partial W_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and W_1

$$\beta_1 = \beta_1 - step\_size_{\beta_1} \quad W_1 = W_1 - step\_size_{W_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

$$Z_1 = f(W_1 X + \beta_1)$$
$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$

Gradient Descent

Step 1: Start with initial values for $W_1, W_2, \beta_1, \beta_2$

Step 2a: Compute… $Z_1 = f(W_1 X + \beta_1)$

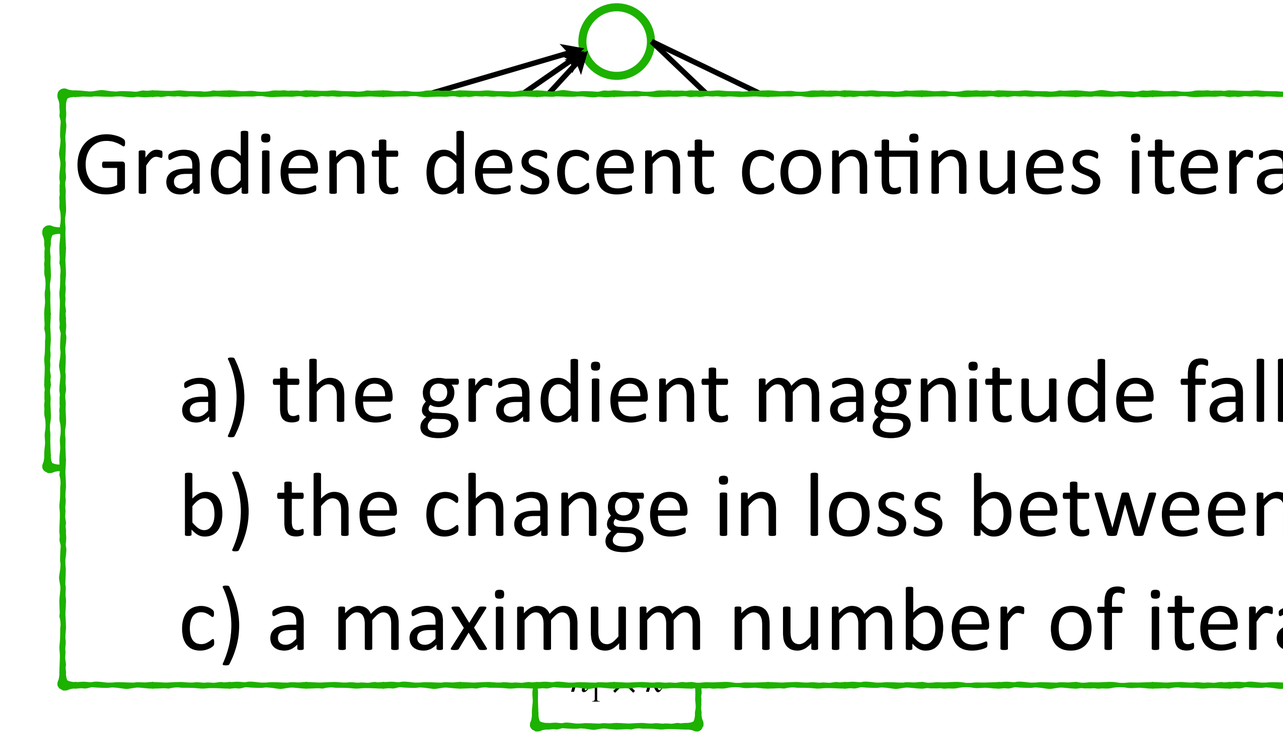
Step 2b: Compute… $\hat{Y} = f(W_2 Z_1 + \beta_2)$

Gradient descent continues iteratively until either…

a) the gradient magnitude falls below a specified threshold,

b) the change in loss between iterations becomes negligible,

c) a maximum number of iterations is reached



$$step\_size_{\beta_1} = \frac{\partial}{\partial \beta_1} cost \times learning\_rate$$

$$step\_size_{W_1} = \frac{\partial}{\partial W_1} cost \times learning\_rate$$

Step 4c: Compute new values for β_1 and W_1

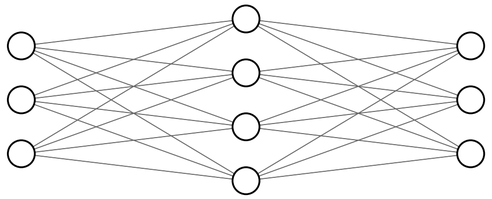
$$\beta_1 = \beta_1 - step\_size_{\beta_1} \qquad W_1 = W_1 - step\_size_{W_1}$$

Step 5: Go to step 2 and repeat

Neural Networks

Let recap and generalize the equations for Forward and Back Propagation in Neural Networks

Neural Networks



3 Layers

Forward Propagation

$$Z_1 = f(W_1 X + \beta_1)$$

$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$

Back Propagation

$$\beta_2 = \beta_2 - learning\_rate \times \frac{\partial}{\partial \beta_2} cost$$

$$W_2 = W_2 - learning\_rate \times \frac{\partial}{\partial W_2} cost$$

$$\beta_1 = \beta_1 - learning\_rate \times \frac{\partial}{\partial \beta_1} cost$$

$$W_1 = W_1 - learning\_rate \times \frac{\partial}{\partial W_1} cost$$

4 Layers

$$Z_1 = f(W_1 X + \beta_1)$$

$$Z_2 = f(W_2 Z_1 + \beta_2)$$

$$\hat{Y} = f(W_3 Z_2 + \beta_3)$$

$$\beta_3 = \beta_3 - learning\_rate \times \frac{\partial}{\partial \beta_3} cost$$

$$W_3 = W_3 - learning\_rate \times \frac{\partial}{\partial W_3} cost$$

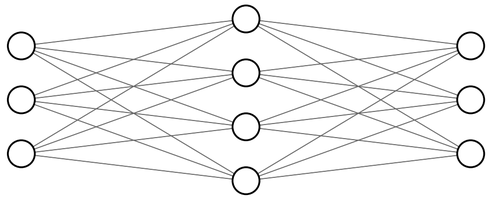
$$\beta_2 = \beta_2 - learning\_rate \times \frac{\partial}{\partial \beta_2} cost$$

$$W_2 = W_2 - learning\_rate \times \frac{\partial}{\partial W_2} cost$$

$$\beta_1 = \beta_1 - learning\_rate \times \frac{\partial}{\partial \beta_1} cost$$

$$W_1 = W_1 - learning\_rate \times \frac{\partial}{\partial W_1} cost$$

Neural Networks



3 Layers

Forward Propagation

$$Z_1 = f(W_1 X + \beta_1)$$

$$\hat{Y} = f(W_2 Z_1 + \beta_2)$$

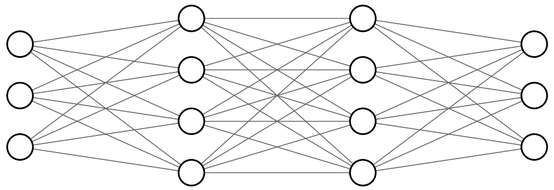
Back Propagation

$$\beta_2 = \beta_2 - learning\_rate \times \frac{\partial}{\partial \beta_2} cost$$

$$W_2 = W_2 - learning\_rate \times \frac{\partial}{\partial W_2} cost$$

$$\beta_1 = \beta_1 - learning\_rate \times \frac{\partial}{\partial \beta_1} cost$$

$$W_1 = W_1 - learning\_rate \times \frac{\partial}{\partial W_1} cost$$



4 Layers

$$Z_1 = f(W_1 X + \beta_1)$$

$$Z_2 = f(W_2 Z_1 + \beta_2)$$

$$\hat{Y} = f(W_3 Z_2 + \beta_3)$$

$$\beta_3 = \beta_3 - learning\_rate \times \frac{\partial}{\partial \beta_3} cost$$

$$W_3 = W_3 - learning\_rate \times \frac{\partial}{\partial W_3} cost$$

$$\beta_2 = \beta_2 - learning\_rate \times \frac{\partial}{\partial \beta_2} cost$$

$$W_2 = W_2 - learning\_rate \times \frac{\partial}{\partial W_2} cost$$

$$\beta_1 = \beta_1 - learning\_rate \times \frac{\partial}{\partial \beta_1} cost$$

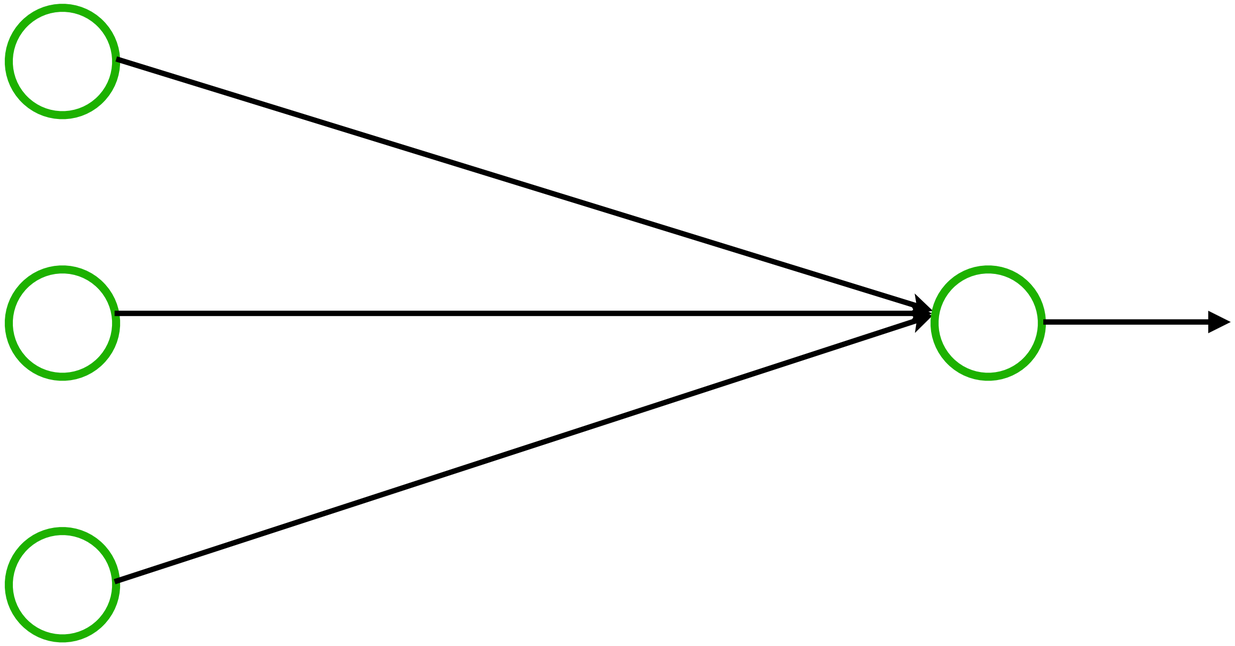
$$W_1 = W_1 - learning\_rate \times \frac{\partial}{\partial W_1} cost$$

Neural Networks

Lets look some Cost Functions…

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$n_0 = 3$

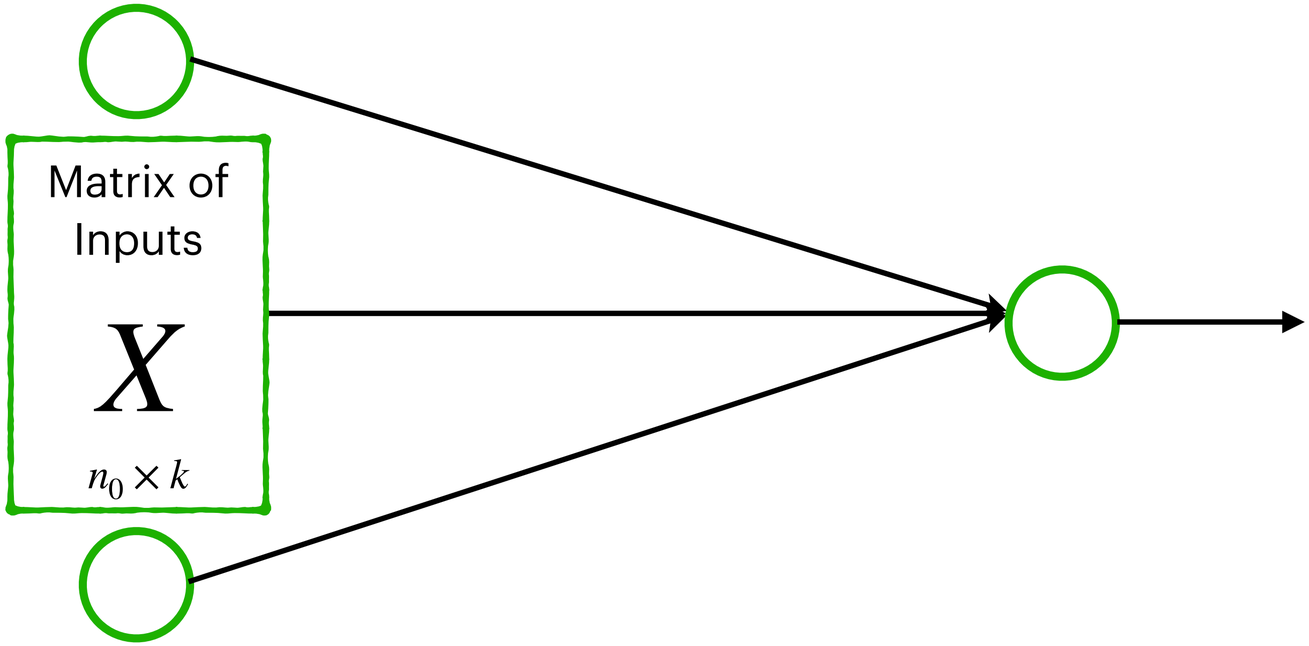
Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$n_0 = 3$

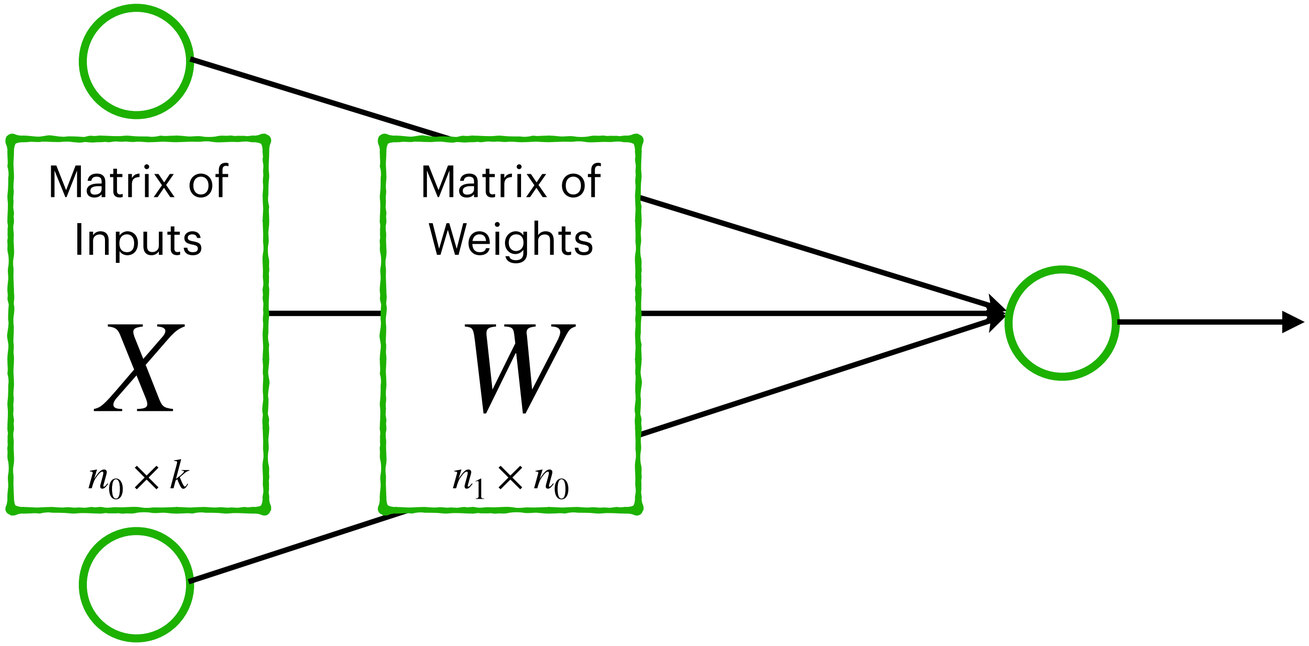
Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$n_0 = 3$

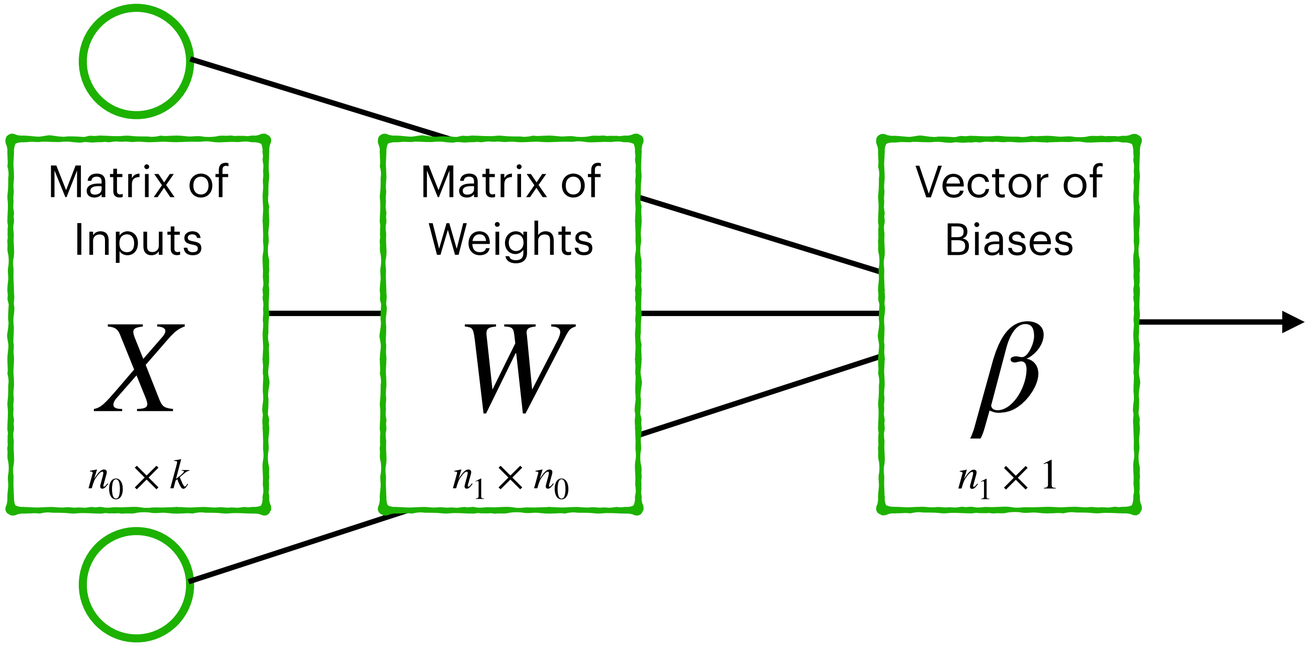
Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$n_0 = 3$

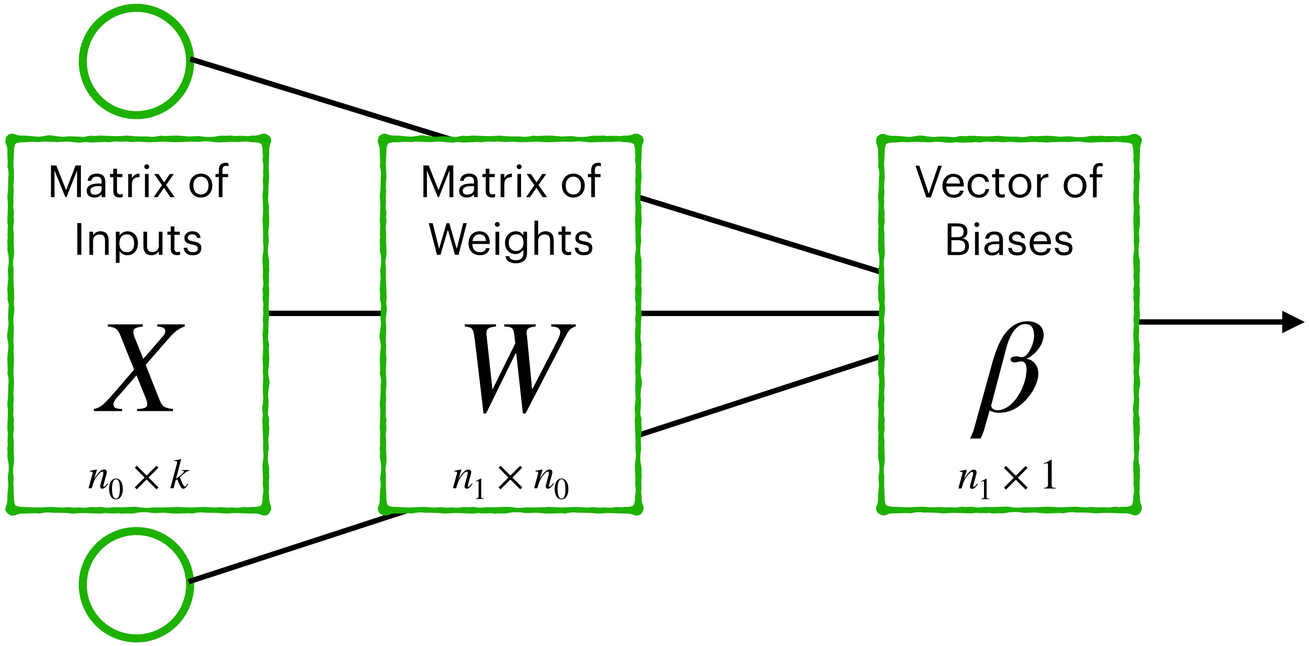
Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$$\hat{Y} = WX + \beta$$

$n_0 = 3$

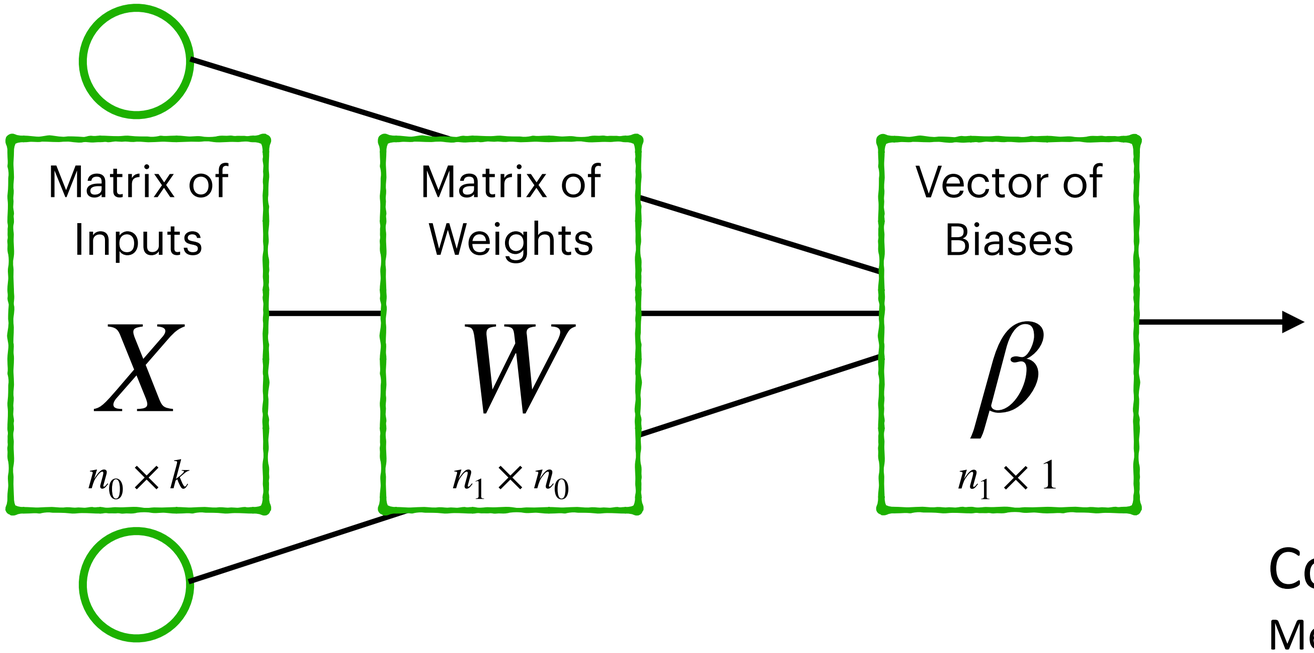
Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$$\hat{Y} = WX + \beta$$

Cost Function

Mean Squared Error (MSE)

$$\frac{1}{2k} \| Y - \hat{Y} \|^2$$

$n_0 = 3$

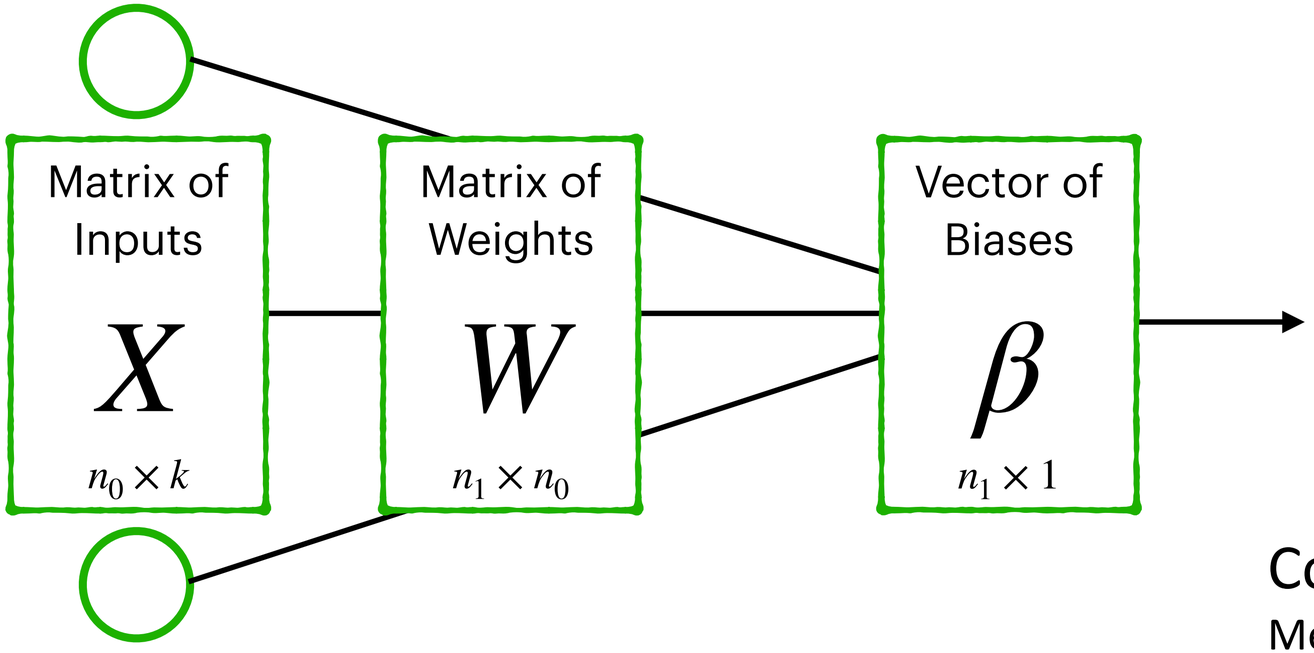
Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Cost Function for Linear Regression

For the details on Multiple Regression see
Multiple Regression



$$\hat{Y} = WX + \beta$$

Cost Function
Mean Squared Error (MSE)

$$\frac{1}{2k} \| Y - \hat{Y} \|^2$$

Cost Function
Mean Absolute Error (MAE)

$$\frac{1}{k} \| Y - \hat{Y} \|$$

$n_0 = 3$
Input Layer (L_0)

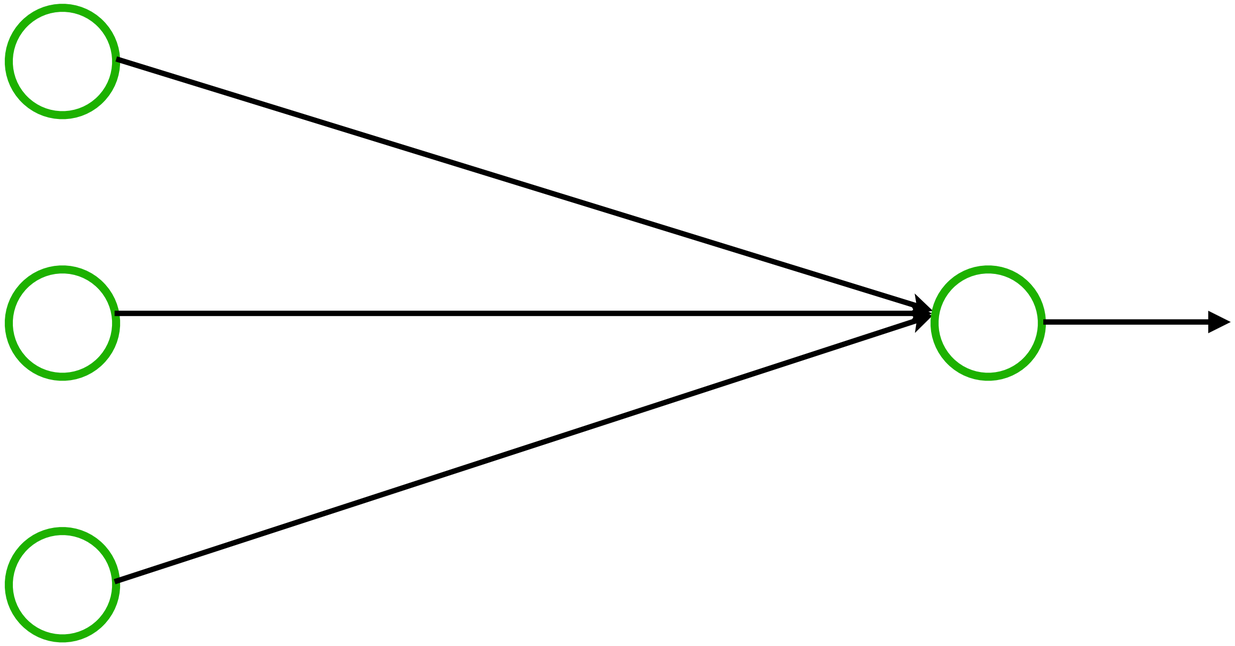
$n_1 = 1$
Output Layer (L_1)

Cost Function for Binary Classification

Logistic Regression

Neural Networks

For the details on Logistic Regression see
Logistic Regression



$n_0 = 3$

Input Layer (L_0)

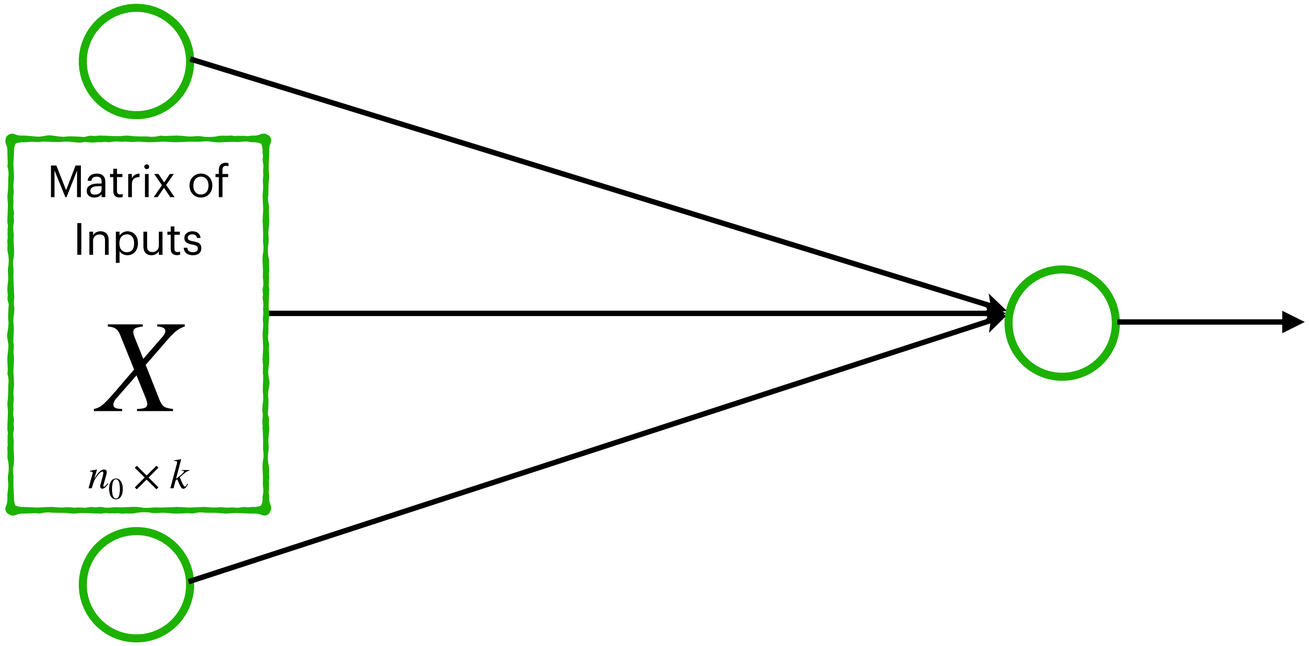
$n_1 = 1$

Output Layer (L_1)

Cost Function for Binary Classification

Logistic Regression

For the details on Logistic Regression see
Logistic Regression



$n_0 = 3$

Input Layer (L_0)

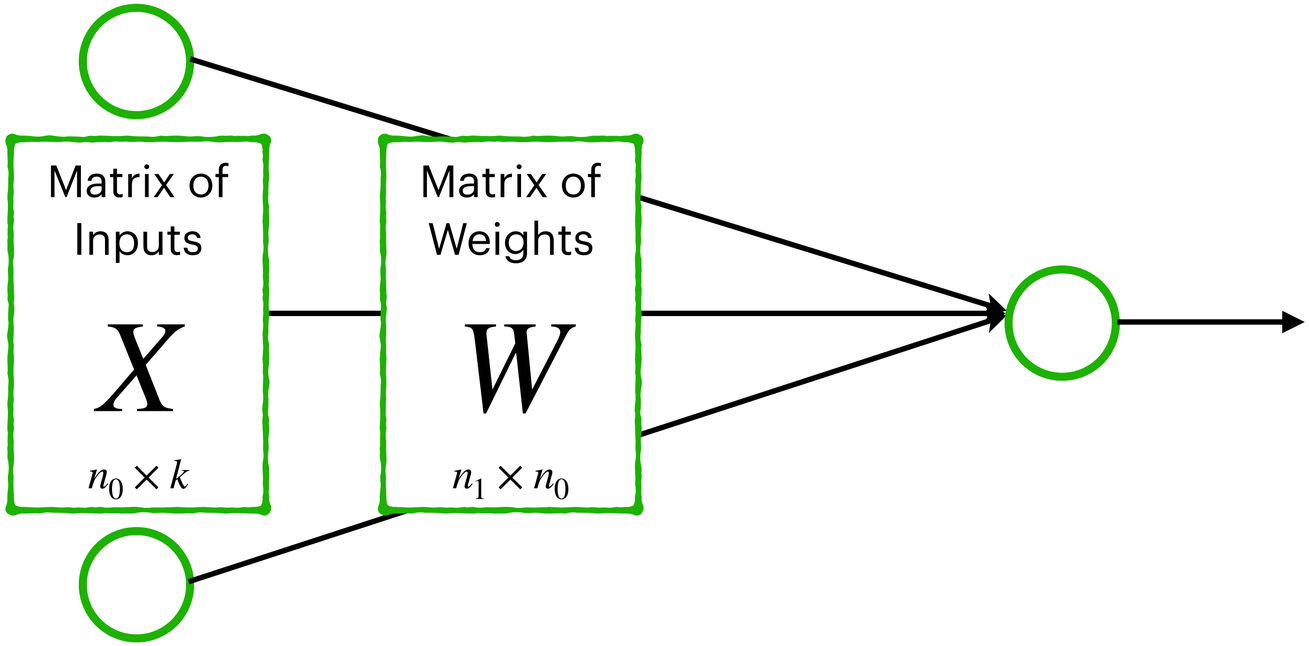
$n_1 = 1$

Output Layer (L_1)

Cost Function for Binary Classification

Logistic Regression

For the details on Logistic Regression see
Logistic Regression



$n_0 = 3$

Input Layer (L_0)

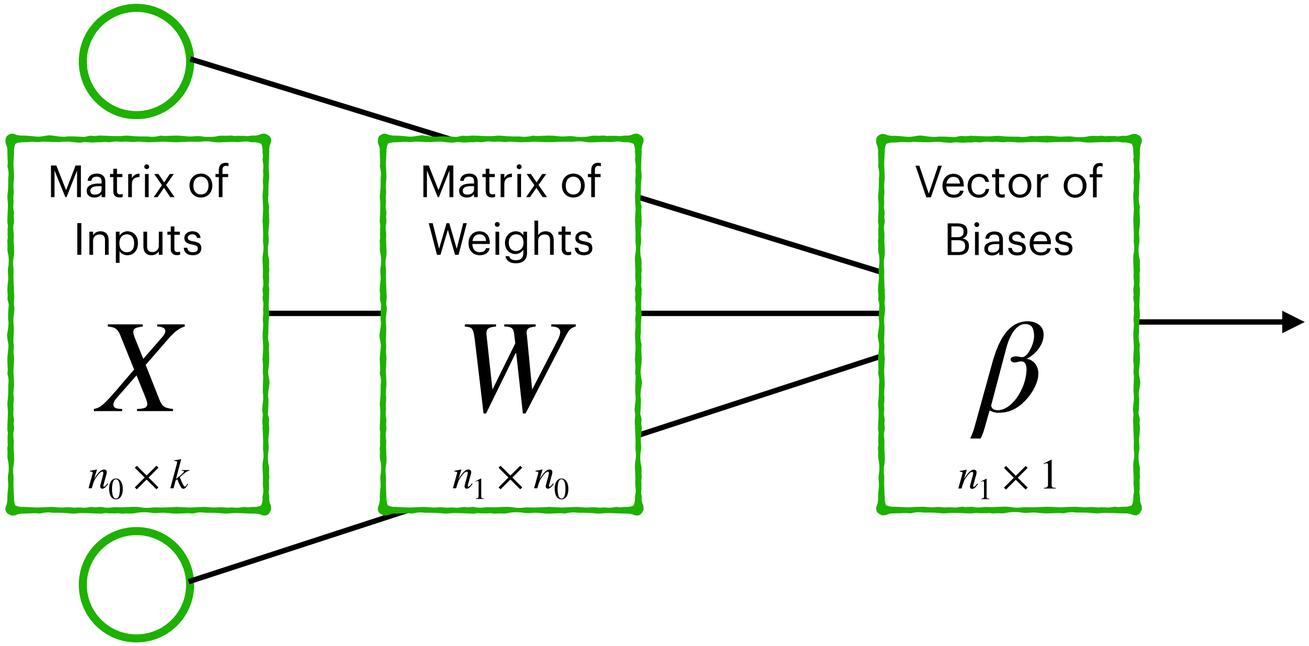
$n_1 = 1$

Output Layer (L_1)

Cost Function for Binary Classification

Logistic Regression

For the details on Logistic Regression see
Logistic Regression



$n_0 = 3$

Input Layer (L_0)

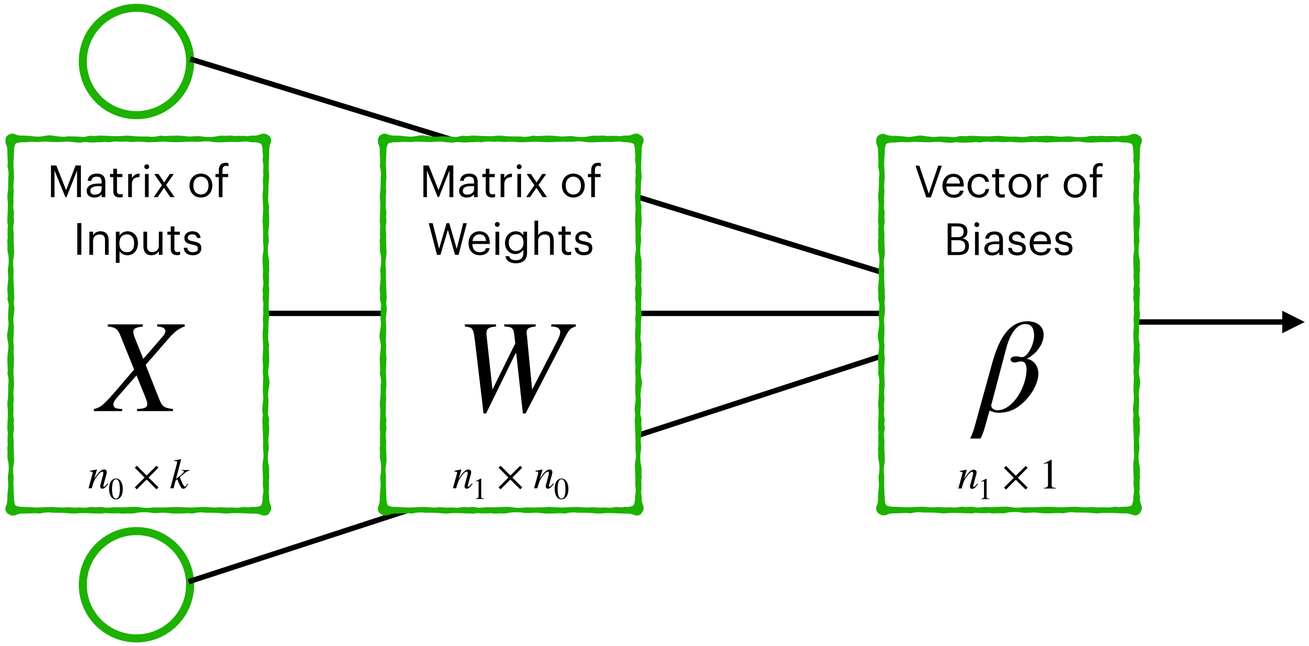
$n_1 = 1$

Output Layer (L_1)

Cost Function for Binary Classification

Logistic Regression

For the details on Logistic Regression see
Logistic Regression



$$\hat{Y} = \sigma(WX + \beta)$$

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

$n_0 = 3$

Input Layer (L_0)

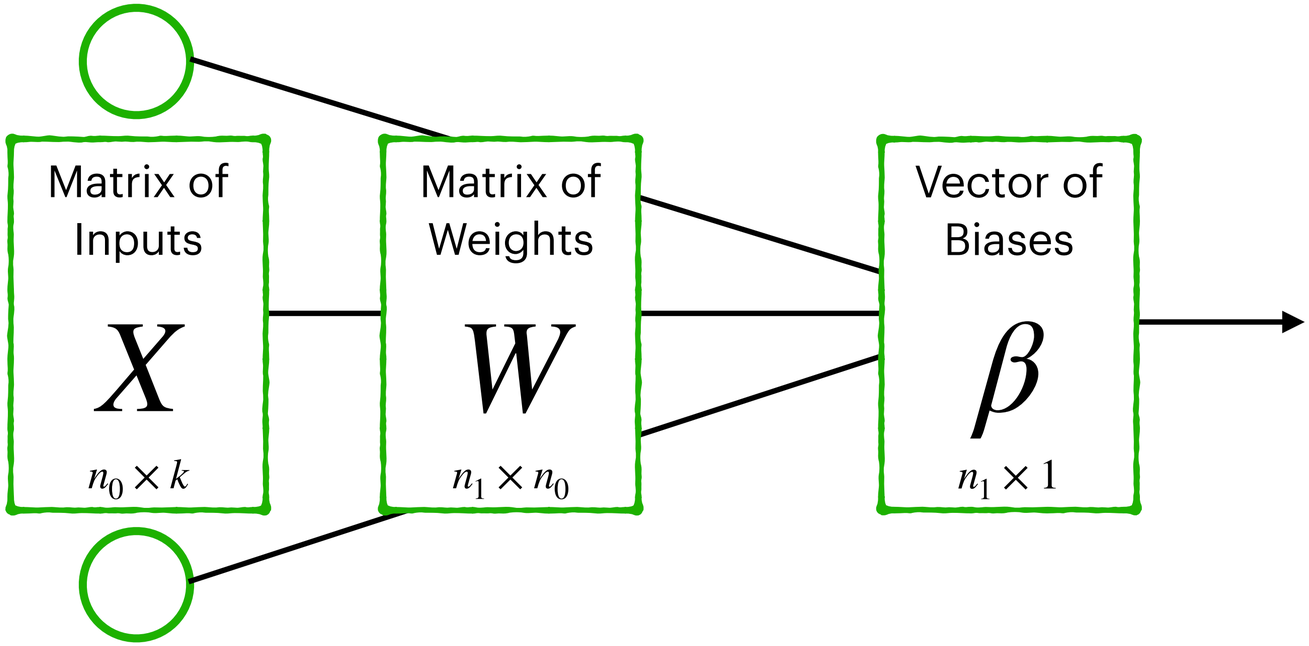
$n_1 = 1$

Output Layer (L_1)

Cost Function for Binary Classification

Logistic Regression

For the details on Logistic Regression see
Logistic Regression



$$\hat{Y} = \sigma(WX + \beta)$$

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

Cost Function

$$-\frac{1}{n}\sum_{i=1}^{n} y\, log_e \hat{y} + (1 - y)\, log_e(1 - \hat{y})$$

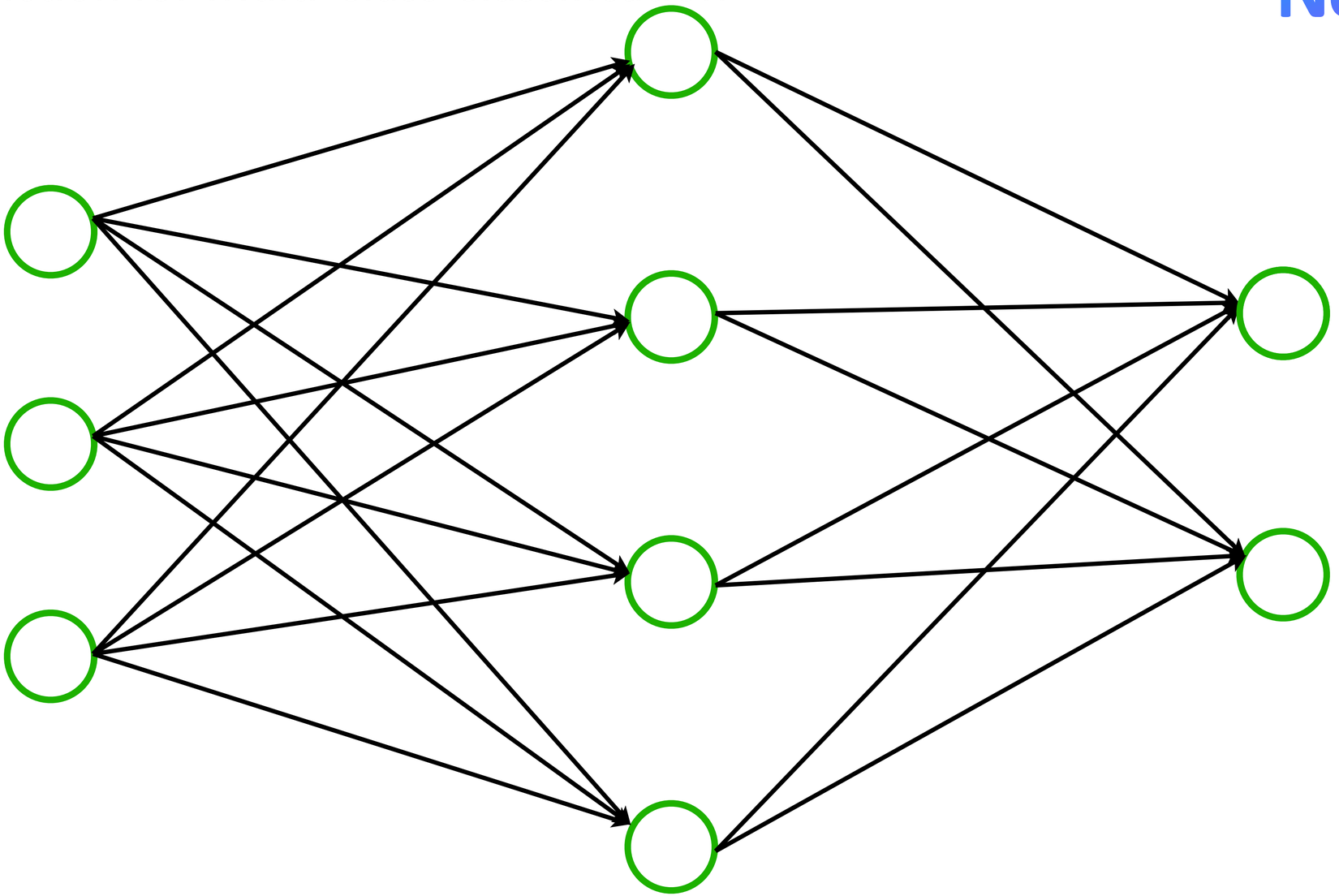
$n_0 = 3$

Input Layer (L_0)

$n_1 = 1$

Output Layer (L_1)

Neural Networks



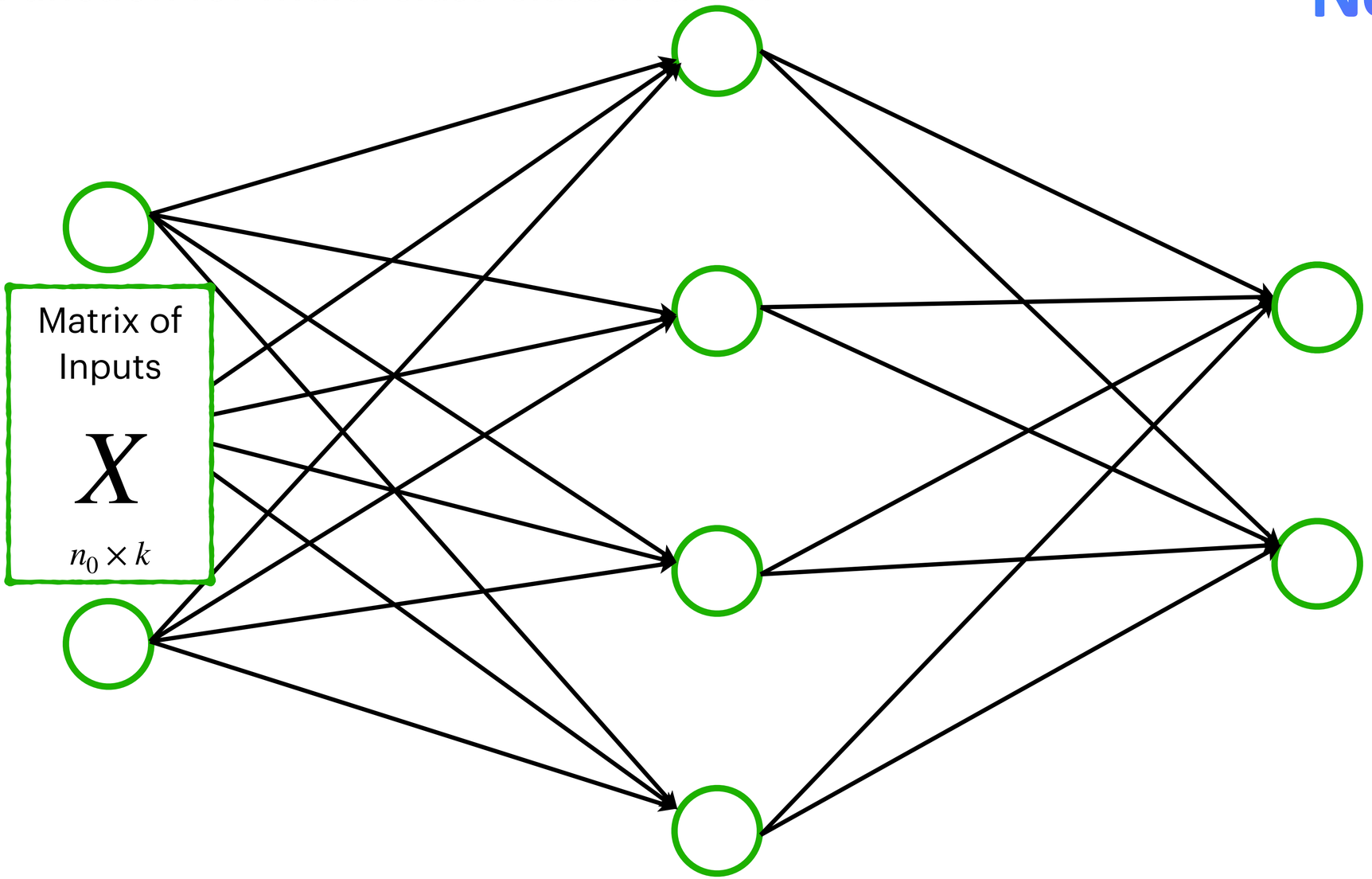
$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks



$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

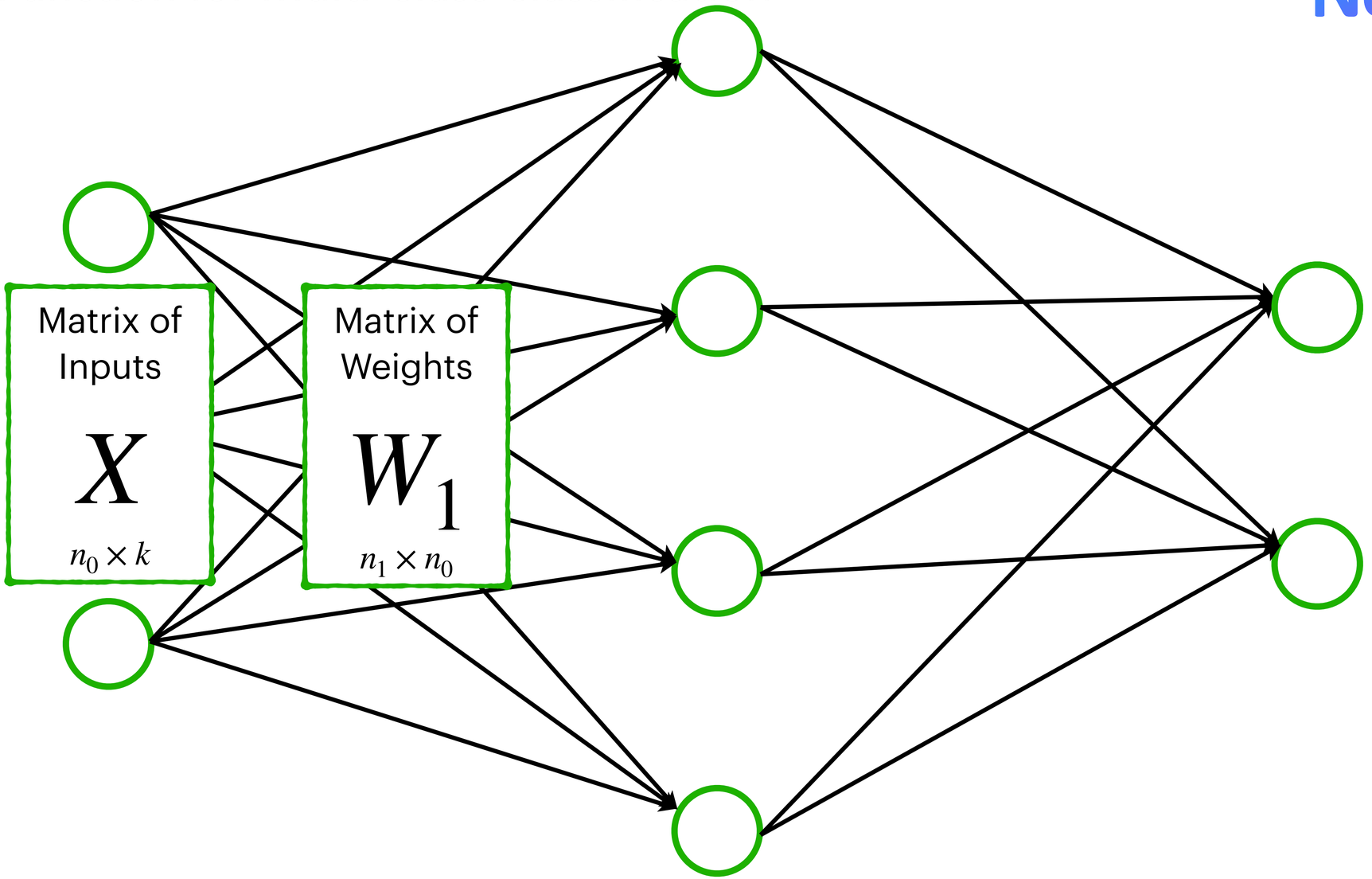
$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Cost Function for Multi-Class Classification

Neural Networks



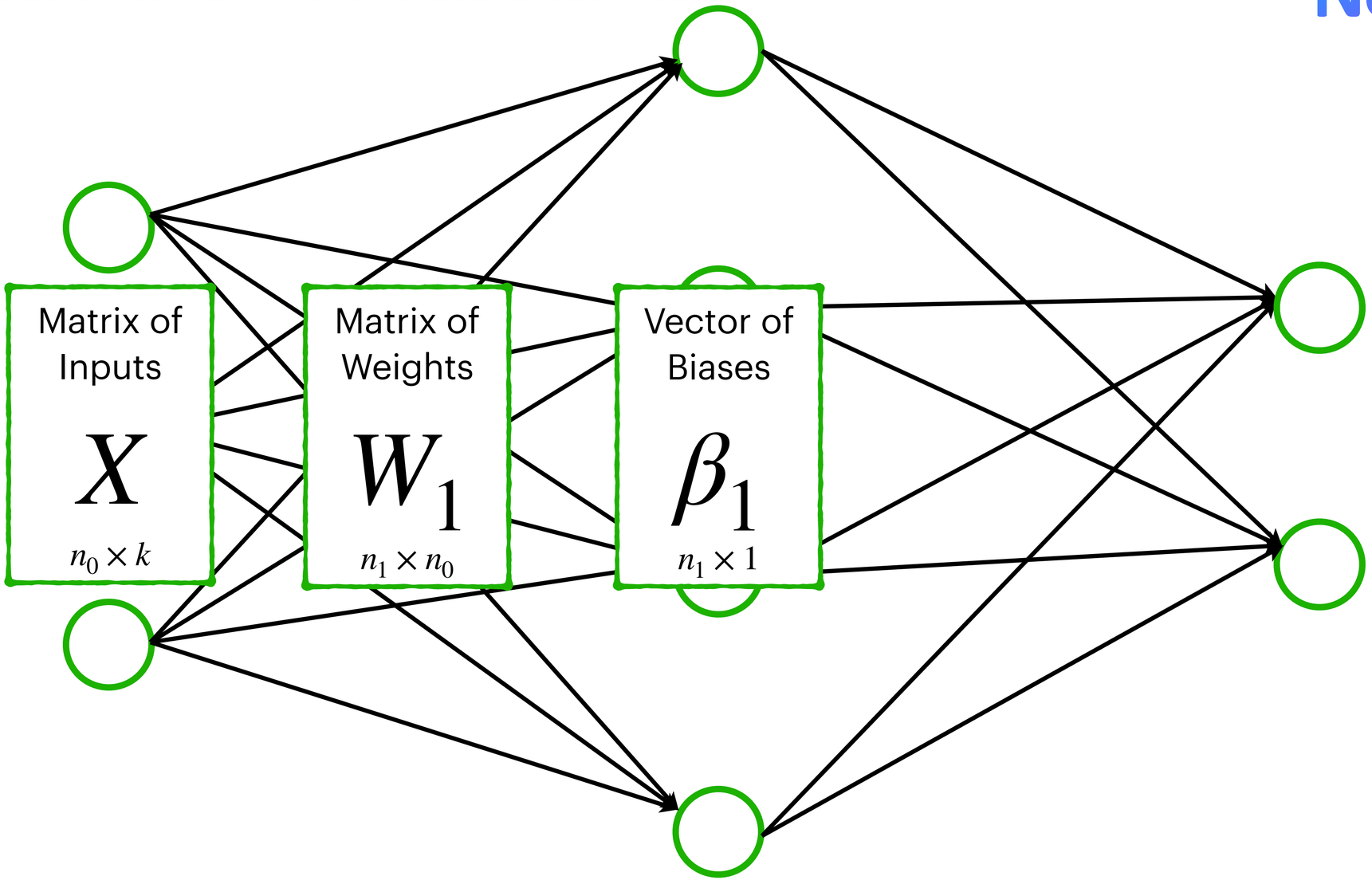
$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks



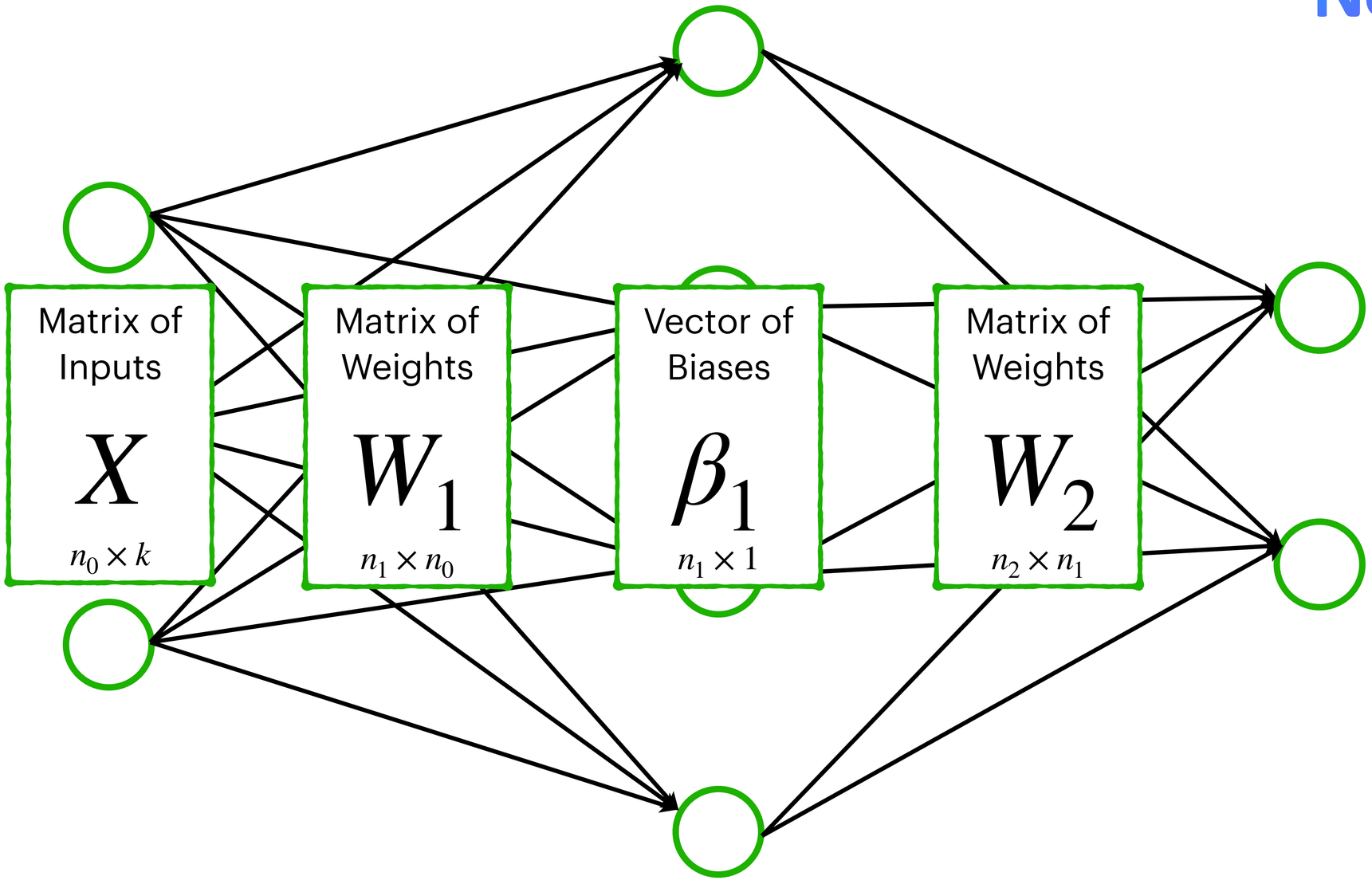
$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks



$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

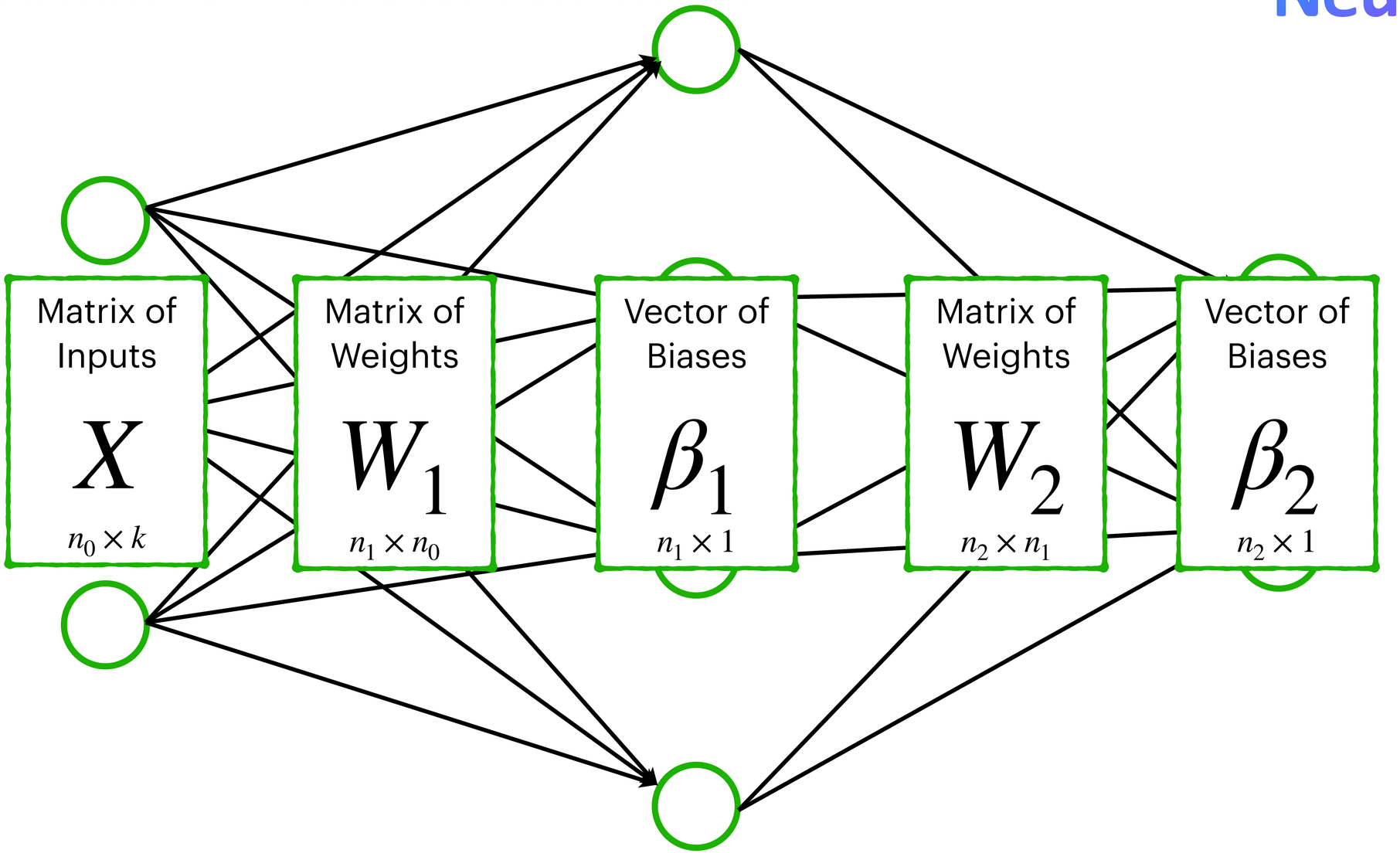
$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Cost Function for Multi-Class Classification

Neural Networks



$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

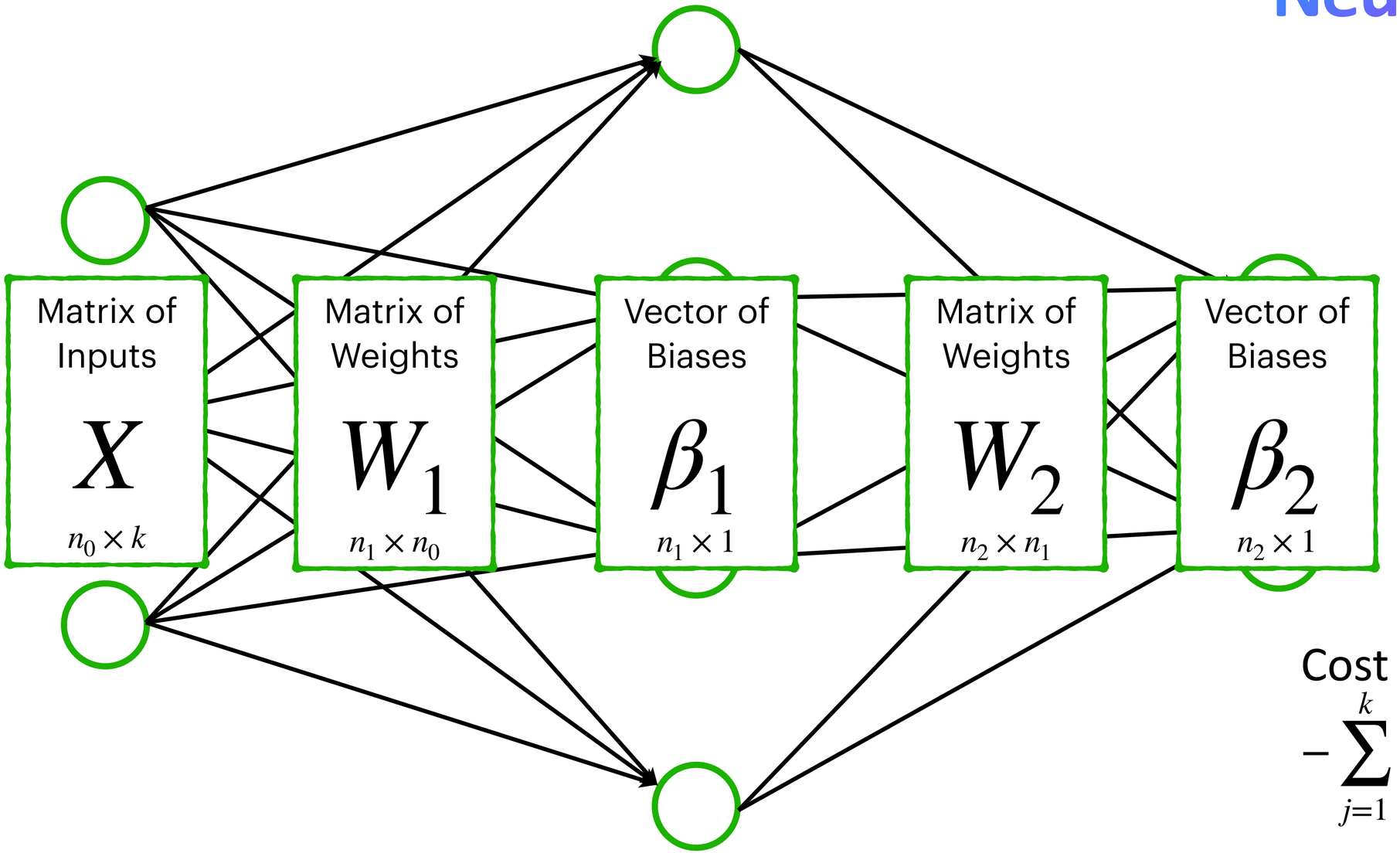
$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Neural Networks

Cost Function for Multi-Class Classification



$$\hat{Y} = \begin{bmatrix} \hat{y}_{11} \\ \hat{y}_{12} \end{bmatrix}$$

Cost Function

$$-\sum_{j=1}^{k}\sum_{i=1}^{n}(y_{ij}\, log_e \hat{y})$$

$n_0 = 3$
Input Layer (L_0)

$n_1 = 4$
Hidden Layer (L_1)

$n_2 = 2$
Output Layer (L_2)

Related Tutorials & Textbooks

Neural Networks

An introduction to Neural Networks starting from a foundation of linear regression, logistic classification and multi class classification models along with the matrix representation of a neural network generalized to *l* layers with *n* neurons

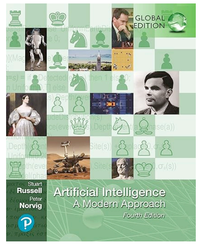
Multiple Regression

Multiple regression extends the two dimensional linear model introduced in Simple Linear Regression to k + 1 dimensions with one dependent variable, k independent variables and k+1 parameters.

Gradient Descent for Multiple Regression

Gradient Descent algorithm for multiple regression and how it can be used to optimize k + 1 parameters for a Linear model in multiple dimensions.

Recommended Textbooks



Artificial Intelligence: A Modern Approach
by Peter Norvig, Stuart Russell

For a complete list of tutorials see:
https://arrsingh.com/ai-tutorials