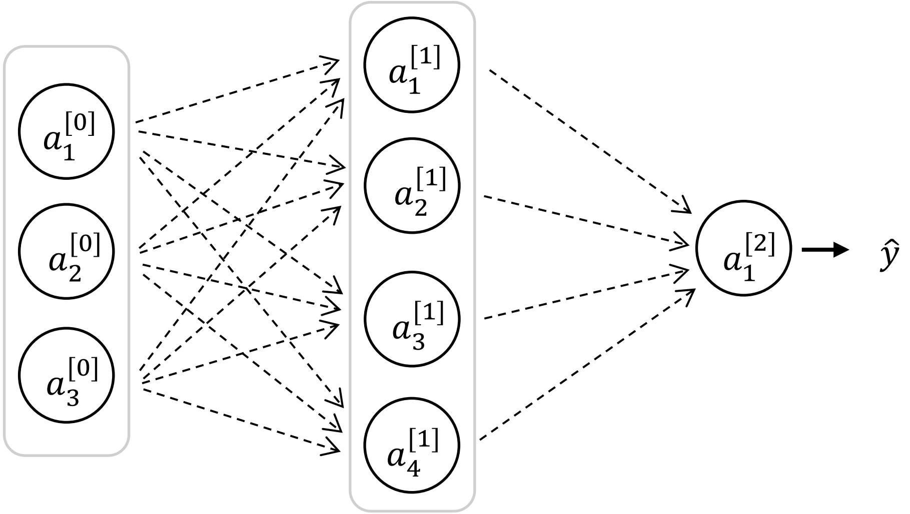
Sixteenth Session

Alireza Moradi

Backpropagation algorithm

ANN learning process

- Suppose we want to use the following Artificial Neural Network (ANN) to predict a disease:



- Before using this architecture for prediction, we must learn the parameters.

✓ In this example, how many parameters do we have?

ANN learning process

 # Parameters = # weights + # biases

- **Backpropagation** algorithm is the standard method for learning weights and biases.

- **Backpropagation** enables the network (ANN) to learn by determining the exact changes to make to weights and biases to produce an accurate result.

Training process => Learning parameters => Backpropagation

- We must initialize the network parameters to something reasonable and use **Backpropagation** algorithm to adjust them.

Backpropagation Algorithm

- You can think of **Backpropagation Algorithm** as a feedback system where, after each round of training, the network reviews its performance on tasks.

- For this purpose, it calculates the difference between its output and the correct answer, known as the **error**.

- Then, it adjusts its internal parameters to reduce this error next time.

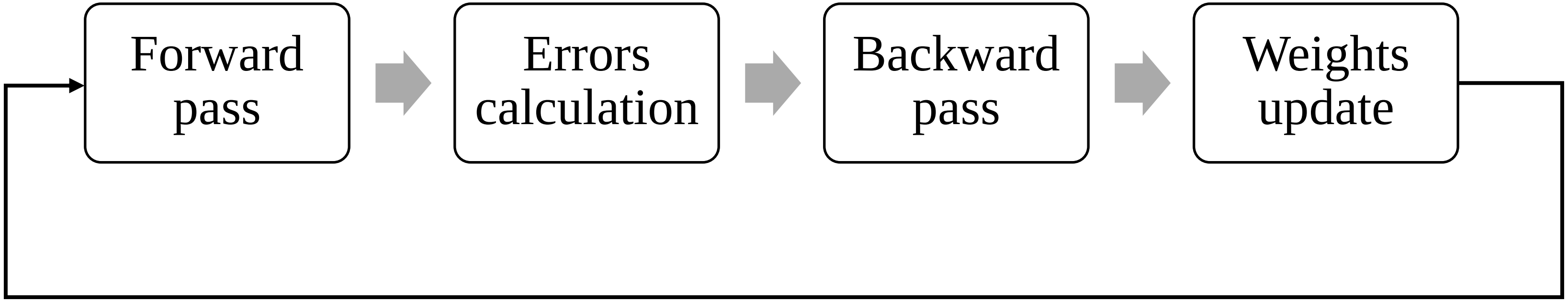
- Backpropagation algorithm was introduced in the **1970s**.

✓ How does **Gradient Descent** help us learn the best parameters in linear regression?

Backpropagation Algorithm

- There are overall four main steps in the backpropagation algorithm:

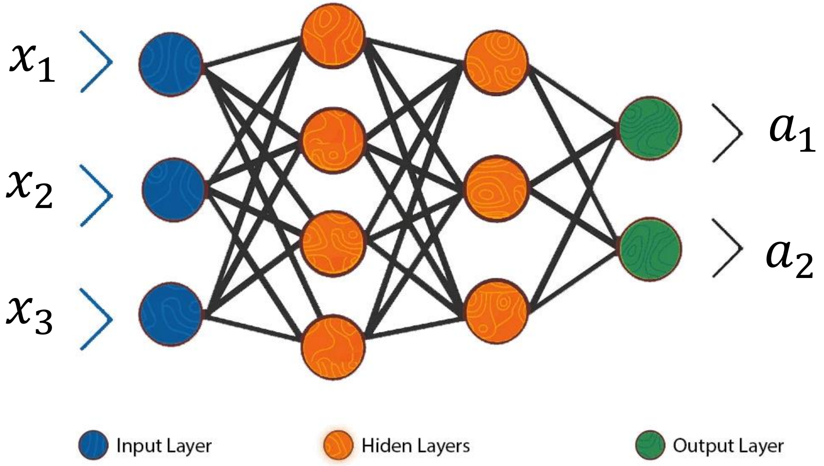
  1. Forward pass
  2. Errors calculation
  3. Backward pass
  4. Parameters update



Forward pass

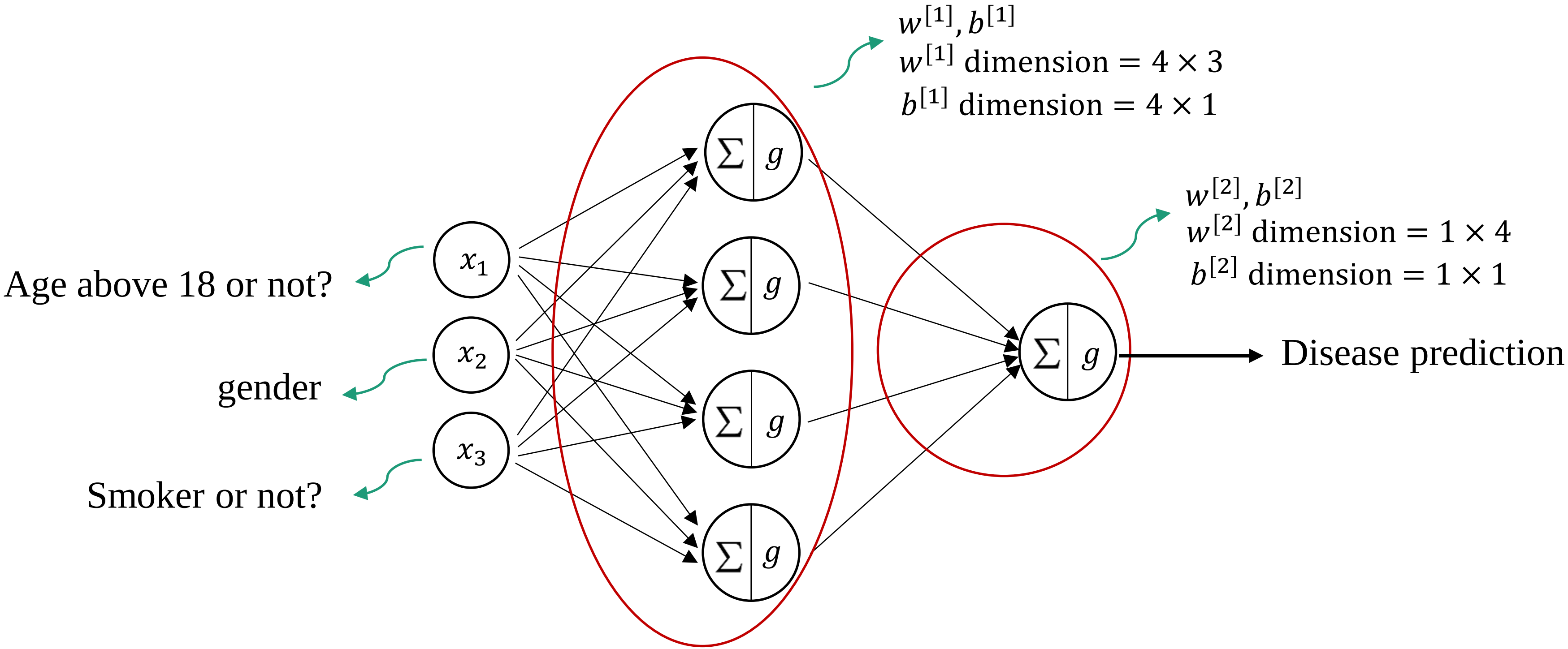
- **Forward propagation**, also known as the **forward pass**, is the process of computing the output of a neural network given a set of input data.

- It involves passing the input data through the network's layers, applying weights and activation functions, and producing predictions or outputs.

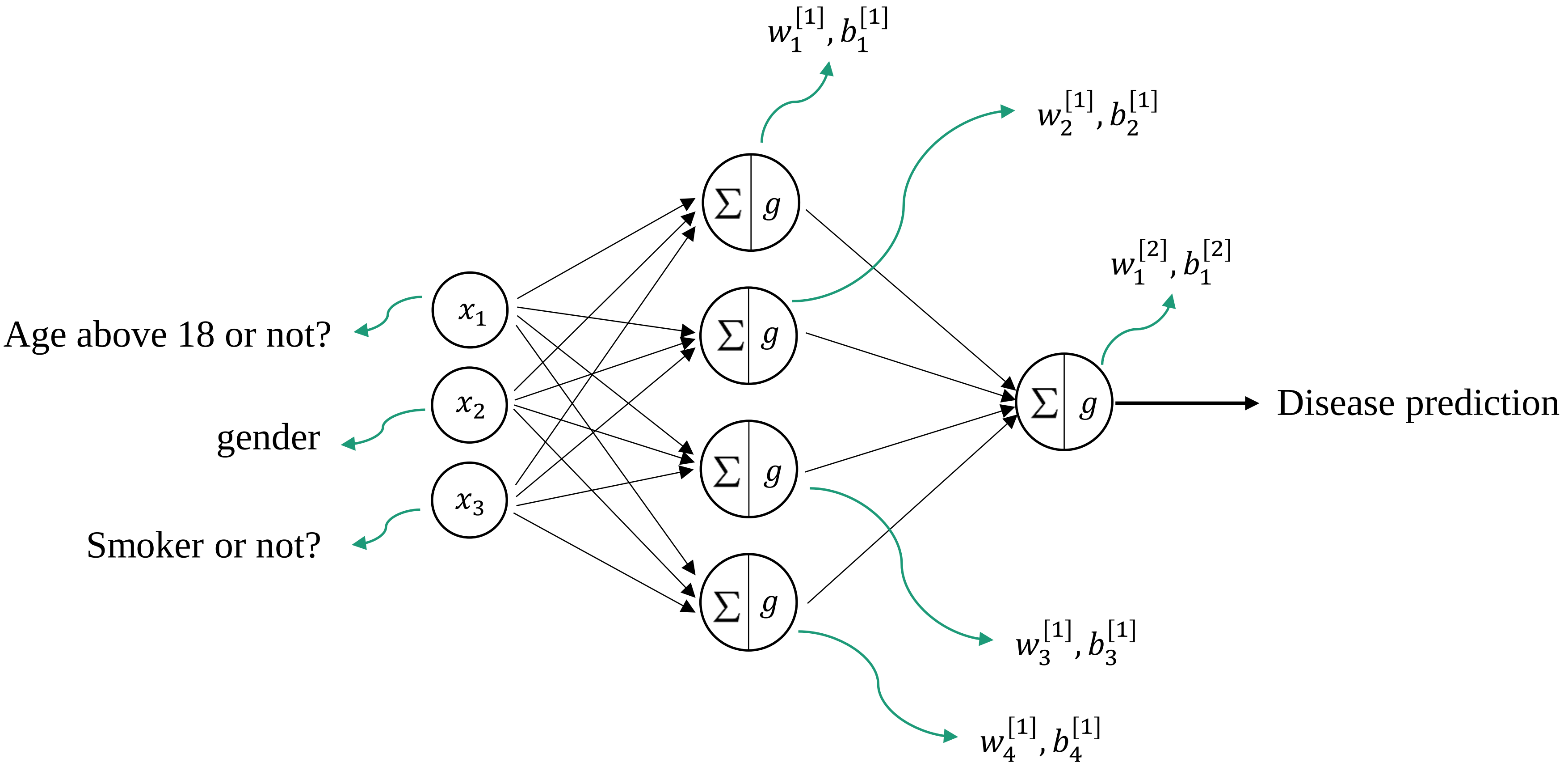


Forward pass; Example

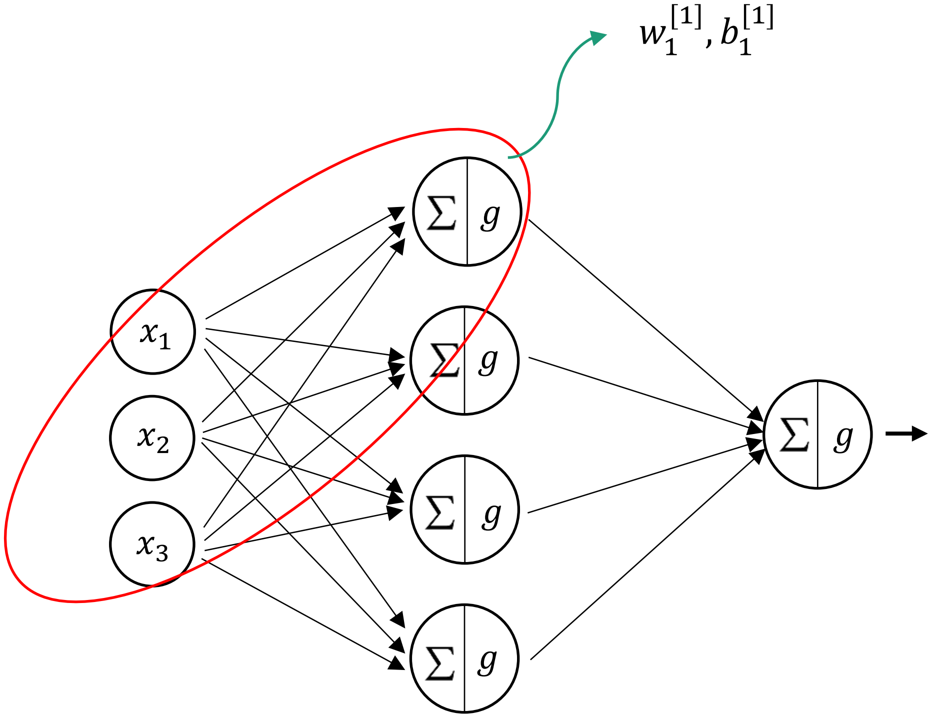
- We want to determine whether a person with certain features has diabetes or not?



Forward pass; Example



Forward pass; Example

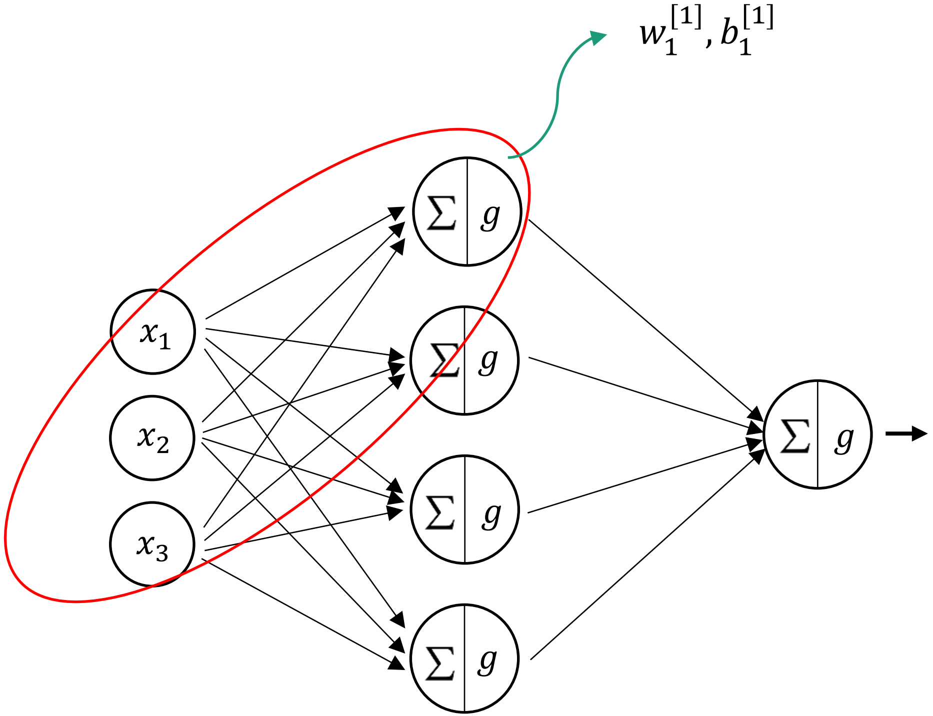


$w_1^{[1]} = [0.1 \quad 0.2 \quad 0.5]$ $\qquad b_1^{[1]} = [0]$

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$g = sigmoid$

Forward pass; Example



$w_1^{[1]} = [0.1 \quad 0.2 \quad 0.5]$ $\qquad b_1^{[1]} = [0]$

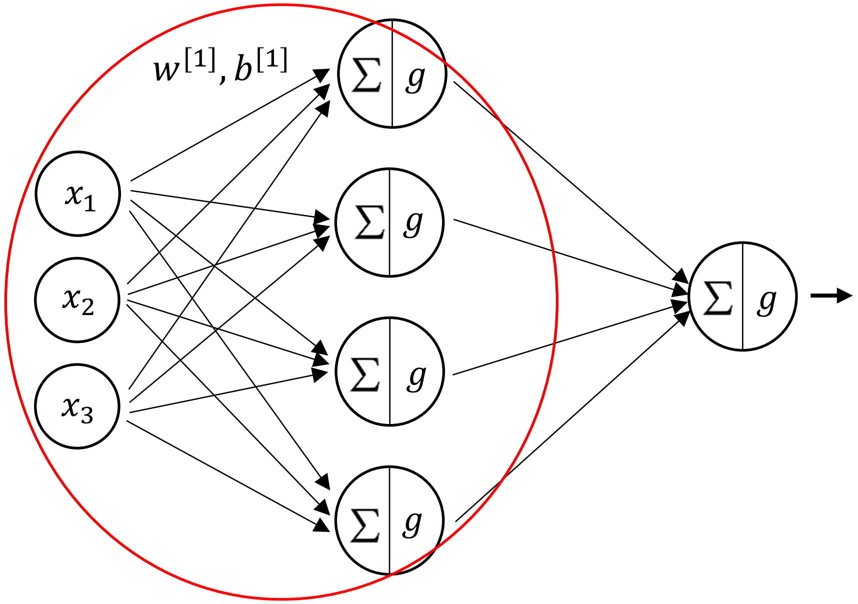
$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$g = sigmoid$

$z_1^{[1]} = w_1^{[1]}.\mathrm{x} + b_1^{[1]} =$
$0.1 \times 1 + 0.2 \times 1 + 0.5 \times 1 = 0.8$

$a_1^{[1]} = sigmoid(0.8) = \mathbf{0.68}$

Forward pass; Example

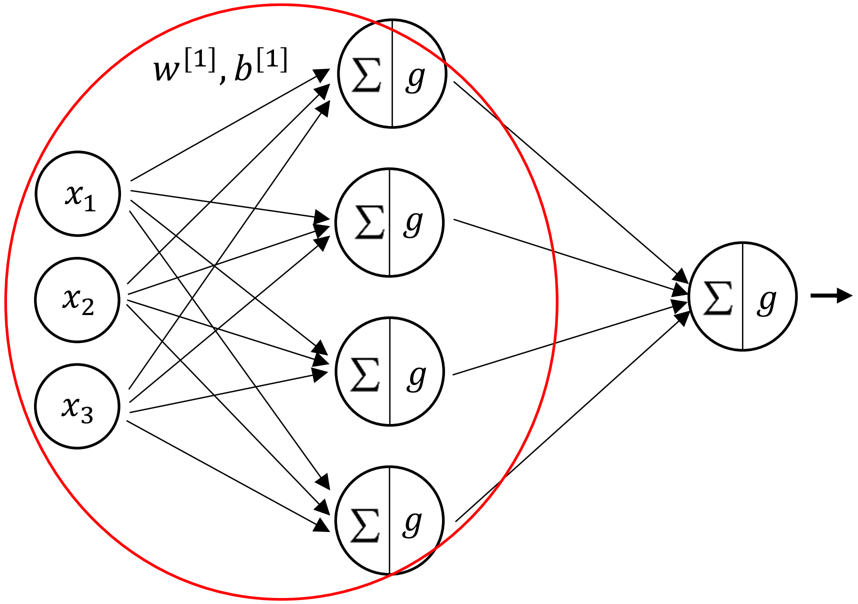


$$w^{[1]} = \begin{bmatrix} 0.1 & 0.2 & 0.5 \\ 0.2 & 0.3 & 0.1 \\ 0.1 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.9 \end{bmatrix} \qquad b^{[1]} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$g = sigmoid$$

Forward pass; Example



$$w^{[1]} = \begin{bmatrix} 0.1 & 0.2 & 0.5 \\ 0.2 & 0.3 & 0.1 \\ 0.1 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.9 \end{bmatrix} \qquad b^{[1]} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

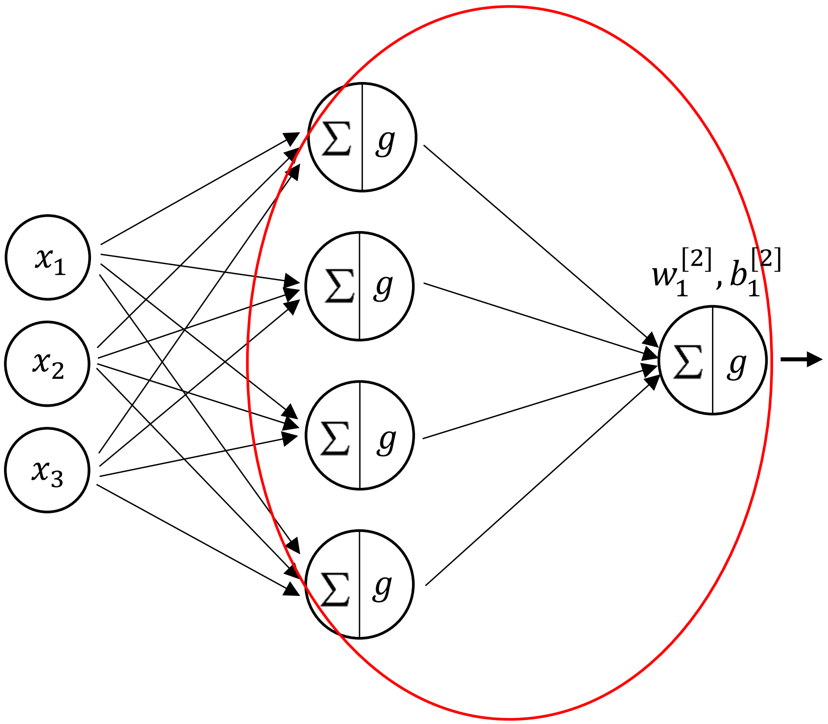
$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad g = sigmoid$$

$$z^{[1]} = w^{[1]}.x + b^{[1]} =$$

$$\begin{bmatrix} 0.1 & 0.2 & 0.5 \\ 0.2 & 0.3 & 0.1 \\ 0.1 & 0.4 & 0.6 \\ 0.2 & 0.3 & 0.9 \end{bmatrix} . \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.6 \\ 1.1 \\ 1.4 \end{bmatrix}$$

$$a^{[1]} = sigmoid\left(z^{[1]}\right) = \begin{bmatrix} 0.68 \\ 0.64 \\ 0.75 \\ 0.8 \end{bmatrix}$$

Forward pass; Example

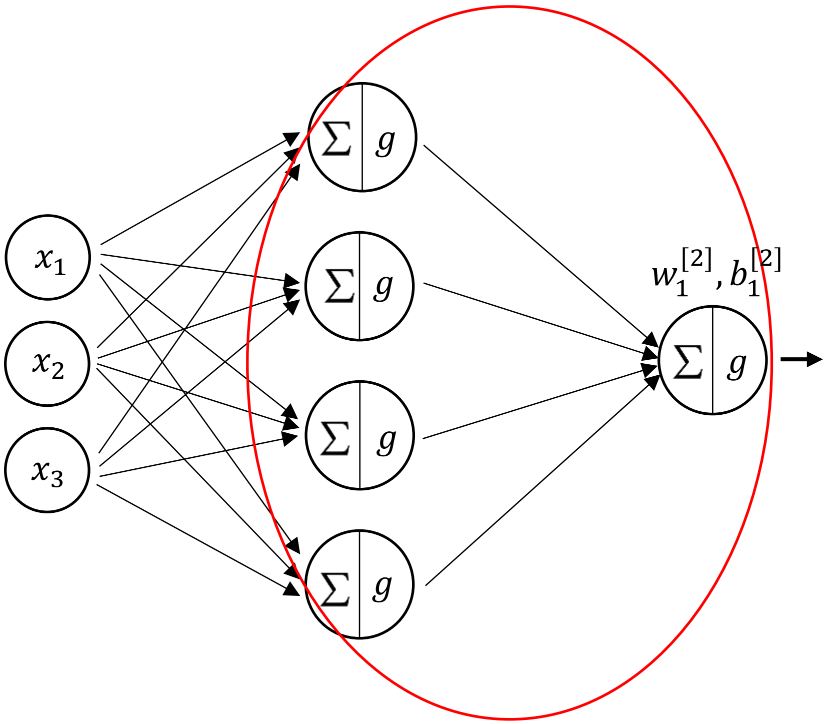


$w^{[2]} = [0.4 \quad 1.2 \quad 4.1 \quad 0.1]$ $\qquad b^{[2]} = 1.4$

$g = sigmoid$

$\hat{y} = ?$

Forward pass; Example



$$w^{[2]} = [0.4 \quad 1.2 \quad 4.1 \quad 0.1] \qquad b^{[2]} = 1.4$$

$$g = sigmoid$$

$$z^{[2]} = w^{[2]}.a^{[1]} + b^{[2]} =$$

$$[0.4 \quad 1.2 \quad 4.1 \quad 0.1].\begin{bmatrix} 0.68 \\ 0.64 \\ 0.75 \\ 0.8 \end{bmatrix} + 1.4 = 5.595$$

$$a^{[2]} = sigmoid\left(z^{[2]}\right) = \mathbf{0.996}$$

$$Prediction? \qquad 0.996 > 0.5\ then\ \hat{y} = 1$$

Error calculation

- The **error calculation** phase of backpropagation involves determining the difference between the predicted output of the neural network and the desired output, which is used to quantify the error or loss.

$$\hat{y} \text{ or } P(X) \Longleftrightarrow y$$

- This error is later then propagated backward through the network to update the weights.

- To achieve this goal of learning weights and incorporating error, we need a **cost function**.

✓ What do you think cost functions should be?

Index	y	$P(X)$	$\hat{y}$
1	1	0.996	1
2	0	0.3	0
3	0	0.001	0
…	…	…	…
n	1	0.01	0

Error calculation

- The specific method for calculating the error depends on the task and the type of network.

- For example, in **regression** problems, the mean squared error (**MSE**) or mean absolute error (**MAE**) may be used.

$$J(w,b) = \frac{1}{2n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

- In **binary classification** problems, **binary cross-entropy loss** or other appropriate loss functions may be employed.

$$J(\mathrm{w},\mathrm{b}) = -\frac{1}{n}\sum\nolimits_{i=1}^{n}\left[y^{(i)}\log\left(P_{\mathrm{w,b}}\left(X^{(i)}\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - P_{\mathrm{w},b}\left(X^{(i)}\right)\right)\right]$$

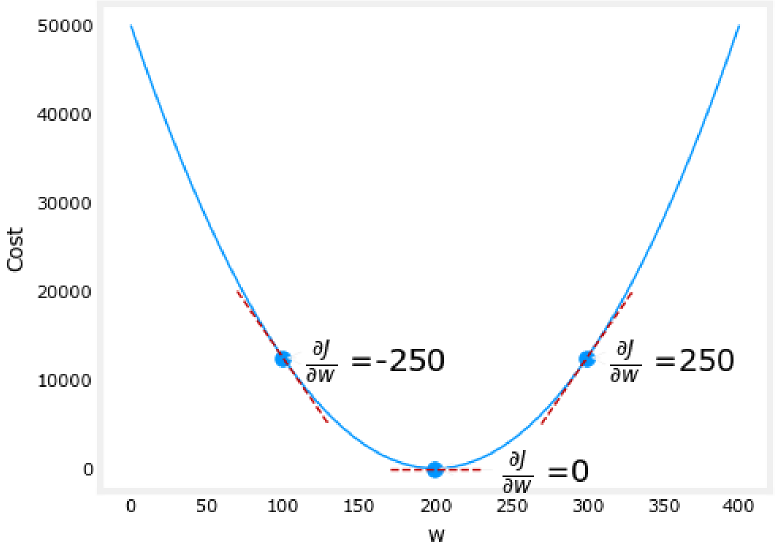
Backward pass

- We minimize our cost function to get the best parameters. Remember that cost function is a function of parameters.

$$Min\ J(w,b)$$

- Recall that in **Gradient Descent:**

If $\frac{\partial J}{\partial\, w_i} > 0$, then increasing w_i increases J.
If $\frac{\partial J}{\partial w_i} < 0$, then increasing w_i decreases J.



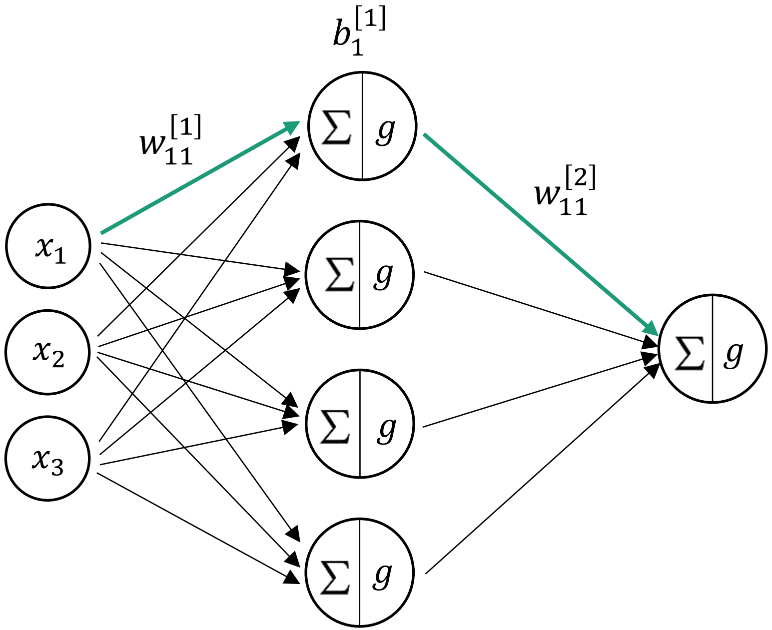
- So, the following update **decreases the cost function**:

$$w_{i,new} = w_{i,old} - \alpha \frac{\partial J}{\partial w_i}$$

Backward pass

- We must compute the gradients for each parameter.

$$w_{i,new} = w_{i,old} - \alpha \frac{\partial J}{\partial w_i}$$



$$\begin{cases} w_{11,new}^{[1]} = w_{11,old}^{[1]} - \alpha \dfrac{\partial J}{\partial w_{11}^{[1]}} \\ b_{1,new}^{[1]} = b_{1,old}^{[1]} - \alpha \dfrac{\partial J}{\partial b_1^{[1]}} \\ w_{11,new}^{[2]} = w_{11,old}^{[2]} - \alpha \dfrac{\partial J}{\partial w_{11}^{[2]}} \end{cases}$$

✓ How to compute the gradients?

Backward pass; Chain rule

- Example:

$$f(u) = 7u + u^2$$
$$u(x) = x^3$$
$$f(x) = f\big(u(x)\big) = 7x^3 + x^6$$

$$\frac{\partial f}{\partial x} = 21x^2 + 6x^5$$

$$\frac{\partial f}{\partial u} = 7 + 2u$$

$$\frac{\partial u}{\partial x} = 3x^2$$

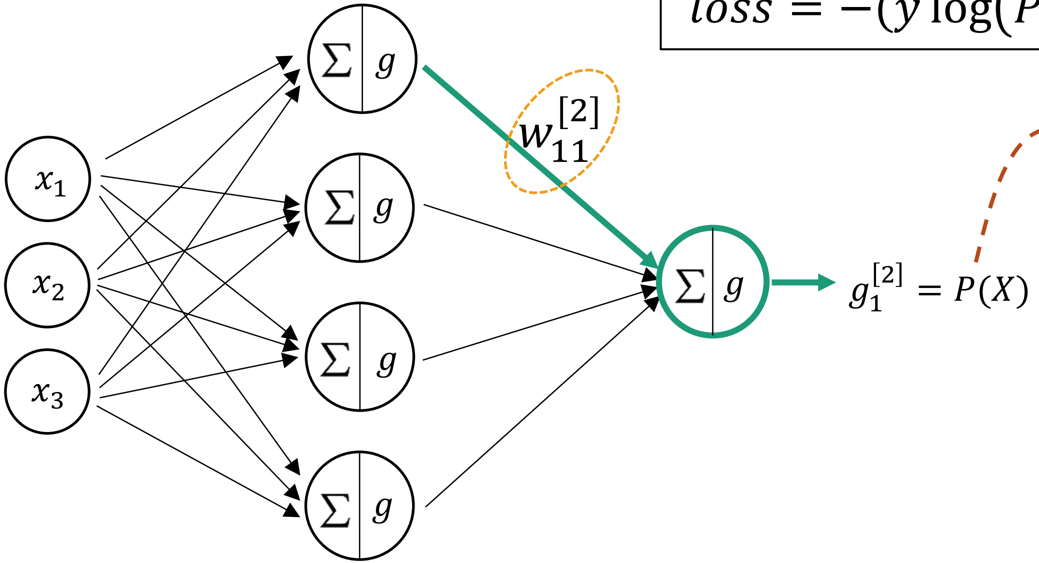
$$\boxed{\frac{\partial f}{\partial x} = \frac{\partial f}{\partial u} \times \frac{\partial u}{\partial x}}$$

$$\frac{\partial f}{\partial x} = (7 + 2u)\left(3x^2\right) = \left(7 + 2x^3\right)\left(3x^2\right) = 21x^2 + 6x^5$$

Backward pass; Example

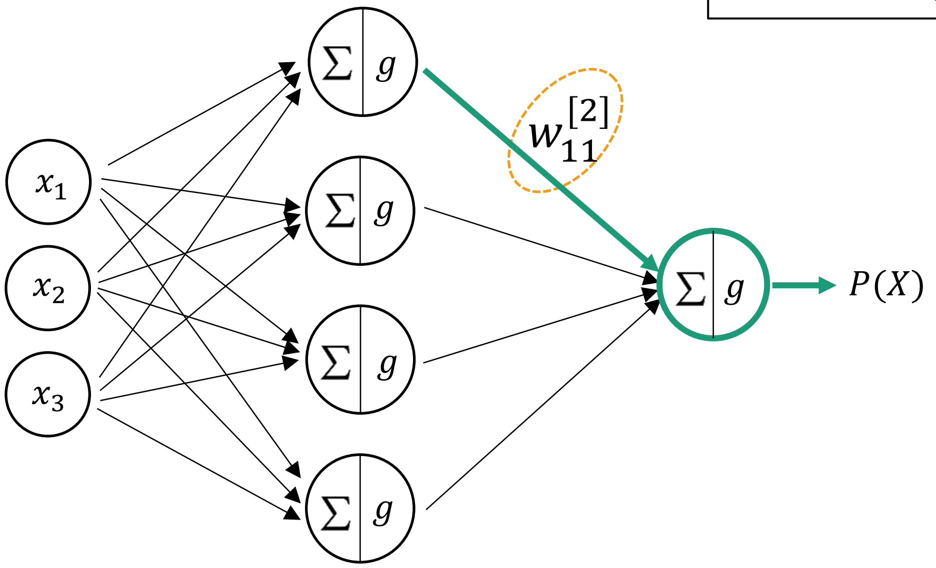
- **Backpropagation** uses the chain rule for differentiation.
- Suppose we have one observation => $n = 1$

$$\boxed{loss = -\left(y \log\big(P(X)\big) + (1 - y) \log\big(1 - P(X)\big)\right)}$$



Backward pass; Example

$$loss = -(y \log(P(X)) + (1-y)\log(1-P(X)))$$



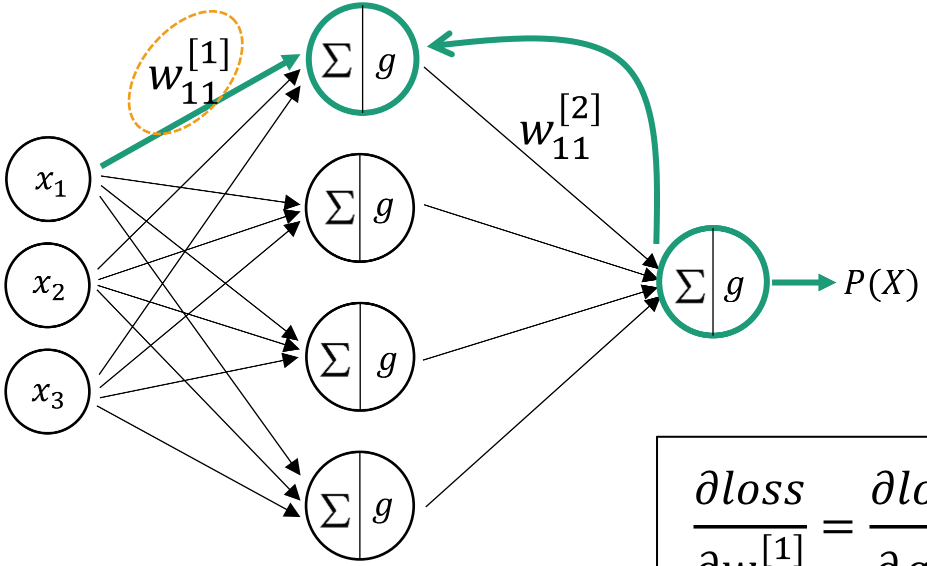
Chain rule

$$\frac{\partial loss}{\partial w_{11}^{[2]}} = \frac{\partial loss}{\partial g} \times \frac{\partial g}{\partial z_1^{[2]}} \times \frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}}$$

Backward pass

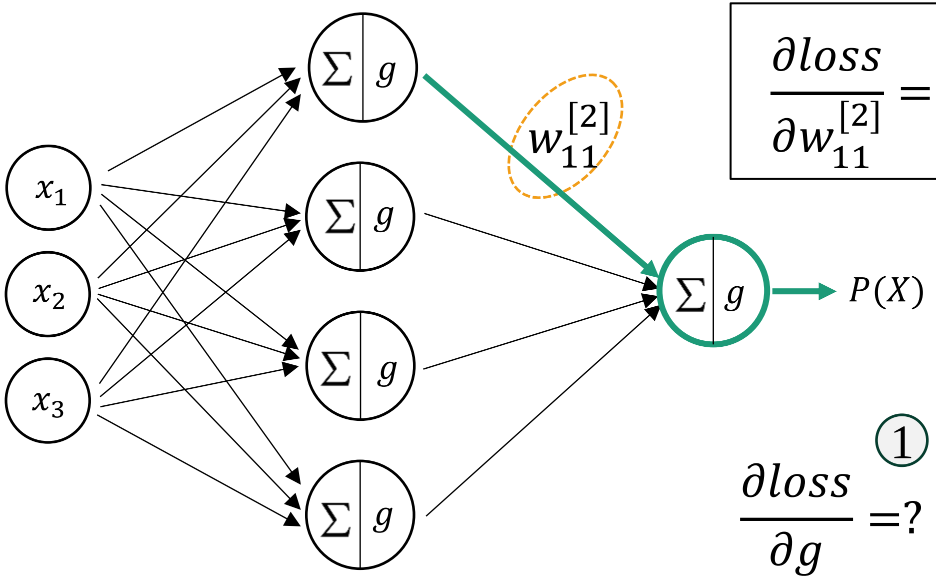
✓ Another example:

$$\boxed{loss \Rightarrow g \Rightarrow z \Rightarrow w \text{ or } b}$$



$$\boxed{\frac{\partial loss}{\partial w_{11}^{[1]}} = \frac{\partial loss}{\partial g_1^{[2]}} \times \frac{\partial g_1^{[2]}}{\partial z_1^{[2]}} \times \frac{\partial z_1^{[2]}}{\partial g_1^{[1]}} \times \frac{\partial g_1^{[1]}}{\partial z_1^{[1]}} \times \frac{\partial z_1^{[1]}}{\partial w_{11}^{[1]}}}$$

Backward pass; Example



$$loss = -(y\log(P(X)) + (1-y)\log(1-P(X)))$$

$$\frac{\partial loss}{\partial w_{11}^{[2]}} = \frac{\partial loss}{\partial g} \times \frac{\partial g}{\partial z_1^{[2]}} \times \frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}}$$

1 $\frac{\partial loss}{\partial g} = ?$

2 $\frac{\partial g}{\partial z_1^{[2]}} = ?$

3 $\frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}} = ?$

Differentiation; Review

$$\boxed{\frac{d\ \log_a x}{d\ x} = \frac{1}{x.\ln(a)}} \Longrightarrow$$

1. $$\frac{\partial loss}{\partial g} = -(\frac{y}{g.\ln(10)} + \frac{y-1}{g.\ln(10)}) = \frac{2y-1}{g.\ln(10)}$$

$$\boxed{\frac{d\ \sigma(x)}{d\ x} = \sigma(x)(1-\sigma(x))}$$

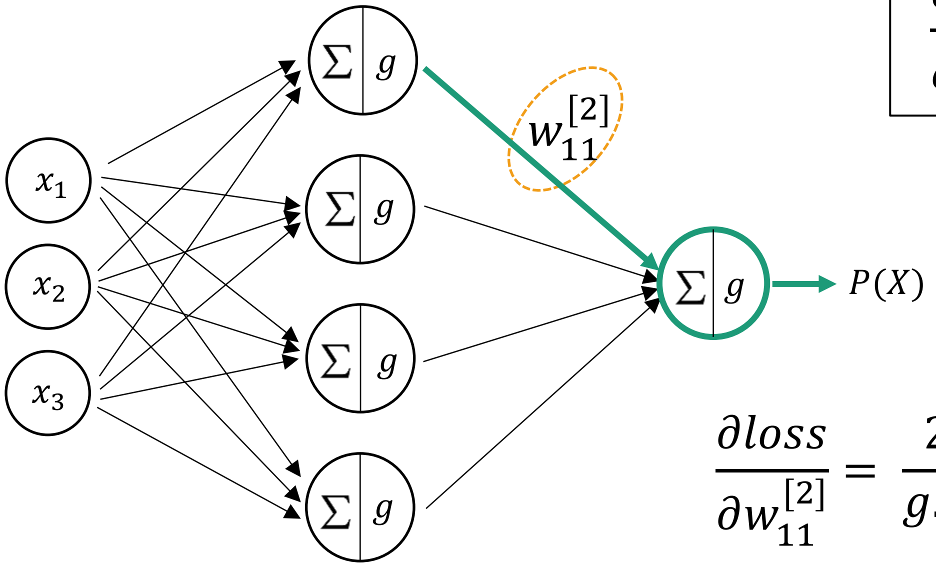
2. $$\frac{\partial g}{\partial z_1^{[2]}} = \sigma\left(z_1^{[2]}\right)(1-\sigma(z_1^{[2]}))$$

$$\boxed{\frac{\partial z_j^{[k]}}{\partial w_{ij}^{[k]}} = a_i^{[k-1]}} \Longrightarrow$$

$$\boxed{\frac{\partial z_j^{[k]}}{\partial b_j^{[k]}} = 1}$$

3. $$\frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}} = a_1^{[1]}$$

Backward pass; Example



$$\boxed{\frac{\partial loss}{\partial w_{11}^{[2]}} = \frac{\partial loss}{\partial g} \times \frac{\partial g}{\partial z_1^{[2]}} \times \frac{\partial z_1^{[2]}}{\partial w_{11}^{[2]}}}$$

$$\frac{\partial loss}{\partial w_{11}^{[2]}} = \frac{2y-1}{g.\ln(10)} \times \sigma\left(z_1^{[2]}\right)(1-\sigma(z_1^{[2]})) \times a_1^{[1]}$$

Parameters updates

- We **update** all parameters (weights and biases) according to:

$$w_{i,new} = w_{i,old} - \alpha \frac{\partial J}{\partial w_i}$$

- For our example:

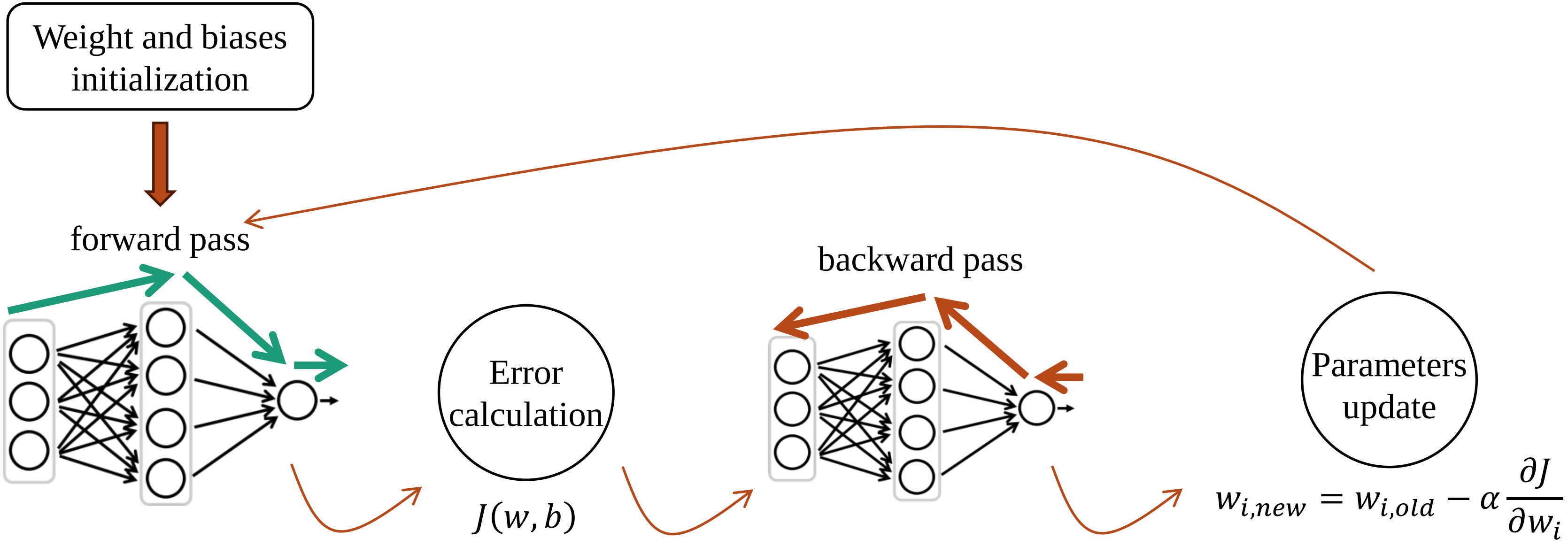
$$w_{11,new}^{[2]} = w_{11,old}^{[2]} - \alpha \frac{\partial l}{\partial w_{11}^{[2]}}$$

$$w_{11,new}^{[2]} = w_{11,old}^{[2]} - \alpha (\frac{2y-1}{g.\ln(10)} \times \sigma\left(z_1^{[2]}\right)(1-\sigma(z_1^{[2]})) \times a_1^{[1]})$$

✓ What about α ?

Backpropagation; Summary

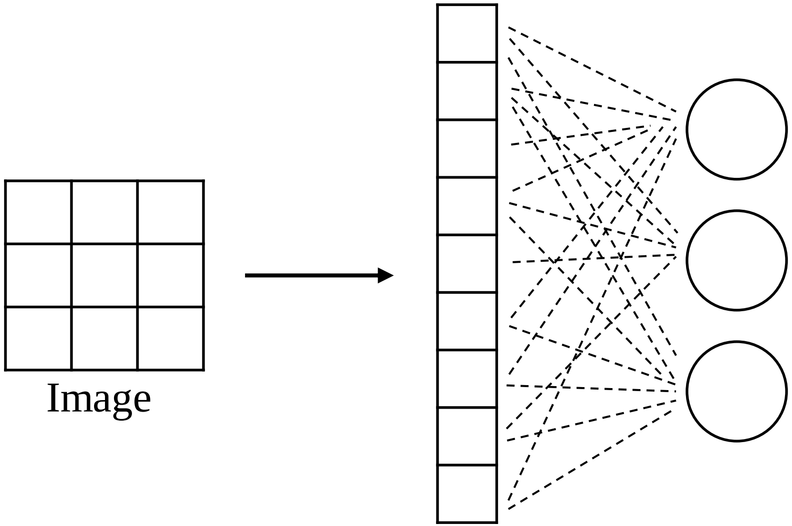
- This process of **forward pass**, **error calculation**, **backward pass**, and weights **update** continues for multiple epochs until the network performance reaches a satisfactory level or stops improving significantly.



SoftMax

SoftMax

- Suppose we have a **multiclass classification** problem.



$$P_{cat} = a_1^{[1]} = \sigma(w_1^{[1]}.x + b_1^{[1]})$$

?

$$P_{horse} = a_2^{[1]} = \sigma(w_2^{[1]}.x + b_2^{[1]})$$

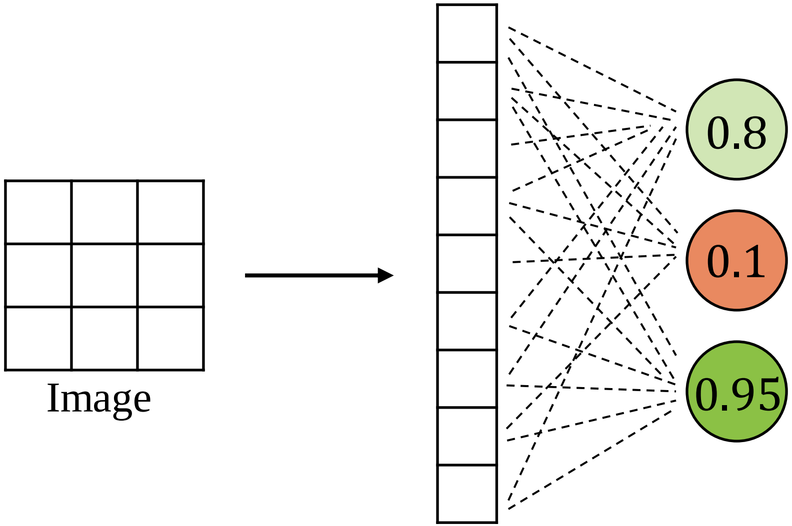
?

$$P_{lion} = a_3^{[1]} = \sigma(w_3^{[1]}.x + b_3^{[1]})$$

?

SoftMax

- ✓ Considering a **threshold** of **0.5**, what label would you assign in this situation?



$$P_{cat} = a_1^{[1]} = \sigma(w_1^{[1]}.x + b_1^{[1]})$$

$$P_{horse} = a_2^{[1]} = \sigma(w_2^{[1]}.x + b_2^{[1]})$$

$$P_{lion} = a_3^{[1]} = \sigma(w_3^{[1]}.x + b_3^{[1]})$$

SoftMax

- The **SoftMax** activation function, also known as the **normalized exponential function**, is particularly useful within the context of multi-class classification problems.

- The **SoftMax** function is a function that turns a vector of K real values into a vector of K real values that sum to 1.

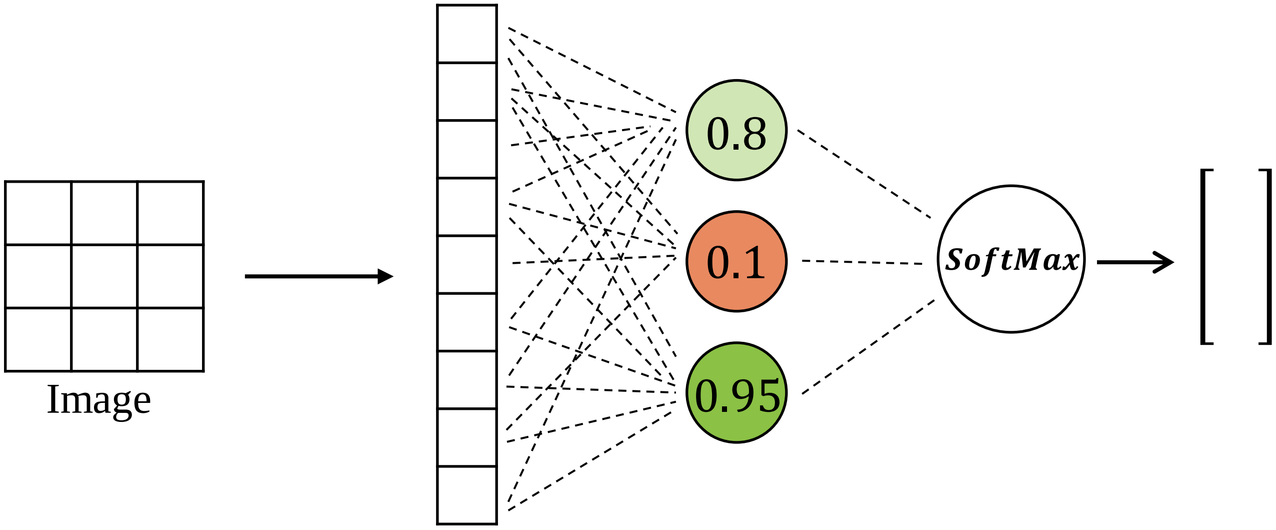
$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \dots \\ a_K \end{bmatrix} \Rightarrow \boxed{\textbf{SoftMax}} \Rightarrow \begin{bmatrix} a_1{}' \\ a_2{}' \\ a_3{}' \\ \dots \\ a_K{}' \end{bmatrix}$$

$$\sum_{i=1}^{K} a_i{}' = 1$$

SoftMax

- For input vector z with elements $z_1, z_2, \ldots, z_k$ the **SoftMax** function is defined as:

$$g(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

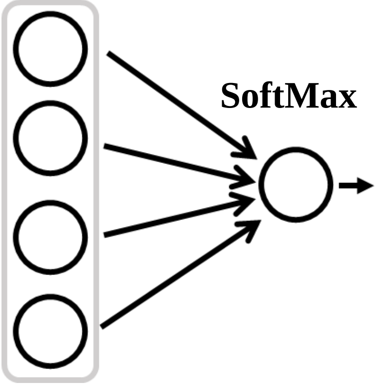


SoftMax; Architecture

- For the **SoftMax** layer to function correctly, we need a neuron for each class in the previous layer.

- Consequently, the SoftMax output will also be a vector of this size.

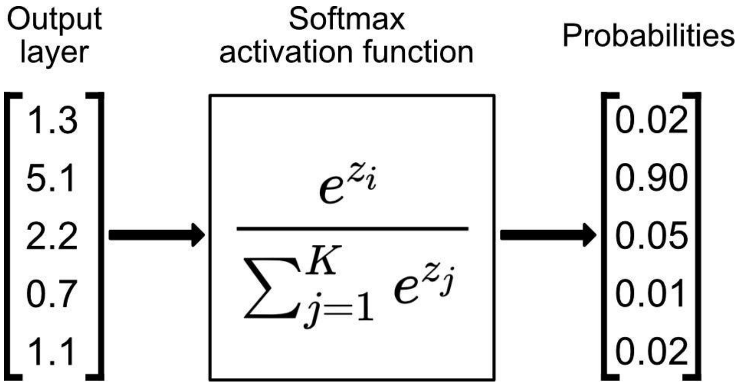
- For example, in a four-class classification:



$$\begin{bmatrix} a_1' \\ a_2' \\ a_3' \\ a_4' \end{bmatrix}$$

SoftMax

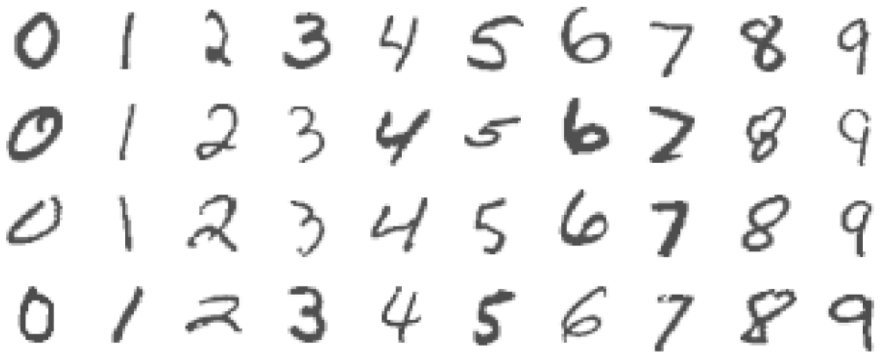
- The input values can be positive, negative, zero, or greater than one, but the **SoftMax** transforms them into values between 0 and 1, so that they can be interpreted as probabilities.

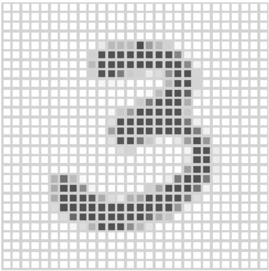


- There's no necessity to use another activation function like **sigmoid** before **SoftMax**.In fact, Since **SoftMax** already handles the conversion to probabilities, using sigmoid beforehand is redundant in the context of multi-class classification.

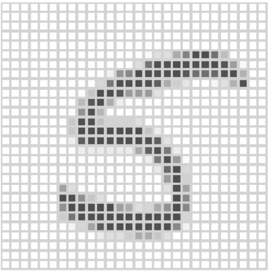
Multi-class Classification; example

- MNIST is digit recognition dataset.
  - Handwritten digits
  - 28×28 grayscale images
  - Grayscale values: [0,255]
  - Features are the 784 pixel
  - 60K train, 10K test images
  - Labels are the digit class 0 to 9

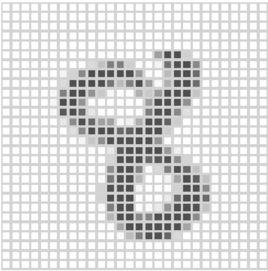




3



5

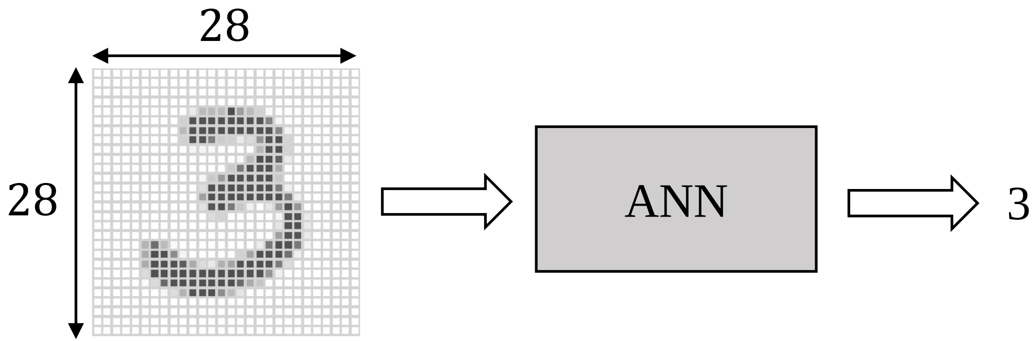


8

Multi-class Classification; example

- Our goal is to build a classier to predict the image class.

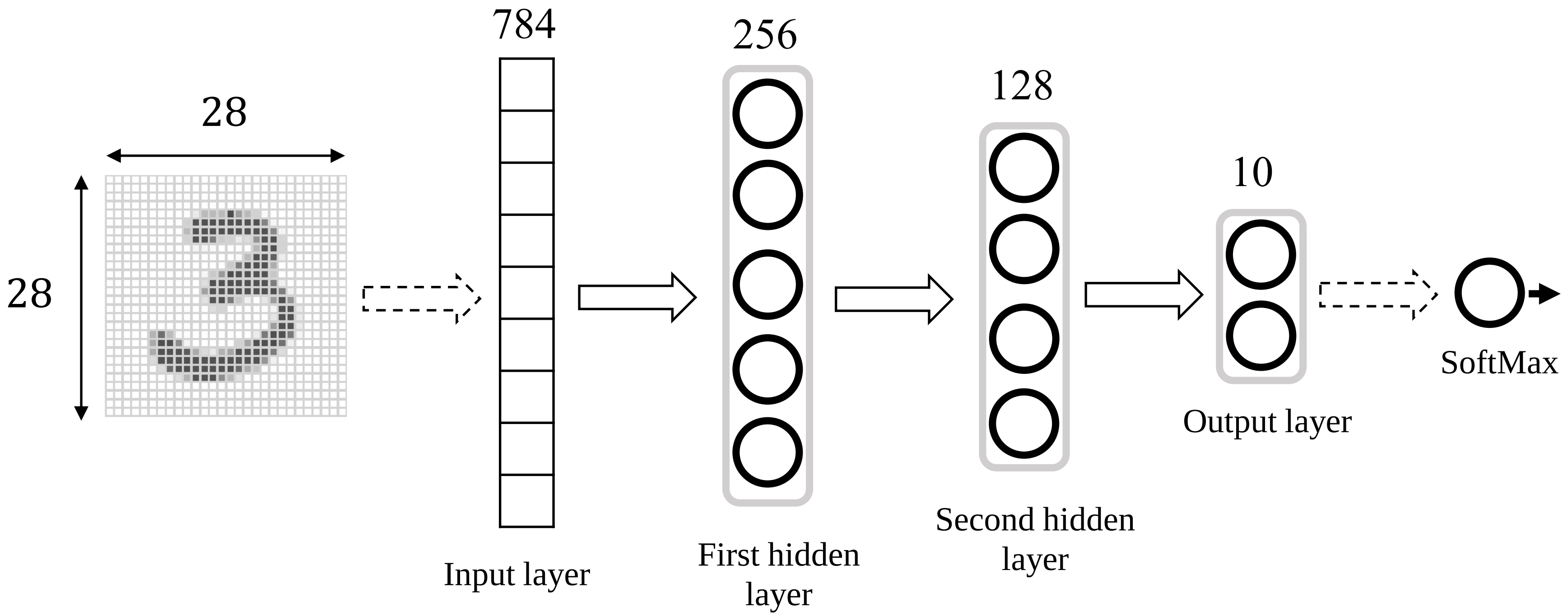
- We build a **two-layer network** with 256 units at first layer, 128 units at second layer, and 10 units at output layer.



- Along with biases, there are 235,146 parameters.

✓ Comparing number of parameters with previous simpler models, what can you conclude?

Multi-class Classification; example



Multi-class Classification; example

- We fit the model by minimizing **cross-entropy.**

$$J(w,b) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{K} y_j^{(i)} \,.\log(P_{\mathrm{w,b}}(X^{(i)}))$$

- Model performance: **error rate < 0.5%**

- For comparison, human error rate is reported to be around **0.2%**, or 20 of the 10K test images.

- Some of hard examples in MNIST: