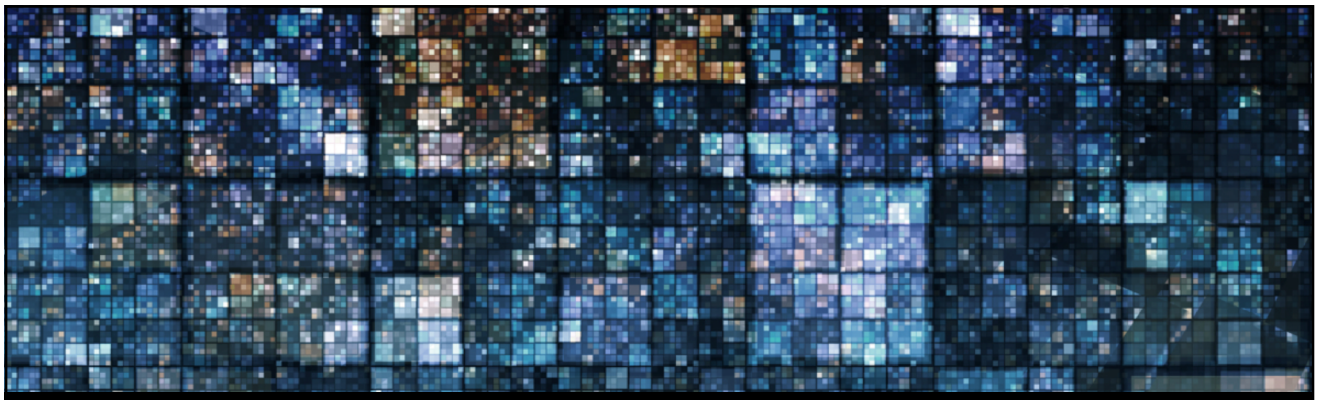
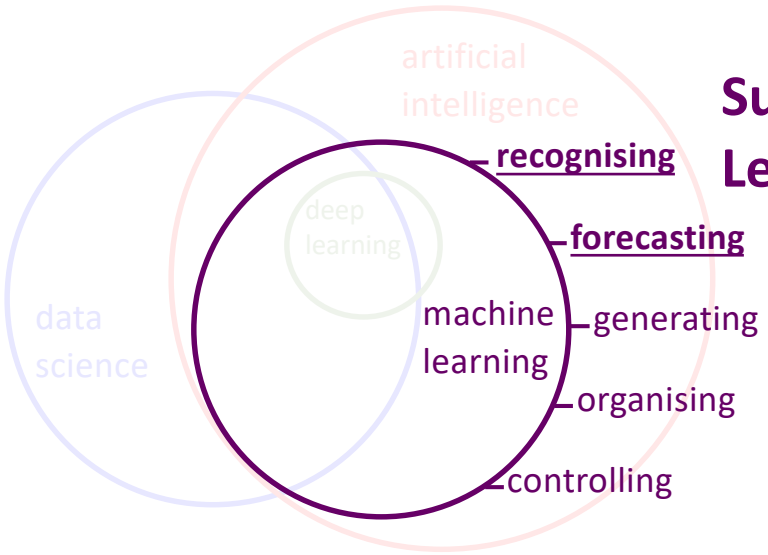


COMP47590
Advanced Machine Learning
Deep Learning - ANNs 2

Dr. Brian Mac Namee

Information

Email:	Brian.MacNamee@ucd.ie
Course Materials:	All material posted on UCD CS moodle https://csmoodle.ucd.ie/moodle/course/view.php?id=756
	Enrolment key **UCDAdvML2019**

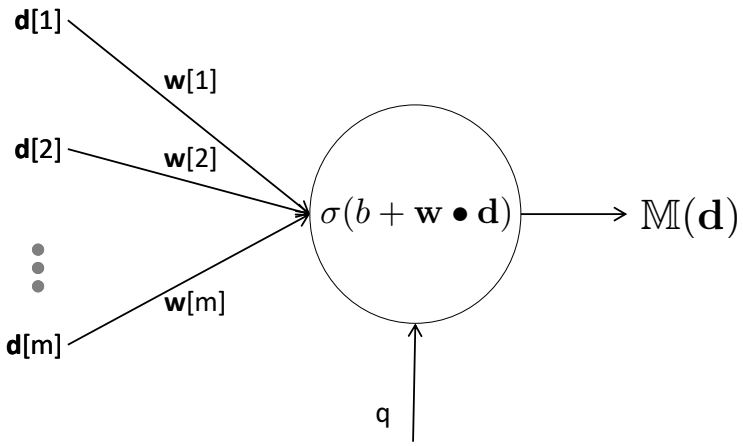


Supervised Learning

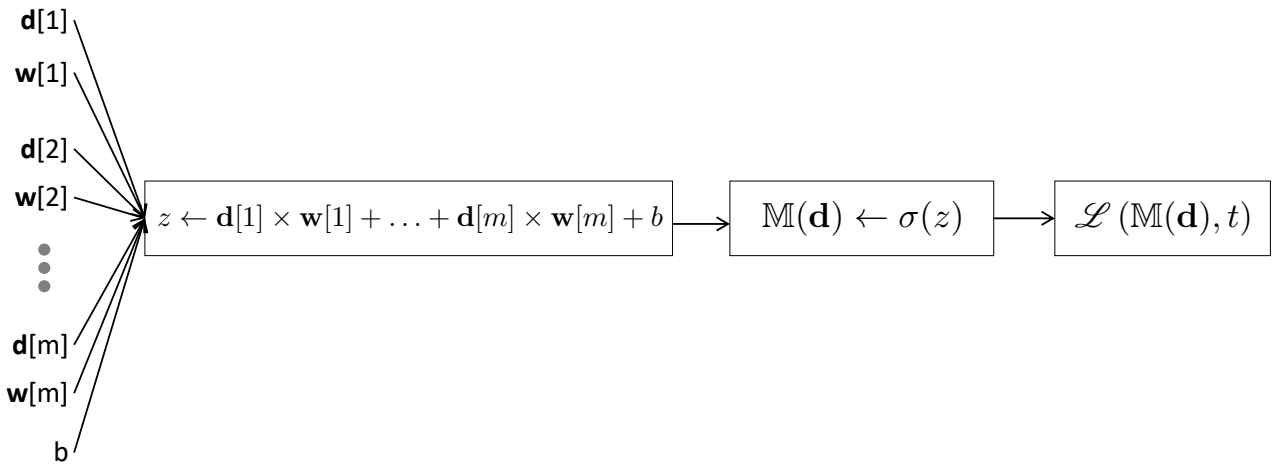
- recognising
- forecasting
- generating
- organising
- controlling

PERCEPTRON

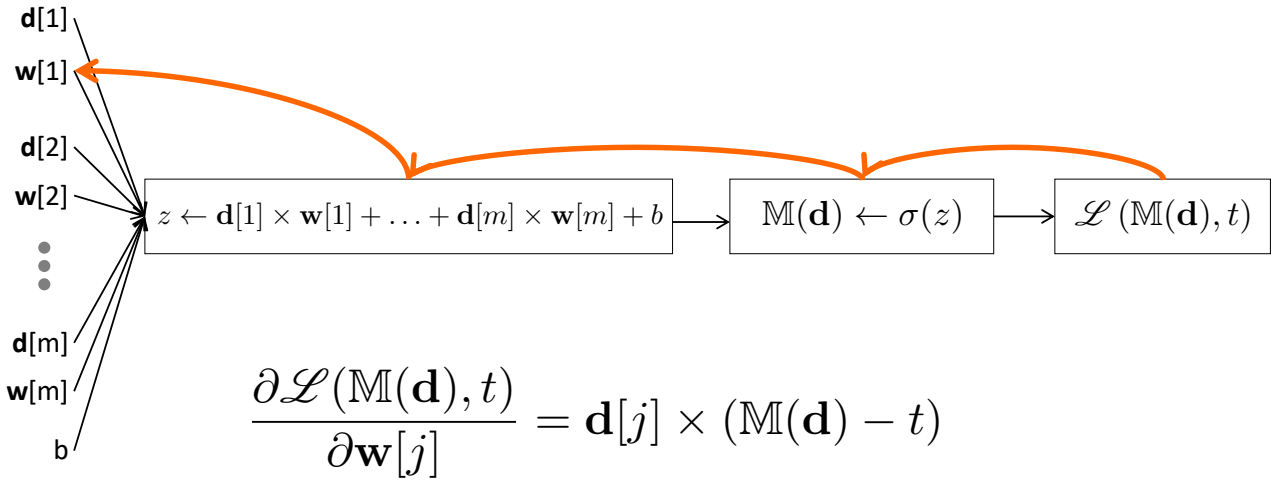
Perceptron



Perceptron

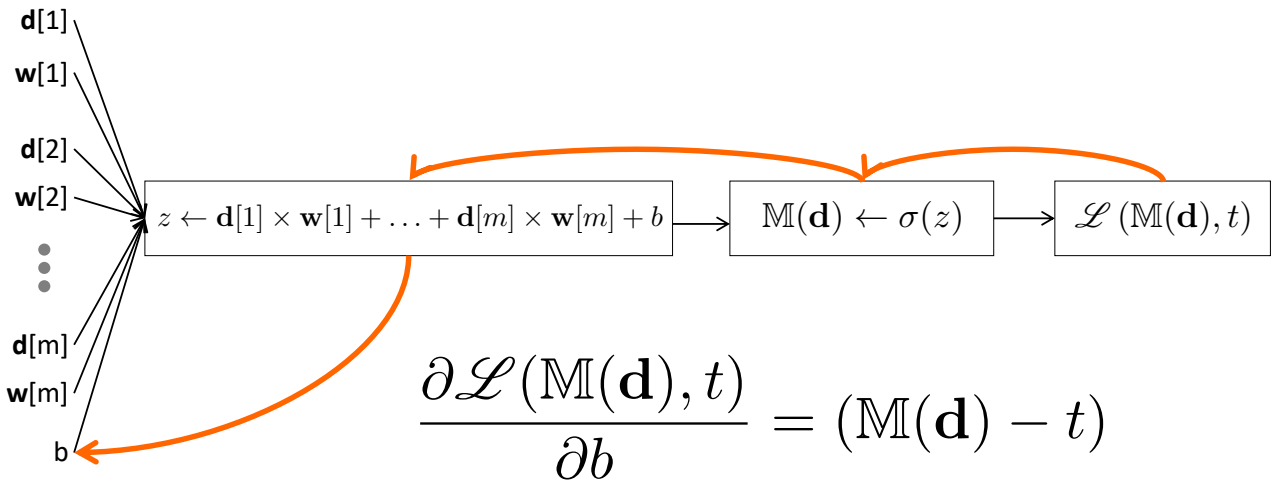


Perceptron



$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{w}[j]} = \mathbf{d}[j] \times (\mathbb{M}(\mathbf{d}) - t)$$

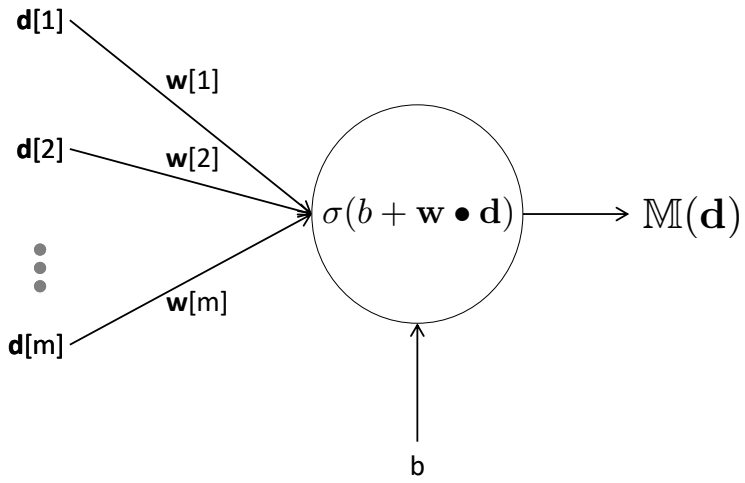
Perceptron



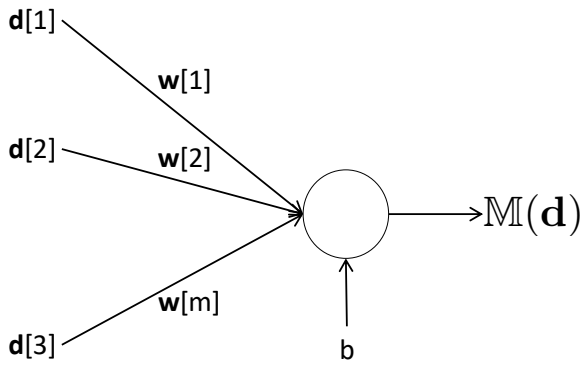
$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial b} = (\mathbb{M}(\mathbf{d}) - t)$$

ARTIFICIAL NEURAL NETWORKS

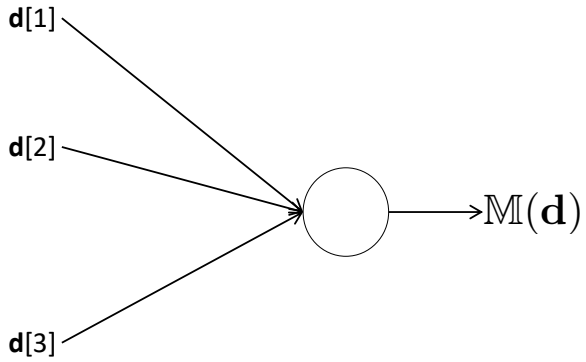
Perceptron



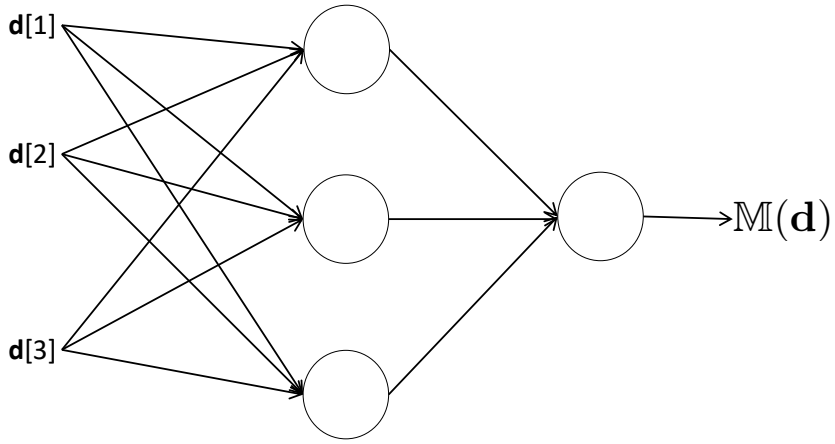
Perceptron



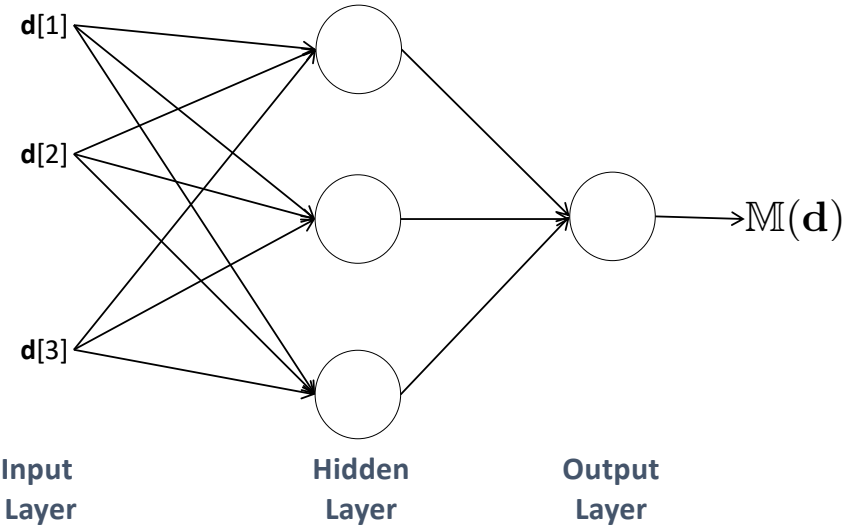
Perceptron



Multi Layer Perceptron

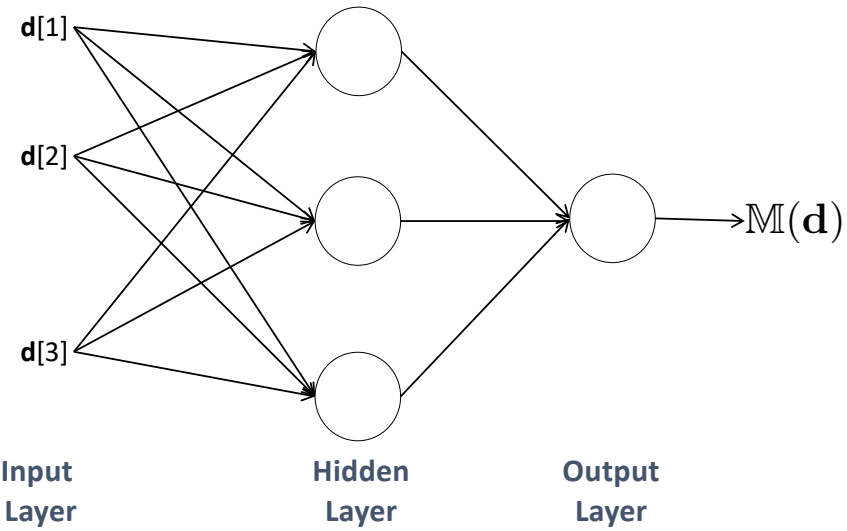


Multi Layer Perceptron



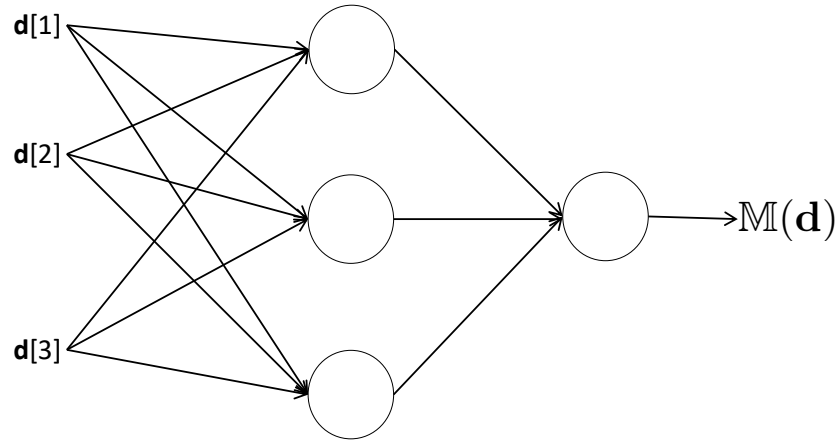
BACKPROPOGATION OF ERRORS

Multi Layer Perceptron



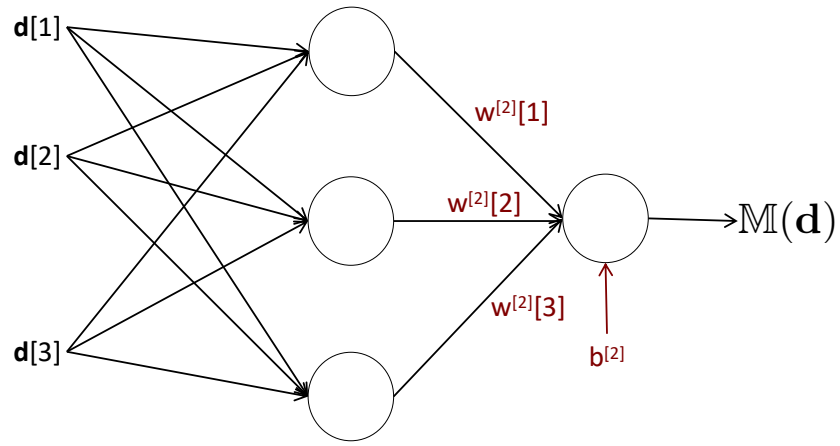
Multi Layer Perceptron

If we remeber what is going on at the output layer



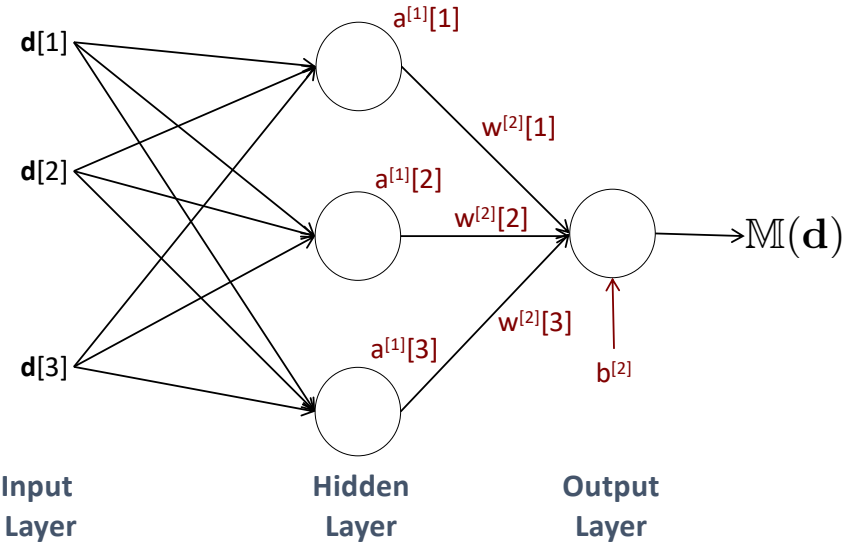
Multi Layer Perceptron

If we remeber what is going on at the output layer



Multi Layer Perceptron

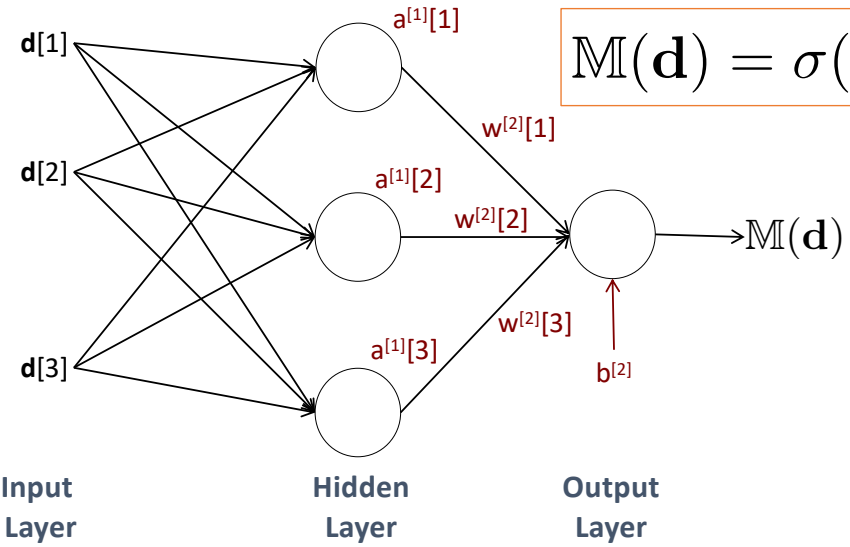
If we remeber what is going on at the output layer



Multi Layer Perceptron

If we remeber what is going on at the output layer

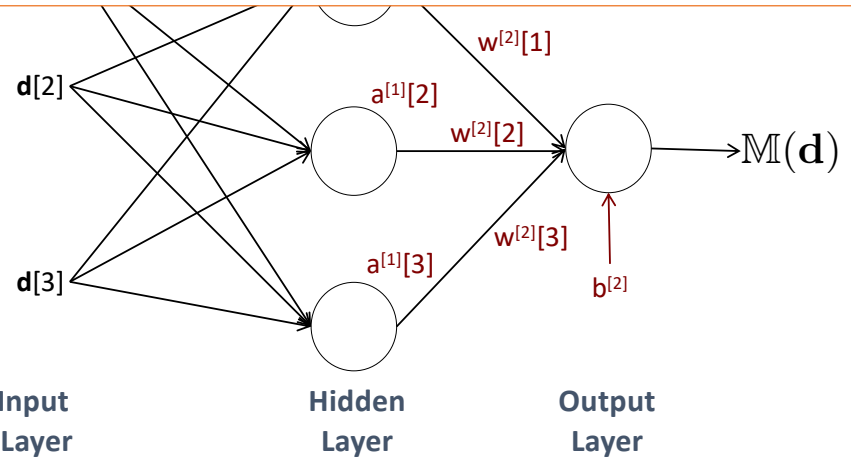
$$\mathbb{M}(\mathbf{d}) = \sigma(z^{[2]})$$



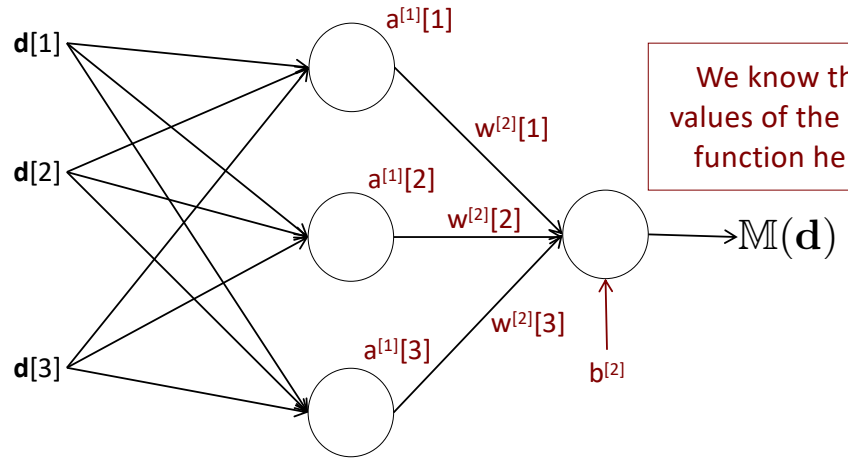
Multi Layer Perceptron

If we remeber what is going on at the output layer

$$z^{[2]} = \mathbf{w}^{[2]}[1] \times \mathbf{a}^{[1]}[1] + \mathbf{w}^{[2]}[2] \times \mathbf{a}^{[1]}[2] + \mathbf{w}^{[2]}[3] \times \mathbf{a}^{[1]}[3]$$



Multi Layer Perceptron

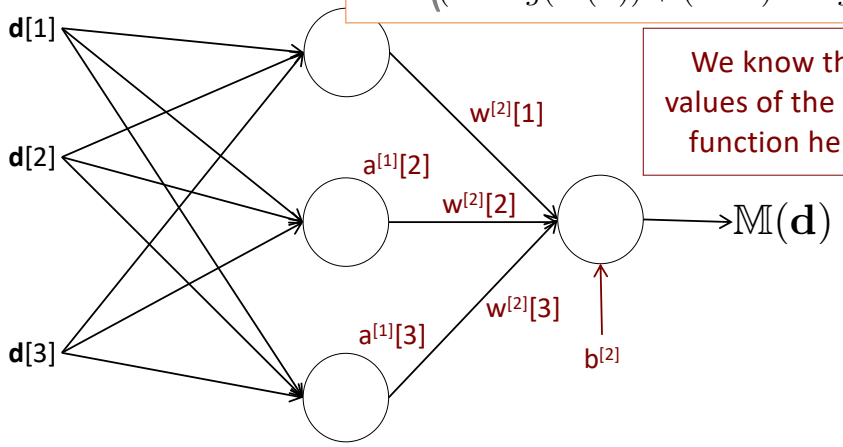


We know the values of the loss function here

Multi Layer Perceptron

$$\mathscr{L}\left(\mathbb{M}(\mathbf{d}), t\right) = -\Big((t \times log(\mathbb{M}(\mathbf{d})) + (1-t) \times log(1-\mathbb{M}(\mathbf{d}))\Big)$$

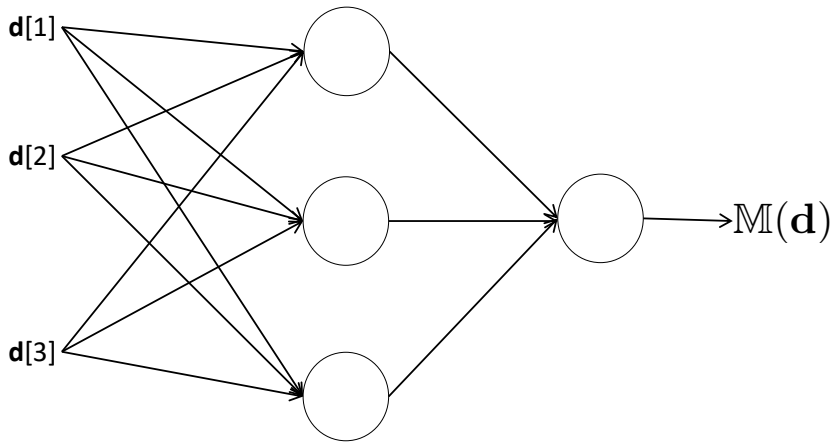
We know the values of the loss function here



Input Layer | Hidden Layer | Output Layer

Multi Layer Perceptron

So we can calculate the derivative of the loss function with respect to the weights coming into this unit

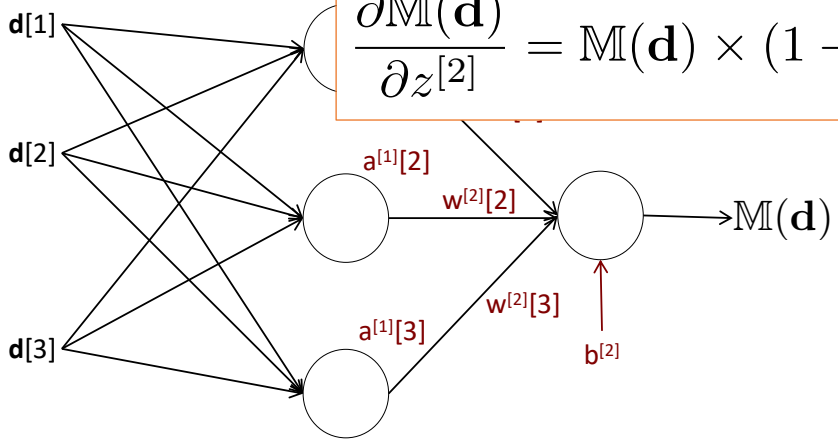


Input Layer | Hidden Layer | Output Layer

Multi Layer Perceptron

So we can calculate the derivative of the loss function with respect to the output of the last unit

$$\frac{\partial \mathbb{M}(\mathbf{d})}{\partial z^{[2]}} = \mathbb{M}(\mathbf{d}) \times (1 - \mathbb{M}(\mathbf{d}))$$

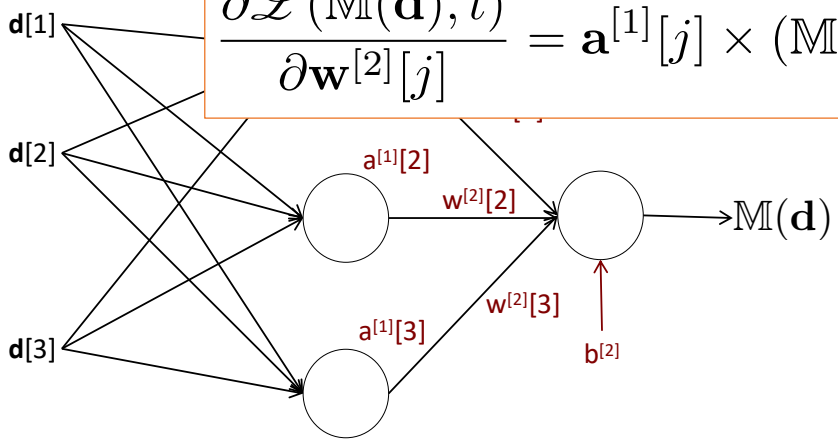


Input Layer | Hidden Layer | Output Layer

Multi Layer Perceptron

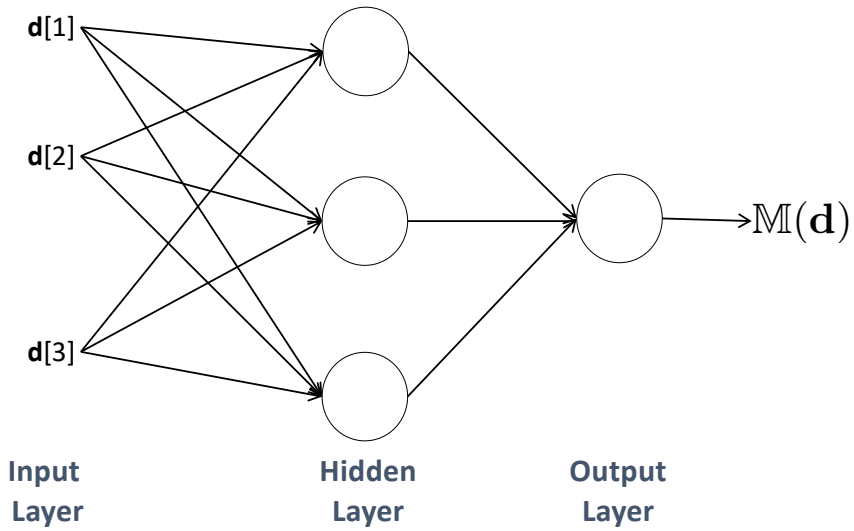
So we can calculate the derivative of the loss function with respect to the weights coming into this unit

$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{w}^{[2]}[j]} = \mathbf{a}^{[1]}[j] \times (\mathbb{M}(\mathbf{d}) - t)$$



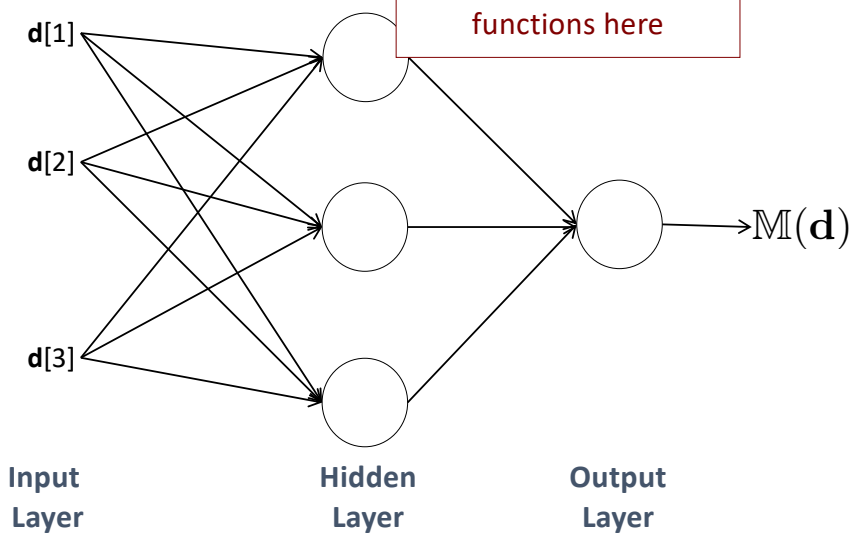
Input Layer | Hidden Layer | Output Layer

Multi Layer Perceptron

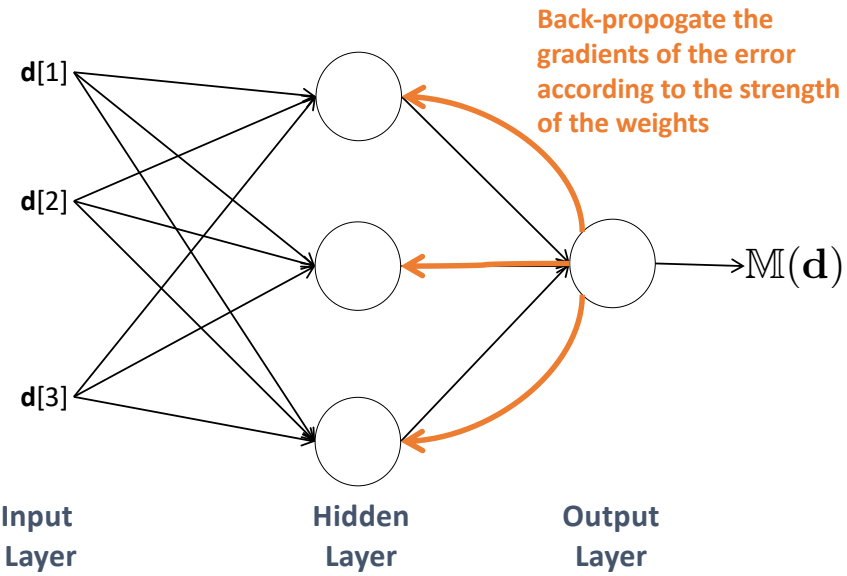


Multi Layer Perceptron

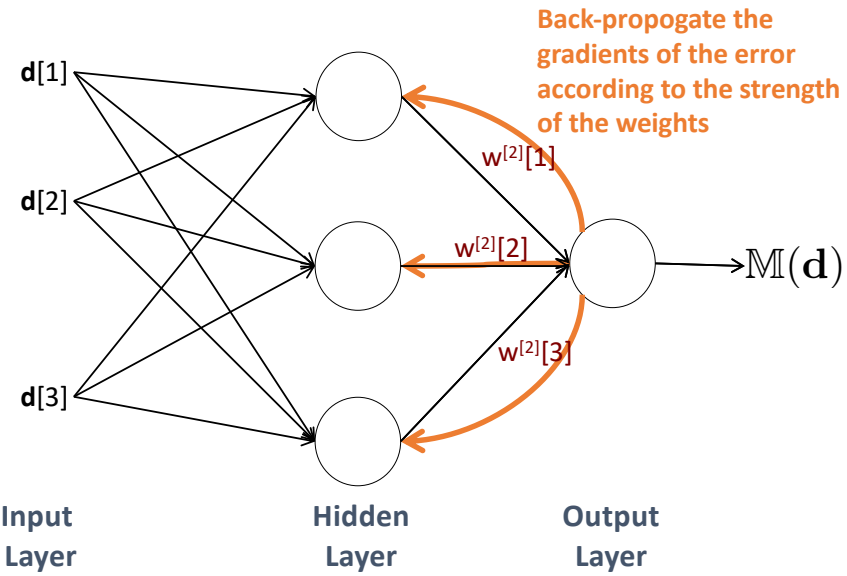
We don't know the value of the cost functions here



Multi Layer Perceptron

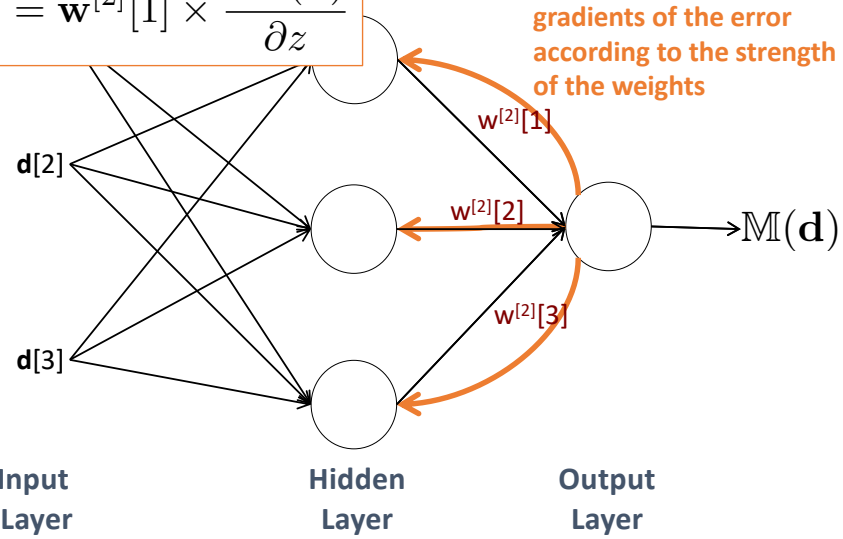


Multi Layer Perceptron



Multi Layer Perceptron

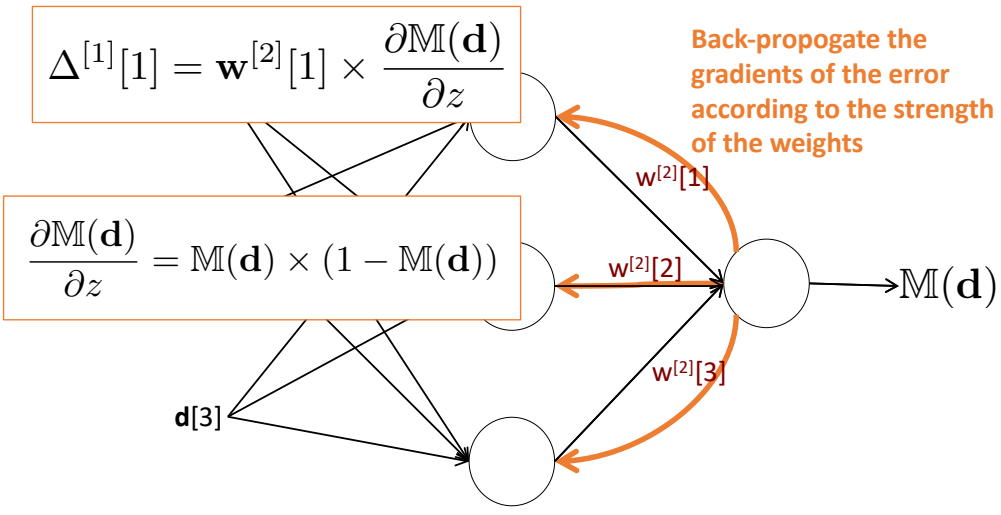
$$\Delta^{[1]}[1] = \mathbf{w}^{[2]}[1] \times \frac{\partial \mathbb{M}(\mathbf{d})}{\partial z}$$



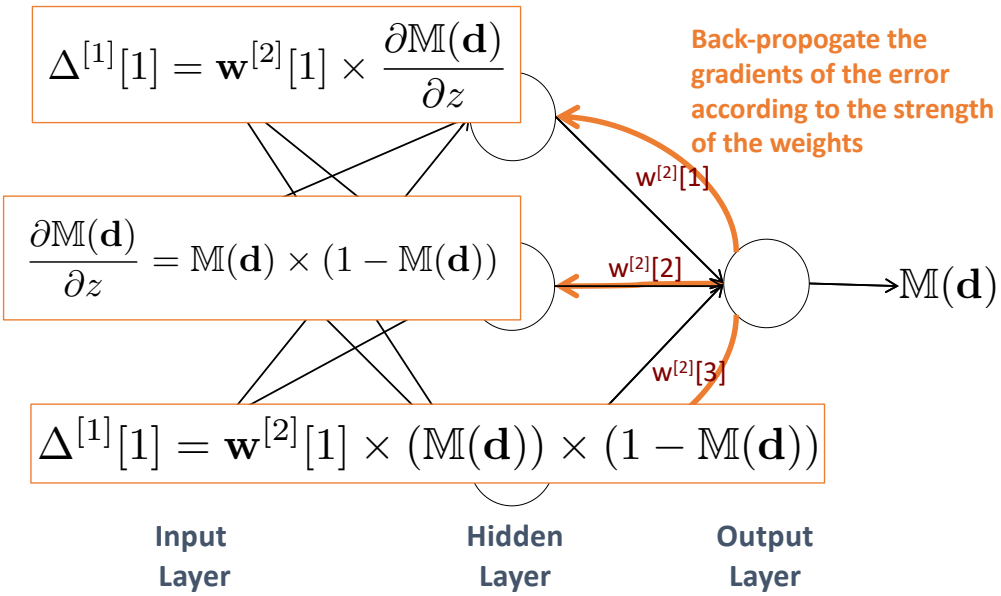
Multi Layer Perceptron

$$\Delta^{[1]}[1] = \mathbf{w}^{[2]}[1] \times \frac{\partial \mathbb{M}(\mathbf{d})}{\partial z}$$

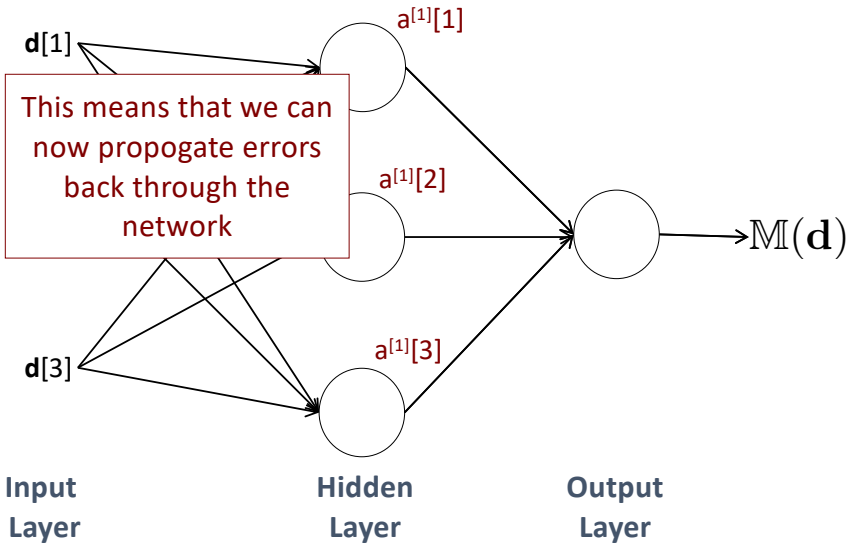
$$\frac{\partial \mathbb{M}(\mathbf{d})}{\partial z} = \mathbb{M}(\mathbf{d}) \times (1 - \mathbb{M}(\mathbf{d}))$$



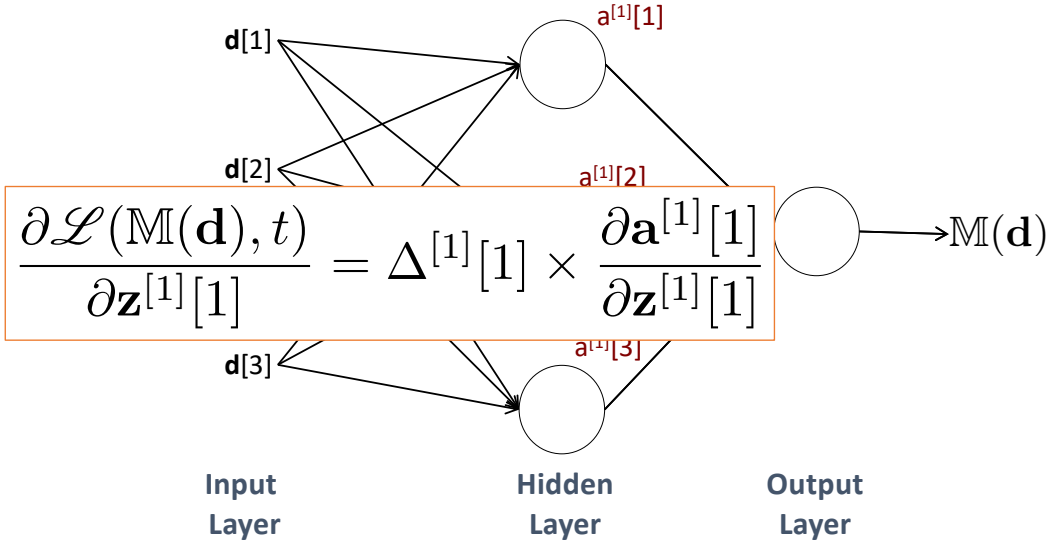
Multi Layer Perceptron



Multi Layer Perceptron

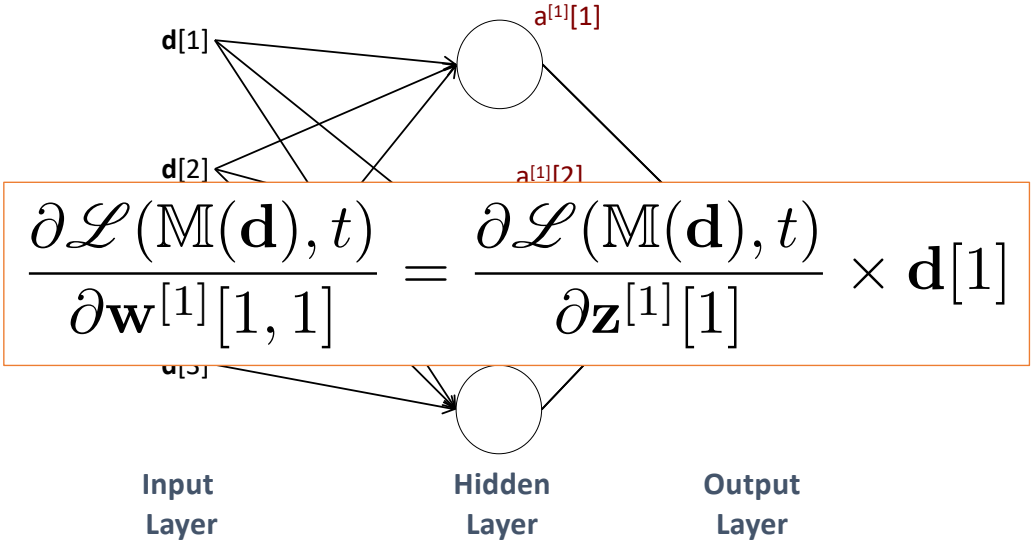


Multi Layer Perceptron



$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{z}^{[1]}[1]} = \Delta^{[1]}[1] \times \frac{\partial \mathbf{a}^{[1]}[1]}{\partial \mathbf{z}^{[1]}[1]}$$

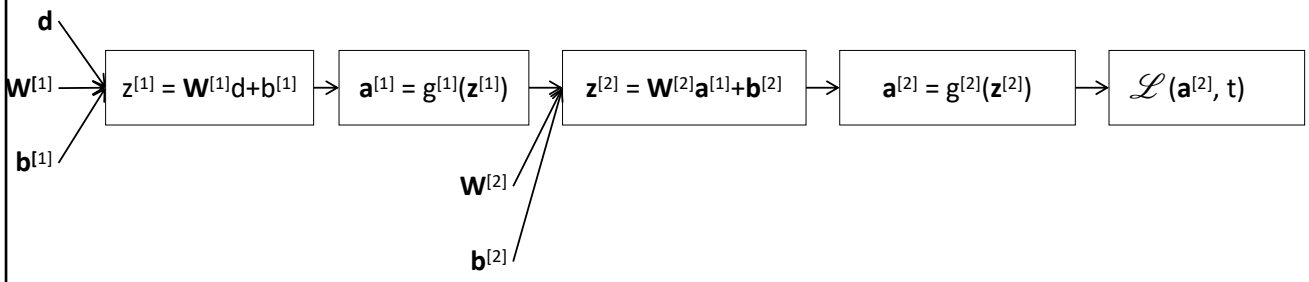
Multi Layer Perceptron



$$\frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{w}^{[1]}[1,1]} = \frac{\partial \mathscr{L}(\mathbb{M}(\mathbf{d}), t)}{\partial \mathbf{z}^{[1]}[1]} \times \mathbf{d}[1]$$

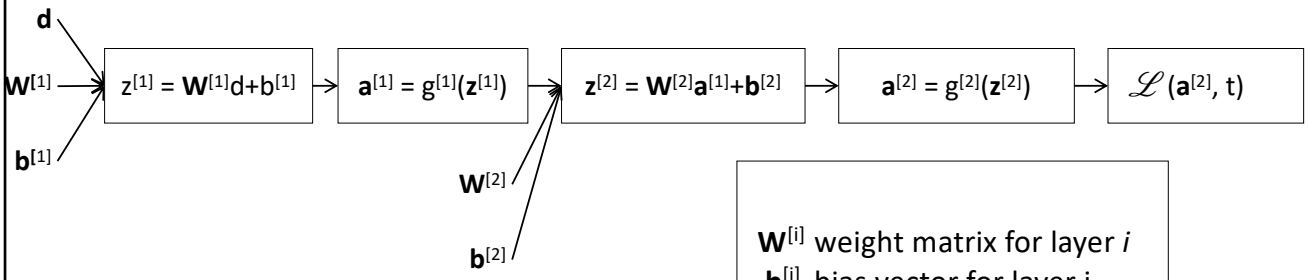
A NEW NOTATION

Multi Layer Perceptron



Multi Layer Perceptron

We adopt a notation here (following Andrew Ng) that makes the layer structure of networks very explicit



$\mathbf{W}^{[i]}$ weight matrix for layer *i*
$\mathbf{b}^{[i]}$ bias vector for layer i

A New Notation

$$d\mathbf{z}^{[2]} = \mathbf{a}^{[2]} - \mathbf{t}$$
$$d\mathbf{W}^{[2]} = d\mathbf{z}^{[2]}\mathbf{a}^{[1]T}$$
$$d\mathbf{b}^{[2]} = d\mathbf{z}^{[2]}$$
$$d\mathbf{z}^{[1]} = \mathbf{W}^{[2]T}d\mathbf{z}^{[2]} * g^{[1]\prime}(\mathbf{z}^{[1]})$$
$$d\mathbf{W}^{[1]} = d\mathbf{z}^{[1]}\mathbf{d}^{T}$$
$$d\mathbf{b}^{[1]} = d\mathbf{z}^{[1]}$$

where

$$d\mathbf{X} = \frac{\partial J(\mathbf{w}, b)}{\partial \mathbf{X}}$$

GRADIENT DESCENT

Gradient Descent

We only need to update our weight update rules to take account of the extra layers

$\mathbf{W}^{[i]} = \mathbf{W}^{[i]} - \alpha \, d\mathbf{W}^{[i]}$

$\mathbf{b}^{[i]} = \mathbf{b}^{[i]} - \alpha \, d\mathbf{b}^{[i]}$

ACTIVATION FUNCTIONS

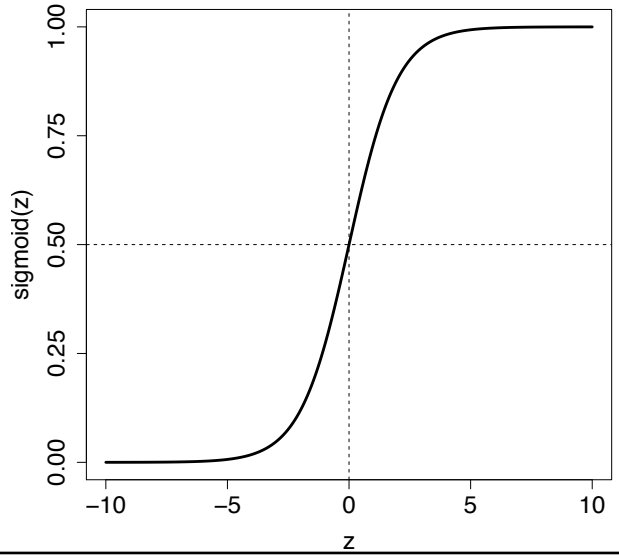
Activation Functions

Unless we introduce a non-linear activation function our networks, no matter how deep, will only be able to learn linear functions

Sigmoid activation units used to be most common

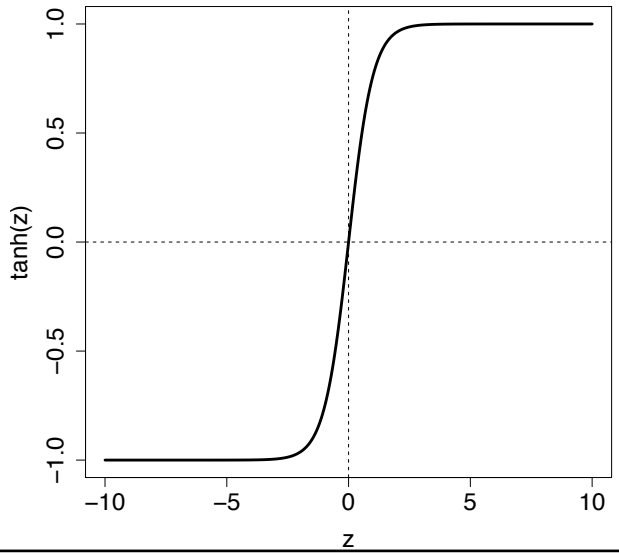
Now replaced with modern alternatives **tanh**, **rectified linear units (relu)**, and **leaky relu**

Activation Functions



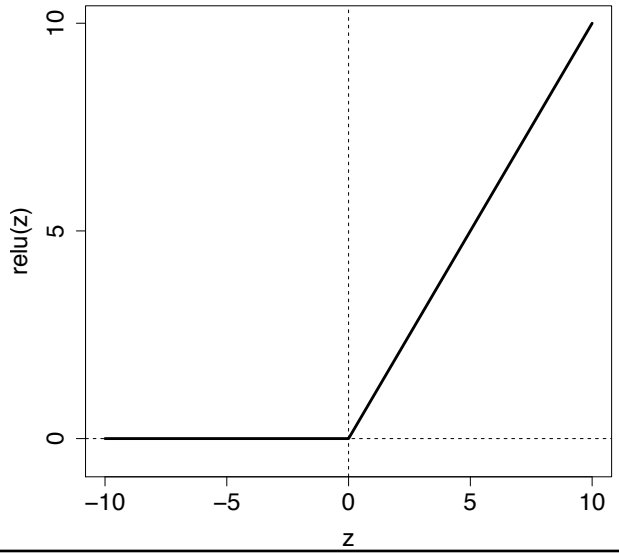
$$sigmoid(z) = \frac{1}{1 + e^{-z}}$$

Activation Functions



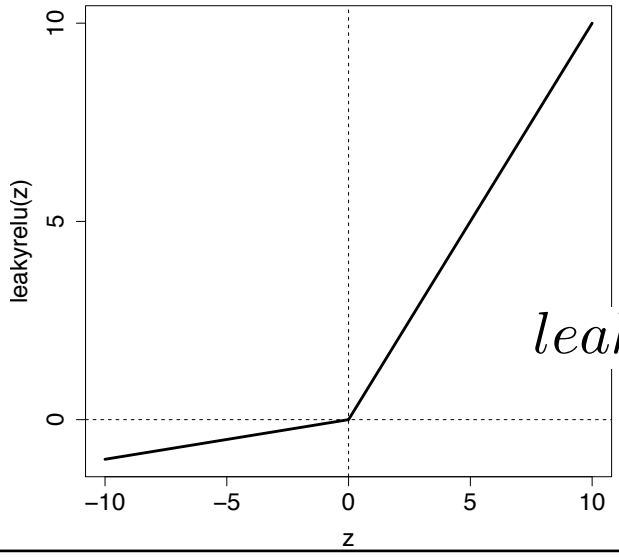
$$tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Activation Functions



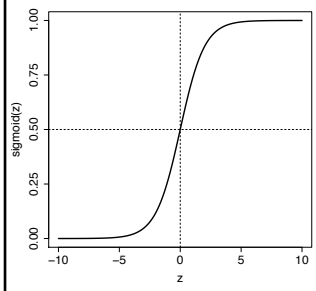
$$relu(z) = max(0, z)$$

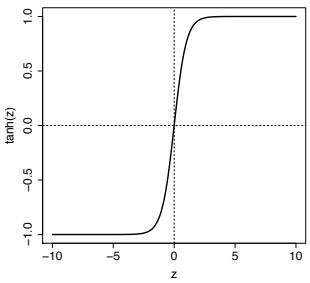
Activation Functions

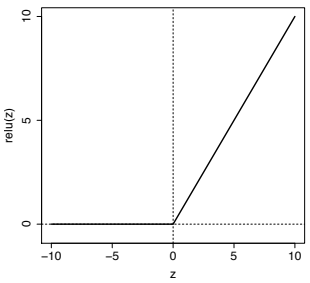


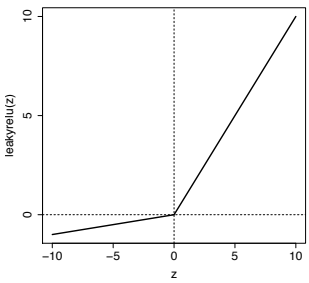
$$leakyrelu(z) = max(0.1\,z, z)$$

Activation Functions









SUMMARY

Summary

Expadning the perceptron apporach to multiple layers of computational units is straight-forward

The gradient descent algorithm does not require any major changes

The key is the **backpropogation of error** algorithm which allows us to propogate the gradient of the error from the output layer back through the earlier layers in a network

Questions