



# Your First MLP

Recitation 1, part 1  
Fall 2022



# Overview

- Neural Networks
- Perceptrons
- Multilayer perceptrons
  - Forward Pass
  - Backpropagation
  - Update Weights

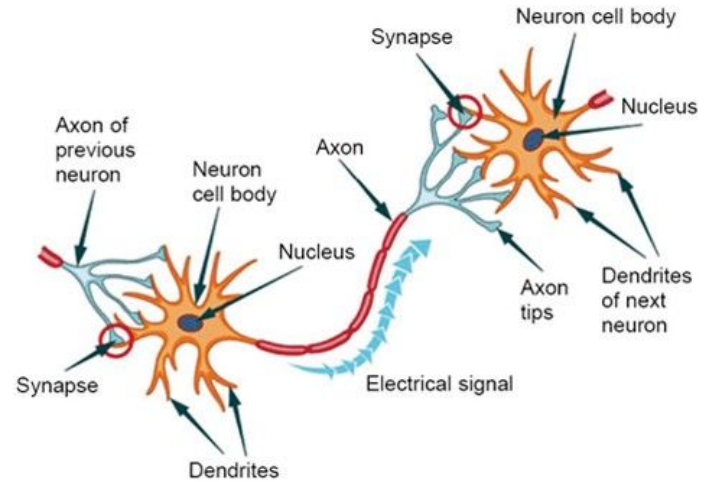
# Neural Networks

- The brain, made up of connected neurons, are the inspirations for artificial neural networks.



# Neural Networks

- A neuron is a node with many inputs and one output.
- A neural network consists of many interconnected neurons -- a “simple” device that receives data at the input and provides a response.
- Information are transmitted from one neuron to another by electrical impulses and chemical signals.



# Perceptrons

- Perceptron is a single layer neural network.

# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.

# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
  - Input values



A vertical diagram showing input values. It consists of four light blue circles arranged vertically. The top circle contains the symbol  $x_1$ . Below it are three black dots, indicating a continuation of the sequence. The third circle from the top contains the symbol  $x_{n-1}$ . The bottom circle contains the symbol  $x_n$ .

$x_1$

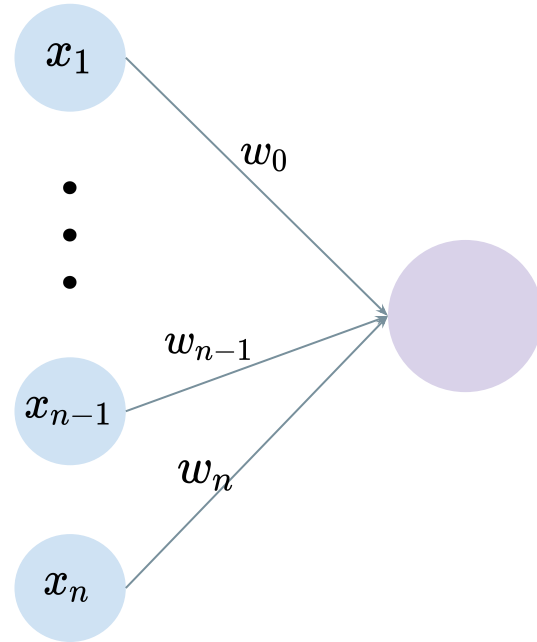
•  
•  
•

$x_{n-1}$

$x_n$

# Perceptrons

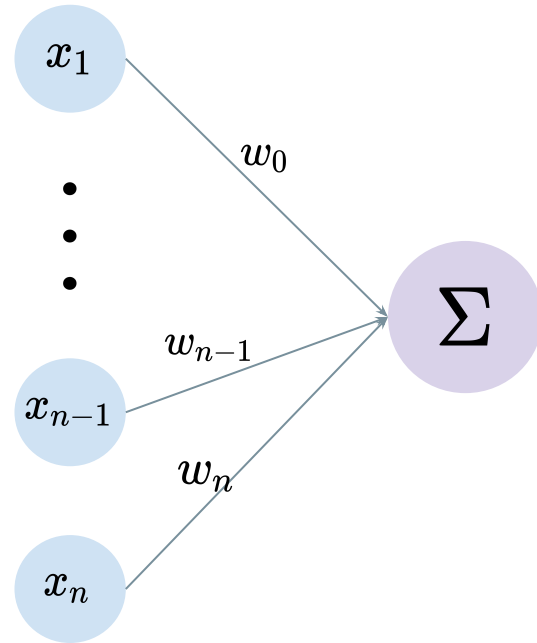
- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
  - Input values
  - Weights





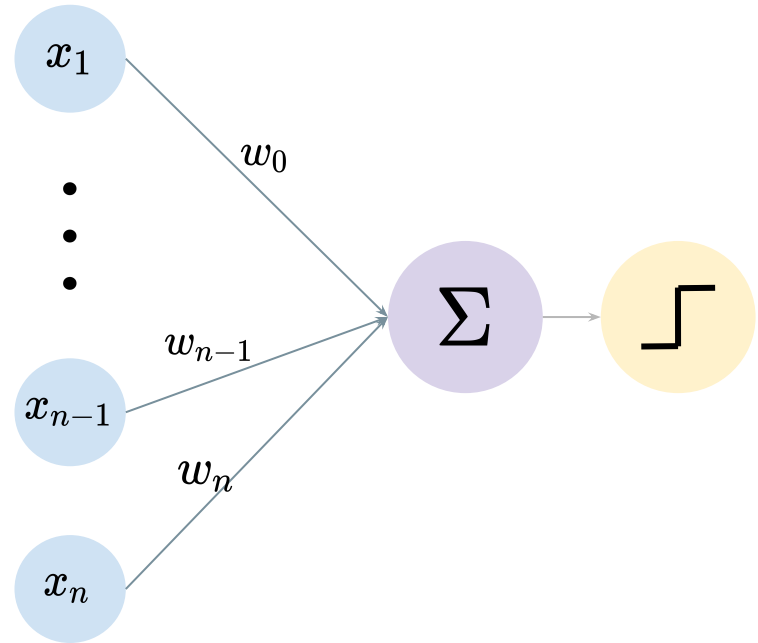
# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
  - Input values
  - Weights
  - Weighted sums



# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
  - Input values
  - Weights
  - Weighted sums
  - Threshold / Activation functions



# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
- The perceptron works on the following steps:
  - Multiply all inputs with their weights

$$x_1 \times w_0$$

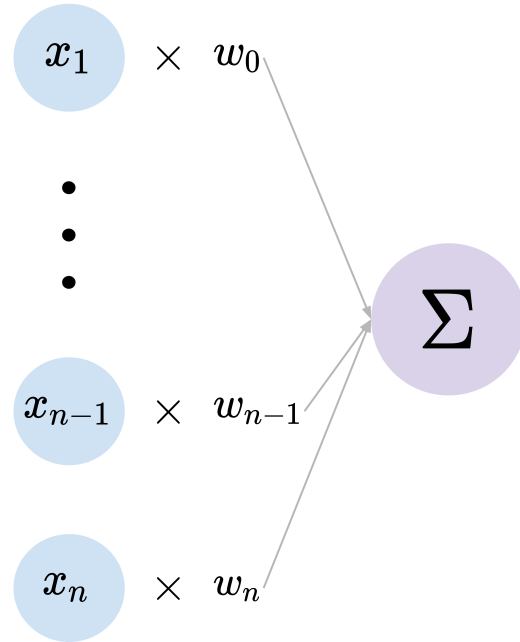
•  
•  
•

$$x_{n-1} \times w_{n-1}$$

$$x_n \times w_n$$

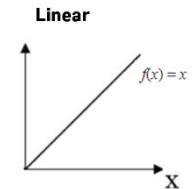
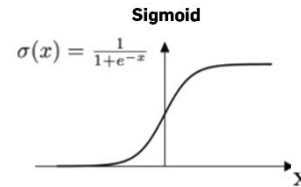
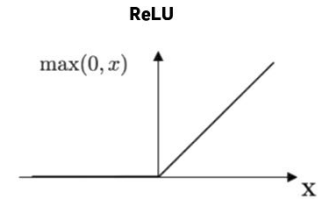
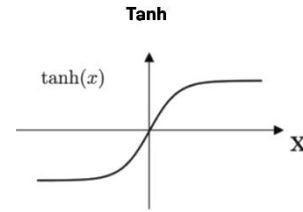
# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
- The perceptron works on the following steps:
  - Multiply all inputs with their weights
  - Add all multiplies values  $\rightarrow$  weighted sum



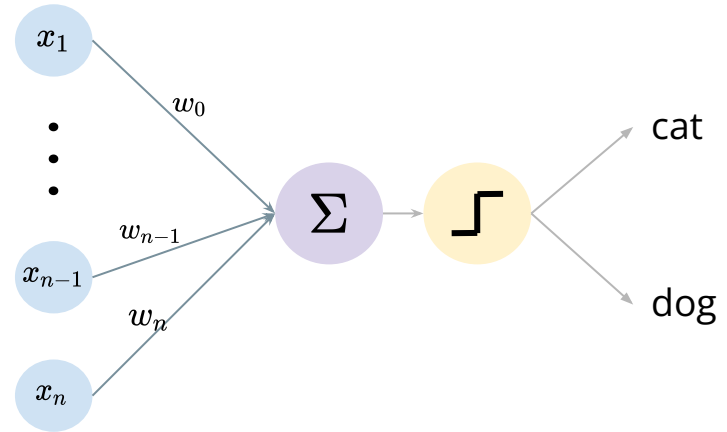
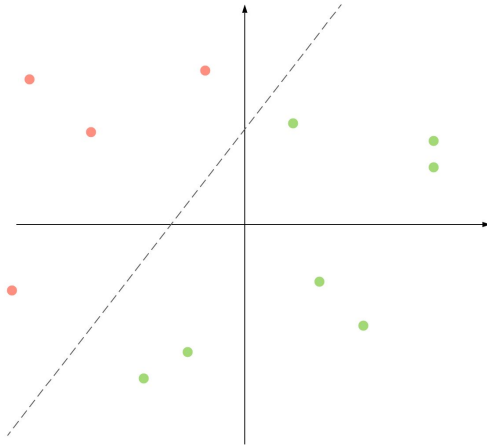
# Perceptrons

- Perceptron is a single layer neural network.
- The perceptron consists of 4 parts.
- The perceptron works on the following steps:
  - Multiply all inputs with their weights
  - Add all multiplied values  $\rightarrow$  weighted sum
  - Apply the weighted sum to activation function



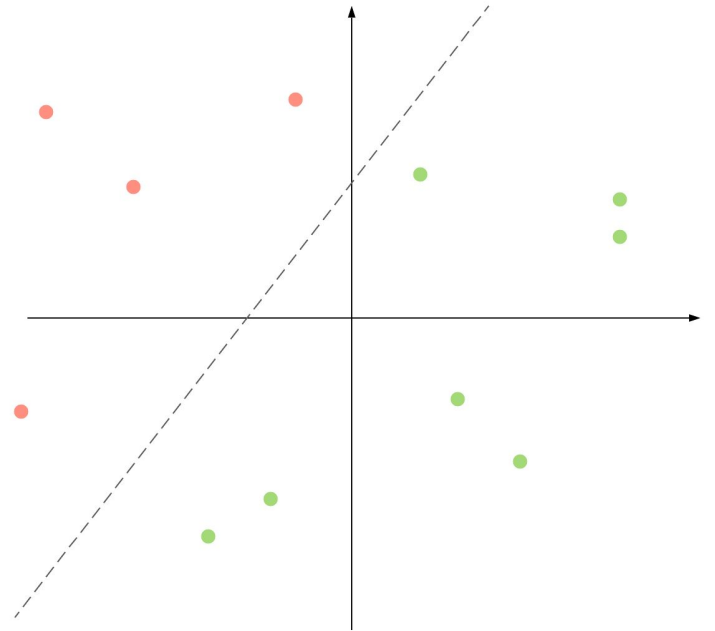
# Perceptrons

- Perceptron is usually used to classify the data into two parts -- **Linear Binary Classifier.**



# Perceptrons

- Perceptron is usually used to classify the data into two parts --  
**Linear Binary Classifier.**
  - **Weights** shows the **strength** of the particular node.
  - **Activation functions** are used to map the input between the required values



# Multilayer Perceptrons

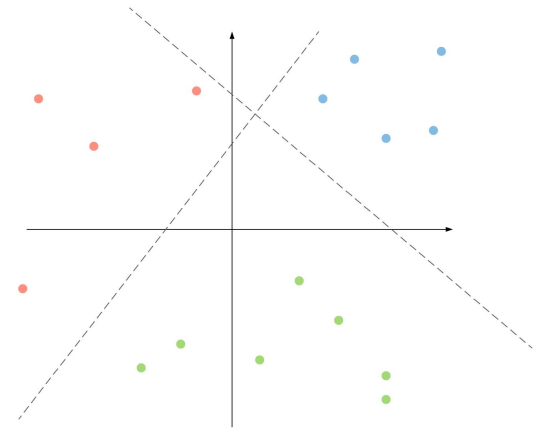
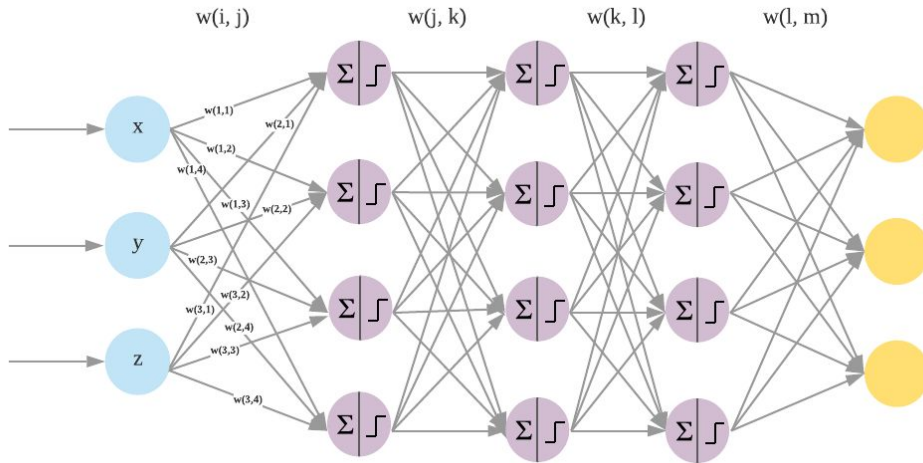
**What if we want to be able to distinguish between more classes?**



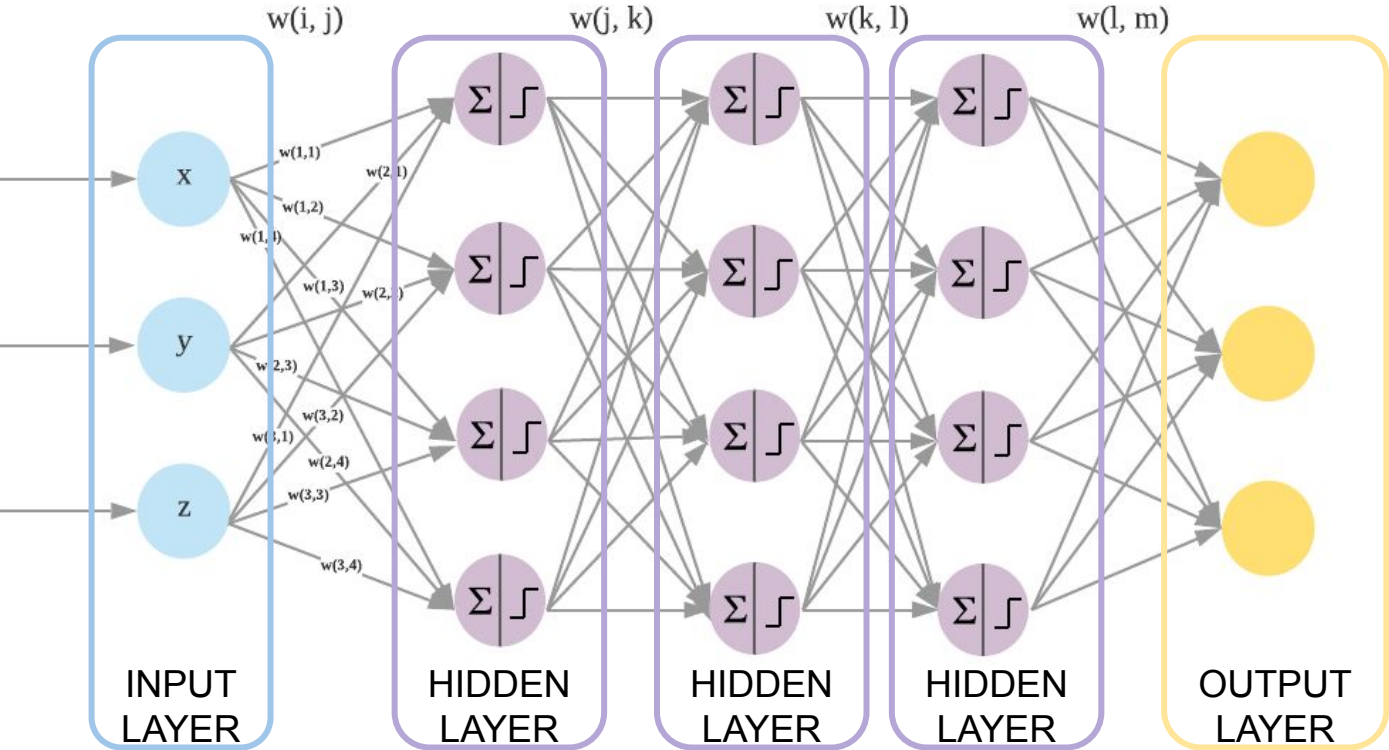
# Multilayer Perceptrons

**What if we want to be able to distinguish between more classes?**

- Introduce more perceptrons !



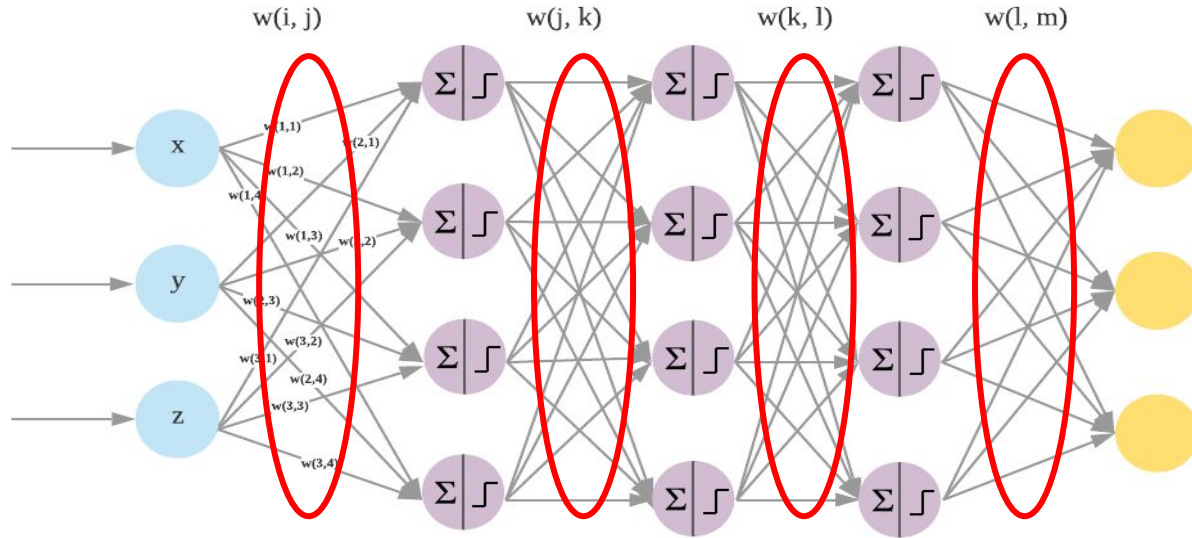
# Multilayer Perceptrons



In order to correctly classify things, the network must be **learned**.

# But first, **what** do we need to learn?

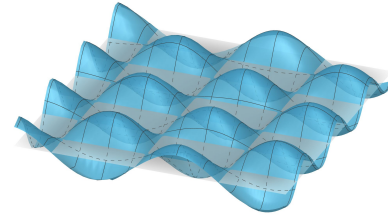
The parameters (or the weights)



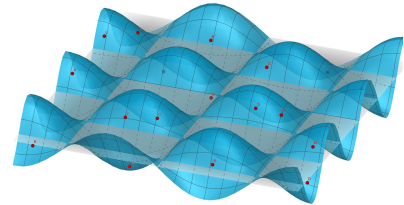
# How do we learn?

→ Actual Function that we are trying to model:

- ◆ Note: We don't know the actual function.



→ We only have several sample data points on this function.



→ Our goal:

- ◆ **Estimate the function with the given samples.**

# How do we learn?

→ A measurement of **error**

- ◆ How much off is the **network output** with respect to the **desired output**

The diagram shows the loss function equation  $Loss(W) = \frac{1}{N} \sum_i div(f(X_i, W), d_i)$  with several annotations. Arrows point from text labels to parts of the equation: 'Number of samples' points to  $\frac{1}{N}$ ; 'For each sample' points to the summation symbol  $\sum_i$ ; 'Divergence function' points to  $div$ ; 'Sample value' points to  $X_i$ ; 'Current weights of estimated function' points to  $W$ ; 'MLP' points to the function  $f$ ; 'Network Output' points to  $f(X_i, W)$ ; and 'Desired Output' points to  $d_i$ . The function  $f(X_i, W)$  is enclosed in a rounded rectangle.

$$Loss(W) = \frac{1}{N} \sum_i div(f(X_i, W), d_i)$$

Annotations:

- Number of samples
- For each sample
- Divergence function
- Sample value
- Current weights of estimated function
- MLP
- Network Output
- Desired Output

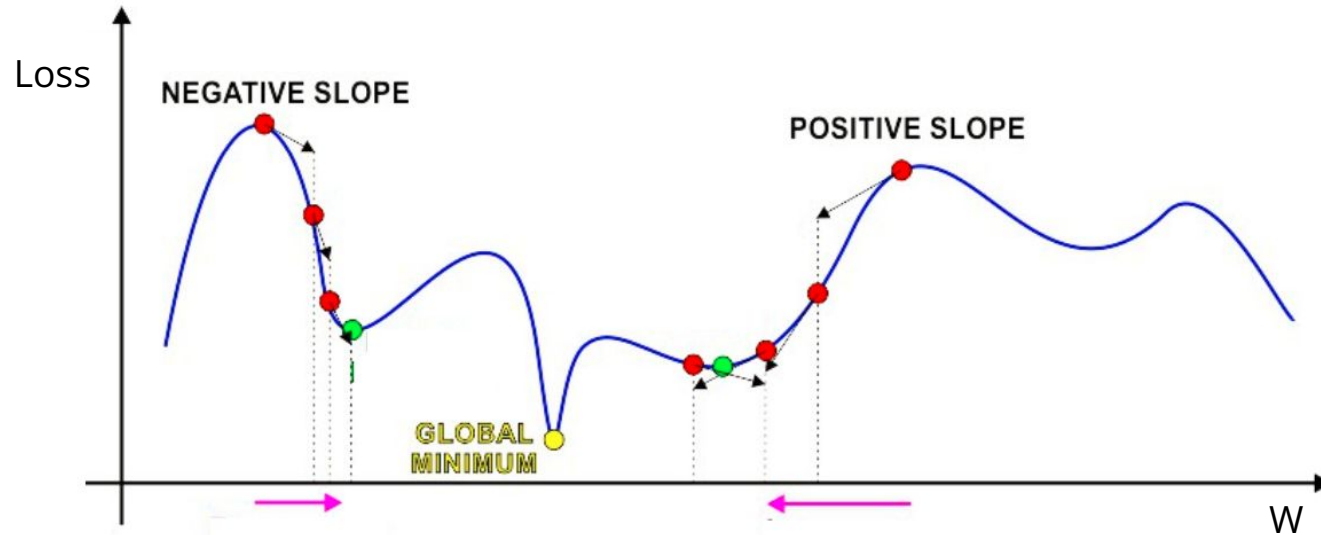
→ Our goal (more specifically):

- ◆ Minimize the loss

$$\hat{W} = \arg \min_W Loss(W)$$

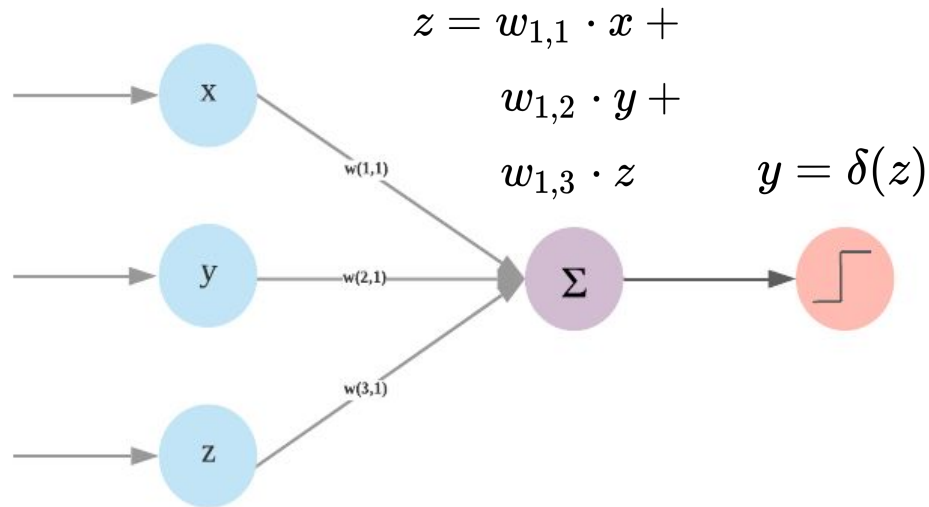
# How do we learn?

→ Gradient Descent



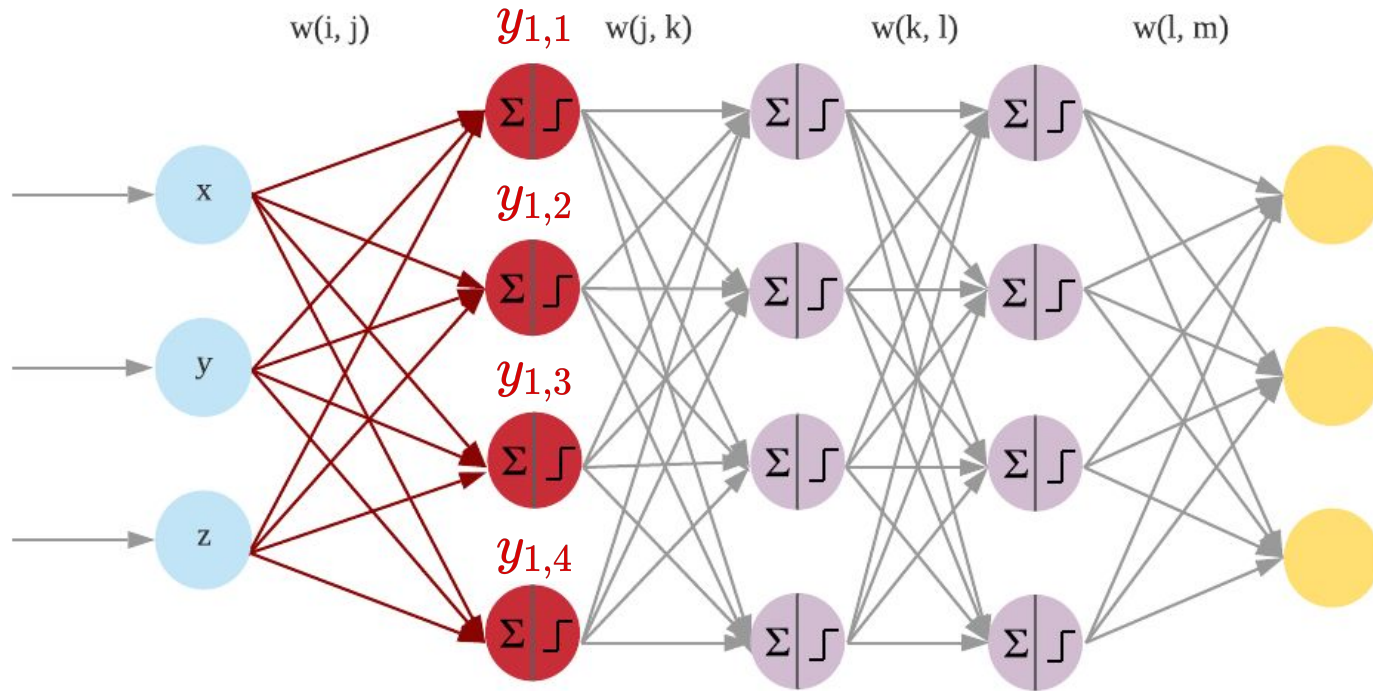
# Forward Pass

- For each single perceptron

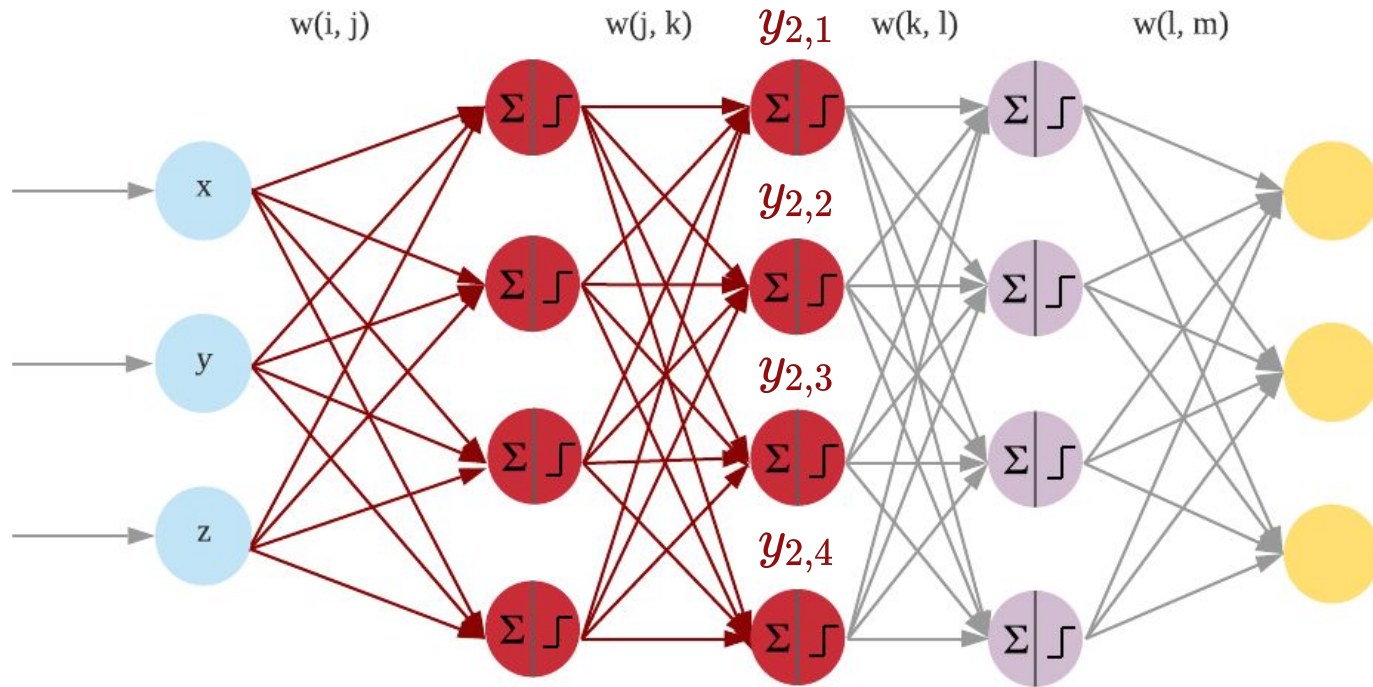




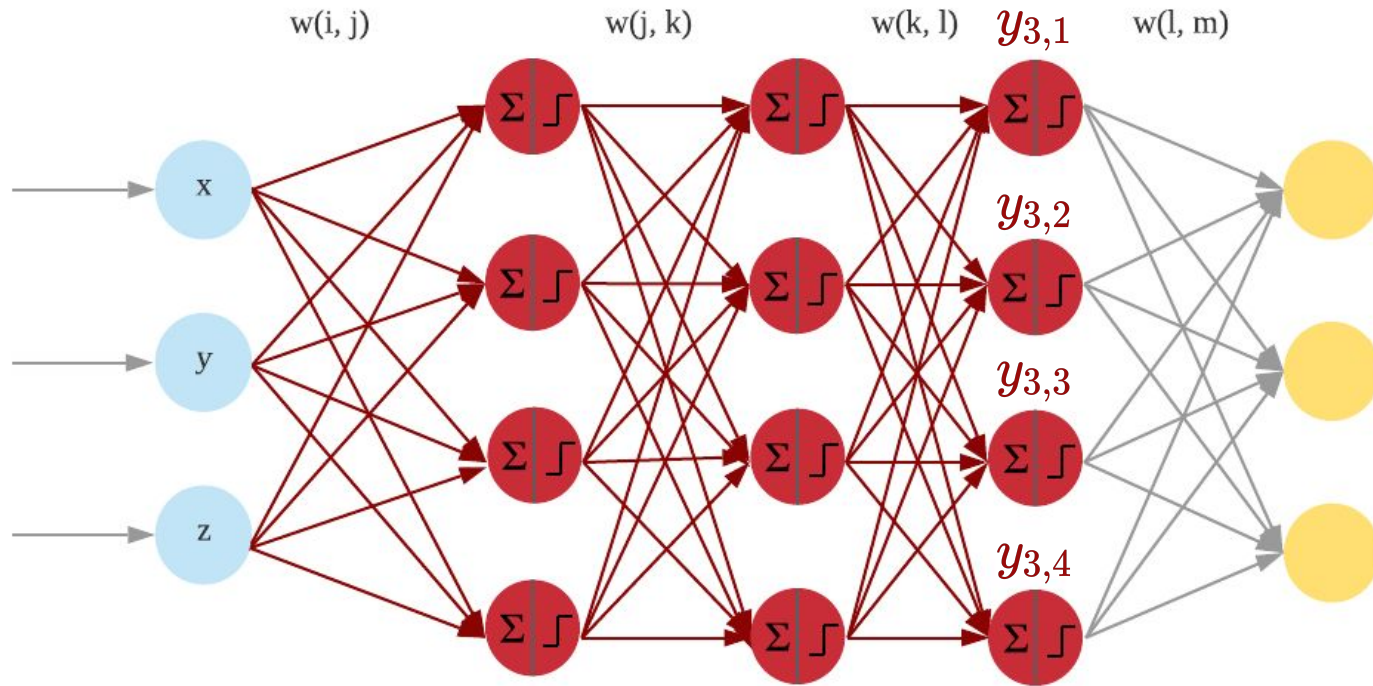
# Forward Pass



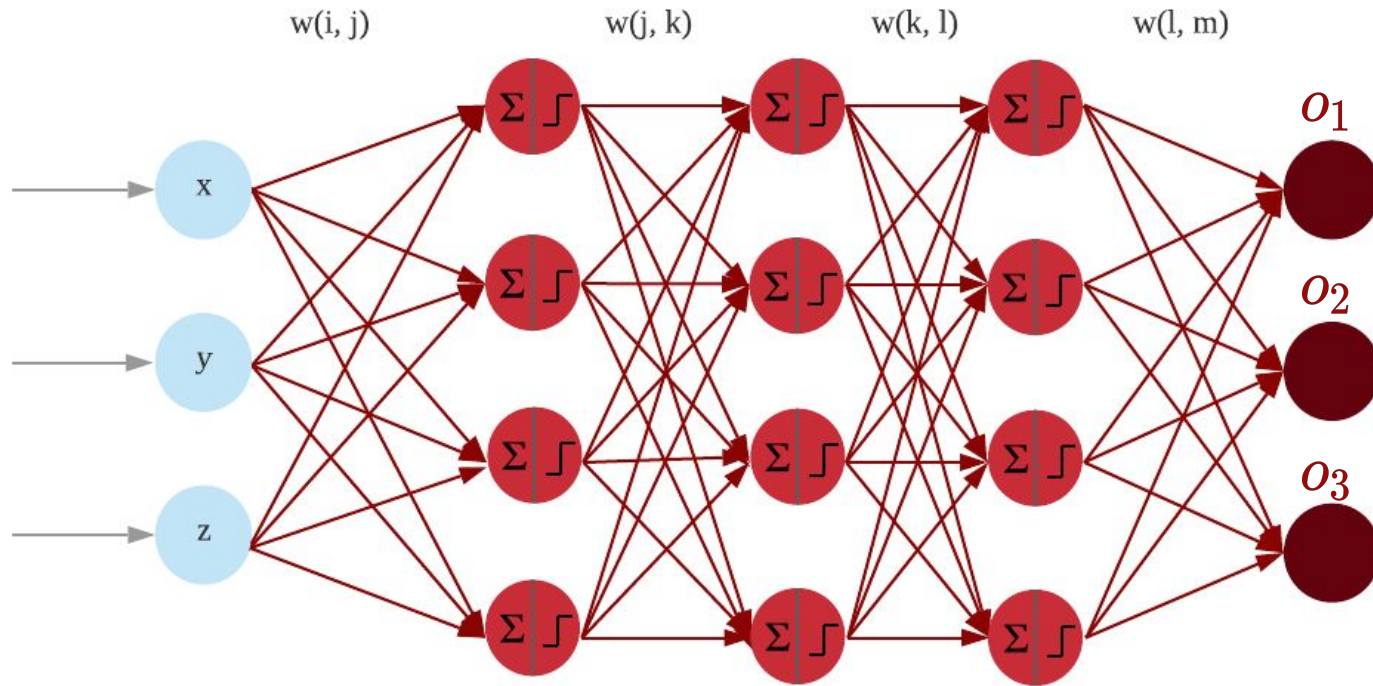
# Forward Pass



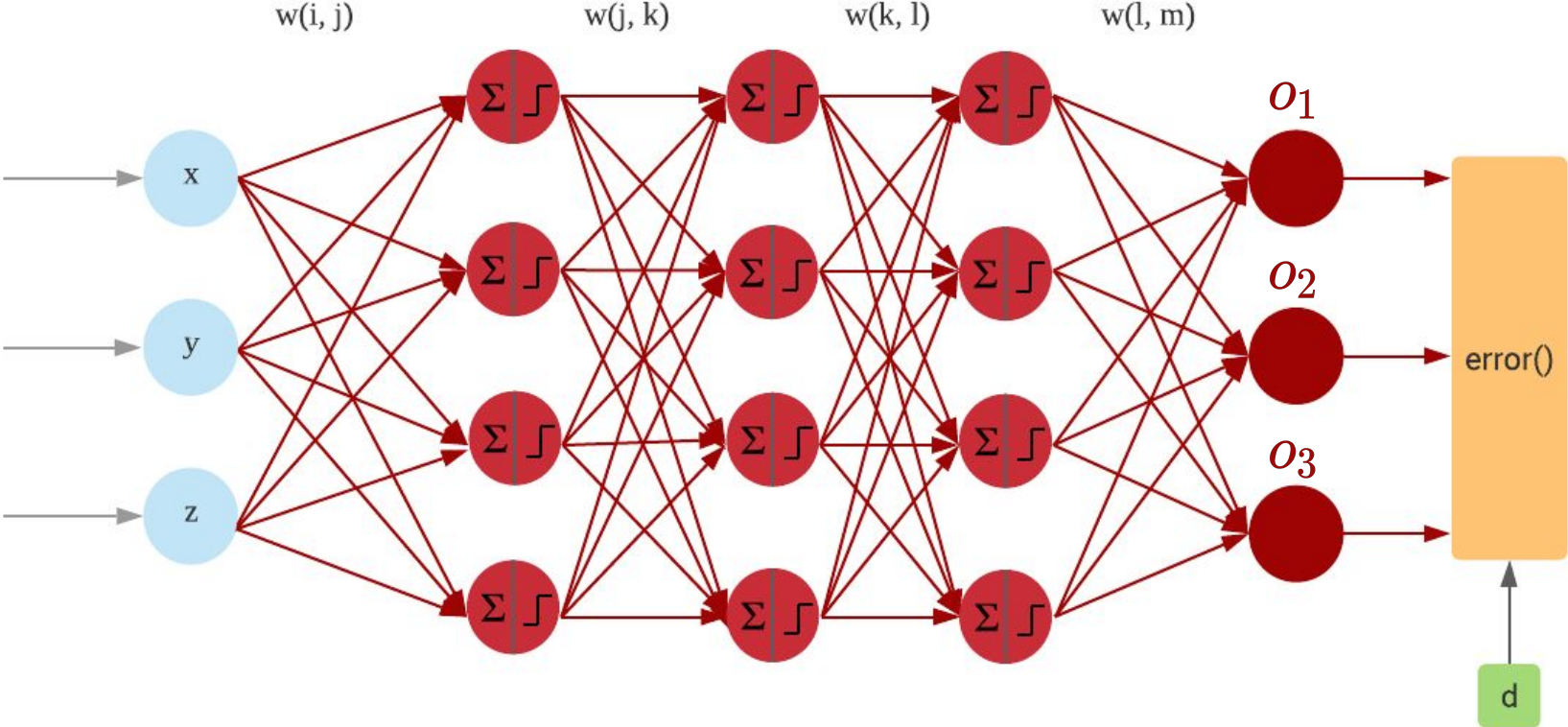
# Forward Pass



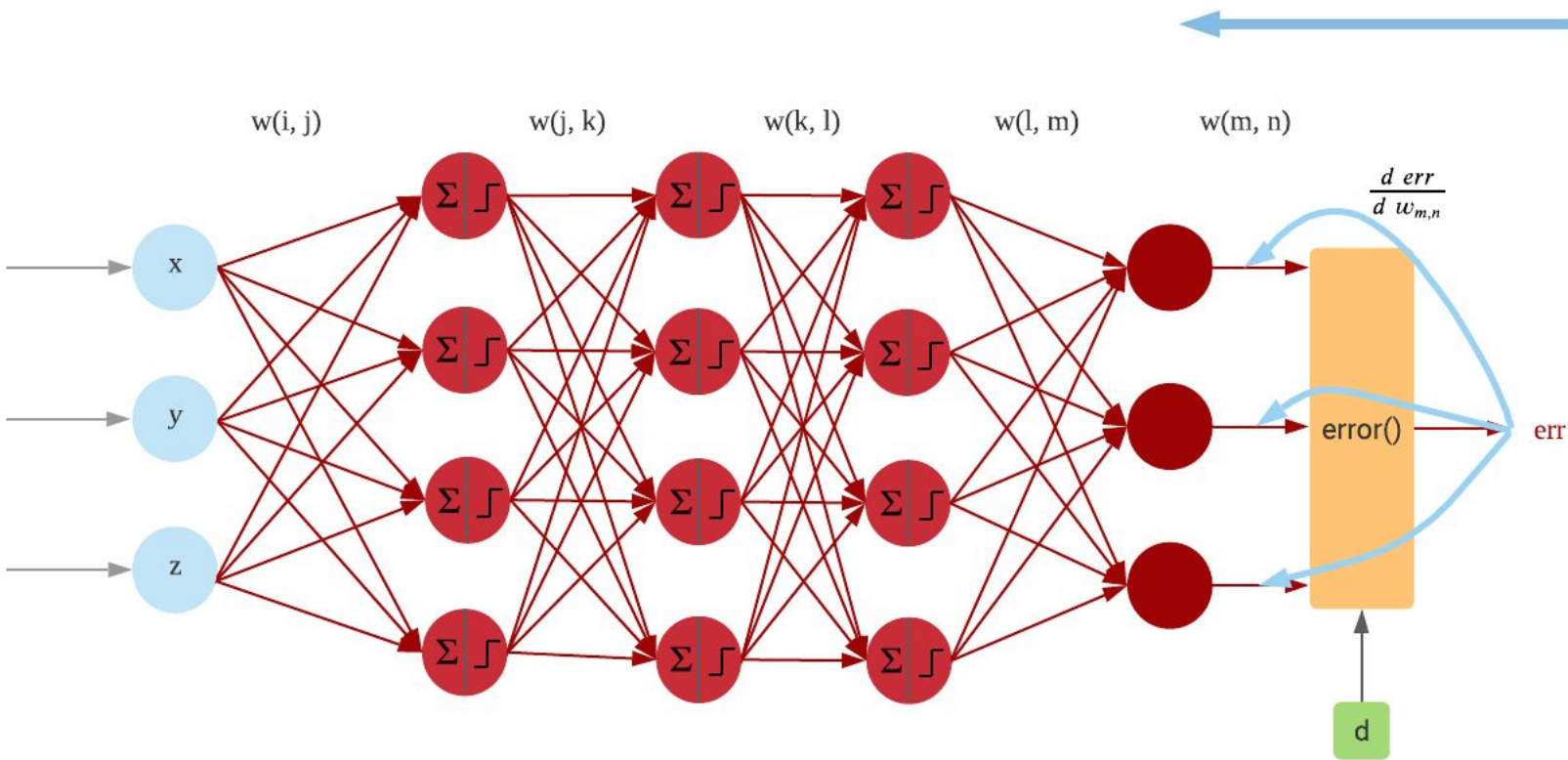
# Forward Pass



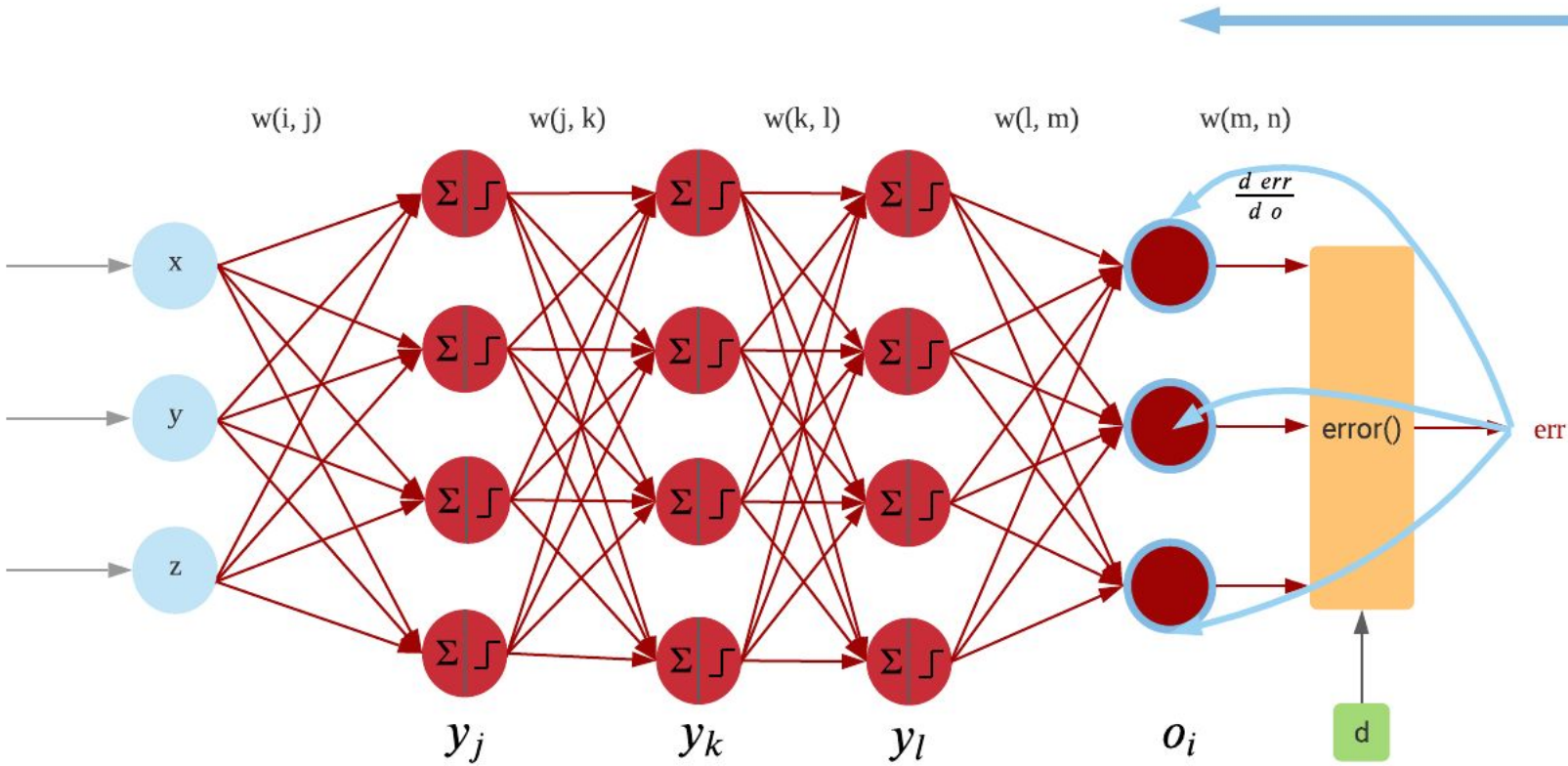
# Error



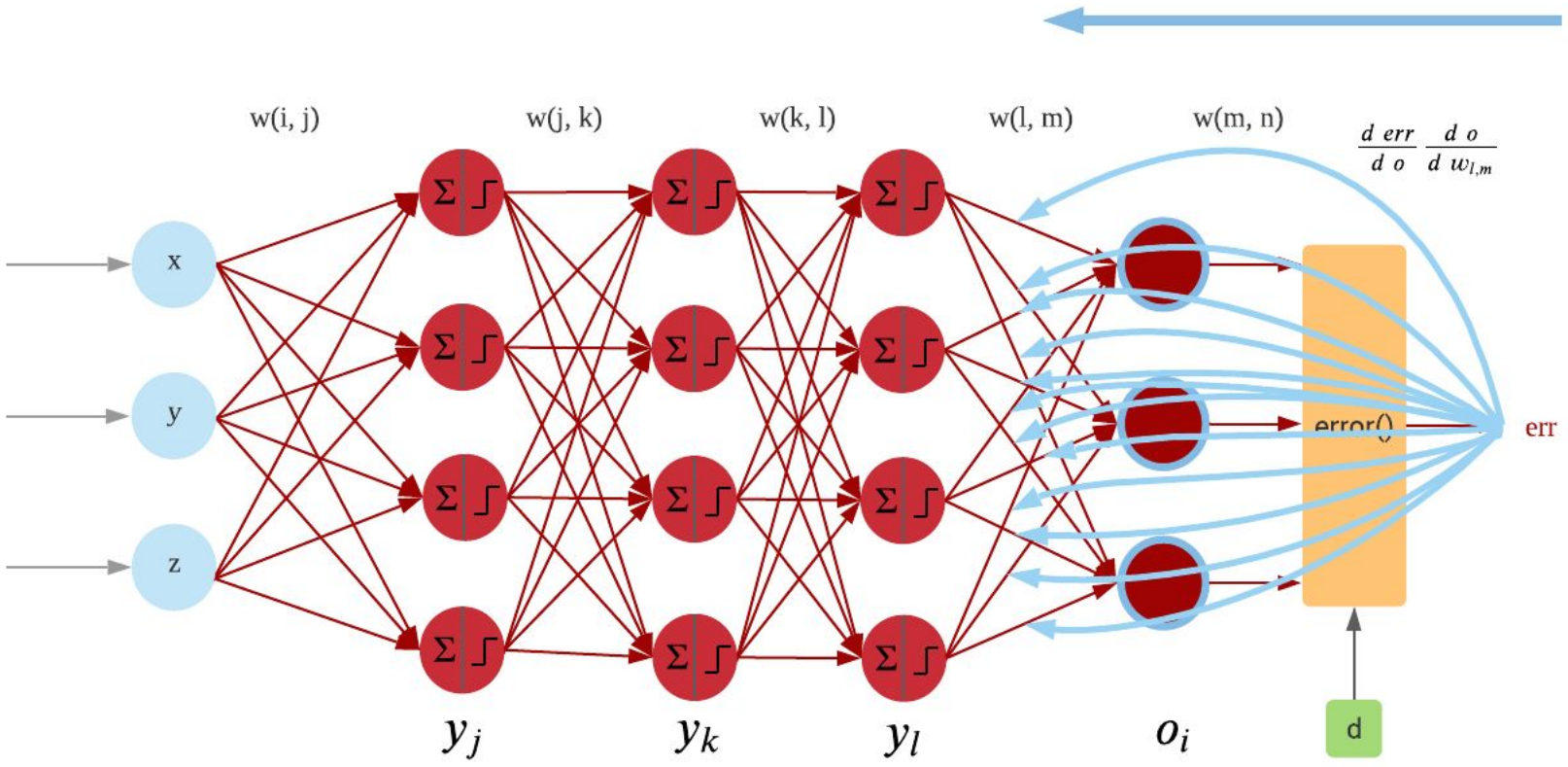
# Backpropagation



# Backpropagation

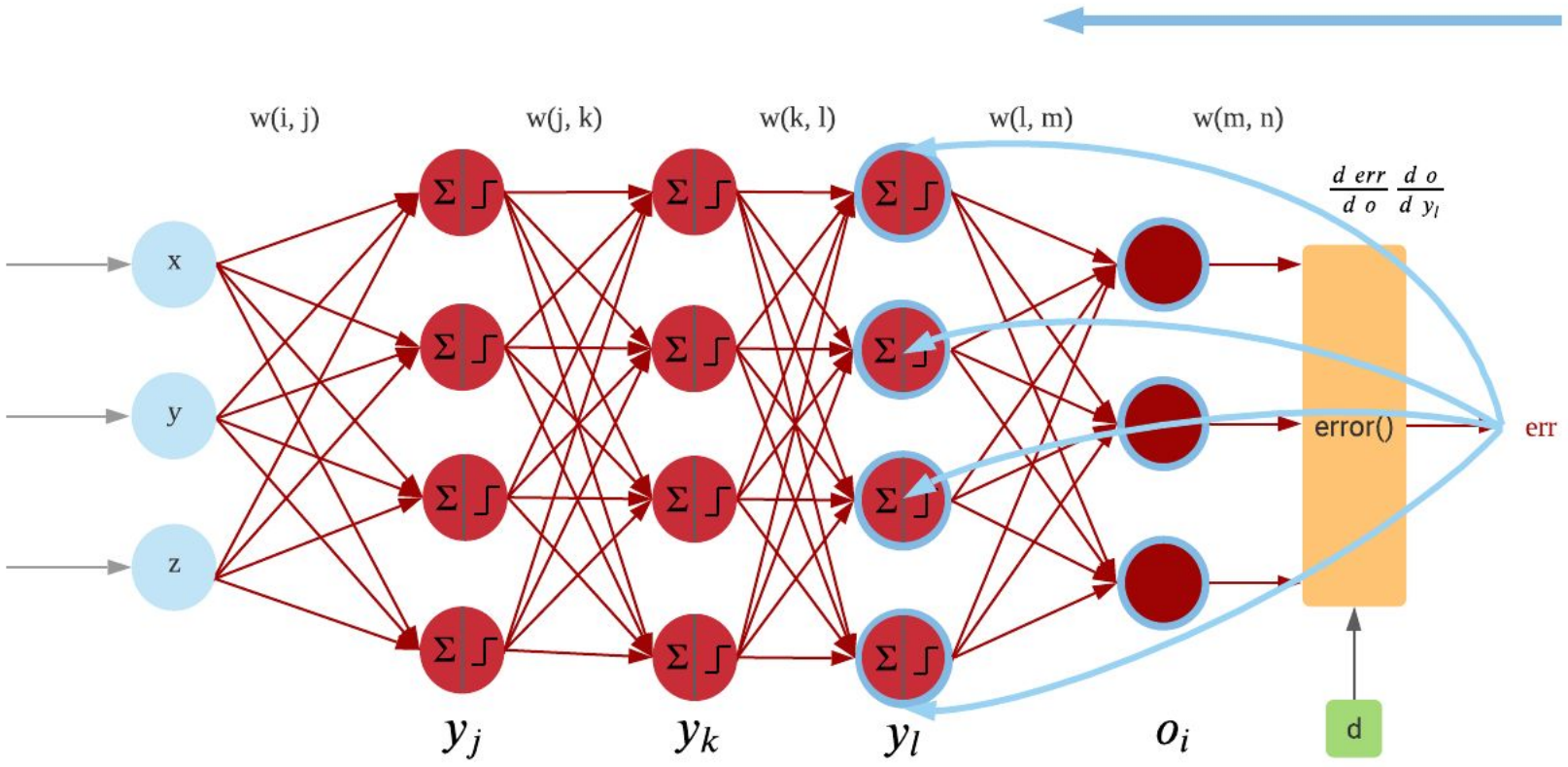


# Backpropagation

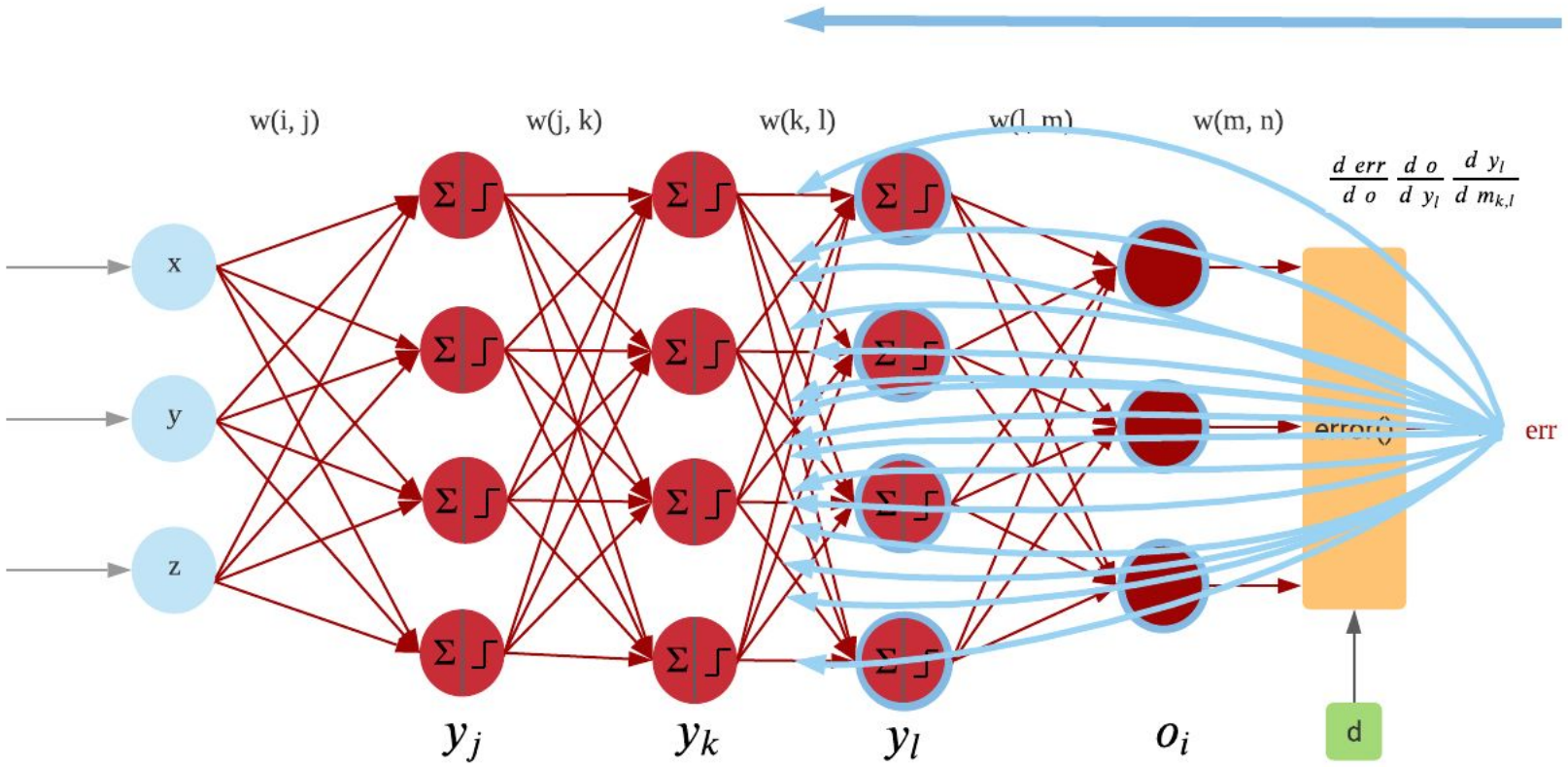




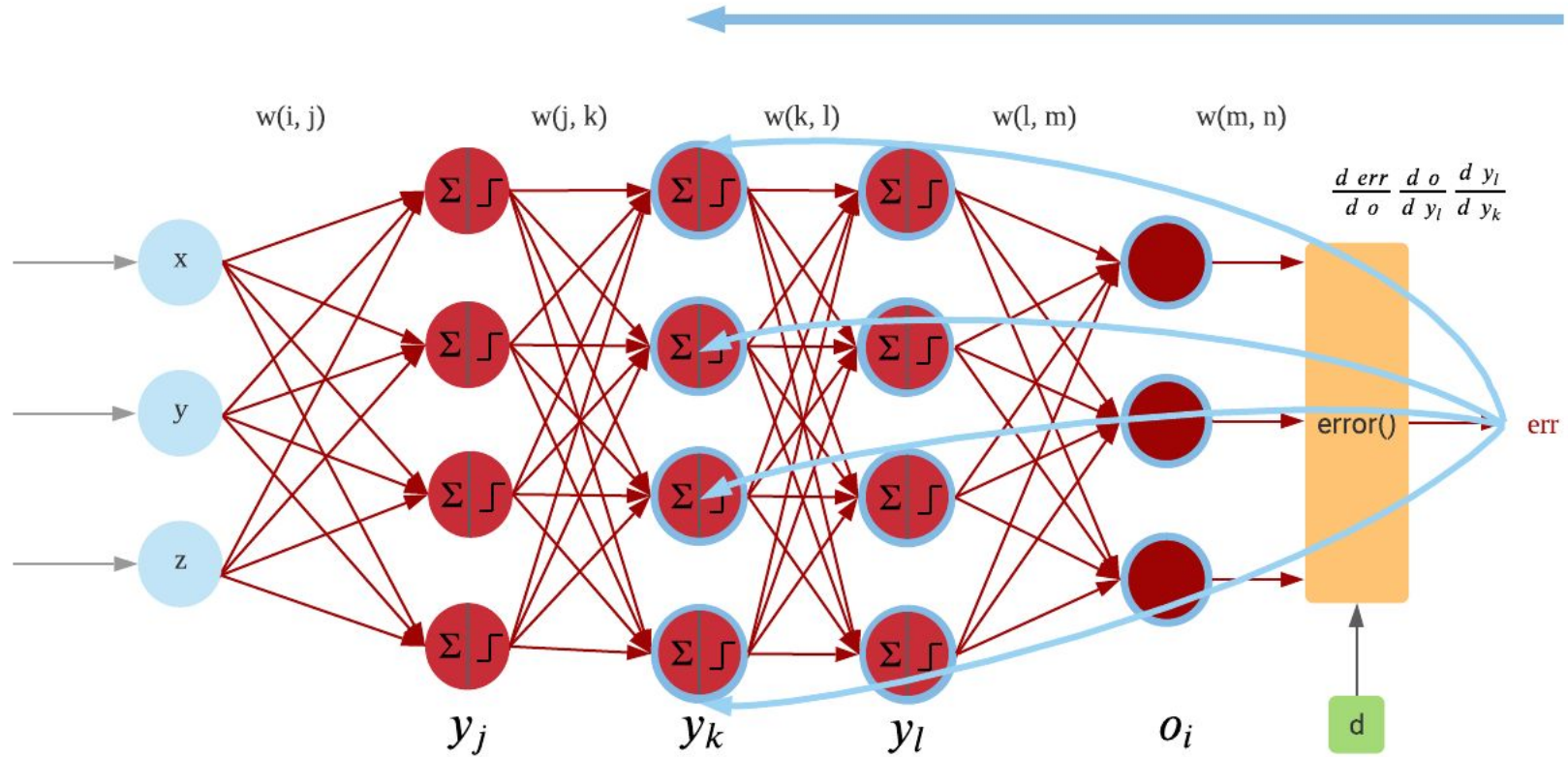
# Backpropagation



# Backpropagation



# Backpropagation



# Backpropagation

.....

# Backpropagation

All gradients of weights w.r.t error are calculated!

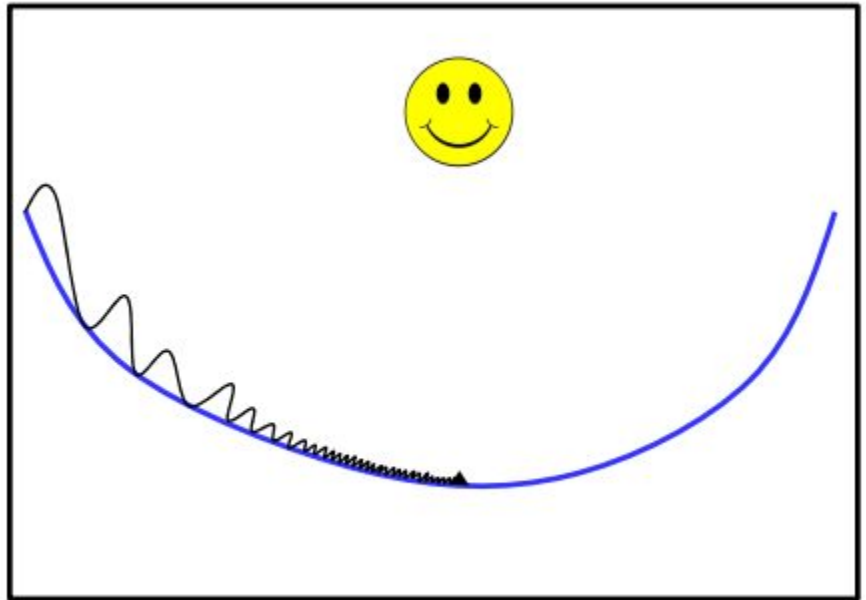
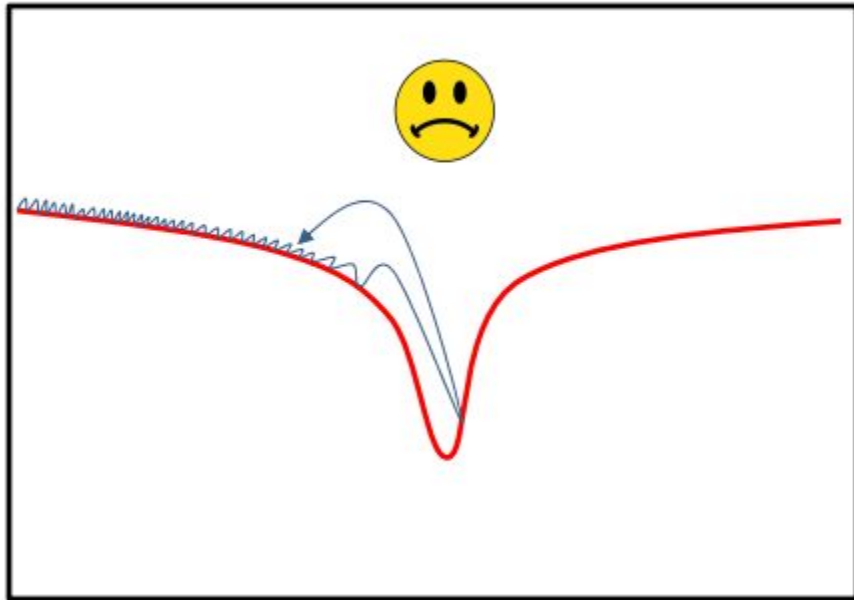
# Update Weights

$$W \leftarrow W - \eta \cdot \nabla_W \text{Loss}(W)$$

learning  
rate

gradient

# What should be the learning rate?



# Optimizers

Gradient Descent:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla_{\theta} J(\theta)$$

Momentum (<http://proceedings.mlr.press/v28/sutskever13.pdf>):

$$\begin{aligned} m_{t+1} &= \mu \cdot m_t + \alpha \cdot \nabla_{\theta} J(\theta) \\ \theta_{t+1} &= \theta_t - m_{t+1} \end{aligned}$$

Adagrad (<https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf>):

$$\begin{aligned} g &\leftarrow \nabla_{\theta} J(\theta) \\ r &\leftarrow r + g^2 \\ \Delta\theta &\leftarrow \frac{\delta}{\sqrt{r + \epsilon}} \cdot g \\ \theta &\leftarrow \theta - \Delta\theta \end{aligned}$$



# Optimizers (Cont')

Adam (<https://arxiv.org/pdf/1412.6980.pdf>):

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

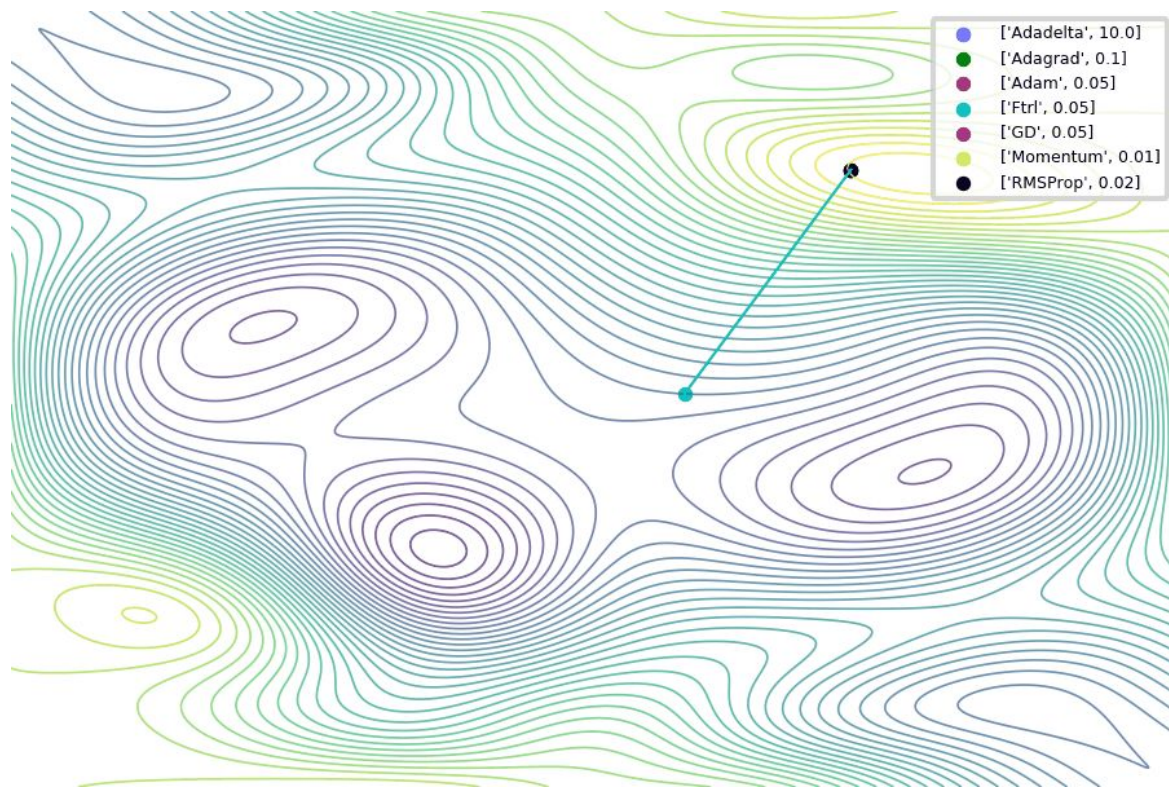
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

$$\hat{m}_t = m_t / (1 - \beta_1^t)$$

$$\hat{v}_t = v_t / (1 - \beta_2^t)$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$$

# Visualization



# Some fun with TF Playground

Tinker With a **Neural Network** Right Here in Your Browser.  
Don't Worry, You Can't Break It. We Promise.

The screenshot displays the TensorFlow Playground interface. At the top, there are controls for Epochs (000,000), Learning rate (0.03), Activation (Tanh), Regularization (None), Regularization rate (0), and Problem type (Classification). Below these are several sliders: Ratio of training to test data (50%), Noise (0), and Batch size (10). A 'REGENERATE' button is at the bottom left.

The main area shows a neural network diagram with the following components:

- DATA:** A section titled 'Which dataset do you want to use?' with several dataset icons.
- FEATURES:** A section titled 'Which properties do you want to feed in?' with a list of features:  $X_1$ ,  $X_2$ ,  $X_1^2$ ,  $X_2^2$ ,  $X_1 X_2$ , and  $\sin(X_1)$ .
- 2 HIDDEN LAYERS:** The first hidden layer has 4 neurons, and the second has 2 neurons. Lines of varying thickness connect neurons between layers, representing weights. A tooltip indicates: 'The outputs are mixed with varying weights, shown by the thickness of the lines.' Another tooltip points to a neuron: 'This is the output from one neuron. Hover to see it larger.'
- OUTPUT:** A scatter plot showing data points (orange and blue) on a 2D plane. The plot is divided into regions by a decision boundary. The output section also displays 'Test loss 0.531' and 'Training loss 0.512'.