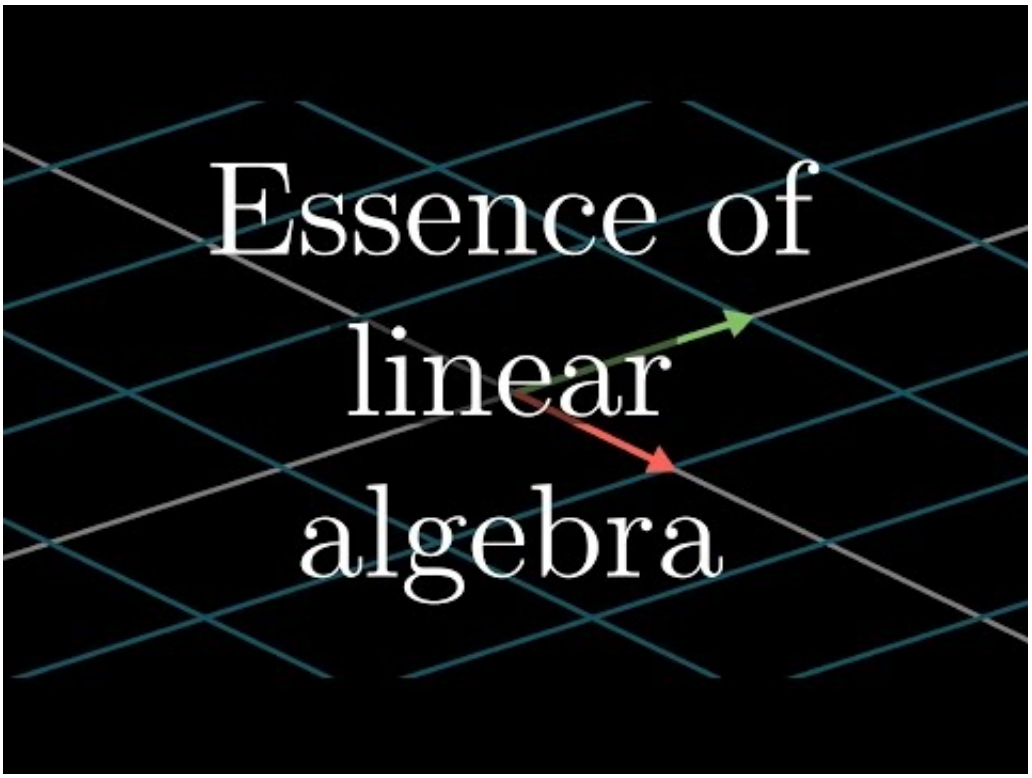
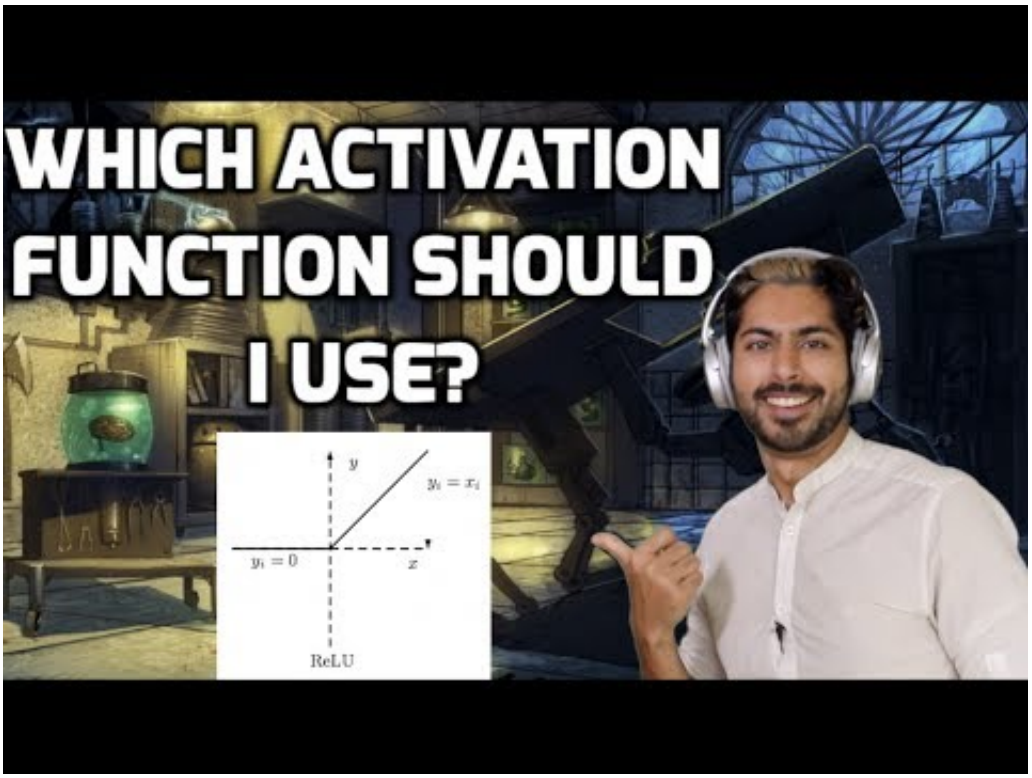
Machine Learning

Videos

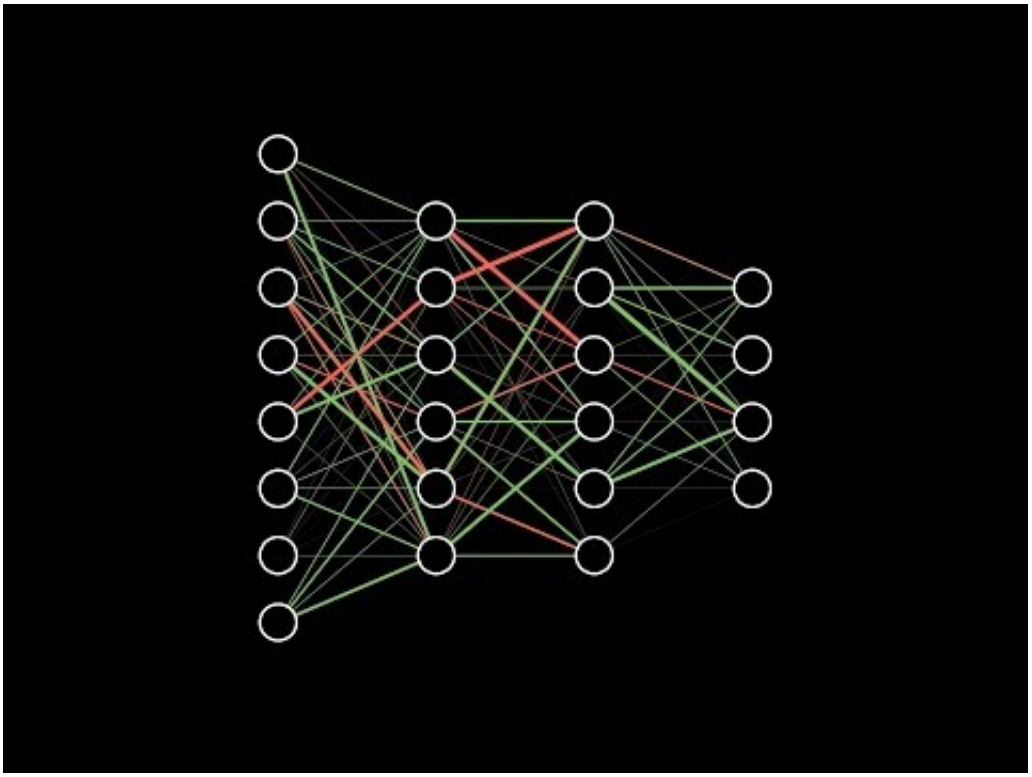
Linear Algebra



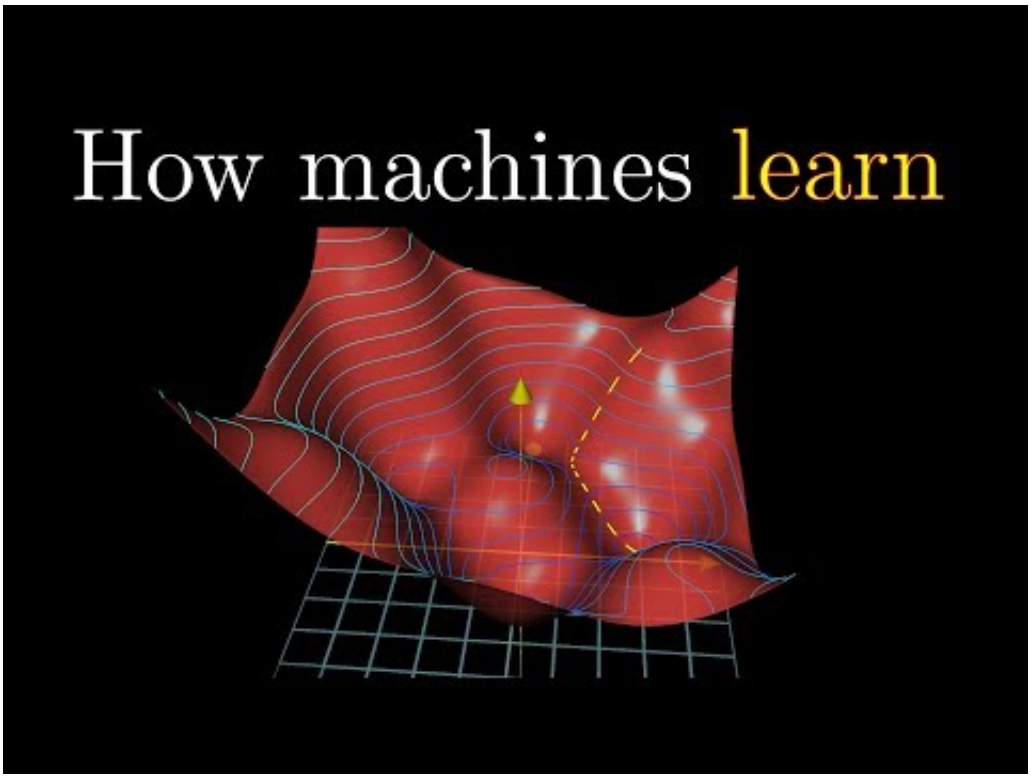
Activation Functions



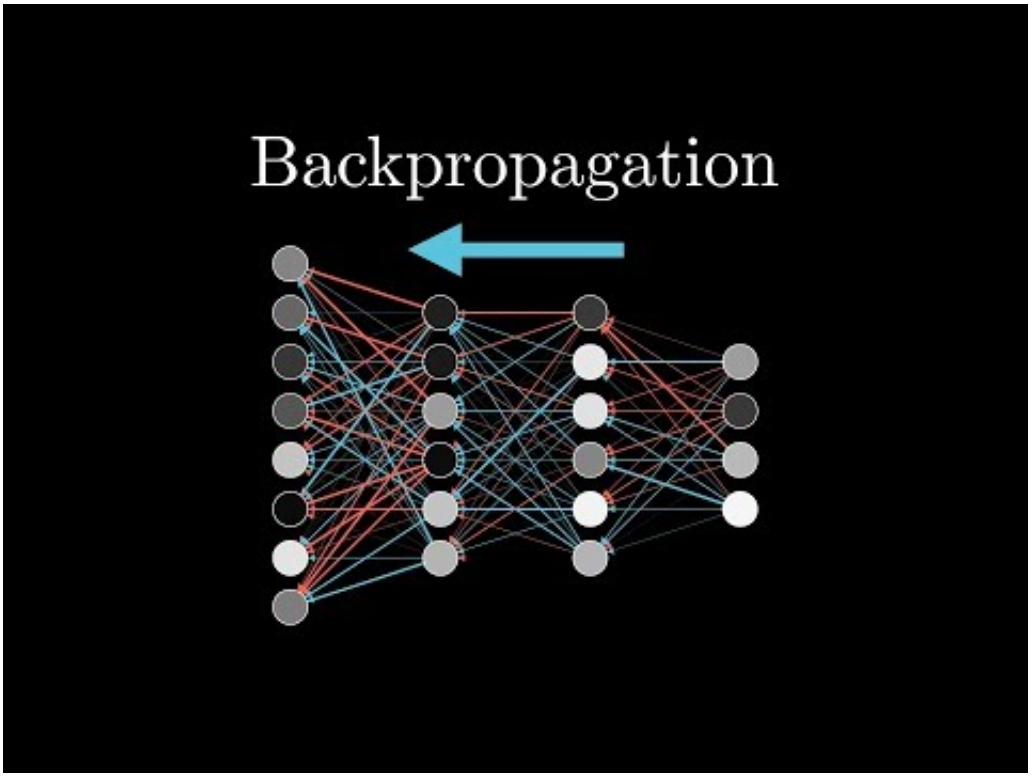
What is a Neural Network



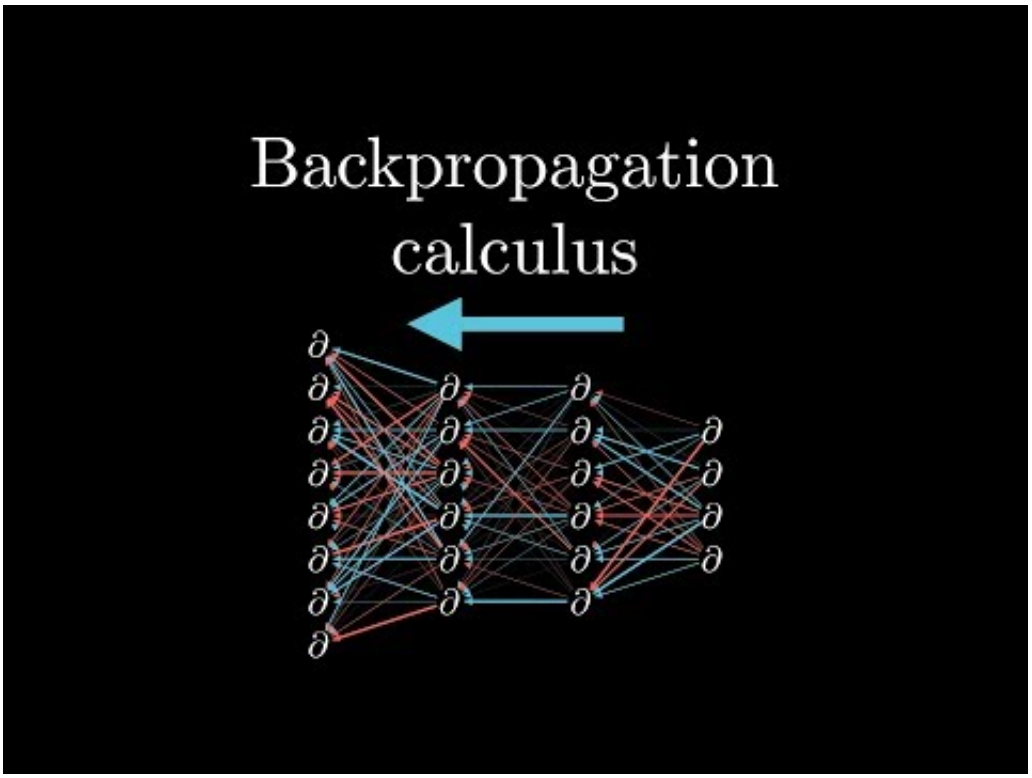
Gradient descent, how neural networks learn



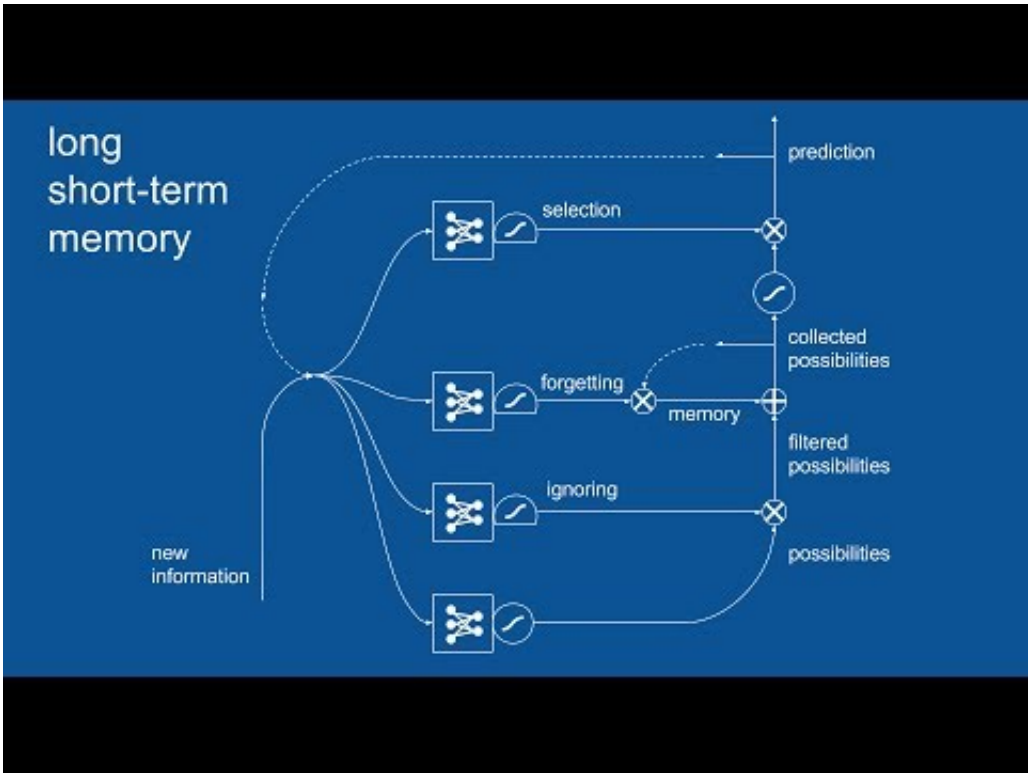
Backpropagation



Backpropagation Calculus



Long Short-Term Memory



Links

Blogs / Articles

The Unreasonable Effectiveness of Recurrent Neural Networks Blog Post

Data Science and Robots Blog

Understanding LSTM Networks

EEG event classification with convolutional neural networks - A First Pass/Brief Overview

Reddit Machine learning

Closing the Simulation-to-Reality Gap for Deep Robotic Learning

pybullet by Alexander Fabisch

Pytransform - lightweight tool to translate between conventions, debug, and visualize

Alexander Fabisch - Robotics Python

Code

Torch-rnn Char-rnn update

Xgboost

Xgboost Node

Frameworks & Websites

TensorFlow

Keras

Try TensorFlow

Colab

Kaggle

Moniel - Interactive Notation for Computational Graphs

OpenAI

OpenAI Github

DeepMind - Lab 3D platform for agent-based AI research

DeepMind - Sonnet

Install Anaconda

Companies

Baidu

DeepMind

Books

Deep learning with python book

Deep learning with python code

Neural Networks and Deep Learning by Michael Nielsen

Programming Robots with ROS

ROS Programming: Building Powerful Robots

ROS Robotics By Example

Lectures

Recurrent Neural Network Lecture

People

Andrej Karpathy

Andrew Ng

Geoffrey Hinton

Notes

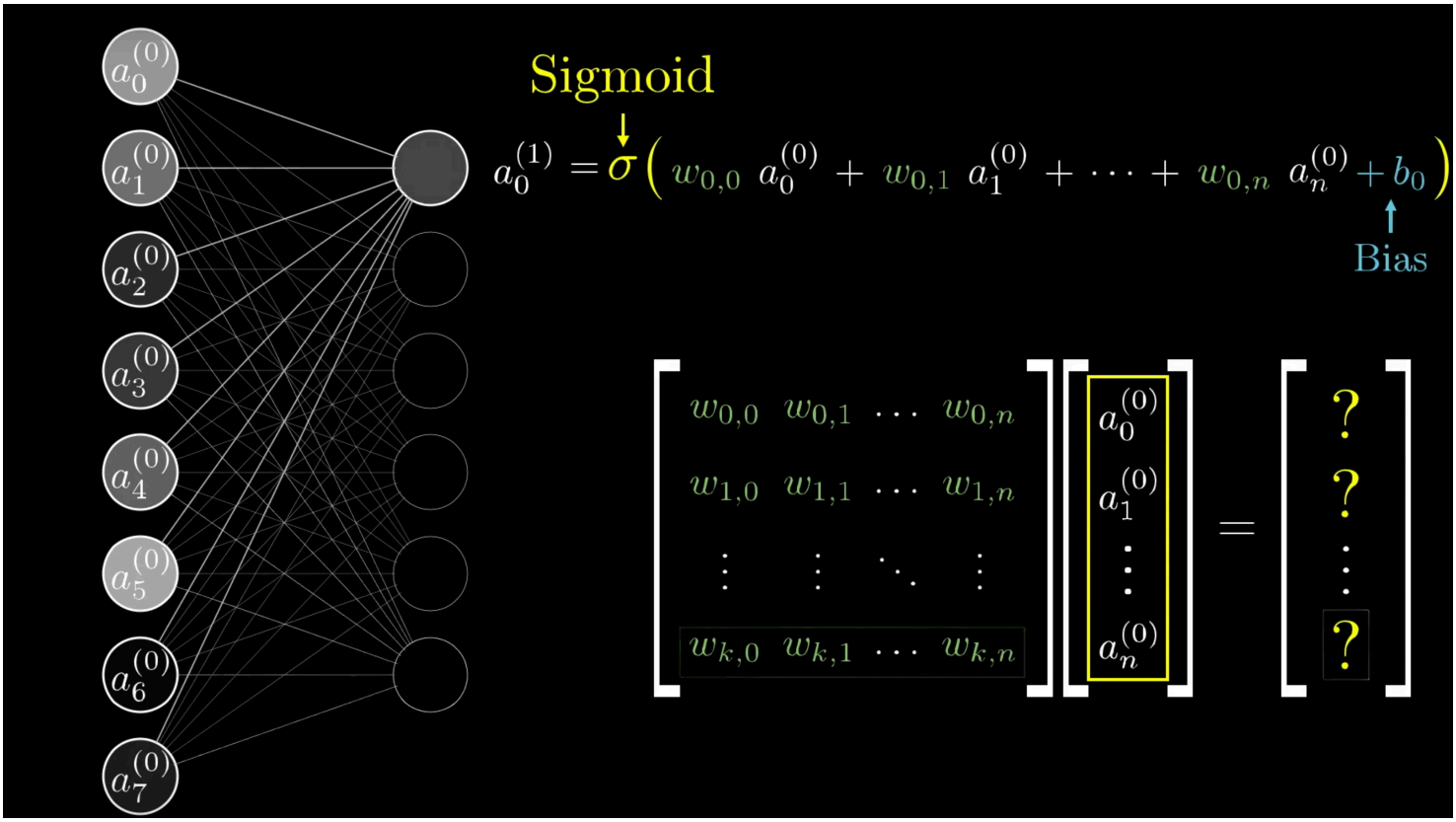
- **Convolutional Neural Network (CNN)** good for **image recognition**
- **Recurrent Neural Networks** and **Long short-term memory** good for **speech recognition**

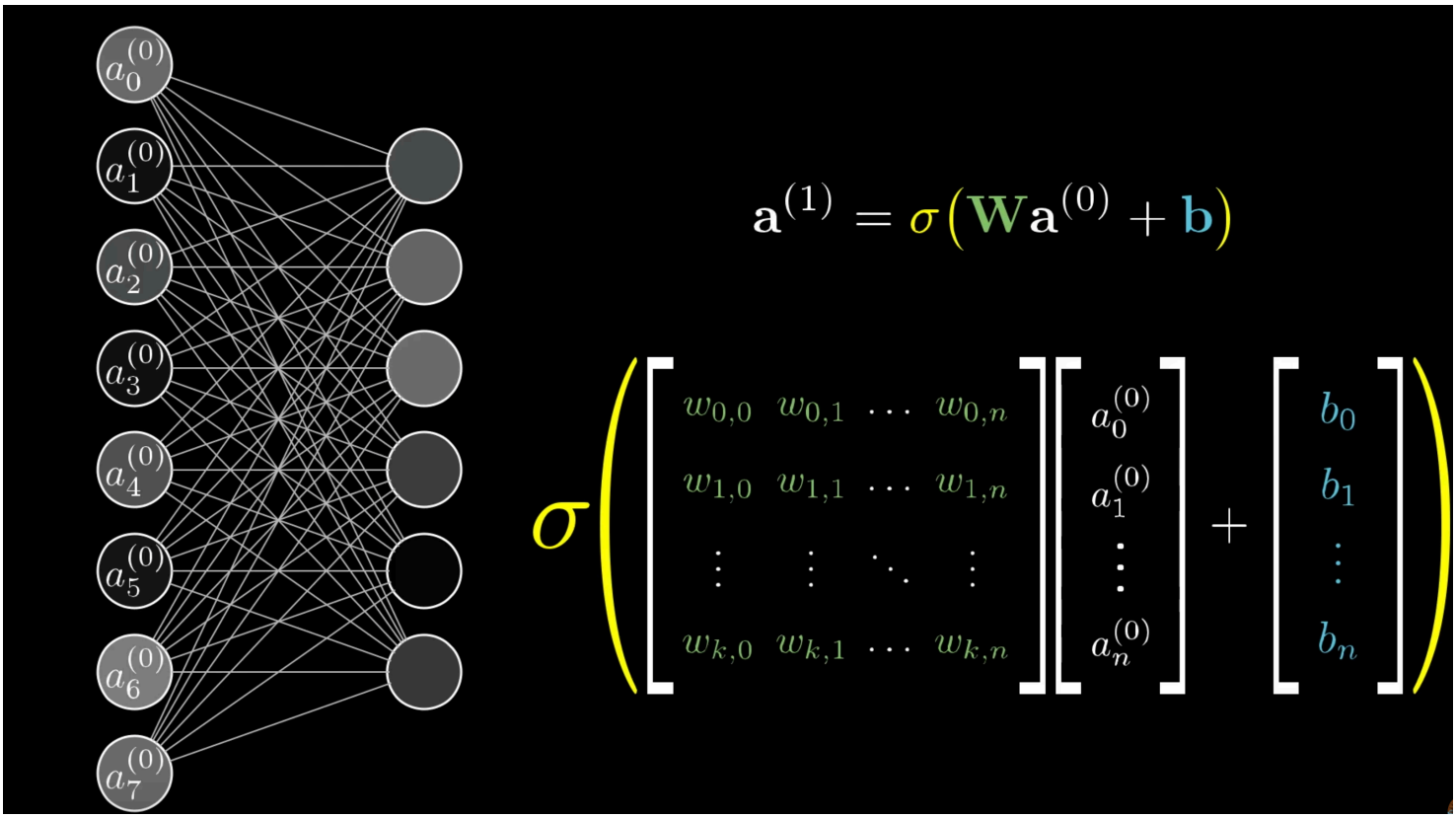
- Transforming a set of real numbers to normalized values **Logistic functions**, **Logistic Curve** - Signmoid Function
- **Weight engineering**? applying weights to areas/components you care about.
- **Cost function** provides feedback to the network
- Learning is minimizing a cost function
- Categorical data is handled using One hot encoding procedures

Activation Function & General Notation

Notation for weights and neuron activations

(Note: Sigmoid approach replaced by RELU activation function)





BackProp

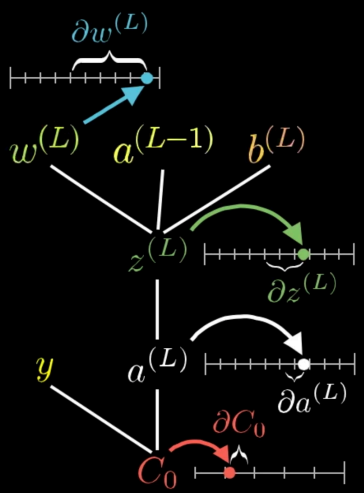
Chain Rule (Calculus) and relevance to cost function in Neural networks (Backprop)(One neuron)

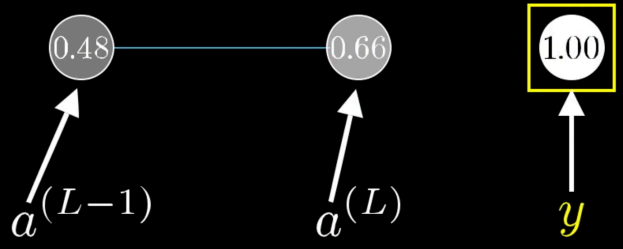
$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$$

$$C_0(\dots) = (a^{(L)} - y)^2$$

$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$





Desired output

$$\frac{\partial C_0}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \frac{\partial a^{(L)}}{\partial z^{(L)}} \frac{\partial C0}{\partial a^{(L)}}$$

$$C_0 = (a^{(L)} - y)^2$$

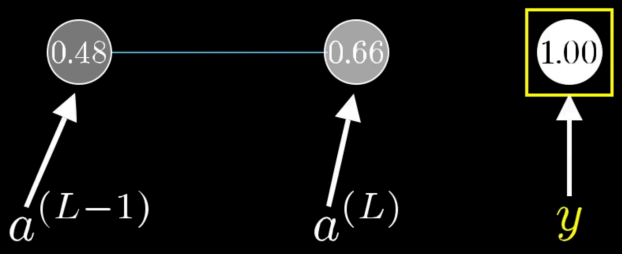
$$z^{(L)} = w^{(L)} a^{(L-1)} + b^{(L)}$$

$$a^{(L)} = \sigma(z^{(L)})$$

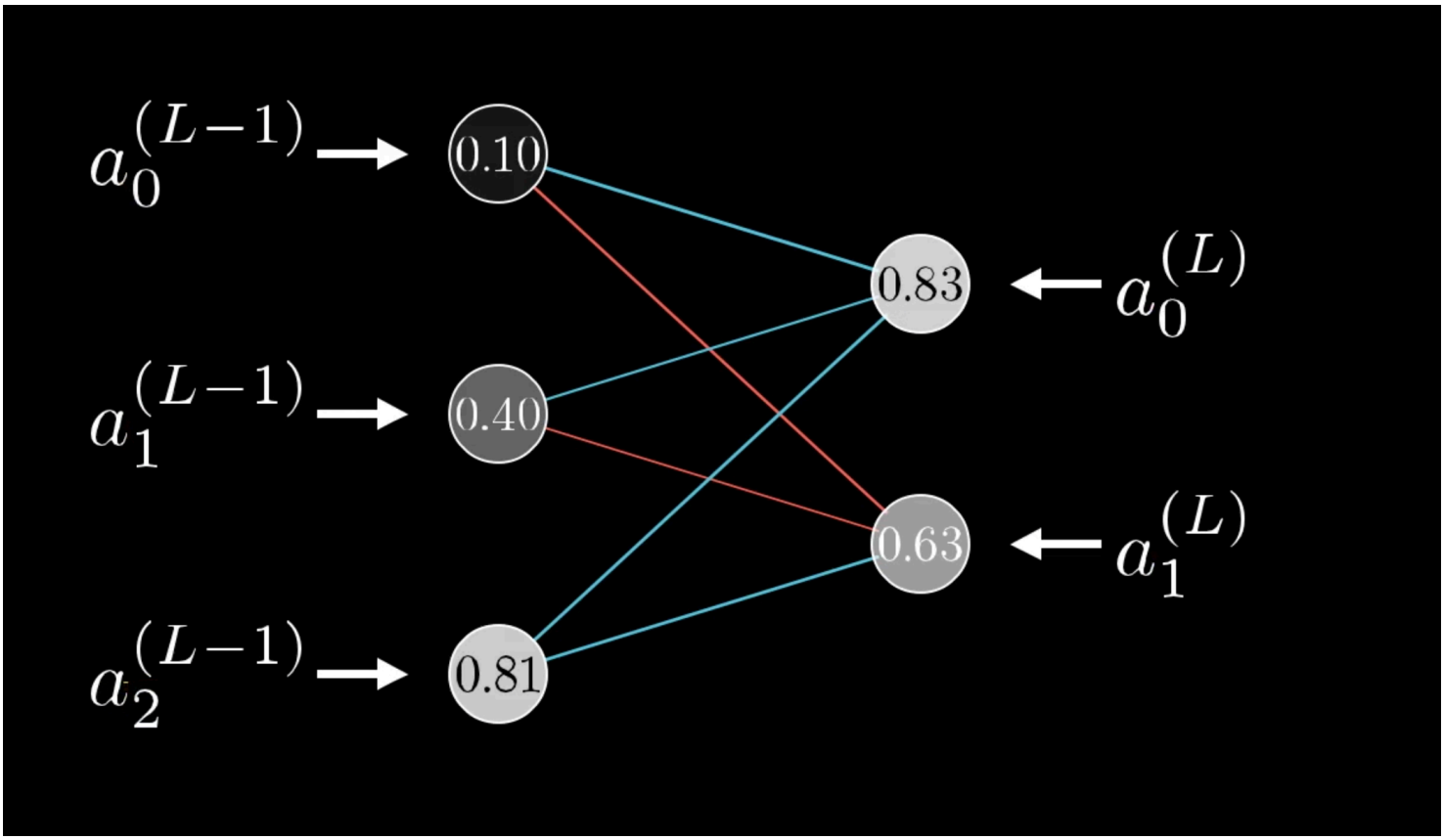
$$\frac{\partial C0}{\partial a^{(L)}} = 2(a^{(L)} - y)$$

$$\frac{\partial a^{(L)}}{\partial z^{(L)}} = \sigma'(z^{(L)})$$

$$\frac{\partial z^{(L)}}{\partial w^{(L)}} = a^{(L-1)}$$



Chain Rule (Calculus) and relevance to cost function in Neural networks (Backprop)(multi neuron)



$$\frac{\partial C_0}{\partial w_{jk}^{(L)}} = \frac{\partial z_j^{(L)}}{\partial w_{jk}^{(L)}} \frac{\partial a_j^{(L)}}{\partial z_j^{(L)}} \frac{\partial C_0}{\partial a_j^{(L)}}$$

$$z_j^{(L)} = \cdots + w_{jk}^{(L)} a_k^{(L-1)} + \cdots$$

$$a_{jj}^{(L)} = \sigma(z_{jj}^{(L)})$$

$$C_0 = \sum_{j=0}^{n_L - 1} (a_j^{(L)} - y_j)^2$$