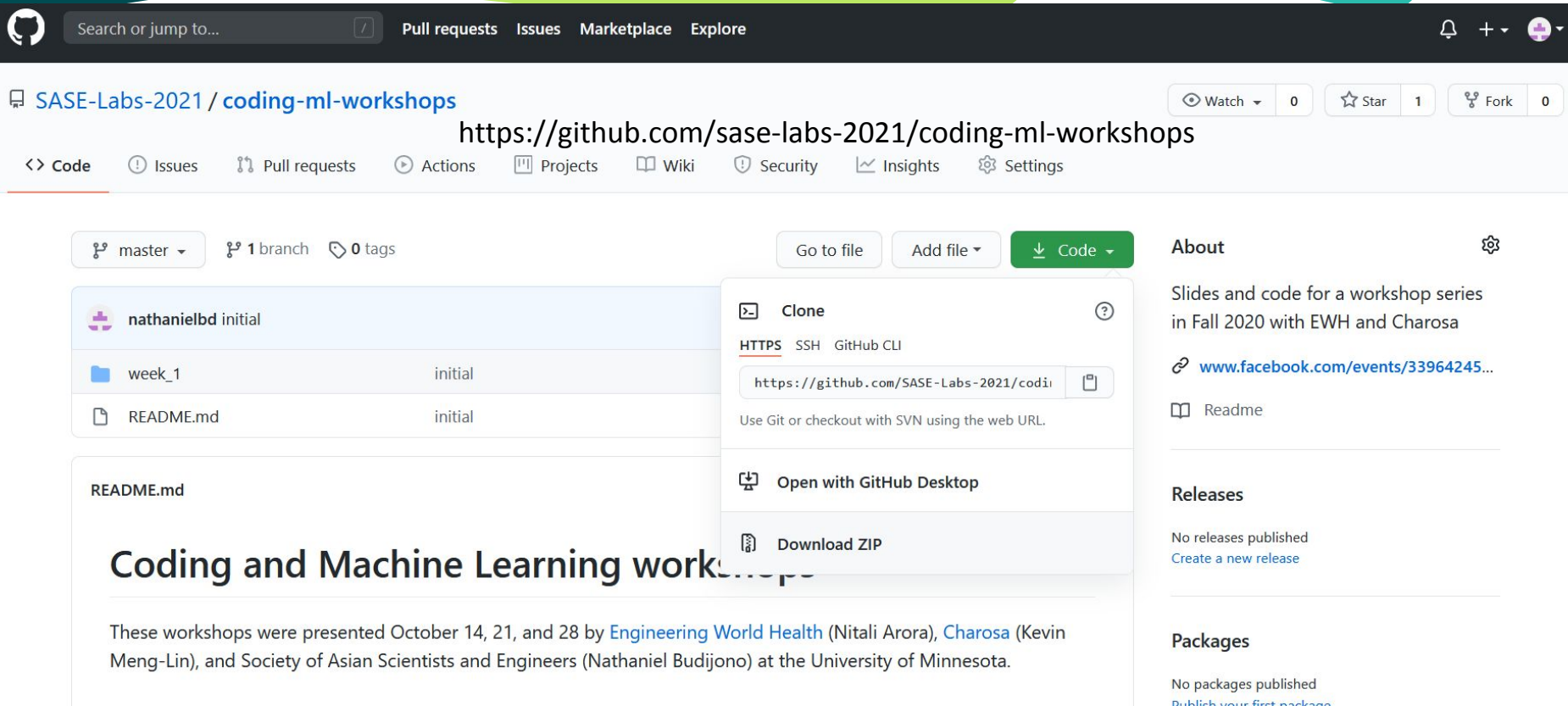
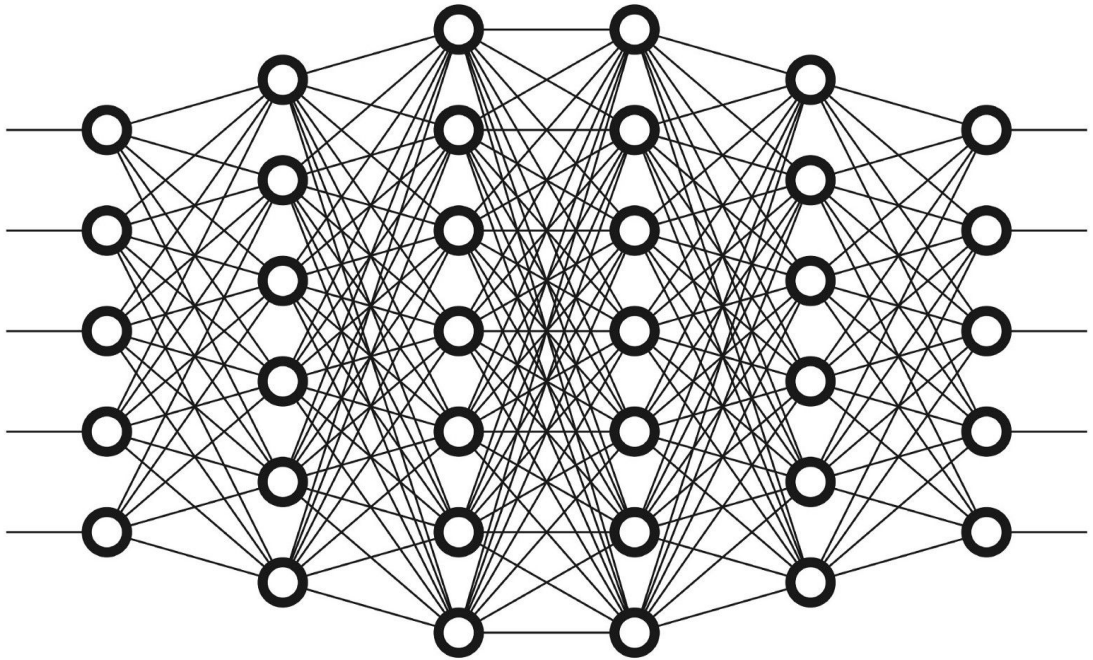
Follow along by downloading from GitHub and get Spyder open with code!



Neural Networks

Charosa & SASE Labs & EWH

Neural Networks? Artificial Neurons?

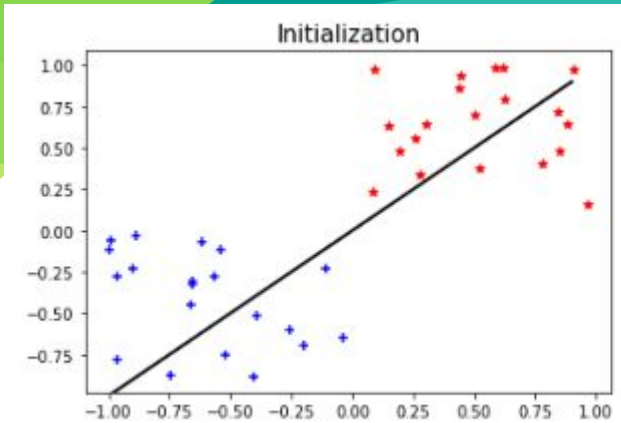


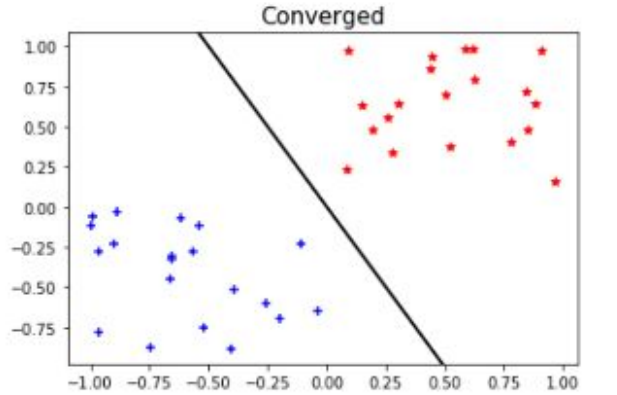
```python
import numpy
import scipy.io as io
import matplotlib.pyplot as plt

def MyPerceptron(X, y, w_0):
    steps = 0
    mistakes = 1
    w = w_0
    n = y.shape[0]
    while mistakes > 0:
        mistakes = 0
        for i in range(0,n):
            steps += 1
            if numpy.inner(w,X[i])*y[i][0] < 0:
                mistakes += 1
                w += y[i][0]*X[i]
    return steps, w
```

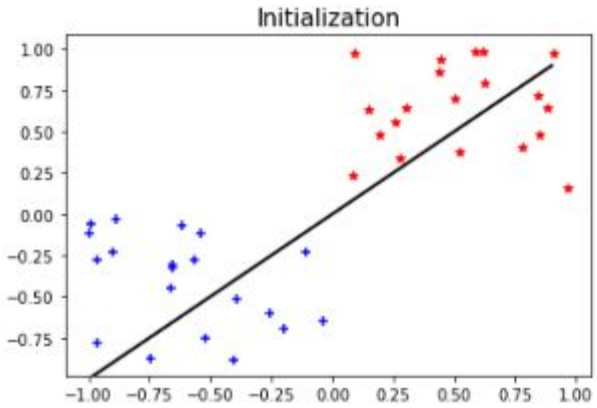
W is the dividing yellow line

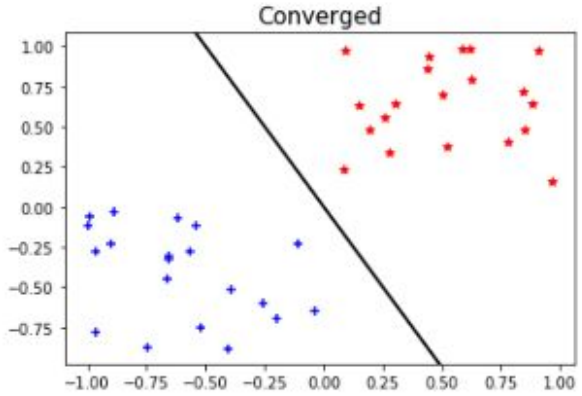
The math to calculate which side of the divider the point is on

Vector addition



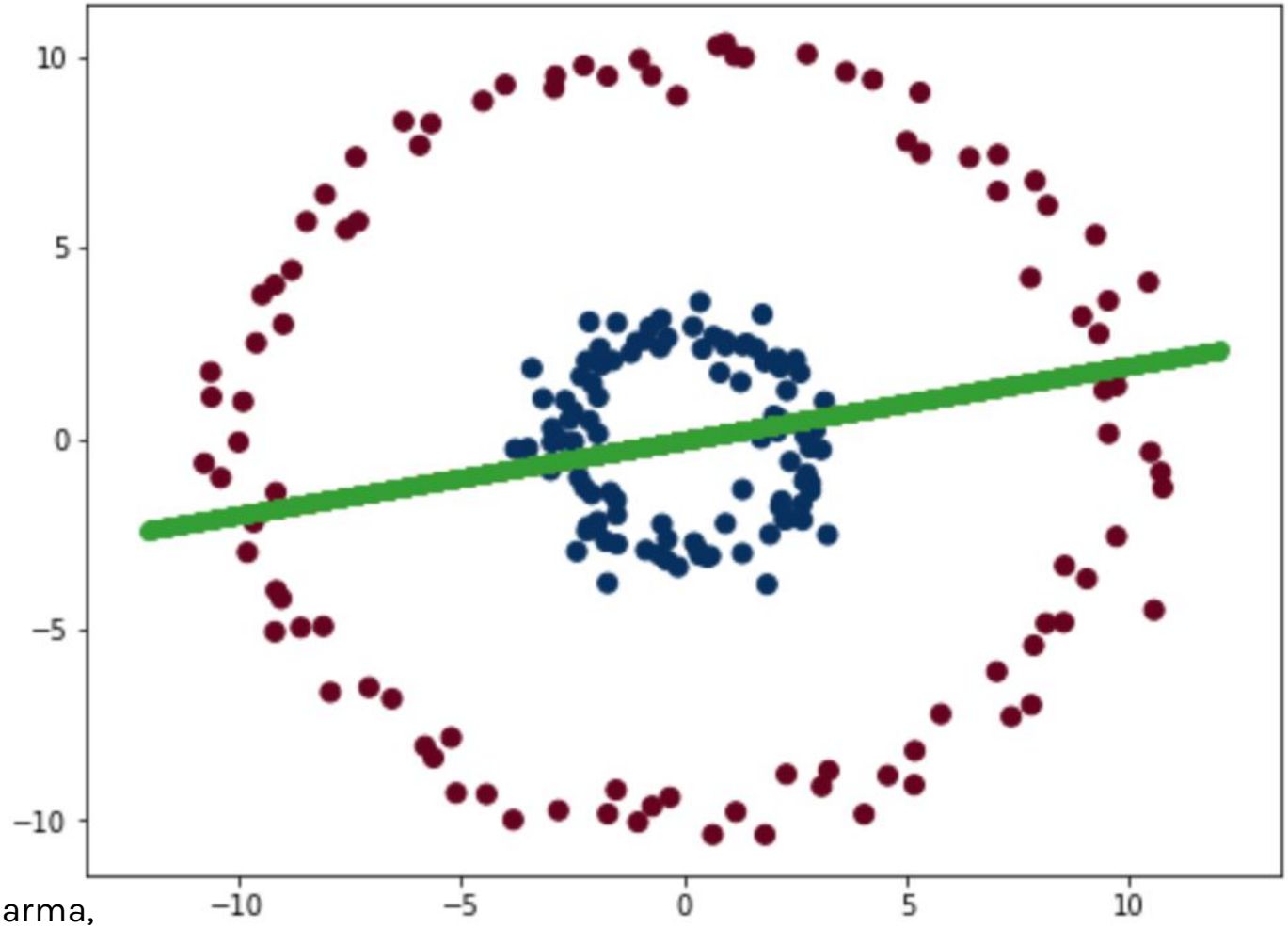




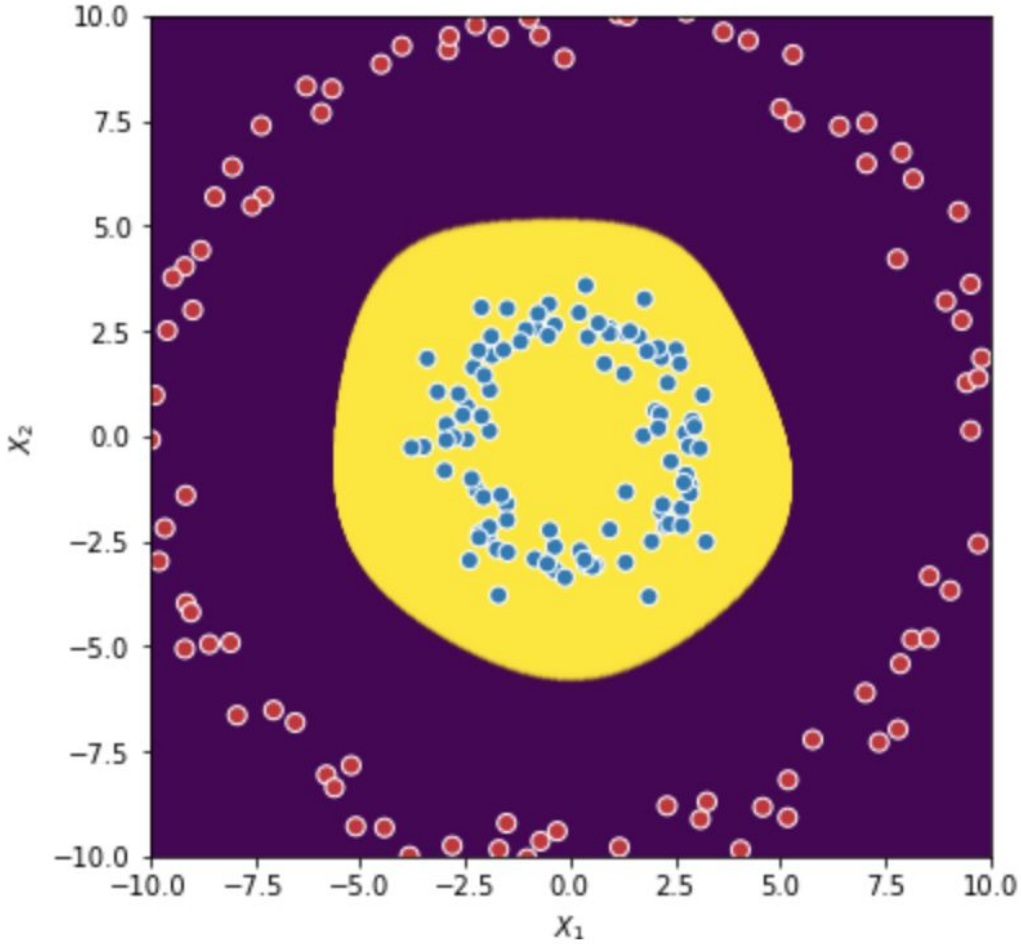


How to generate these figures?

```
20  data = io.loadmat("data1.mat")
21  X = data["X"]
22  y = data["y"]
23  steps, w = MyPerceptron(X, y, numpy.array([1.0,-1.0]))
24  plt.scatter(X[:,0],X[:,1], color=['red' if y[i][0]<0 else 'blue' for i in range(0, y.shape[0])])
25  x_axis = numpy.linspace(-1,1,1000)
26  plt.plot(x_axis, -1*x_axis*w[0]/w[1])
27  plt.xlim(-1,1)
28  plt.ylim(-1,1)
29  plt.show()
```



Credit: Shirkar Sarma, https://towardsdatascience.com/a-visual-introduction-to-neural-networks-68586b0b733b



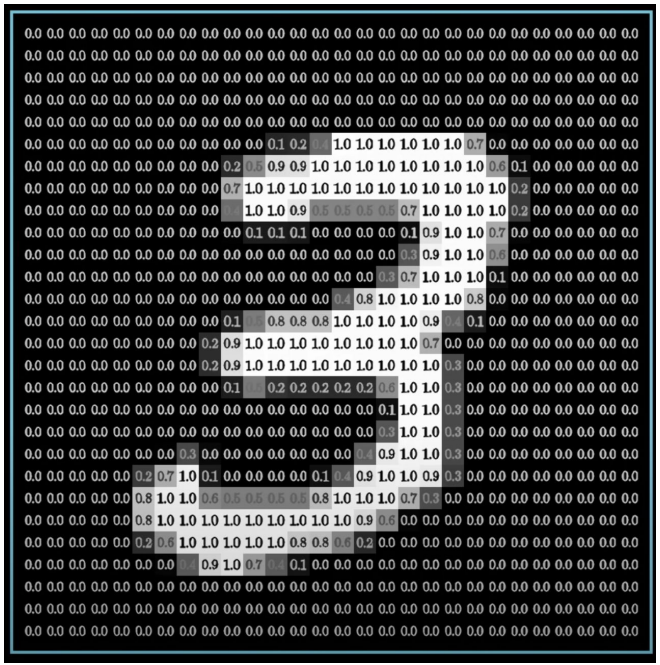
Credit: Shirkar Sarma, https://towardsdatascience.com/a-visual-introduction-to-neural-networks-68586b0b733b

Perceptrons vs. Neural Networks

Perceptrons	**Neural Networks**
Can’t work on data that isn’t linear separable	Can separate any arbitrary data
Corrects mistakes via vector addition	Has a way for correcting mistakes and improving
Starts with a random guess	Starts with a random guess
A single perceptron	Many, many perceptrons

Let's load and preprocess the data!

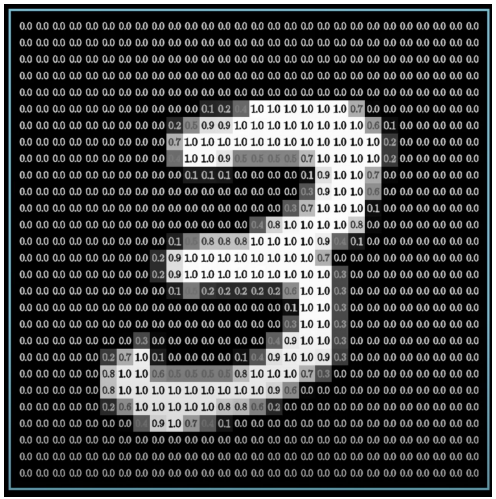
Let's start with Image Recognition



Credit: 3Blue1Brown

Neuron: thing that holds a number

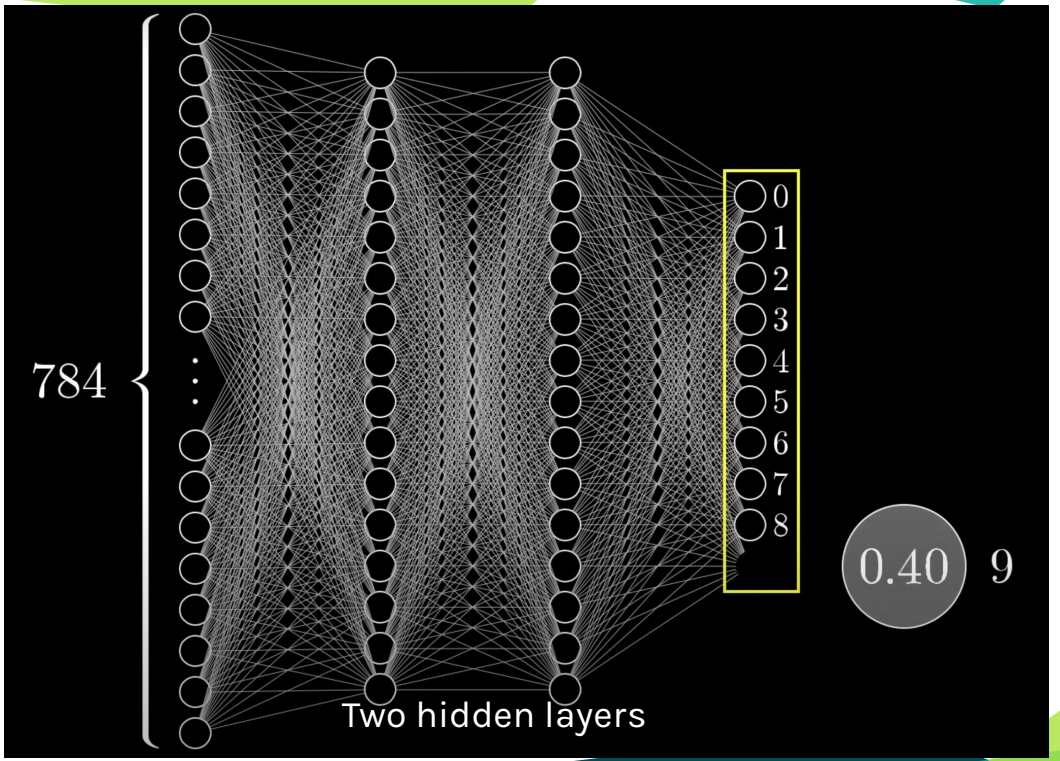
How do they work?



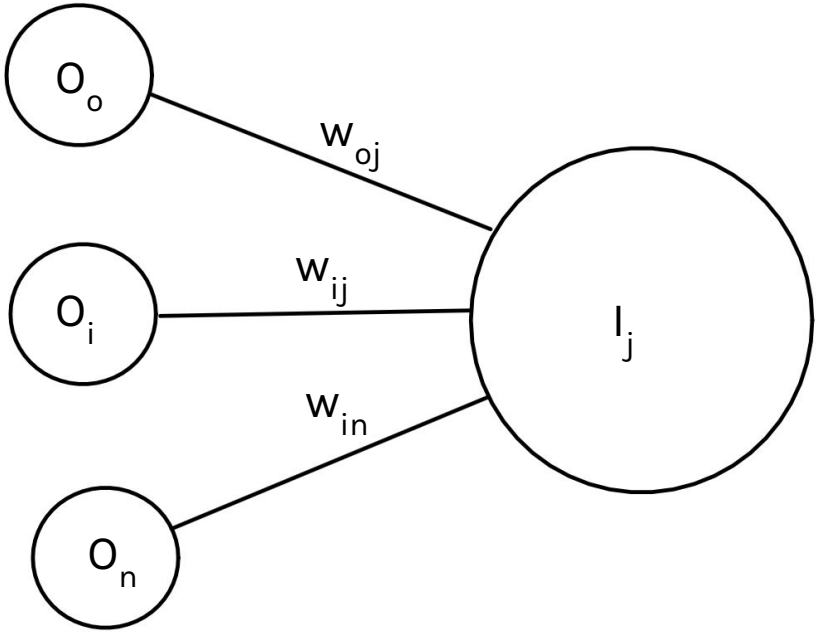
28 by 28 pixels
= 784 total
pixels

Credit: 3Blue1Brown

Activation layers

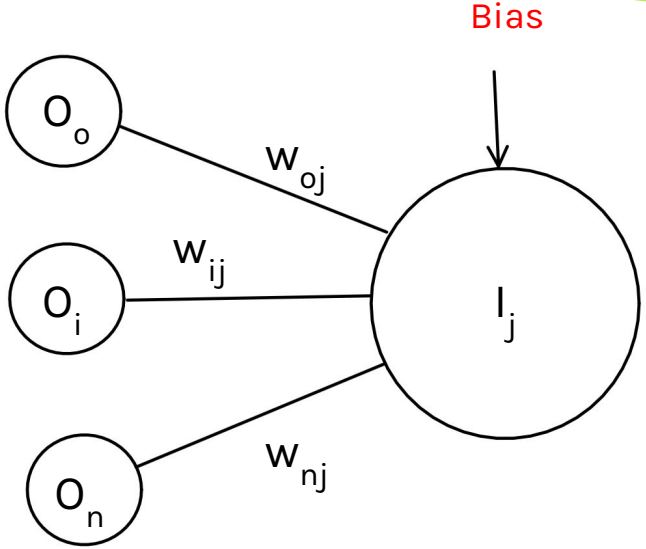


Each node is an equation



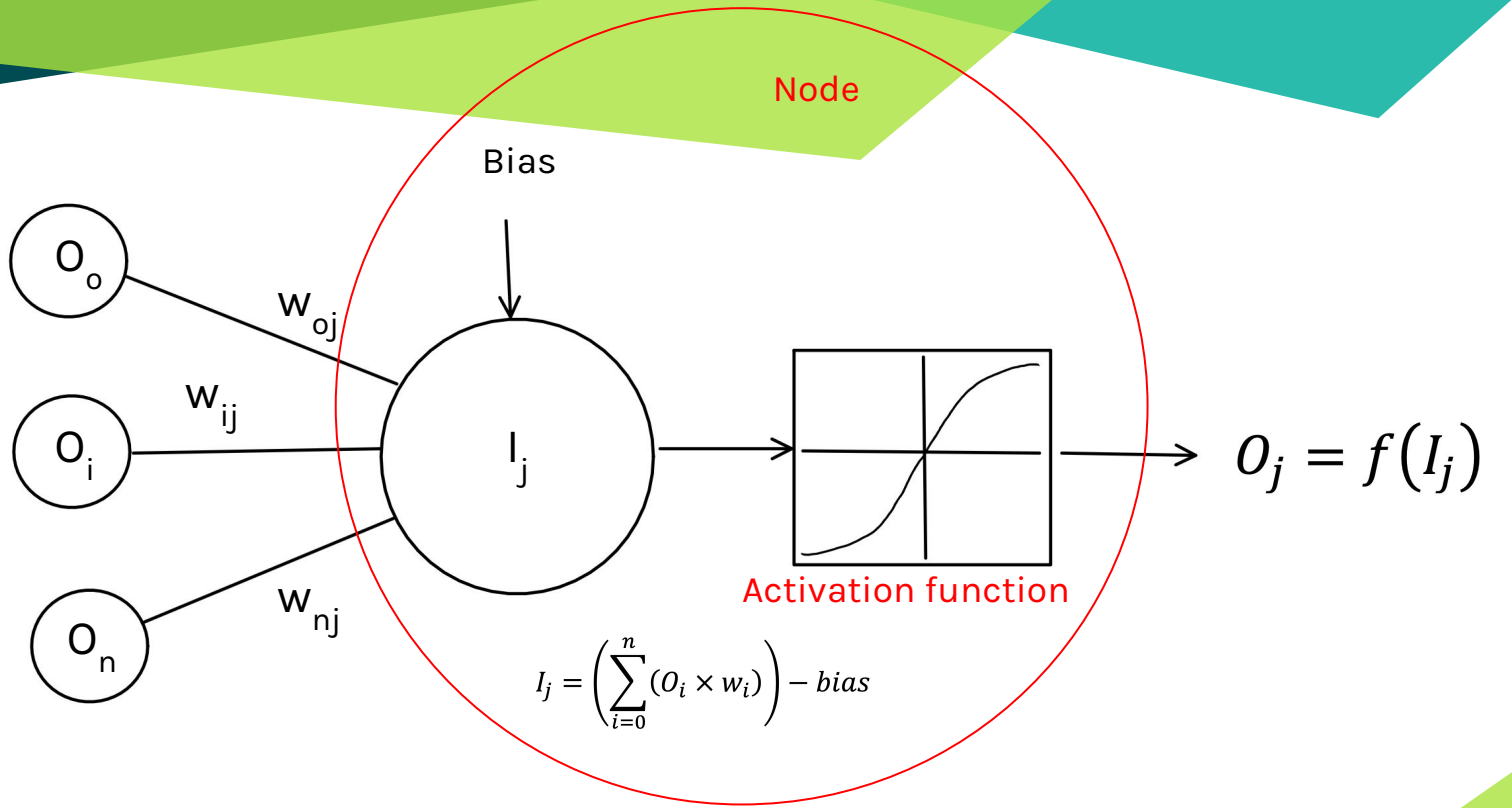
$$I_j = \sum_{i=0}^{n} (O_i \times w_i)$$

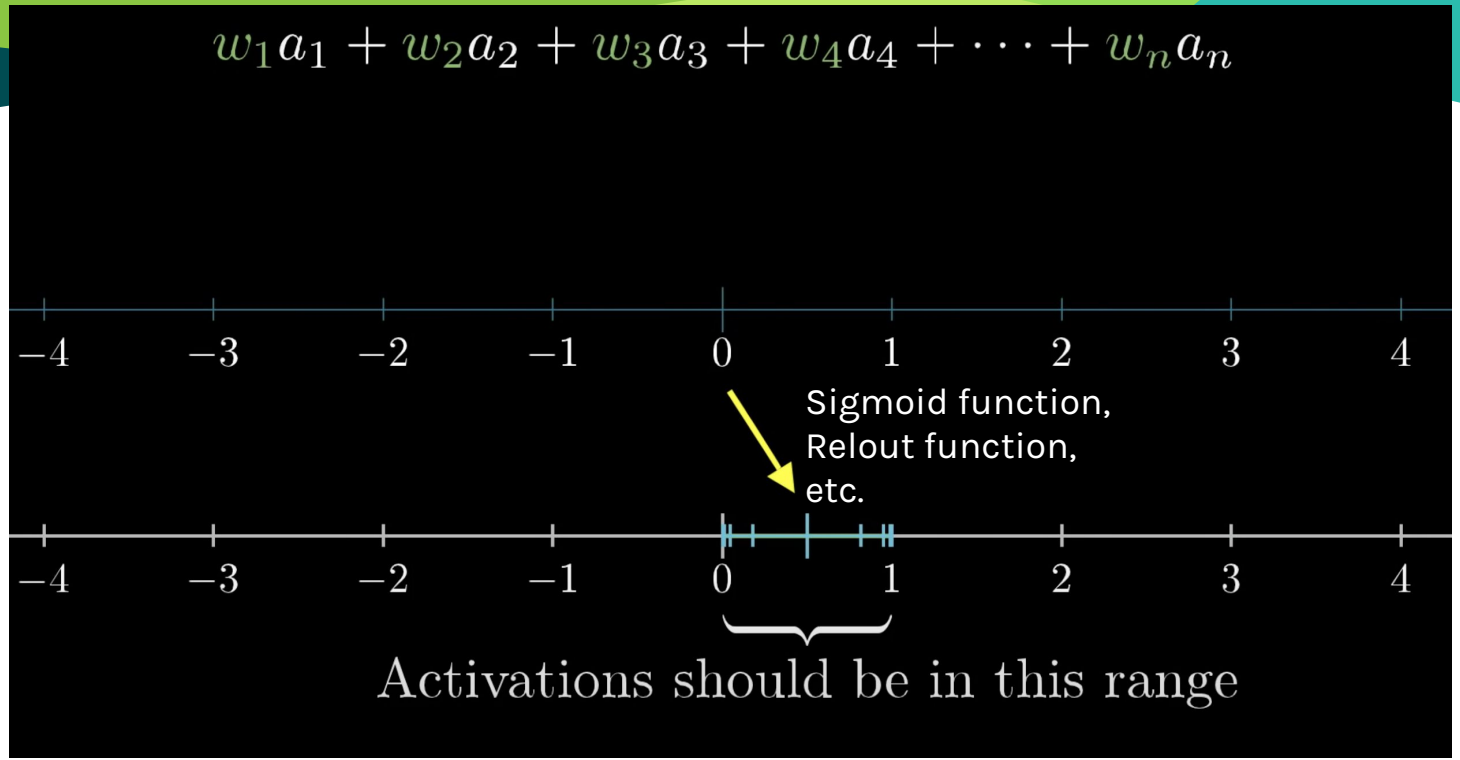
Credit: Karem Chokmani, Research Gate



$$I_j = \left(\sum_{i=0}^{n} (O_i \times w_i) \right) - bias$$

Credit: Intellipat

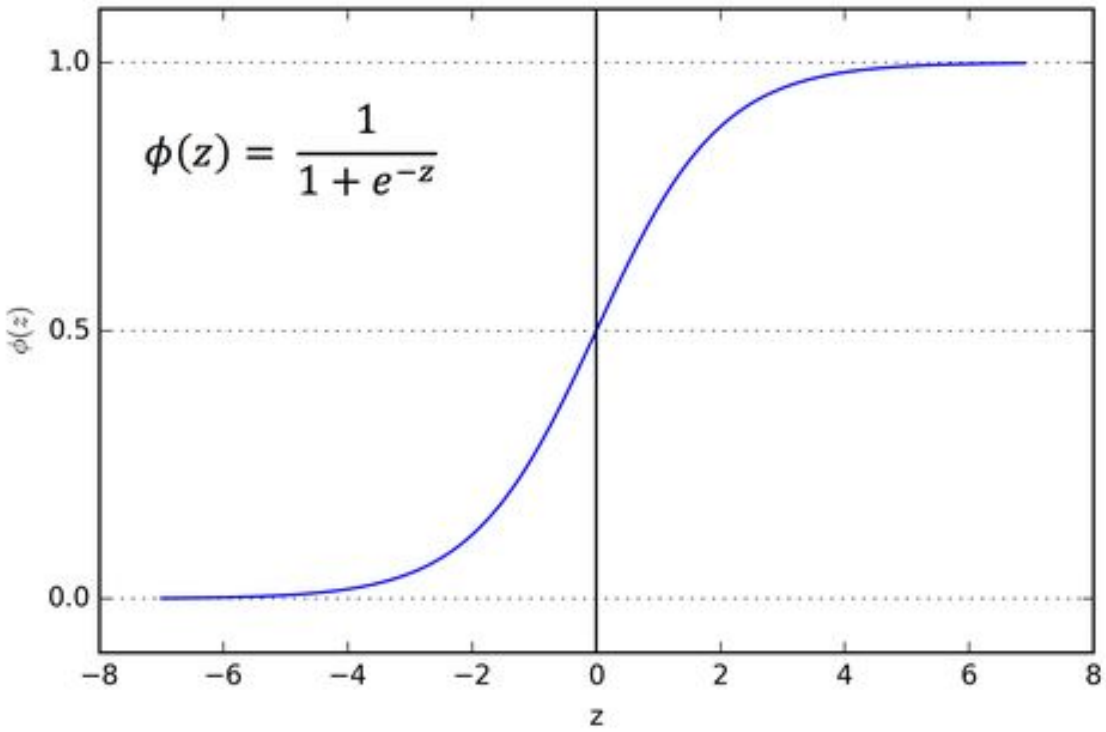

Node
Bias
O o
O i
O n
w oj
w ij
w nj
I j
Activation function
$O_j = f(I_j)$
$I_j = \left(\sum_{i=0}^{n} (O_i \times w_i)\right) - bias$



Credit: 3Blue1Brown

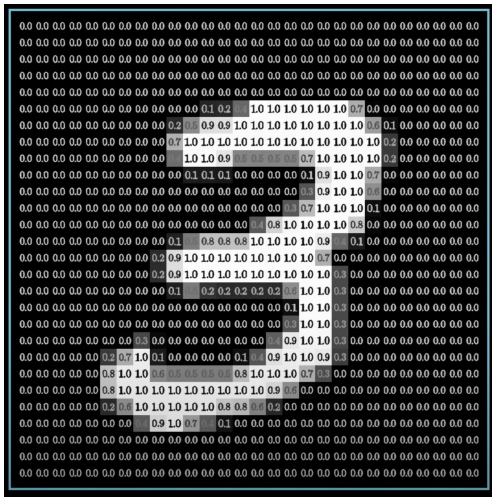
Sigmoid function does this.

Look familiar?

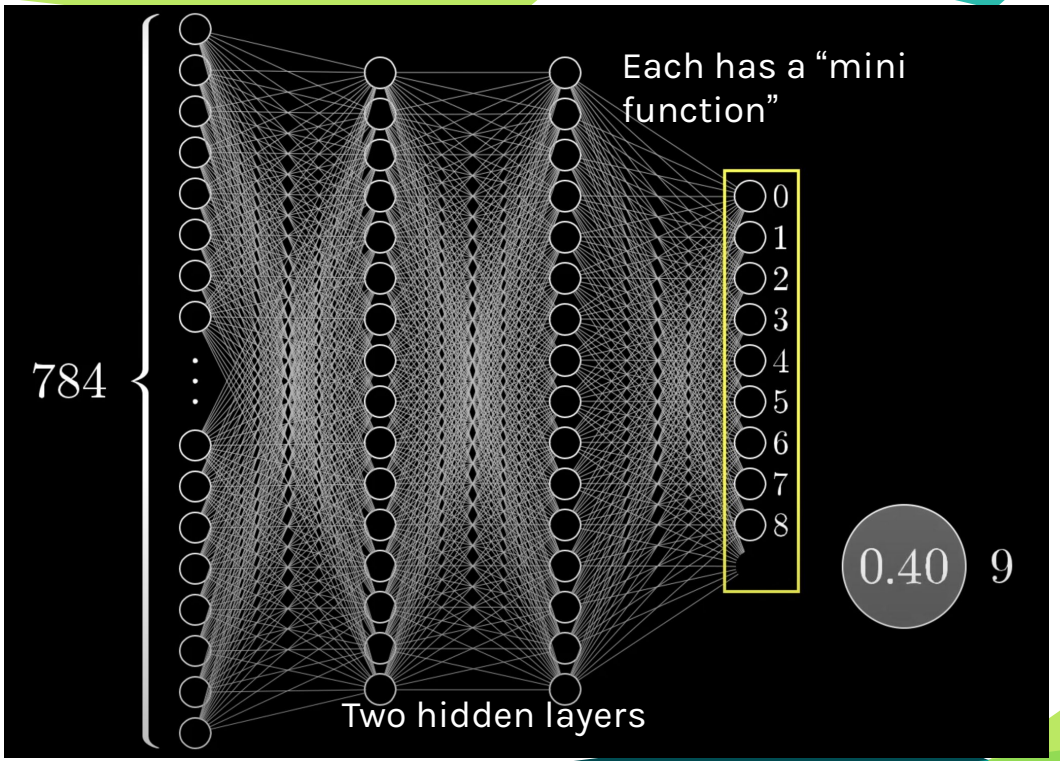


Credit: Sagar Sharma, Towards Data Science

Neural Networks are really powerful



28 by 28 pixels



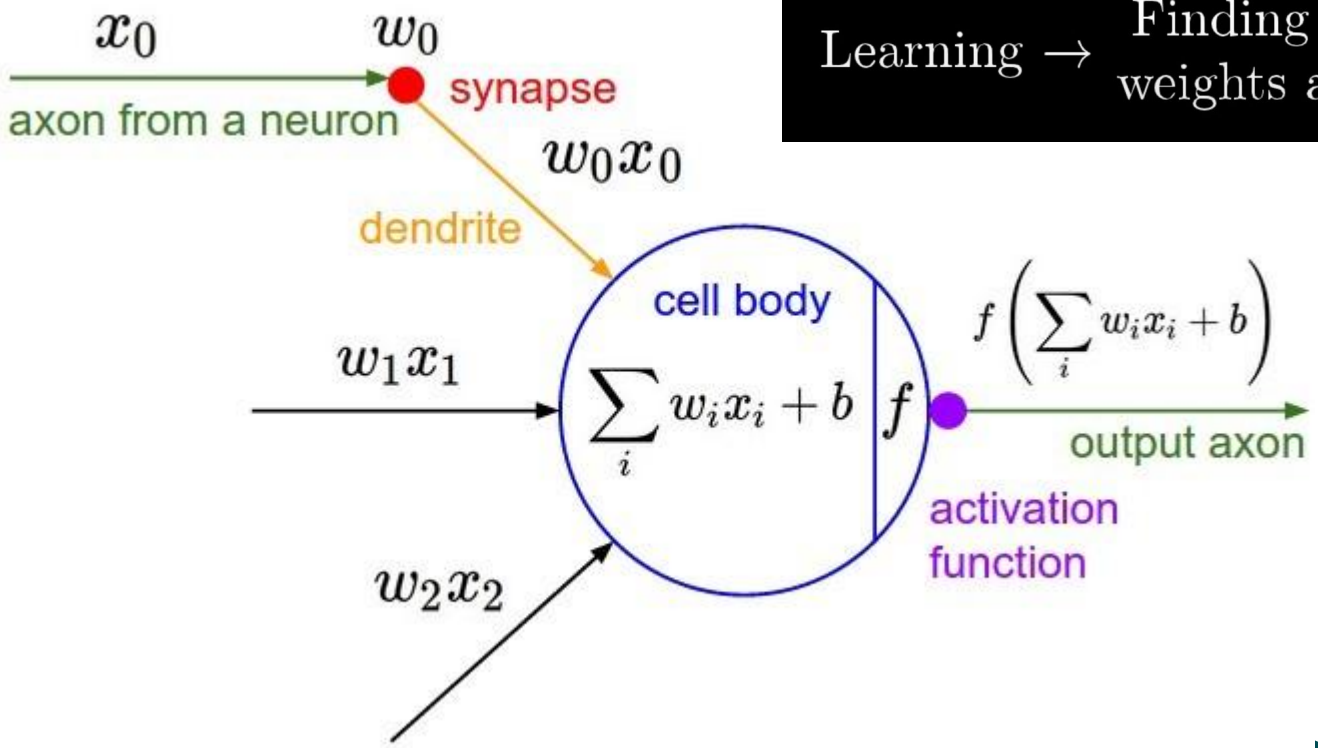
Credit: 3Blue1Brown

Let's Build Our Model!

How do we improve from mistakes? Why ‘Adam’ and ‘binary cross-entropy’?

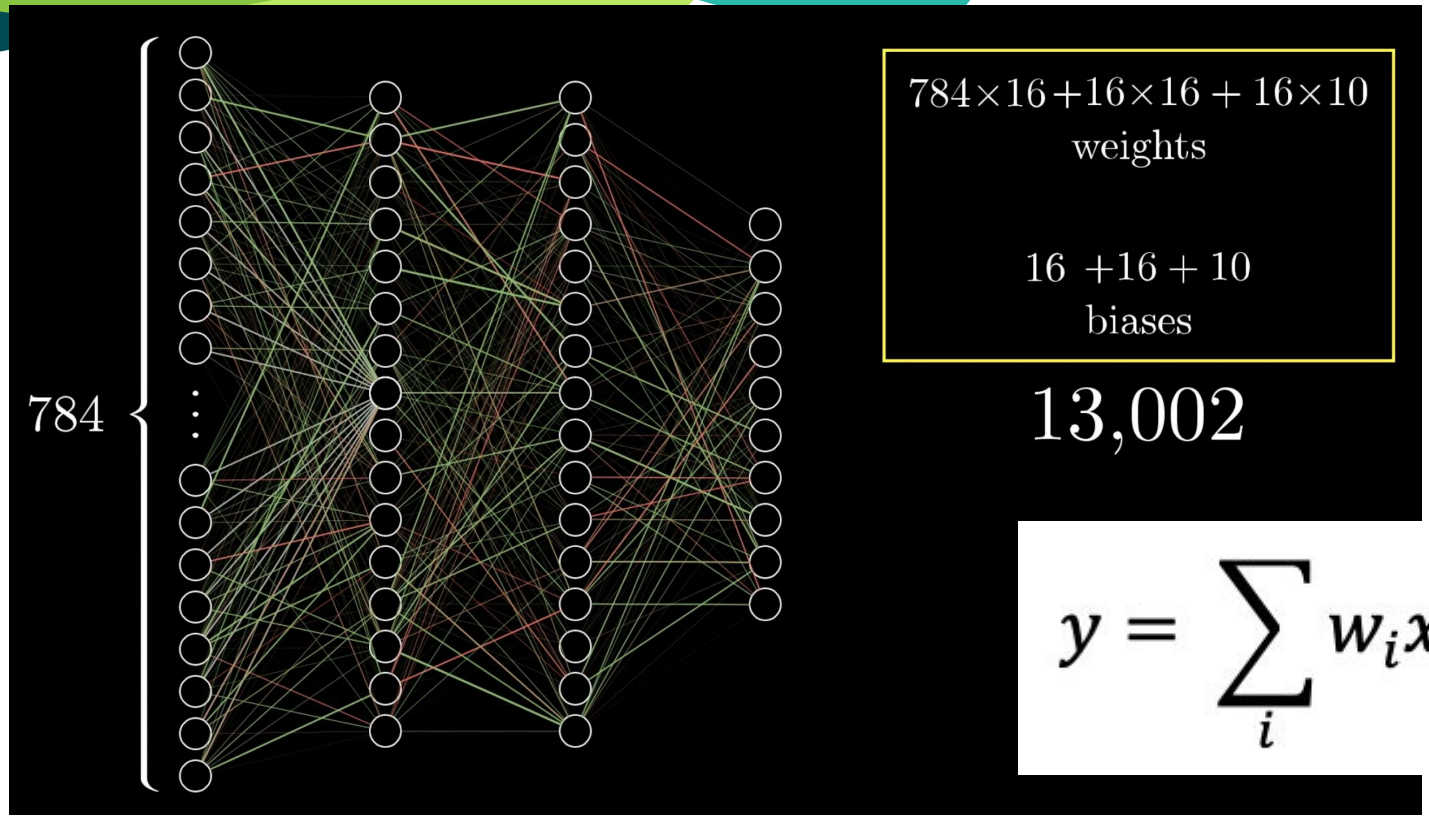
Bias

Learning $\rightarrow$ Finding the right weights and biases



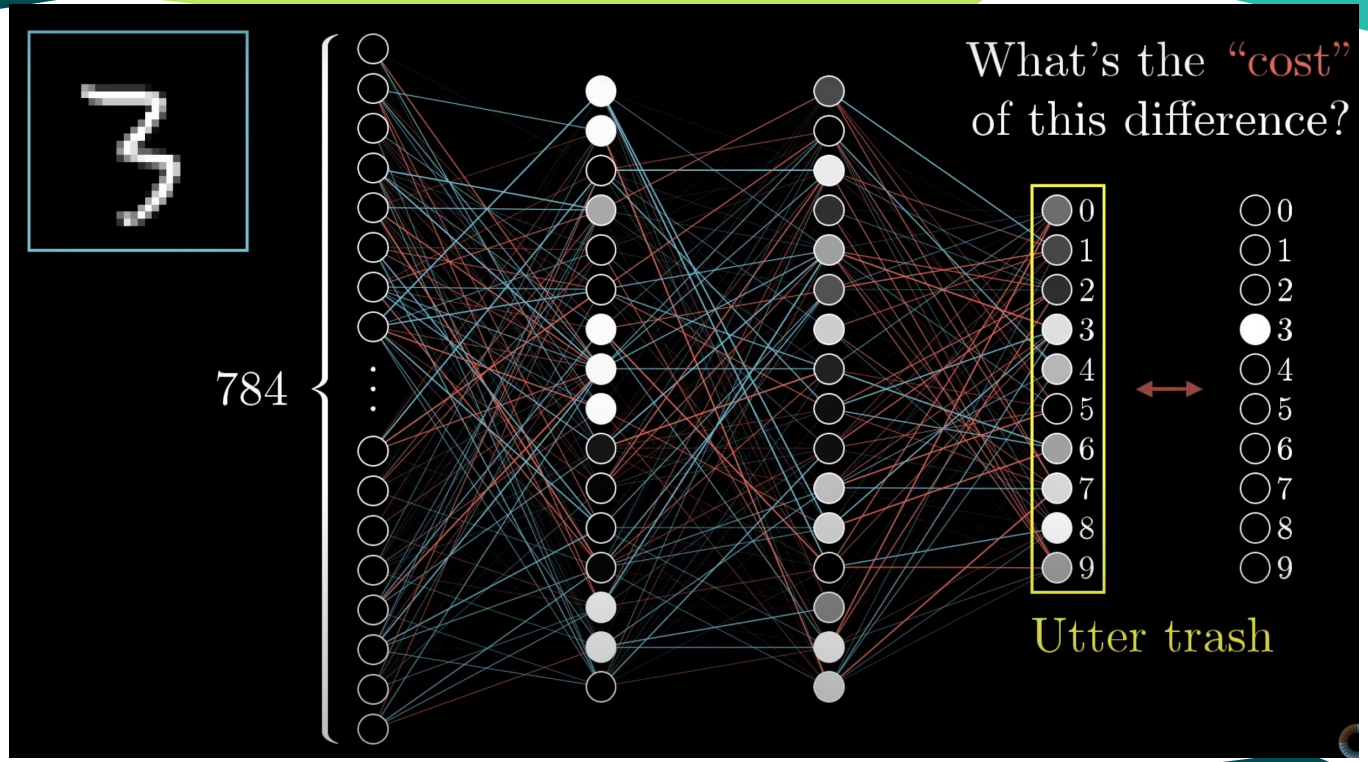
Credit: Cs231n, Stanford

Initially, the weights are all random...



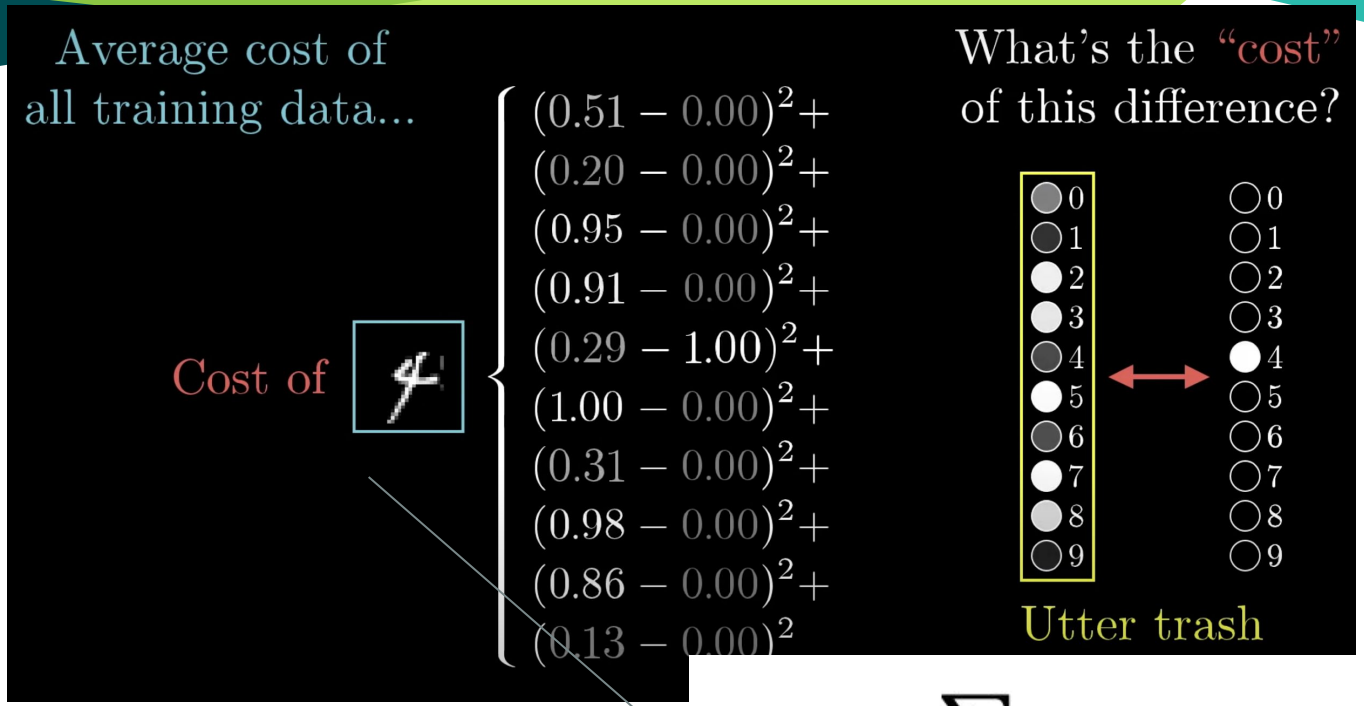
Credit: 3Blue1Brown

Where learning comes in



Credit: 3Blue1Brown

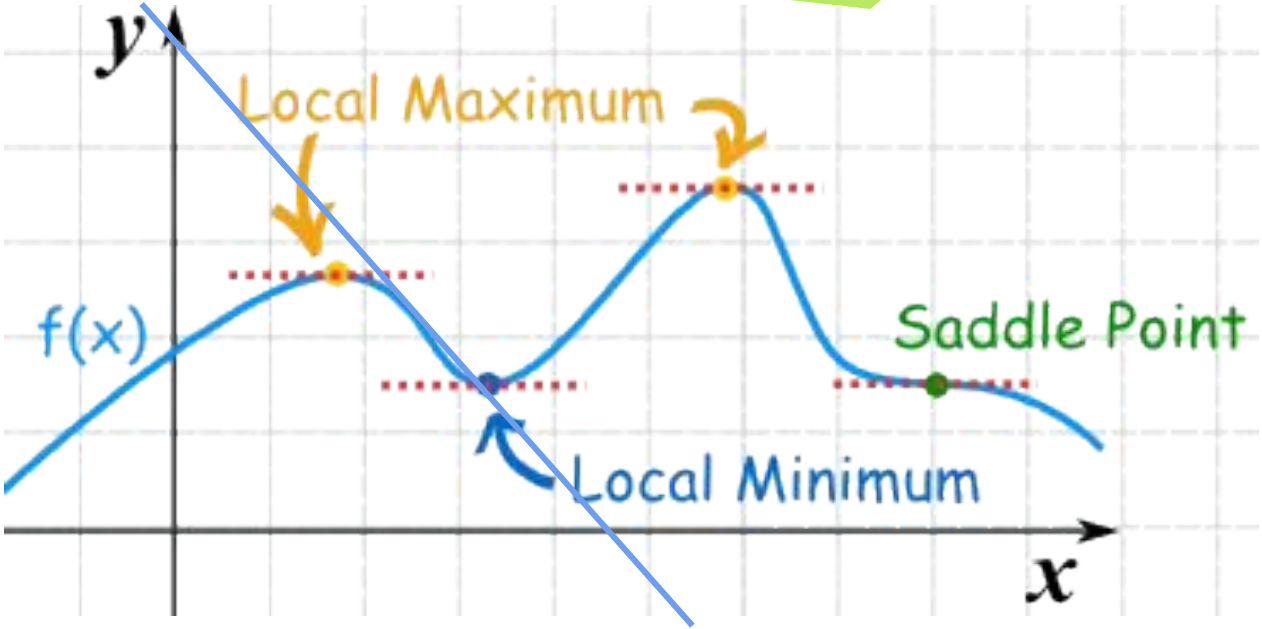
Where learning comes in



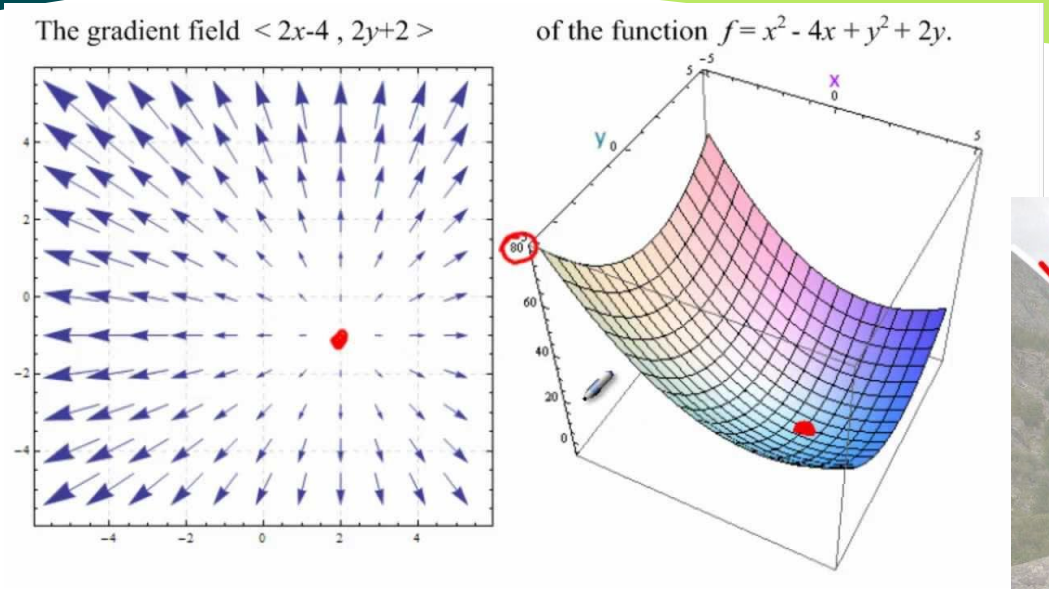
$$cost = \sum_{node} (actual_{node} - expected_{node})^2$$

Credit: 3Blue1Brown

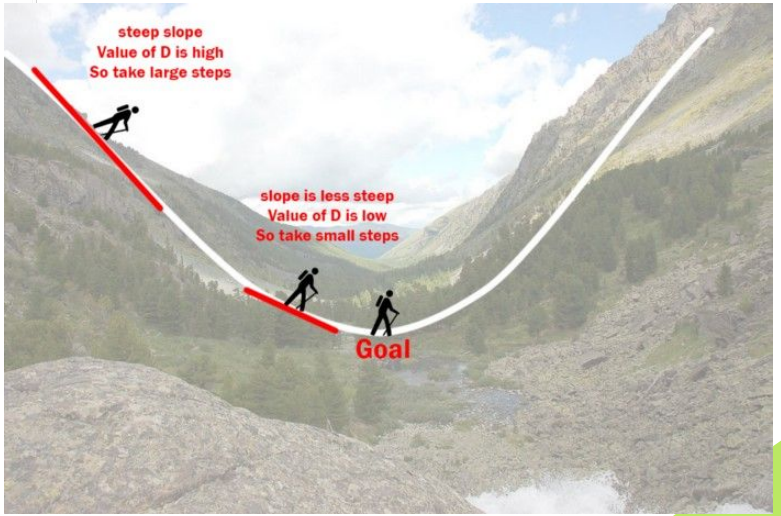
Where the learning comes in



Find the minimum cost

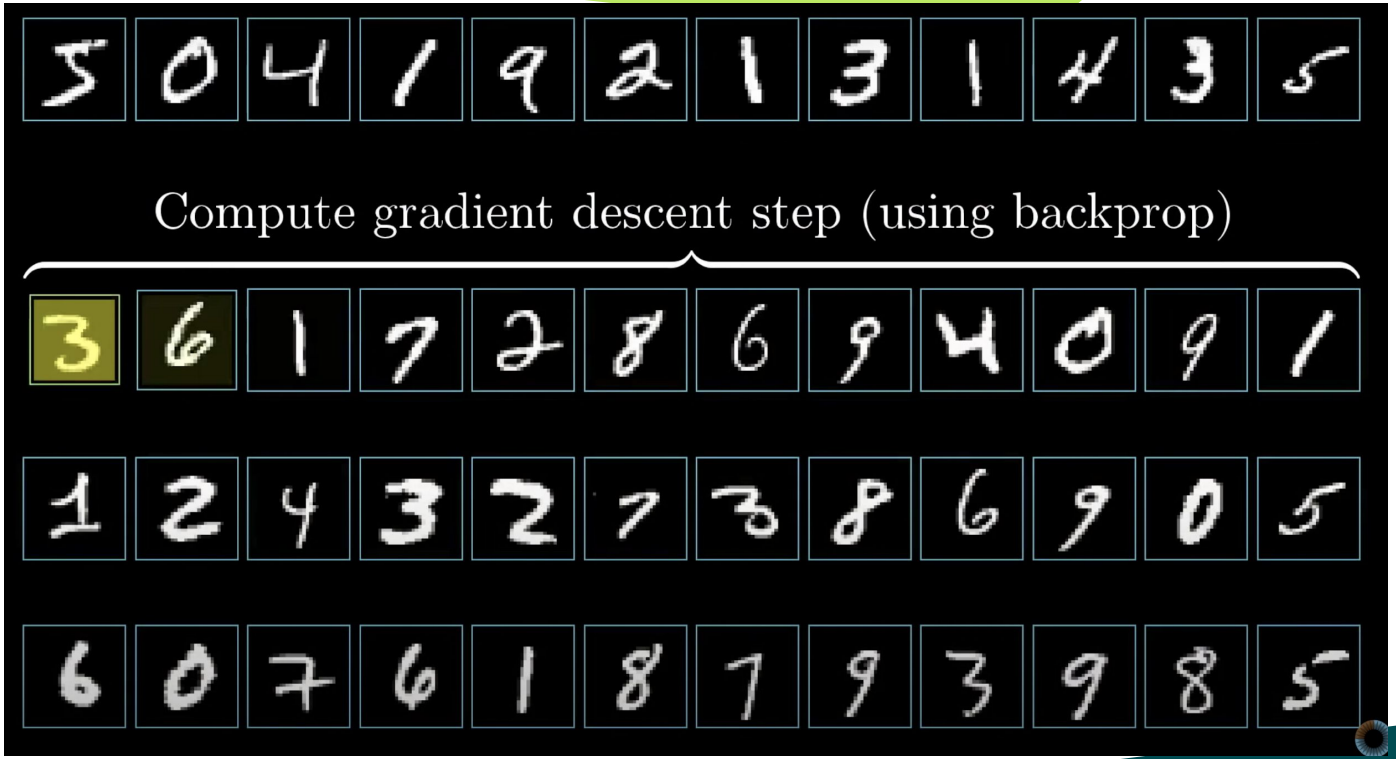


1. Find the Gradient $\nabla C(x,y)$
2. Move in that direction by $-\nabla C(x,y)$



Credit: Linda Fahlberg-Stojanovska

Stochastic Gradient Descent



Do in mini batches

What is an epoch?

Credit: 3Blue1Brown

Training Our Model

Challenge:
improve this NN!

Try changing the activation function!

https://keras.io/api/layers/activations/

Usage of activations

Activations can either be used through an `Activation` layer, or through the `activation` argument supported by all forward layers:

```
model.add(layers.Dense(64, activation=activations.relu))
```

This is equivalent to:

```
from tensorflow.keras import layers
from tensorflow.keras import activations

model.add(layers.Dense(64))
model.add(layers.Activation(activations.relu))
```

Sigmoid, relu, tanh?
Your choice!

Try changing the architecture and parameters!

What happens when I change the optimizer?

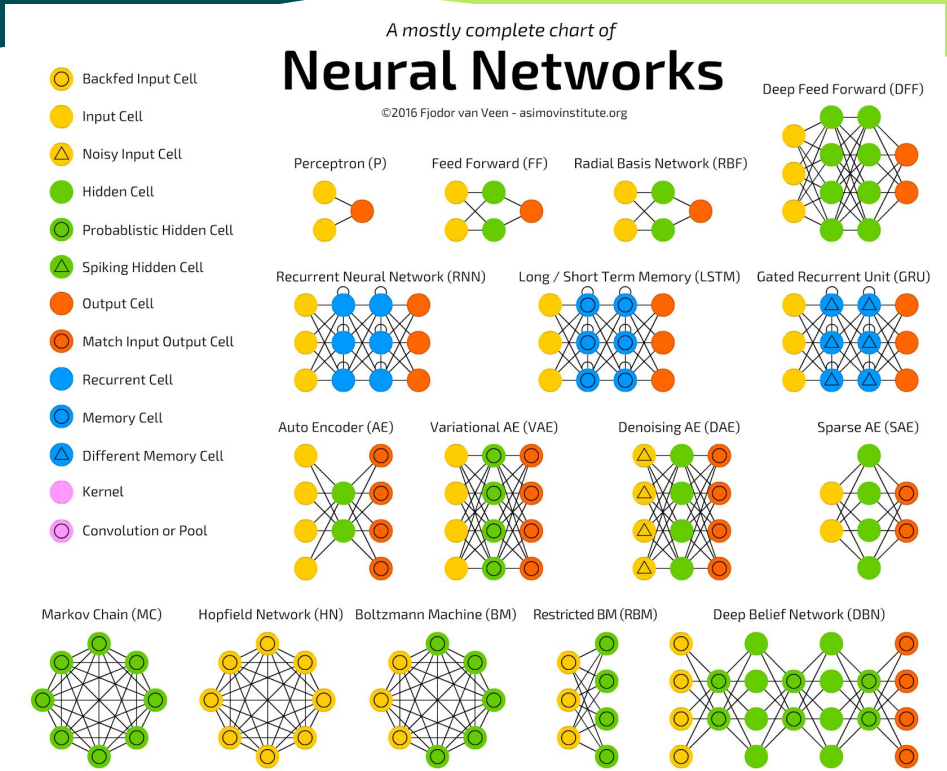
How do I know if I am overfitting?

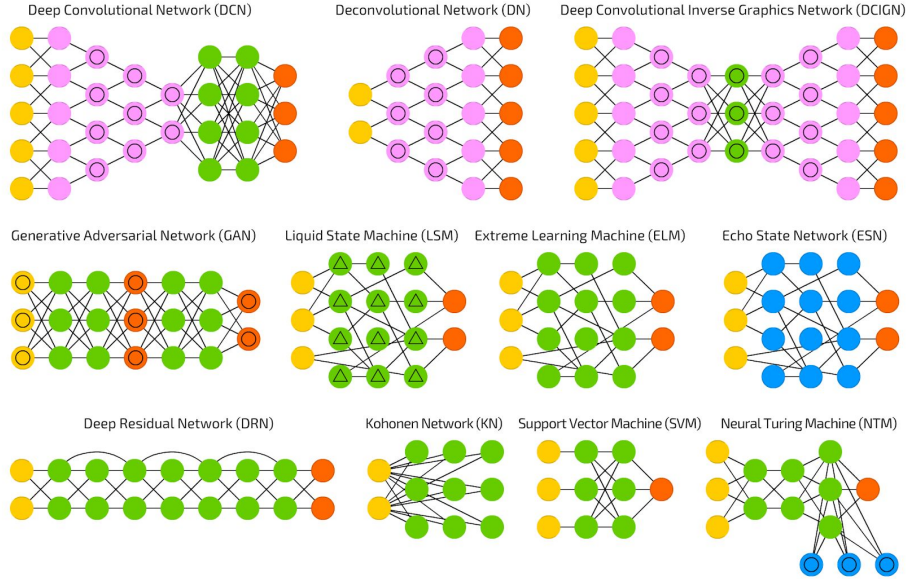
What happens when I add a dropout layer?

Will a deep, narrow network perform better than a shallow, wide network?

Context

Types of Neural Networks...





More ML

- Machine Learning for Absolute Beginners: A Plain English Introduction
  - For the completely uninitiated
- An Introduction to Statistical Learning
  - For those without much code/stat/math background
  - Uses the R language but has been translated to python
    - https://github.com/emredjan/ISL-python
- Andrew Ng's Coursera Course
- Any of the other resource. Keywords: python, data science, machine learning
  - They're all good, you just need to pick one and stick with it