

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

The screenshot shows a GitHub repository page for 'SASE-Labs-2021/coding-ml-workshops'. The repository has 0 stars, 1 fork, and 0 issues. The 'Code' tab is selected. The repository contains one branch ('master') and no tags. The contents of the master branch include a file named 'nathanielbd initial' and two folders: 'week_1' and 'README.md'. A context menu is open over the 'Code' button, showing options like 'Clone', 'Open with GitHub Desktop', and 'Download ZIP'. The URL of the repository is https://github.com/SASE-Labs-2021/coding-ml-workshops.

Search or jump to... / Pull requests Issues Marketplace Explore

SASE-Labs-2021 / coding-ml-workshops <https://github.com/SASE-Labs-2021/coding-ml-workshops>

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

Clone

HTTPS SSH GitHub CLI

https://github.com/SASE-Labs-2021/coding-ml-workshops

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

Slides and code for a workshop series in Fall 2020 with EWH and Charosa

www.facebook.com/events/33964245...

Readme

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

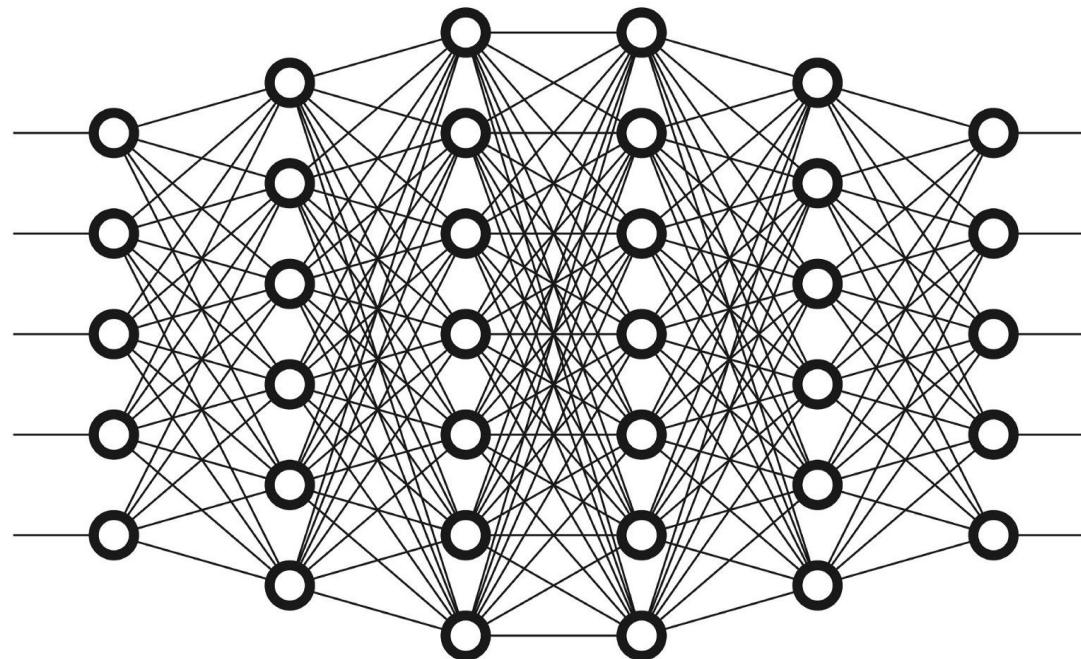
Coding and Machine Learning work...

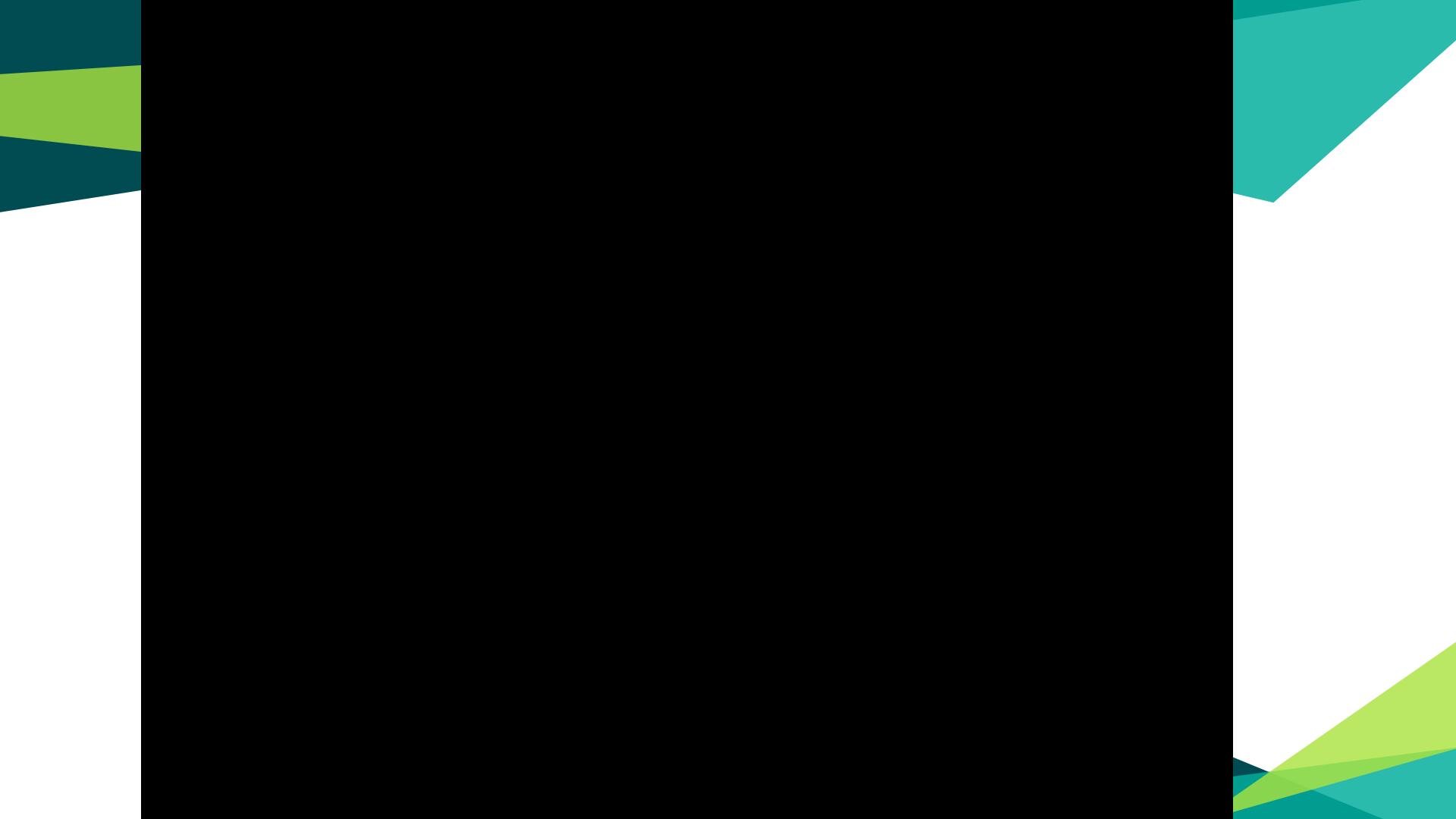
These workshops were presented October 14, 21, and 28 by [Engineering World Health](#) (Nitali Arora), [Charosa](#) (Kevin Meng-Lin), and Society of Asian Scientists and Engineers (Nathaniel Budijono) at the University of Minnesota.

Neural Networks

Charosa & SASE Labs & EWH

Neural Networks? Artificial Neurons?

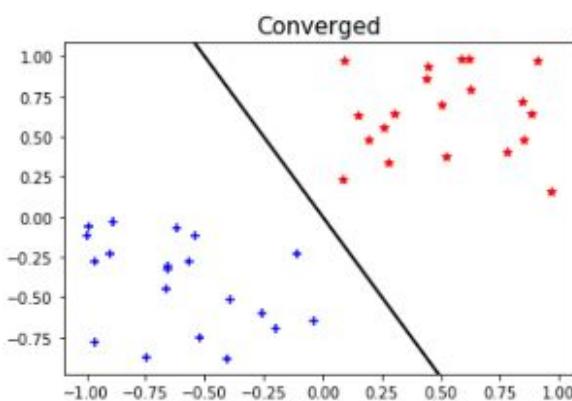
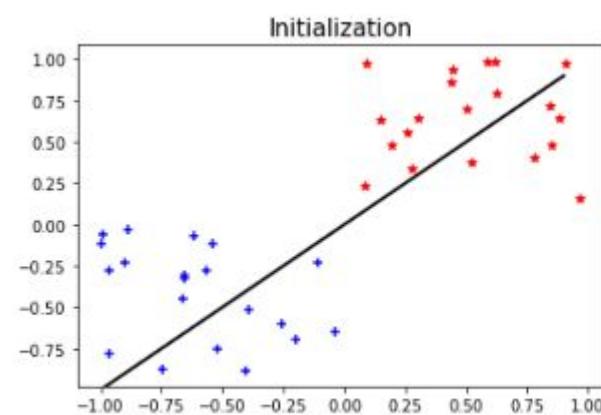




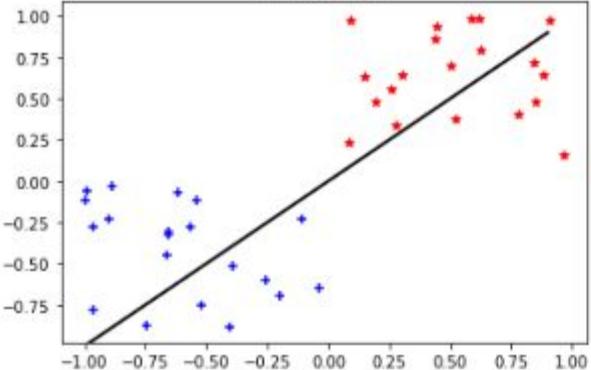
```

1 import numpy
2 import scipy.io as io
3 import matplotlib.pyplot as plt
4
5 def MyPerceptron(X, y, w_0):
6     steps = 0
7     mistakes = 1
8     w = w_0      W is the dividing yellow line
9     n = y.shape[0]
10    while mistakes > 0:
11        mistakes = 0
12        for i in range(0,n):
13            steps += 1          The math to calculate
14            if numpy.inner(w,X[i])*y[i][0] < 0:    which side of the divider
15                mistakes += 1    the point is on
16                w += y[i][0]*X[i]  Vector addition
17
18    return steps, w

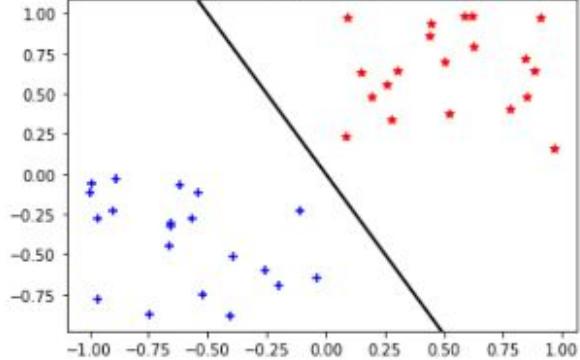
```



Initialization

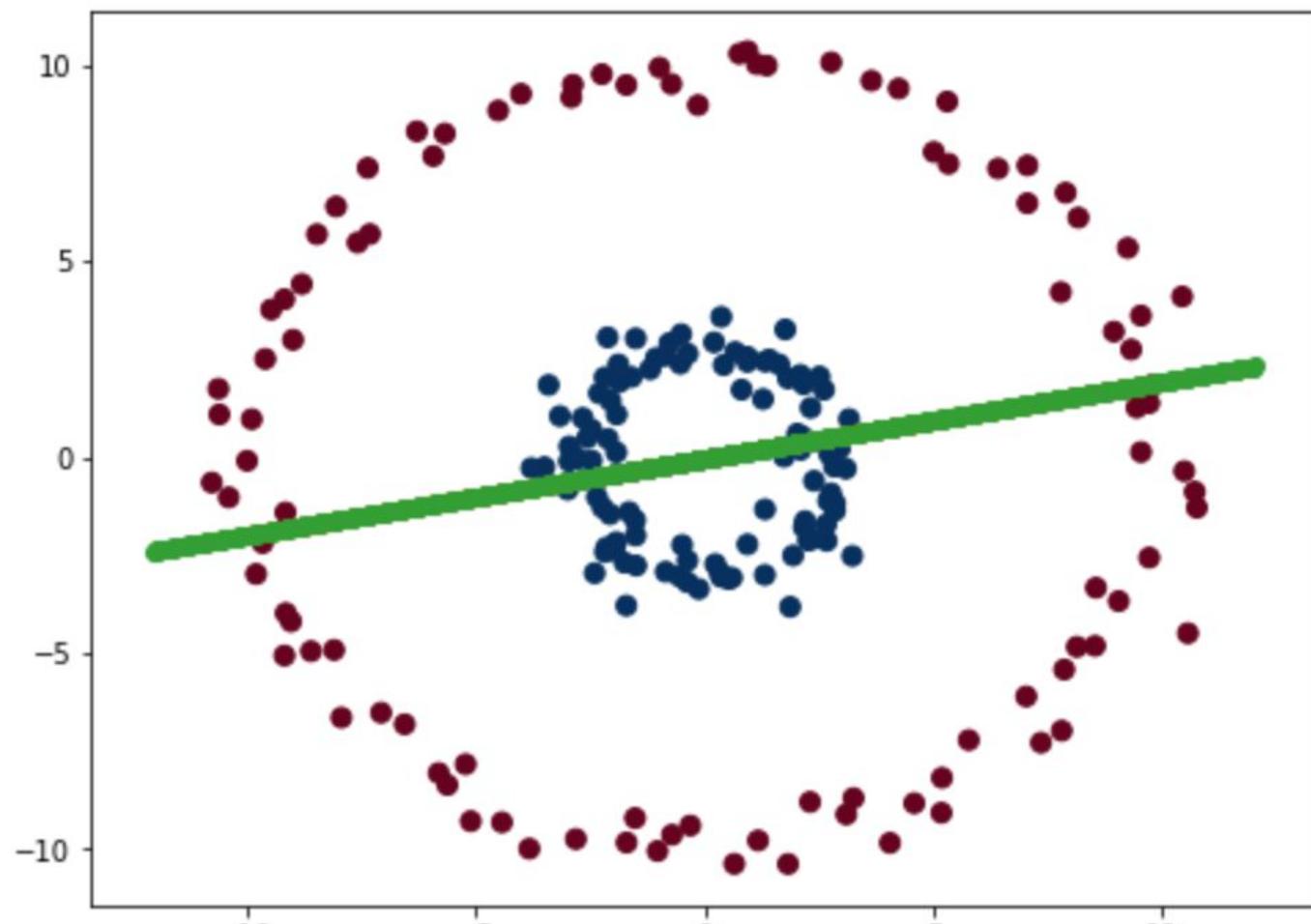


Converged

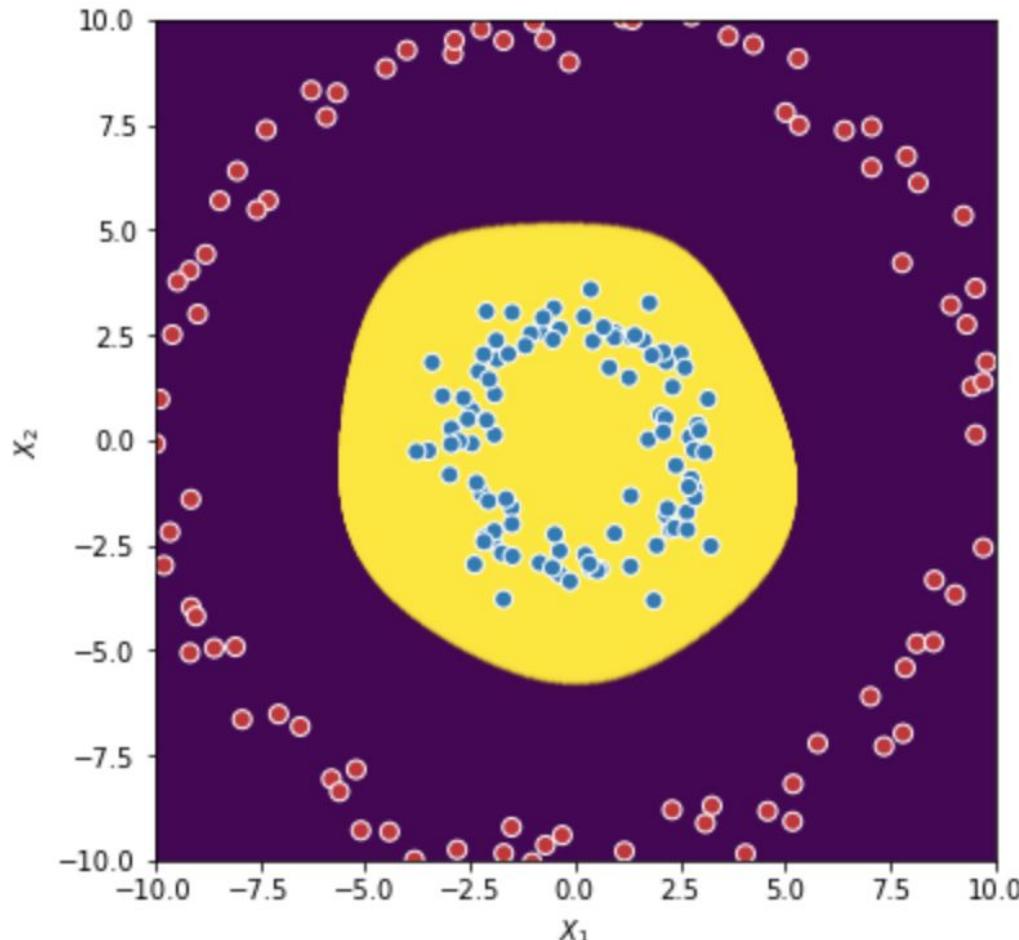


How to generate these figures?

```
0      data = io.loadmat("data1.mat")
1      X = data["X"]
2      y = data["y"]
3      steps, w = MyPerceptron(X, y, numpy.array([1.0,-1.0]))
4      plt.scatter(X[:,0],X[:,1], color=['red' if y[i][0]<0 else 'blue' for i in range(0, y.shape[0])])
5      x_axis = numpy.linspace(-1,1,1000)
6      plt.plot(x_axis, -1*x_axis*w[0]/w[1])
7      plt.xlim(-1,1)
8      plt.ylim(-1,1)
9      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

Neuron: thing that holds a number

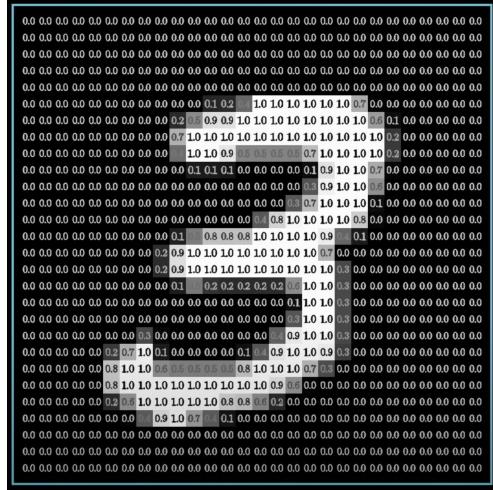
Each pixel



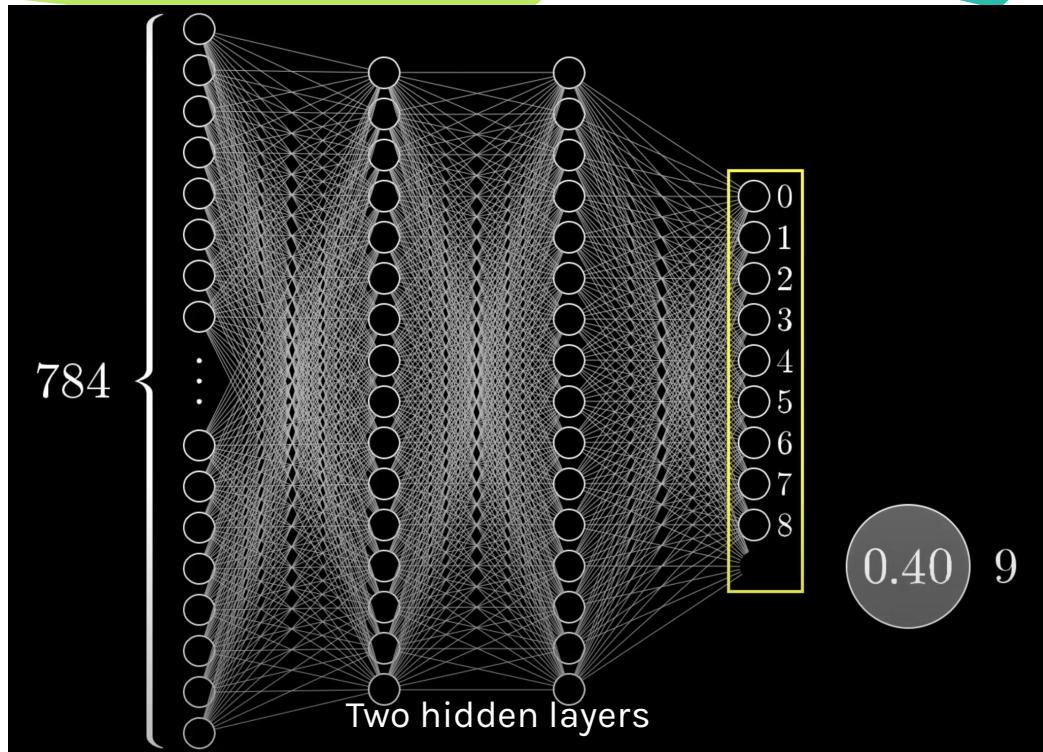
“Activation”

How do they work?

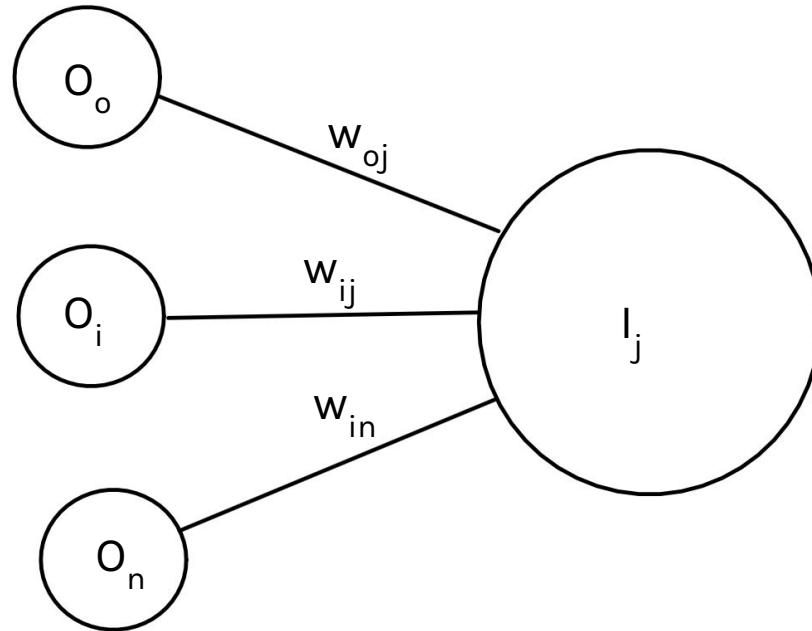
Activation layers



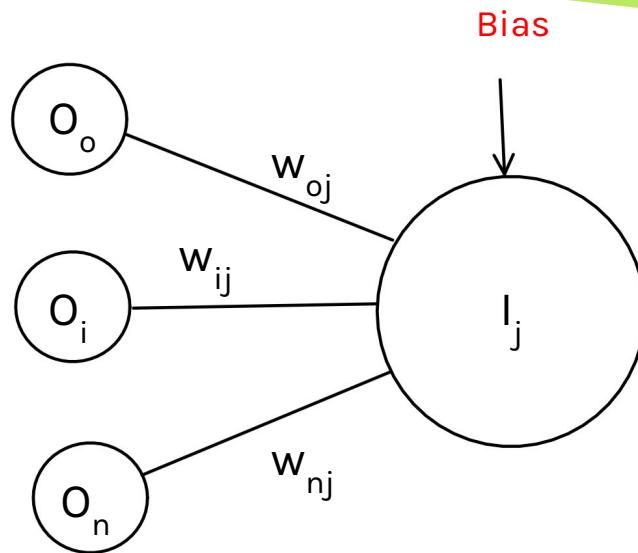
28 by 28 pixels
= 784 total
pixels



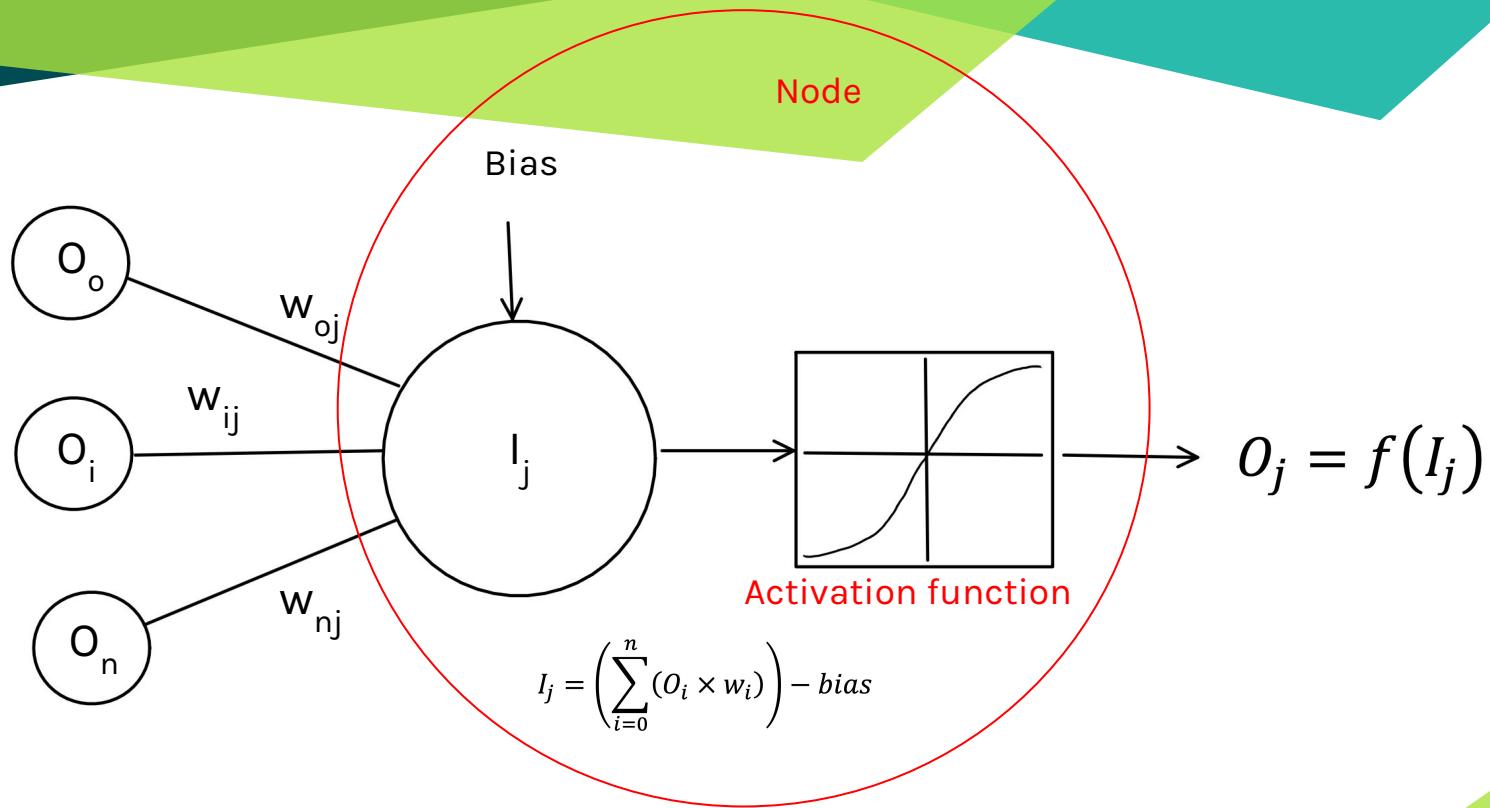
Each node is an equation



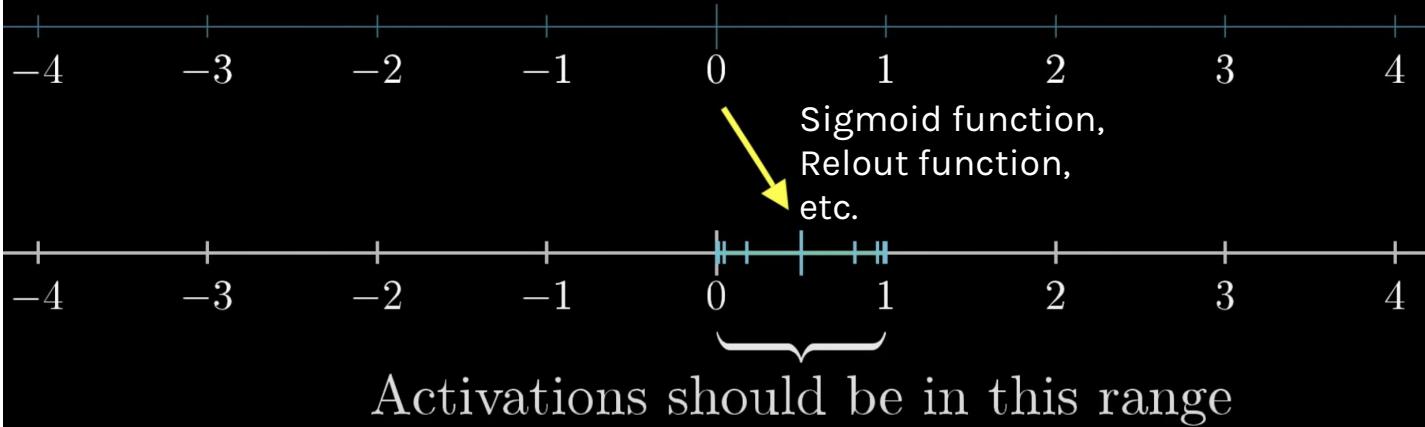
$$I_j = \sum_{i=0}^n (O_i \times w_i)$$



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

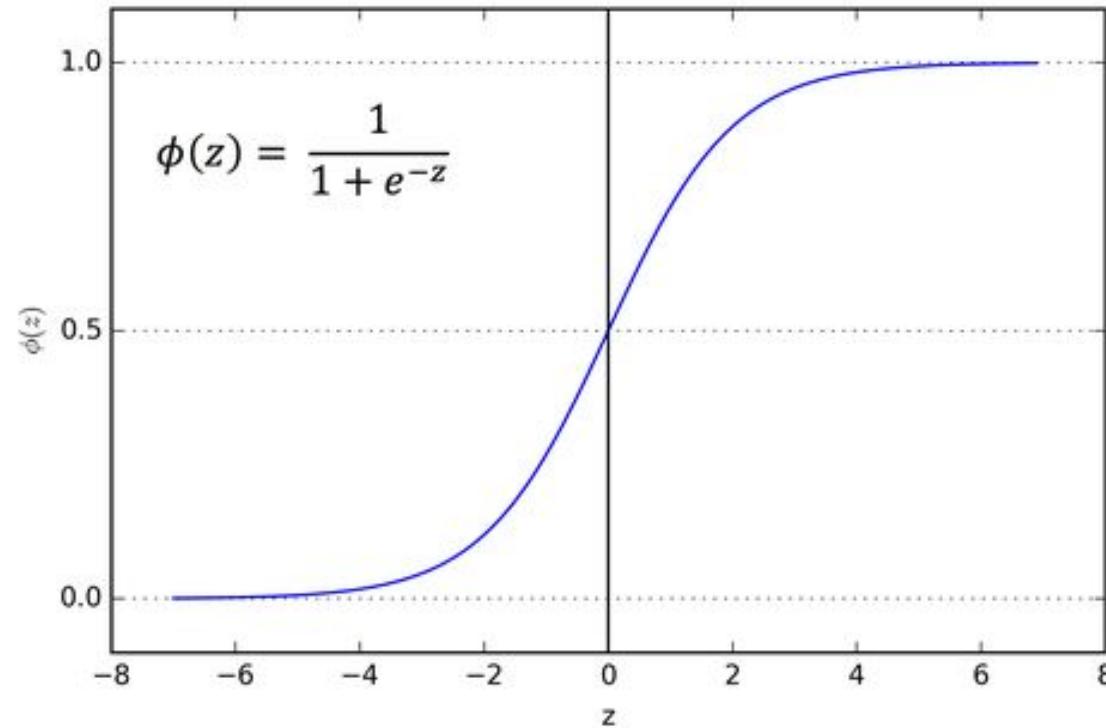


$$w_1a_1 + w_2a_2 + w_3a_3 + w_4a_4 + \cdots + w_na_n$$



Sigmoid function does this.

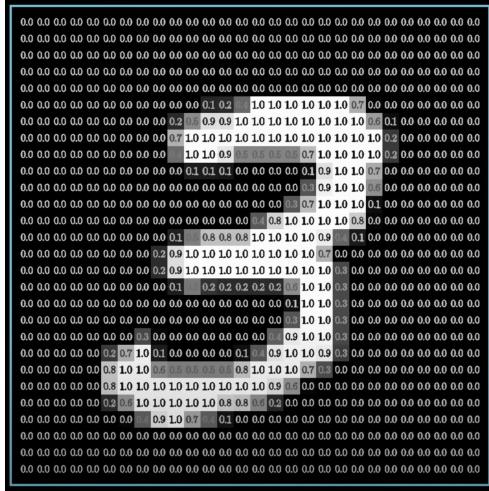
Look familiar?



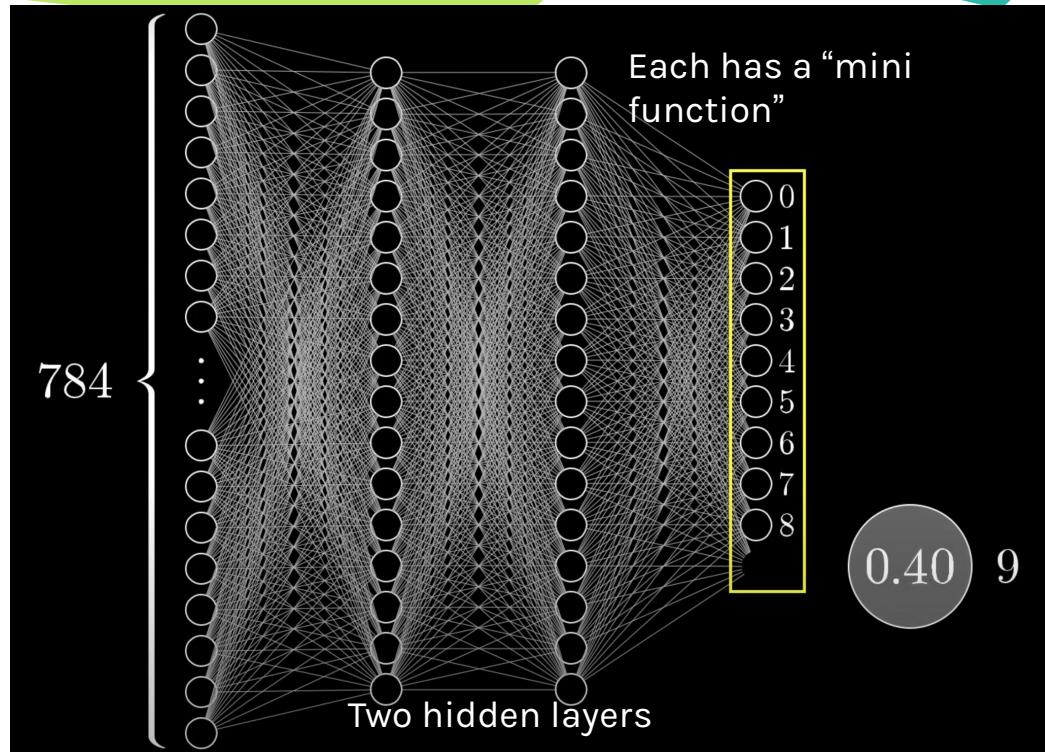
Credit: Sagar Sharma, Towards Data Science

Neural Networks are really powerful

Activation layers



28 by 28 pixels

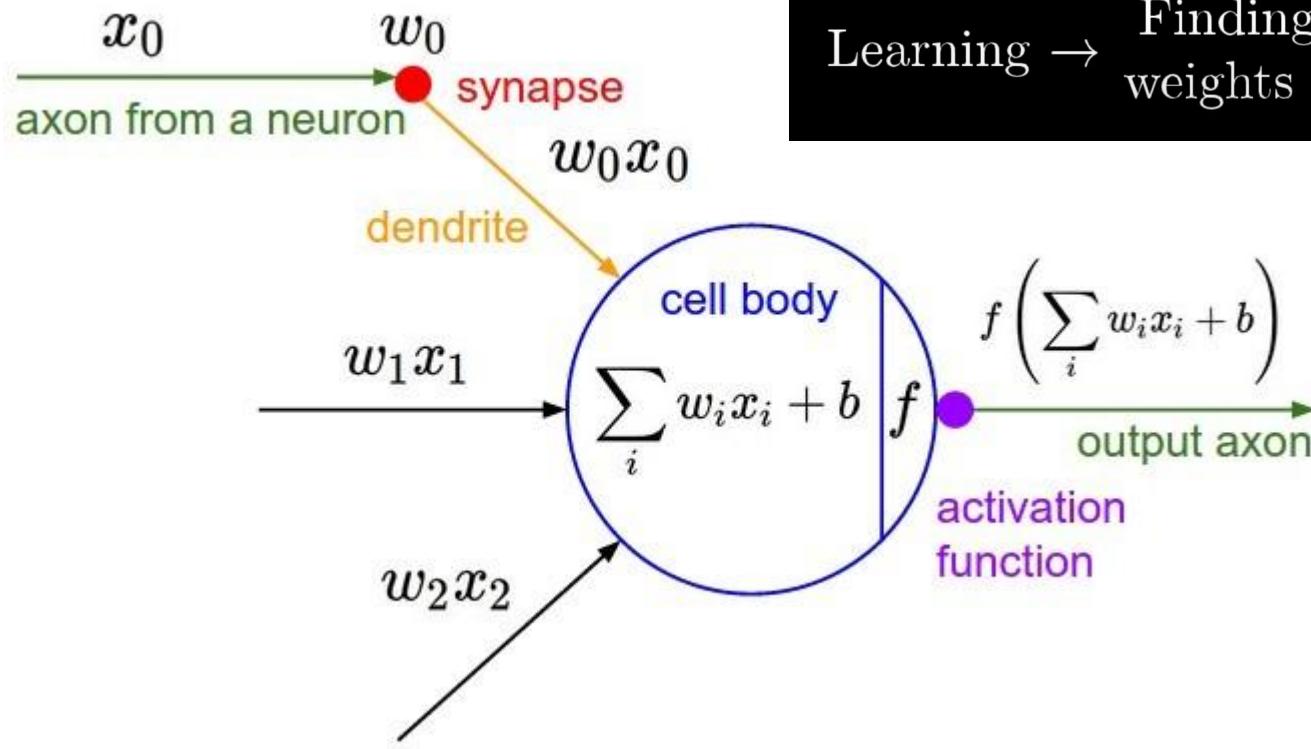


Credit: 3Blue1Brown

Let's Build Our Model!

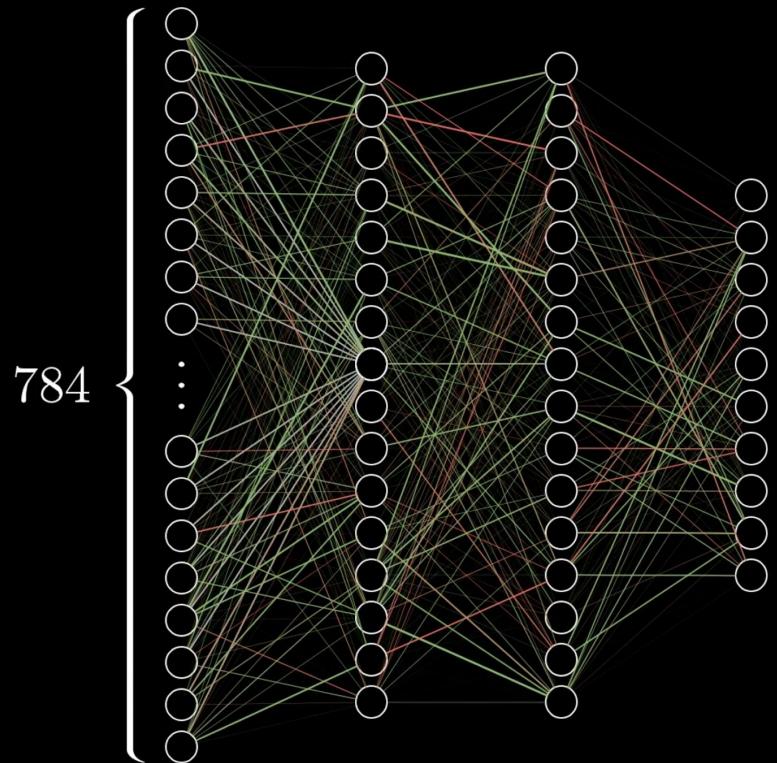
How do we
improve from
mistakes? Why
'Adam' and 'binary
cross-entropy'?

Bias



Learning → Finding the right weights and biases

Initially, the weights are all random...



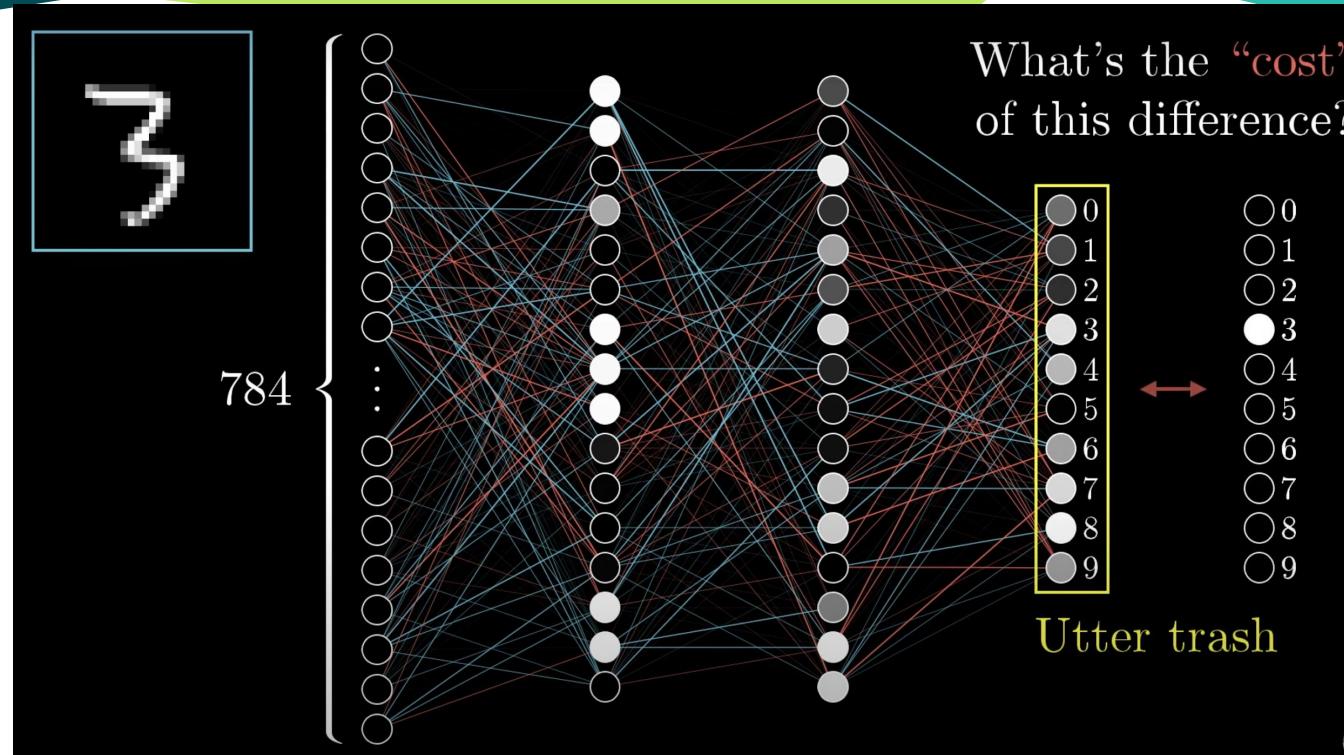
784×16 + 16×16 + 16×10
weights

16 + 16 + 10
biases

13,002

$$y = \sum_i w_i x_i + b$$

Where learning comes in



Credit: 3Blue1Brown

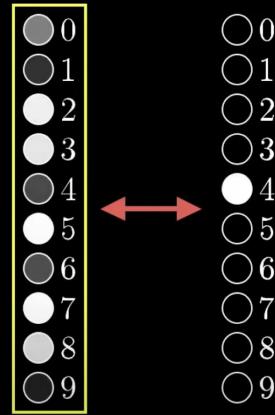
Where learning comes in

Average cost of
all training data...

Cost of 

$$\left\{ \begin{array}{l} (0.51 - 0.00)^2 + \\ (0.20 - 0.00)^2 + \\ (0.95 - 0.00)^2 + \\ (0.91 - 0.00)^2 + \\ (0.29 - 1.00)^2 + \\ (1.00 - 0.00)^2 + \\ (0.31 - 0.00)^2 + \\ (0.98 - 0.00)^2 + \\ (0.86 - 0.00)^2 + \\ (0.13 - 0.00)^2 \end{array} \right.$$

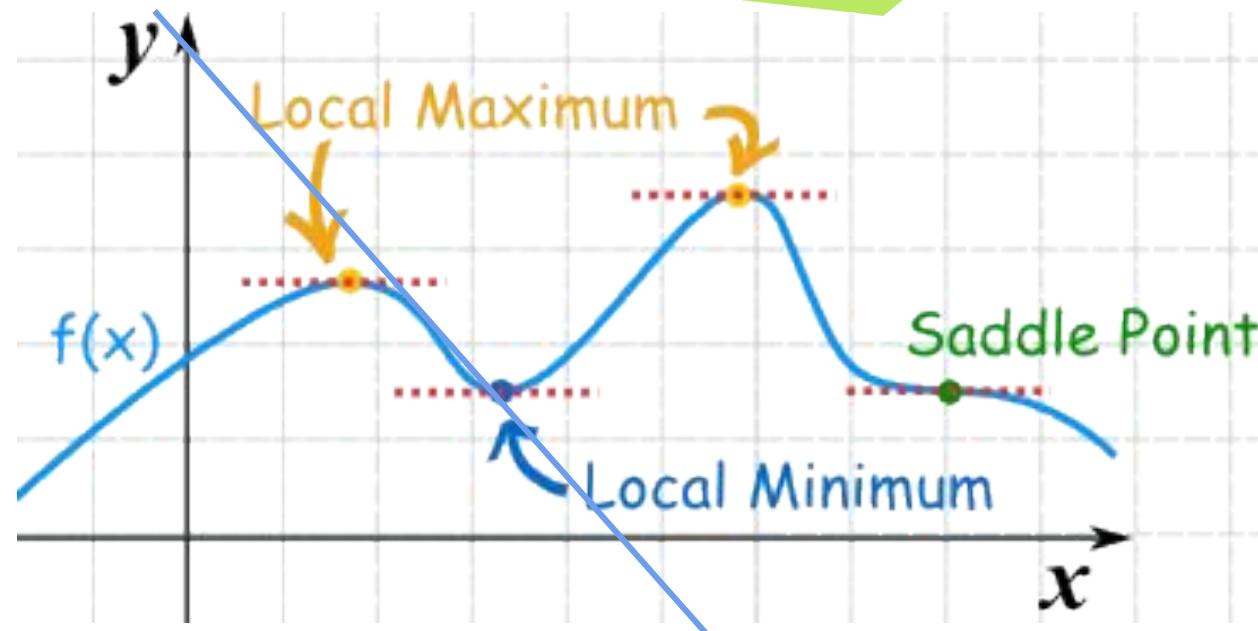
What's the “cost”
of this difference?



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

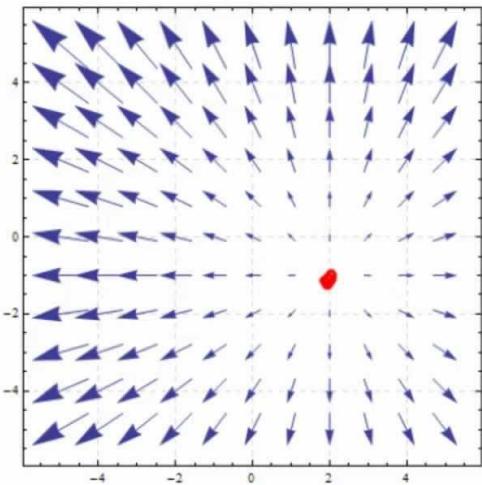
Credit: 3Blue1Brown

Where the learning comes in

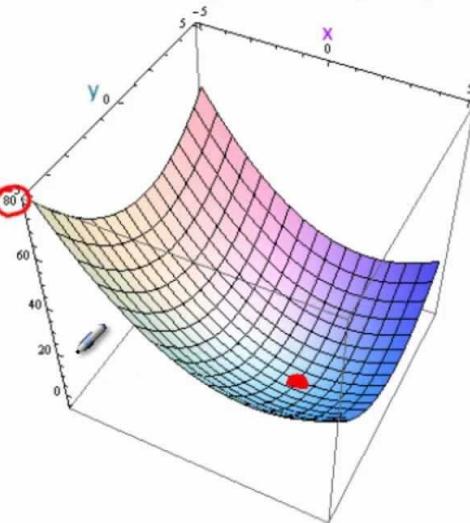


Find the minimum cost

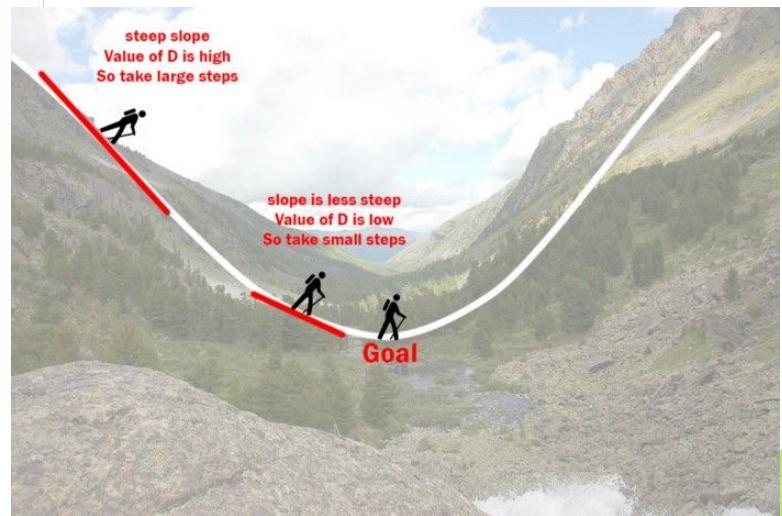
The gradient field $\langle 2x-4, 2y+2 \rangle$



of the function $f = x^2 - 4x + y^2 + 2y$.



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



Stochastic Gradient Descent



Compute gradient descent step (using backprop)



Do in mini batches

What is an epoch?

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

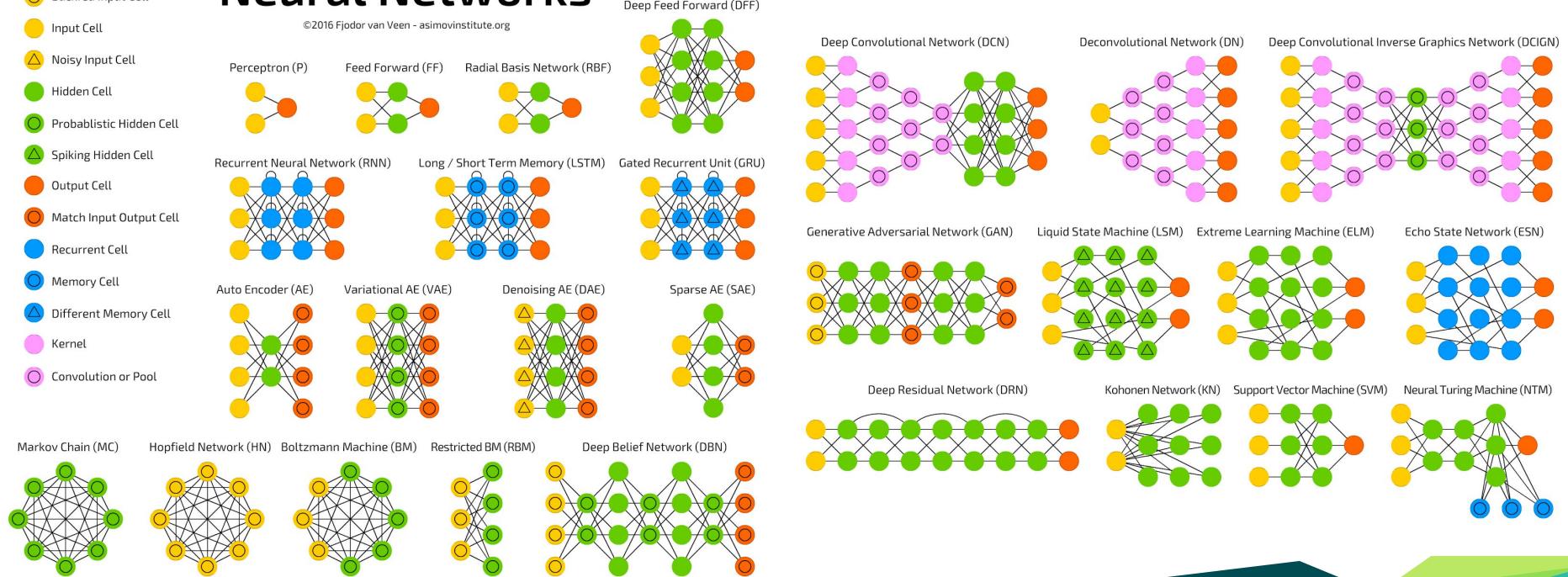
Types of Neural Networks...

A mostly complete chart of

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool



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