Welcome to General Assembly

- **WiFi** GA Guest
- **Password** yellowpencil

GENERAL ASSEMBLY

DATA SCIENCE

DAT11SYD

Lesson 17: Artificial Neural Networks

Course Plan

Date	Class	Lesson	Who
Monday, 19 February 2018	Lesson 1	Introduction to Data Science	Paul
Wednesday, 21 February 2018	Lesson 2	Elements of Data Science	Paul
Monday, 26 February 2018	Lesson 3	Data Visualisation	Paul
Wednesday, 28 February 2018	Lesson 4	Linear Regression	Paul
Monday, 5 March 2018	No Class	* Paul ill	.--
Wednesday, 7 March 2018	Lesson 5	Logistic Regression	Paul
Monday, 12 March 2018	Lesson 6	Model Evaluation	Paul
Wednesday, 14 March 2018	Lesson 7	Regularisation	Paul
Monday, 19 March 2018	Lesson 8	Clustering	Paul
Wednesday, 21 March 2018	Lesson 9	Recommendations	Paul
Monday, 26 March 2018	Lesson 10	SQL + Productivity	Paul
Wednesday, 28 March 2018	Lesson 11	Decision Trees	Greg
Monday, 2 April 2018	No Class	Easter Monday	.--
Wednesday, 4 April 2018	Lesson 12	Ensembles	Greg
Monday, 9 April 2018	Lesson 13	Natural Language Programming	Greg
Wednesday, 11 April 2018	No Class	** Paul ill	.--
Monday, 16 April 2018	Lesson 14	Time Series + R	Paul
Wednesday, 18 April 2018	Lesson 15	Soft Skills + Cloud Computing	Paul
Monday, 23 April 2018	Lesson 16	Network Analysis	Paul
Wednesday, 25 April 2018	No Class	ANZAC Day	.--
Monday, 30 April 2018	Lesson 17	Neural Networks	Paul
Wednesday, 2 May 2018	Lesson 18	GA Data Science Alumni Panel / Final Project Help	Olivia / Paul
Monday, 7 May 2018	Lesson 19	* Final Projects Presentations	Paul
Wednesday, 9 May 2018	Lesson 20	** Final Projects Presentations	Paul

Lesson 17 - Review

FINAL PROJECT

ONE WEEK LEFT!!!

- Final Project split into 4-parts: [Review with James 5mins]

(a) Real-world Problem Identification [Lesson 14]

(b) Data Cleaning [Lesson 15]

(c) Model & Validation [Lesson 16]

(d) Presentation & Storytelling [Lesson 17]

Lesson 18 – any final queries

Lesson 19 & 20 – Presenting final project back to class

Git & GitHub – 1 Pager Guide!

(Part B) EVERY CLASS:

At the START of the class, you'll need to sync the latest materials from the COURSE repo:

(1) **Make sure you are in the dat11syd directory:**
```
cd ~/workspace/dat11syd
```
(2) **Make sure to select the "master" branch of your repo:**
```
git checkout master
```
(3) **Fetch the latest changes from the UPSTREAM repo (i.e the course repo)**
```
git fetch upstream
```
(4) **Merge the changes from the upstream repo to your master branch:**
```
git merge upstream/master
```

DURING the class:

(5) **Before editing, either copy files to your "students/" folder, or rename them**

At the END of every class:

(6) **Make sure you are in the dat11syd directory:**
```
cd ~/workspace/dat11syd
```
(7) **Add any files that you've updated to your git registry:**
```
git add -A
```
(8) **Commit the changes with a sensible comment:**
```
git commit -m "my updates for lesson 7"
```
(9) **Push your changes to your PERSONAL repo:**
```
git push origin master
```

DONE!!!!!

LEARNING

Have you ever questioned what this is?

Kinds of Learning in Nature

1. Individual Learning

   - Individual survival is important (as well as survival of species)
   - High independent intelligence
   - Learning by observation as well as trial & error
   - Individual / localised communication
   - Individual awareness & high-level decisions made consciously

2. Group Learning

   - Hive survival important (individuals are not important)
   - “Hive/Swarm” Intelligence (Low individual intelligence)
   - “Learning” by shotgun approach (feedback loop)
   - Communication by global broadcasting (pheromones) / not individually targeted
   - Complex mathematical problems often solved by optimising something specific

Examples: Learning in Nature

1. **Individual Learning**
   Dolphins - https://www.youtube.com/watch?v=HQ38sycAITQ
   Octopus - https://www.youtube.com/watch?v=GQwJXvITWDw

2. **Group Learning**
   Ants
   Colony Optimisation: https://www.youtube.com/watch?v=GQwJXvITWDw
   Double Bridge Experiment: https://www.youtube.com/watch?v=3oCQ2DWAp4c

   Bees
   Defensive Adaptation: https://www.youtube.com/watch?v=3P8svtzTRuI

   TED Talk: https://www.youtube.com/watch?v=LHgVR0IzFJc

Artificial Learning

Based upon fundamental principles of learning in nature:

- **Specific problem to solve:**
  "Do X more efficiently"

- **INCENTIVISED: REWARD / PENALTY**
  - **Positive behavior reinforced**
  - **Negative behavior discouraged**

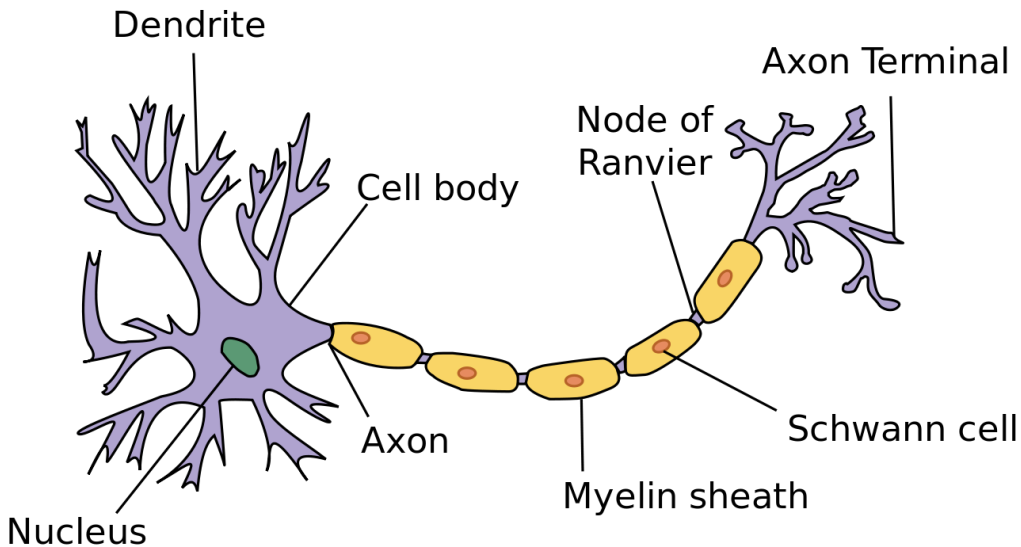
- **FEEDBACK LOOP (Positive Feedback Loop, etc)**

…with a few extra technical tricks thrown in.

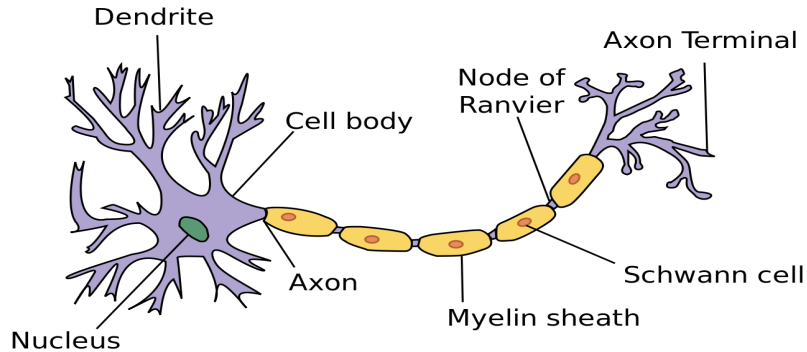
DATA SCIENCE PART TIME COURSE

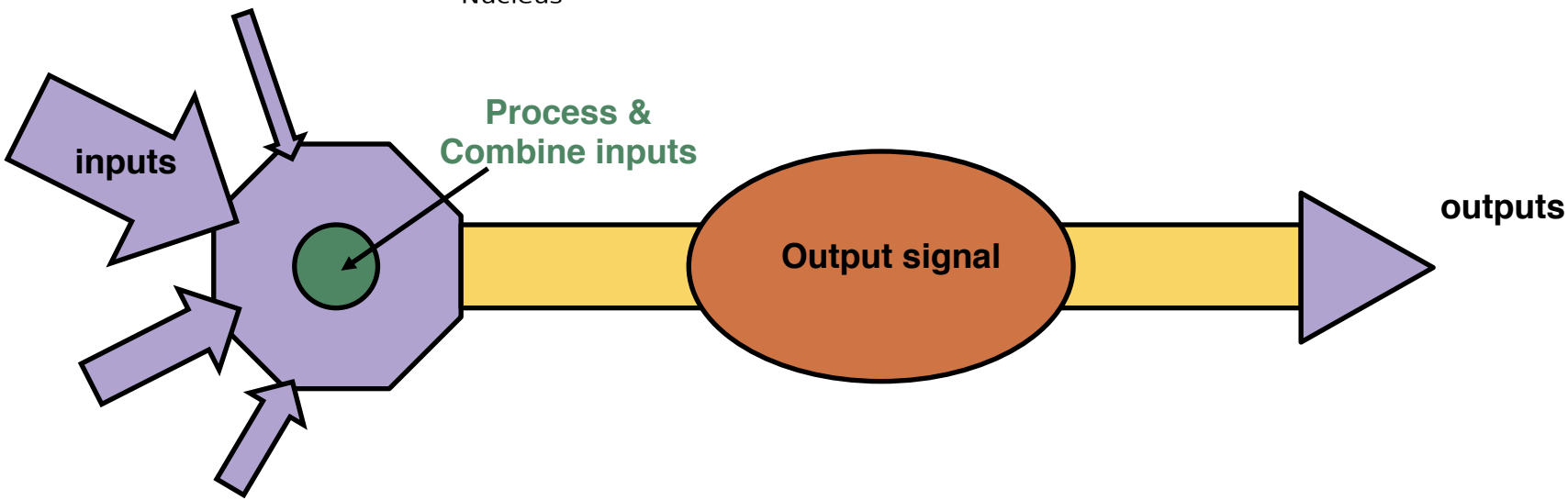
WHAT IS A NEURAL NETWORK?

The NEURON – The basis of a Biological Neural Network

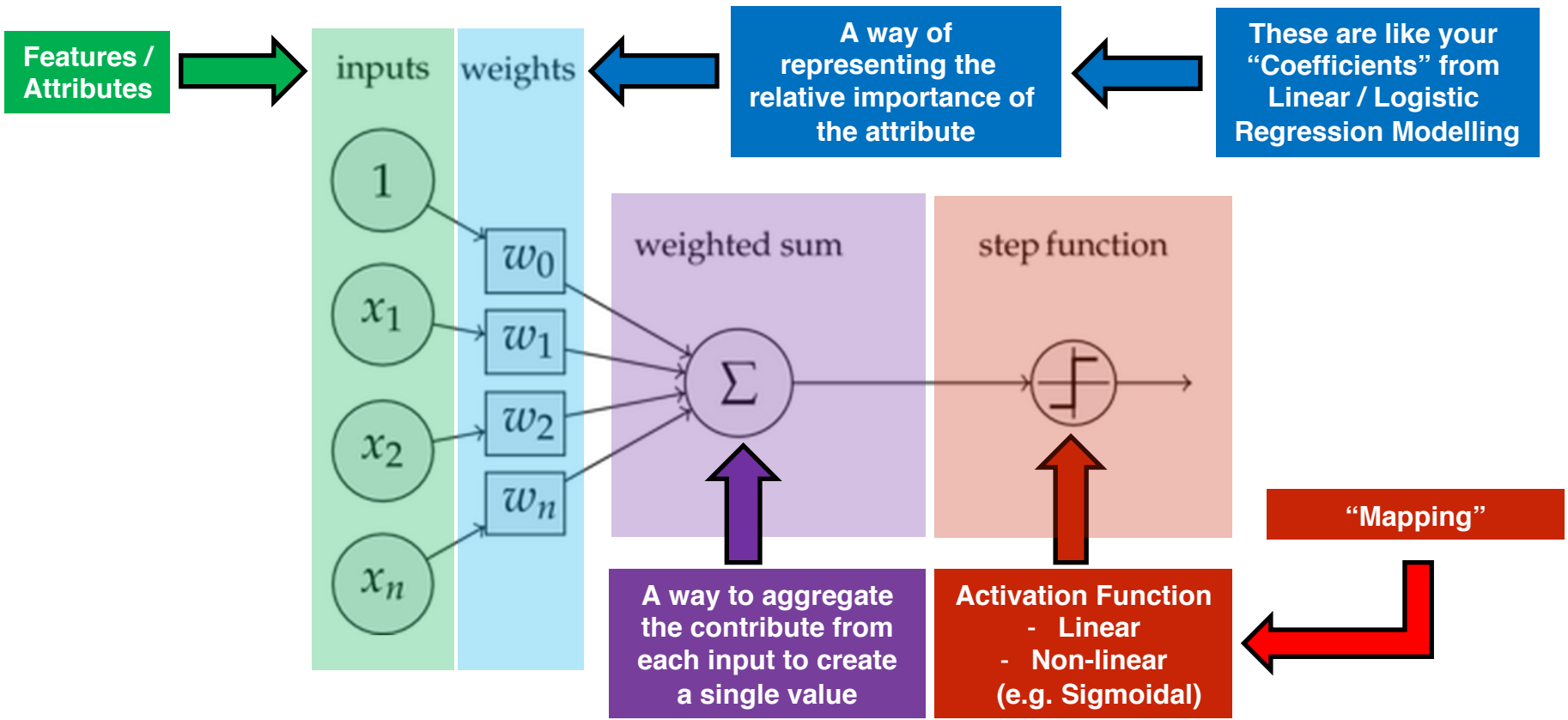


The NEURON - simplified



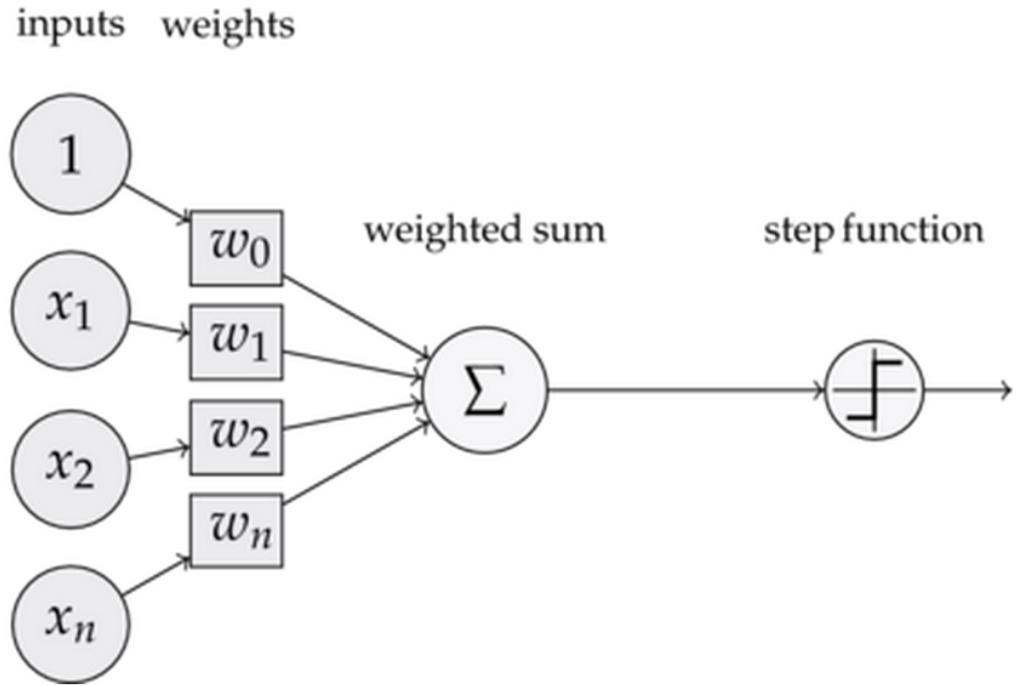


PERCEPTRON – The basis of the Artificial Neural Network

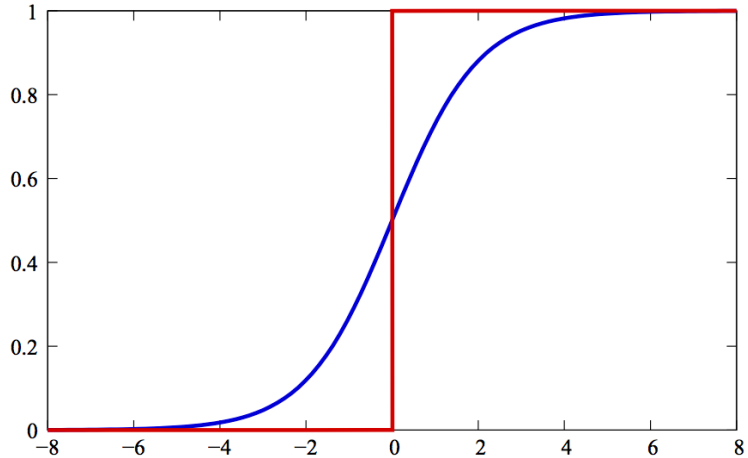


WHAT IS AN ARTIFICIAL NEURAL NETWORK?

A computational system comprised of layers and each layer is built of interconnected perceptrons



Takes in input and uses an activation function in order to output

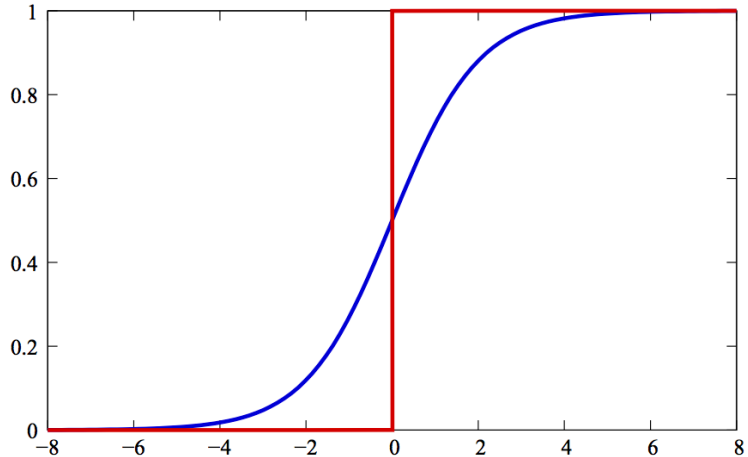


$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

f_{log} is called logistic function

Takes in input and uses an activation function in order to output

What is z?



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

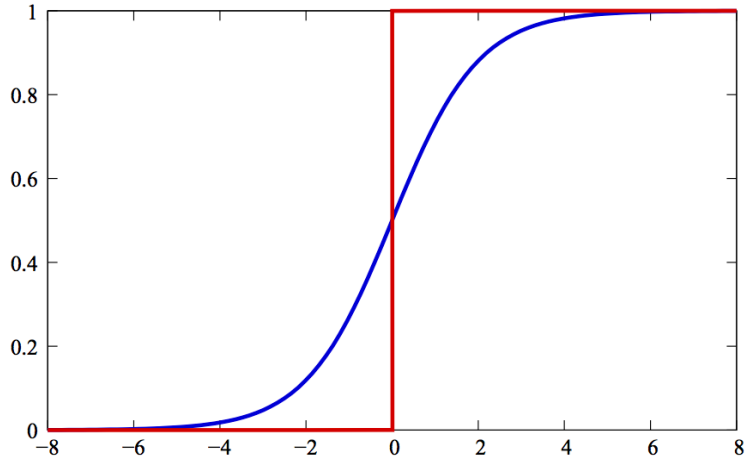
f_{log} is called logistic function

Takes in input and uses an activation function in order to output.

z is a weighted sum on the inputs.

$$z = \sum_{j=0}^{n} w_j * x_j$$

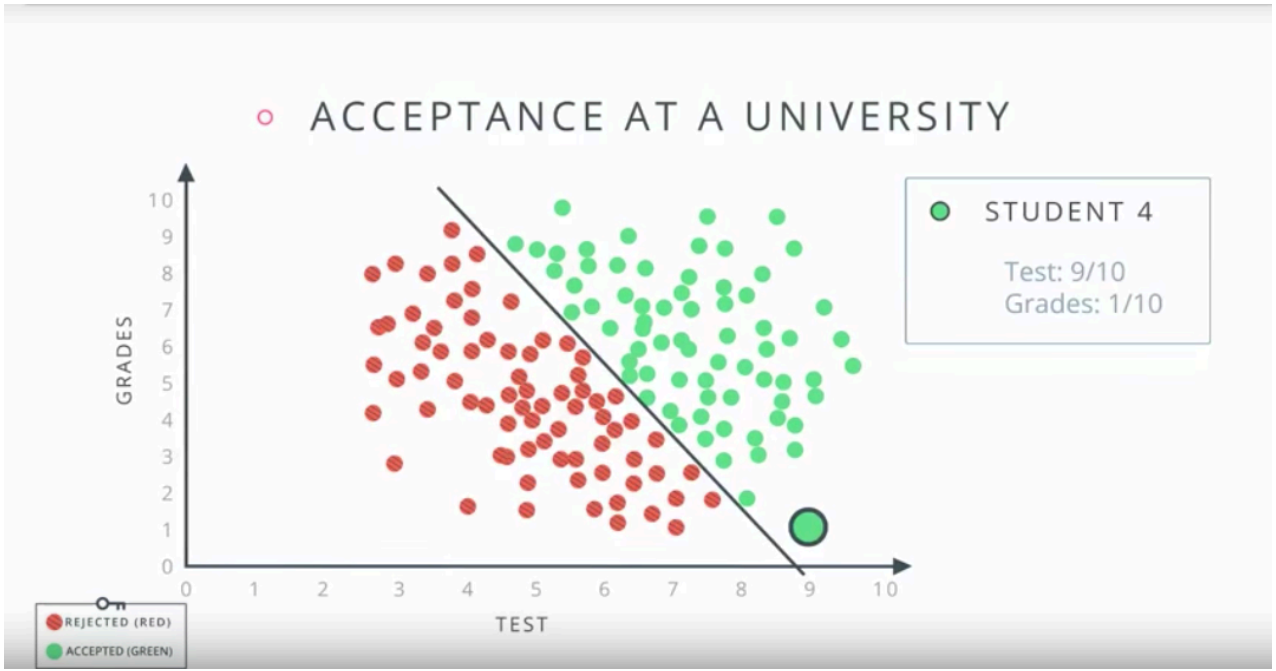
Where w_i is the weight on input x_i



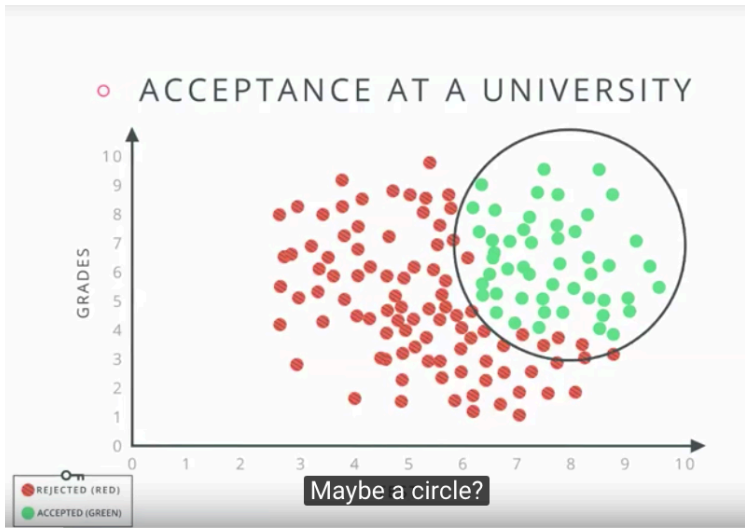
$$f_{log}(z) = \frac{1}{1 + e^{-z}}$$

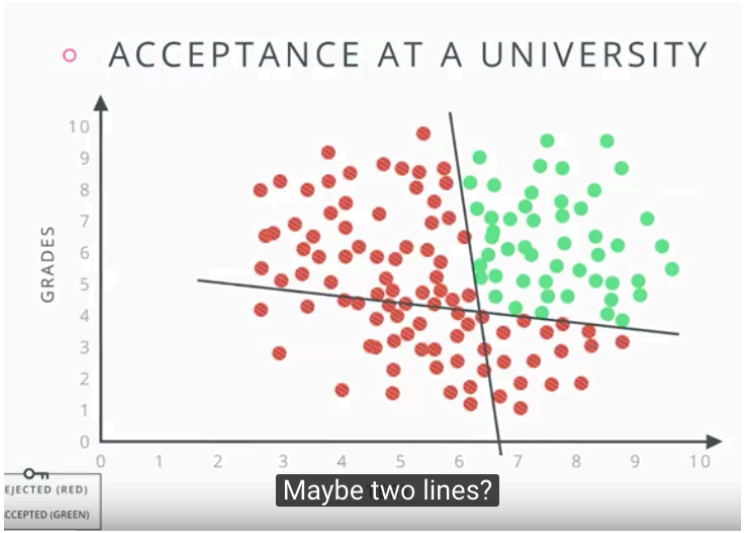
f_{log} is called logistic function

Simple Example

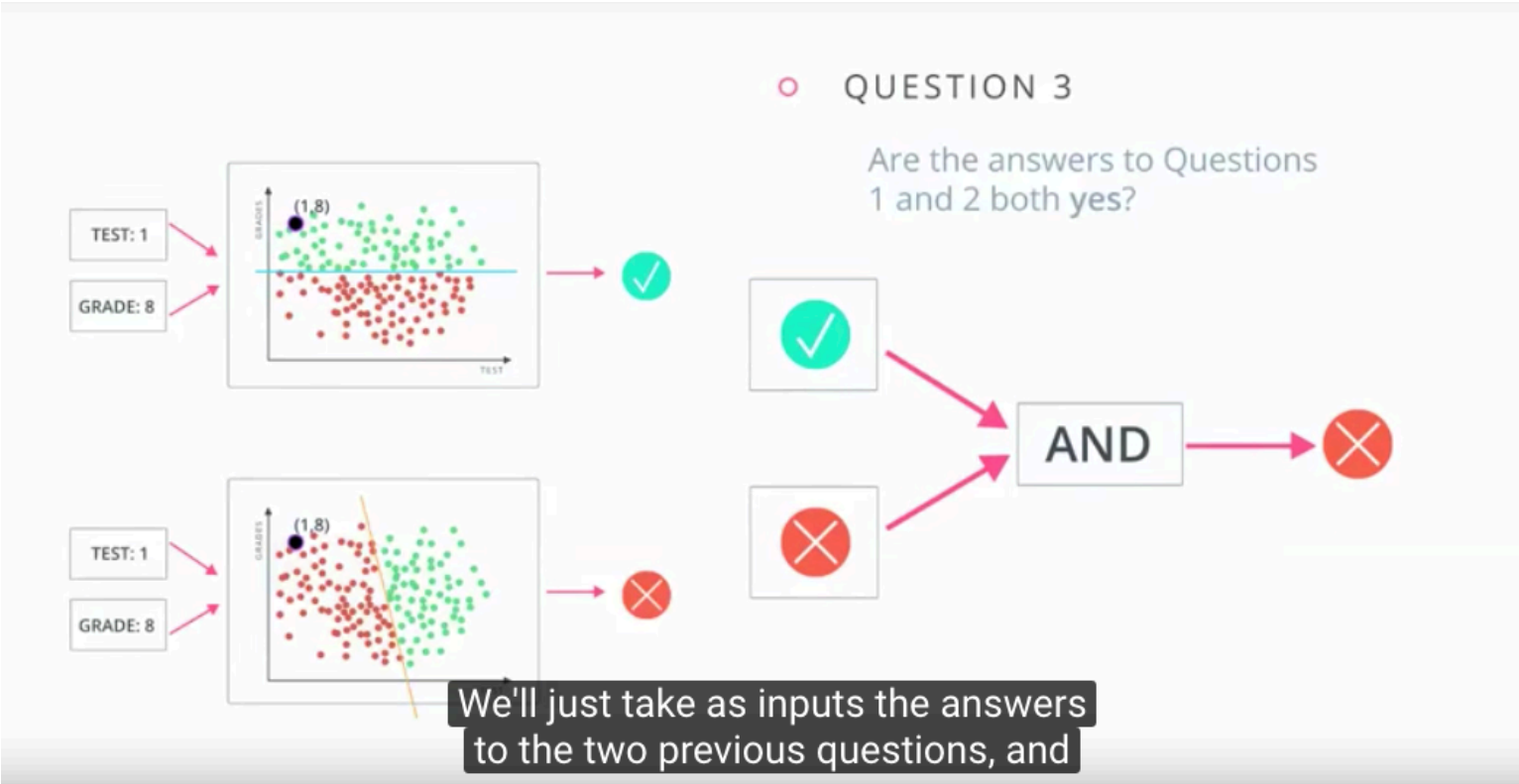


Simple Example

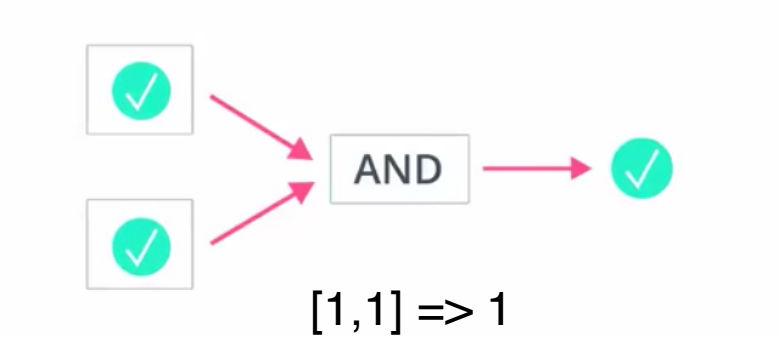


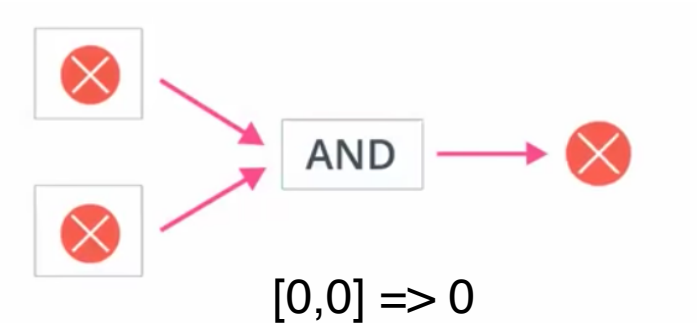


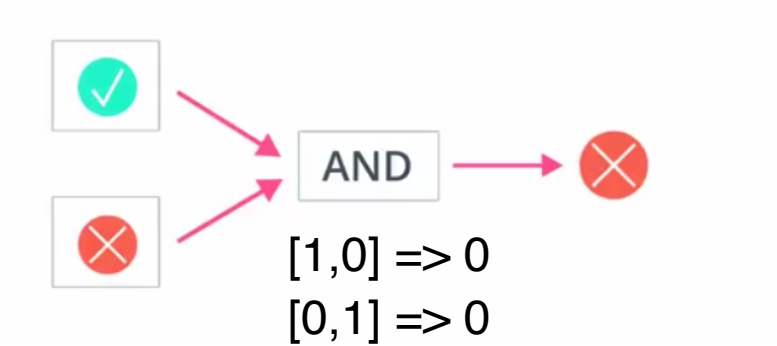
Simple Example



Simple Example

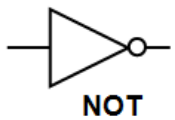






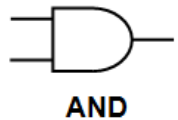
pseudocode: if(sum == 2) then 1 else 0

Some other possible Logic Functions:



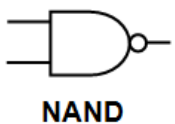
NOT

Input	Output
I	F
0	1
1	0



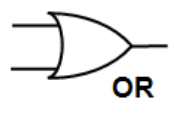
AND

Inputs		Output
A	B	F
0	0	0
1	0	0
0	1	0
1	1	1



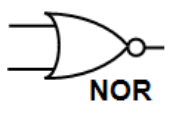
NAND

Inputs		Output
A	B	F
0	0	1
1	0	1
0	1	1
1	1	0



OR

Inputs		Output
A	B	F
0	0	0
1	0	1
0	1	1
1	1	1



NOR

Inputs		Output
A	B	F
0	0	1
1	0	0
0	1	0
1	1	0

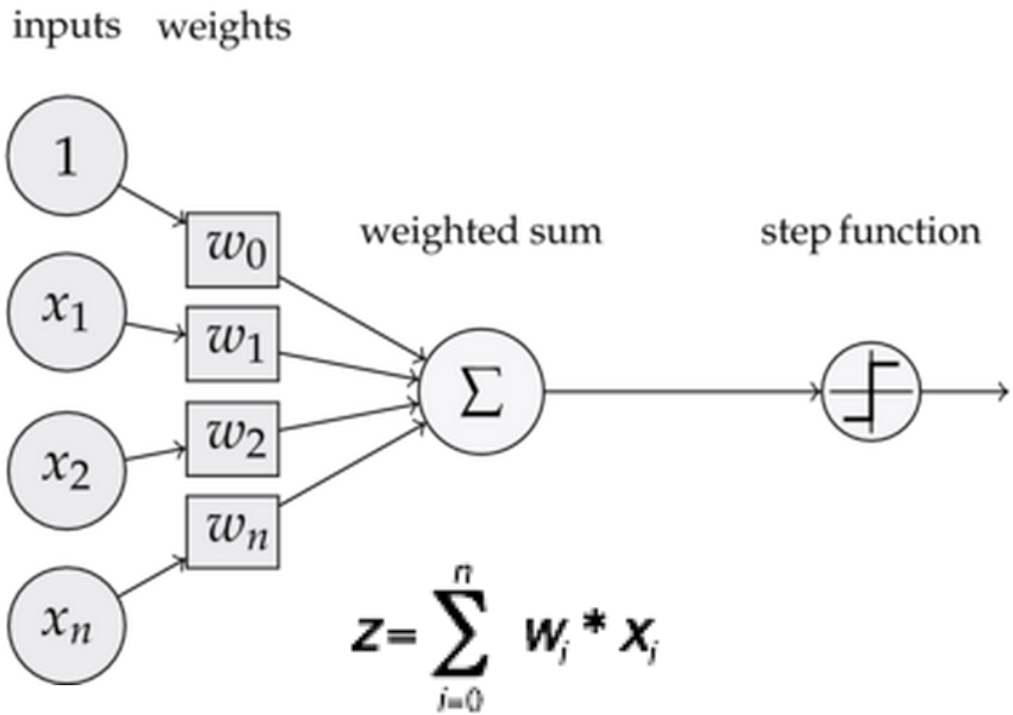
EXCLUSIVE OR

Inputs		Output
A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

EXCLUSIVE NOR

Inputs		Output
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

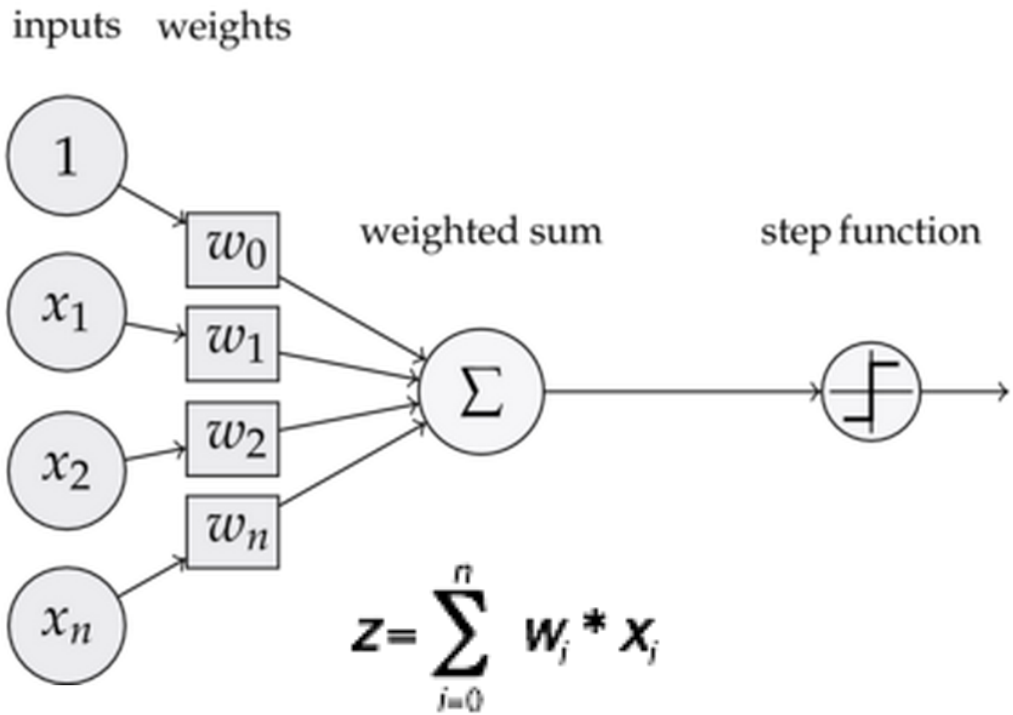
http://www.schoolphysics.co.uk/age16-19/Electronics/Logic%20gates/text/Logic_gates/index.html



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

f_{log} is called logistic function

SINGLE PERCEPTRON



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

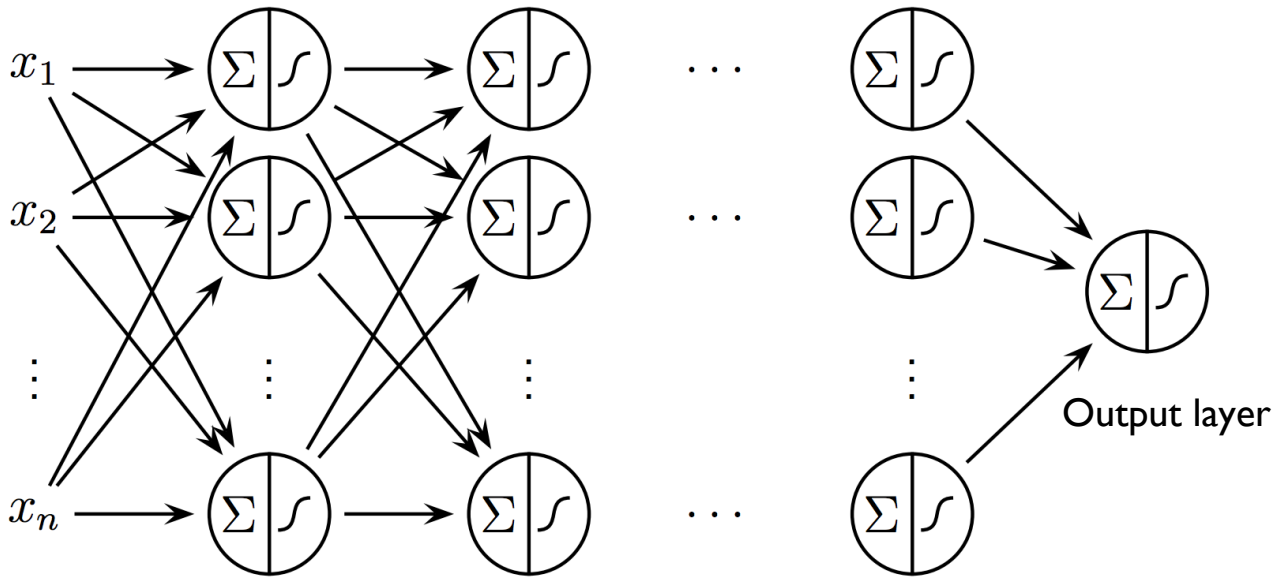
f_{log} is called logistic function

If f(z) if above a threshold, generally called theta, then the neuron "activates"

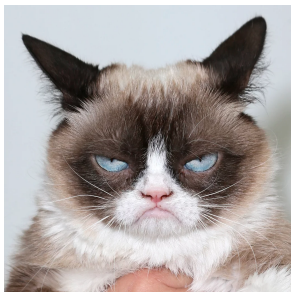
DATA SCIENCE PART TIME COURSE

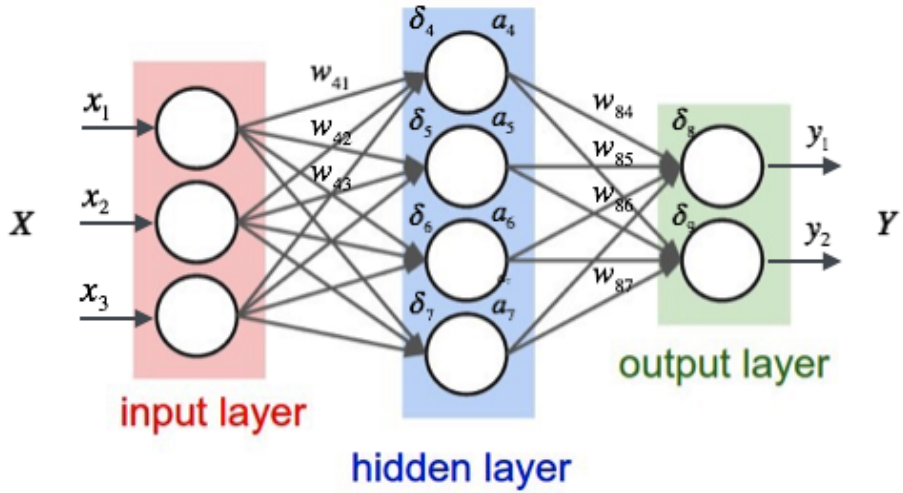
NETWORK STRUCTURE

A multi layer perceptrons (MLP) is a finite acyclic graph. The nodes are neurons with logistic activation.



WHAT IS AN ARTIFICIAL NEURAL NETWORK? 27





Cat

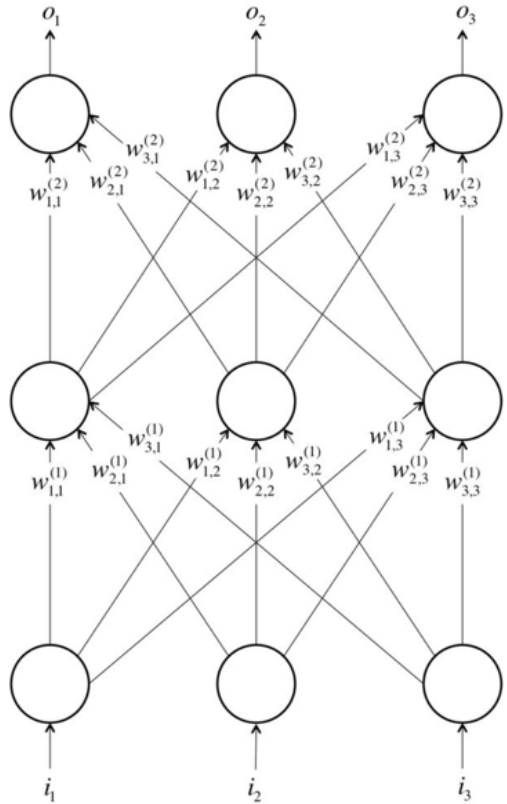
- **Input Layer - the original features of our dataset (our X)**
- **Hidden Layer - these are the derived features of the network. They are called hidden because they are not directly observed.**
- **Output Layer - the final transformation of the inputs into a result**

NETWORK STRUCTURE

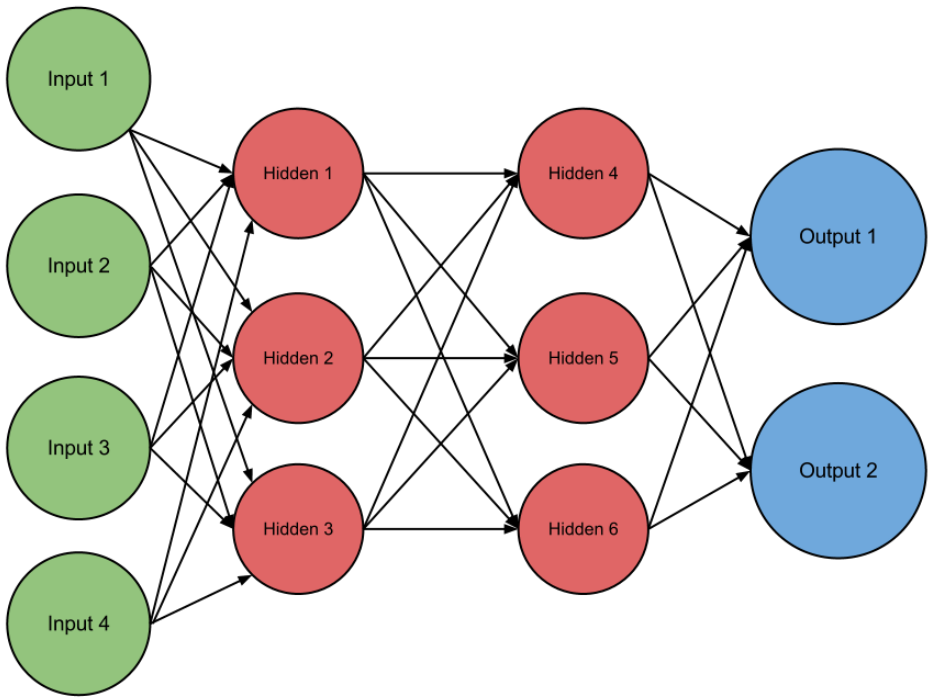
Output Layer ->

Hidden Layers ->

Input Layer ->

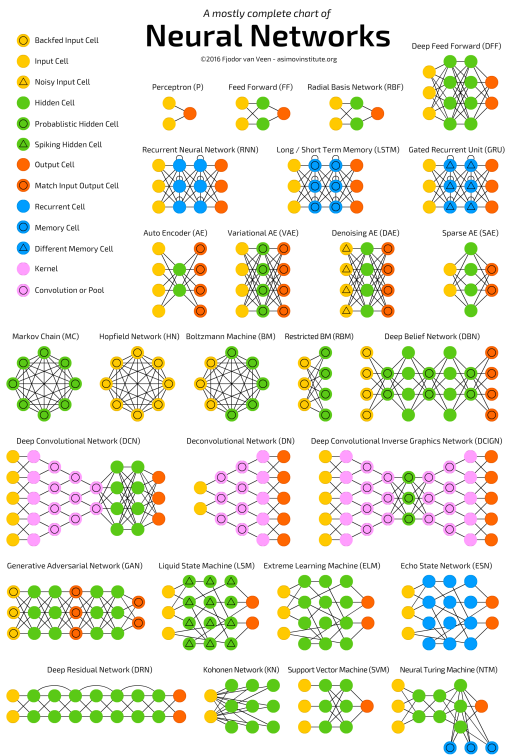


Hidden layers often have fewer neurons than the input layer to force the network to learn compressed representations of the original input.

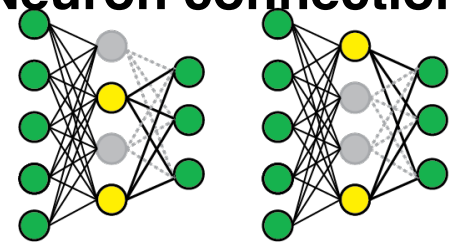


NETWORK STRUCTURE

Architecture

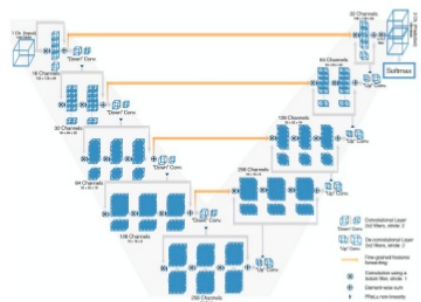


Neuron connections



Deep Residual Learning / Skip connections

V-Net



Milletari, Fausto, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation." In 3D Vision (3DV), 2016 Fourth International Conference on, pp. 565-571. IEEE, 2016.

DATA SCIENCE PART TIME COURSE

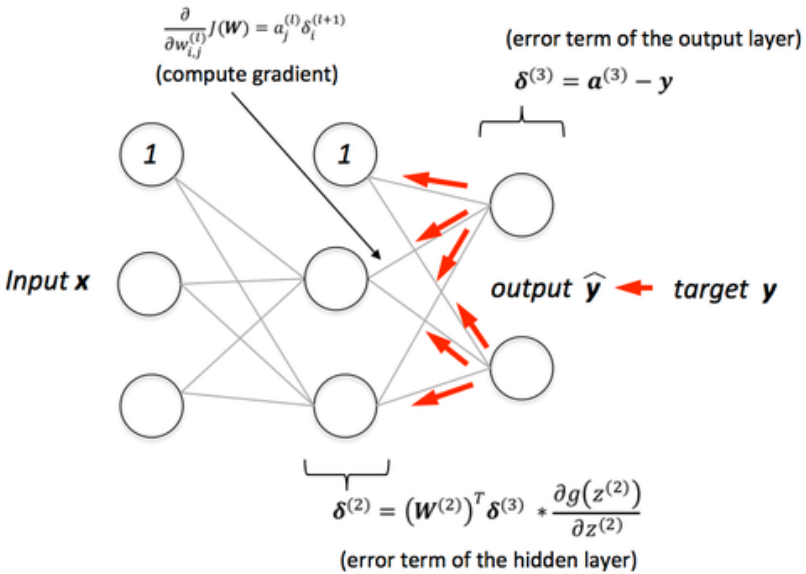
HOW DO WE FIND THE WEIGHTS?

DATA SCIENCE PART TIME COURSE

BACK-PROPAGATION

BACK-PROPOGATION

Training updates the sigmoid function weights to get the best predictions possible.



If an observation goes through the model and is outputted as False when it should have been True the logistic functions in the perceptrons are changed slightly according to the error.

Back-Propogation is a two-pass algorithm.

The Forward pass uses the weights to calculate the prediction.

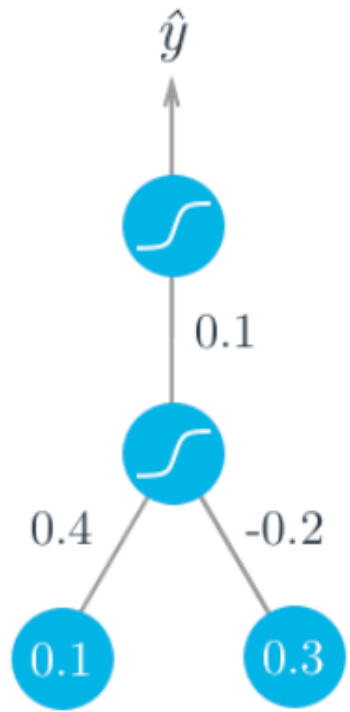
The Backward pass calculates the errors on the output layer and are then back-propagated to give the errors at the hidden layer units.

The weights must then be updated.

Lather, Rinse, Repeat. (Until some criterial is satisfied)

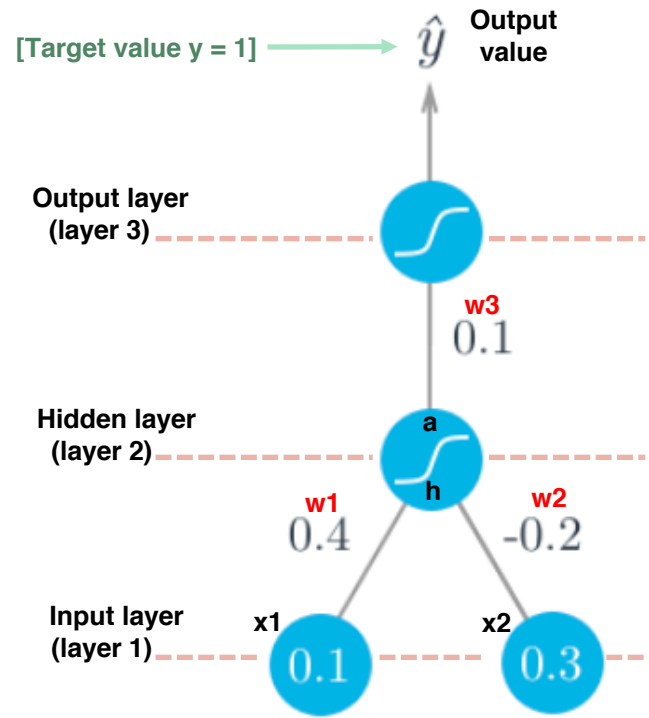
BACK-PROPOGATION: Simple 3-layer example

Assume we're trying to fit some binary data and the target is $y = 1$.



BACK-PROPOGATION: Simple 3-layer example

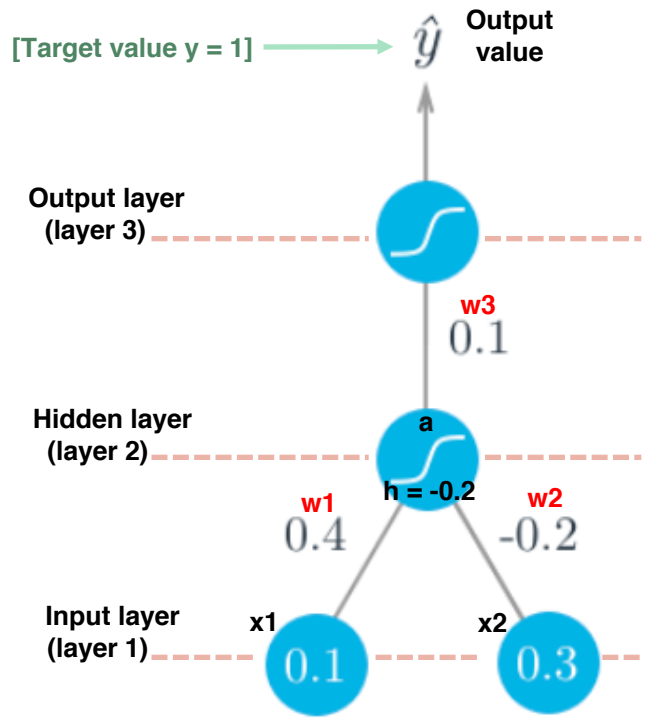
Assume we're trying to fit some binary data and the target is $y = 1$.



BACK-PROPOGATION: Simple 3-layer example

Assume we're trying to fit some binary data and the target is $y = 1$. We'll start with the forward pass, first calculating the input to the hidden unit

$$h = \sum_i w_i x_i = \underset{\text{x1}}{0.1} \times \underset{\text{w1}}{0.4} - \underset{\text{x2}}{0.2} \times \underset{\text{w2}}{0.3} = -0.02$$



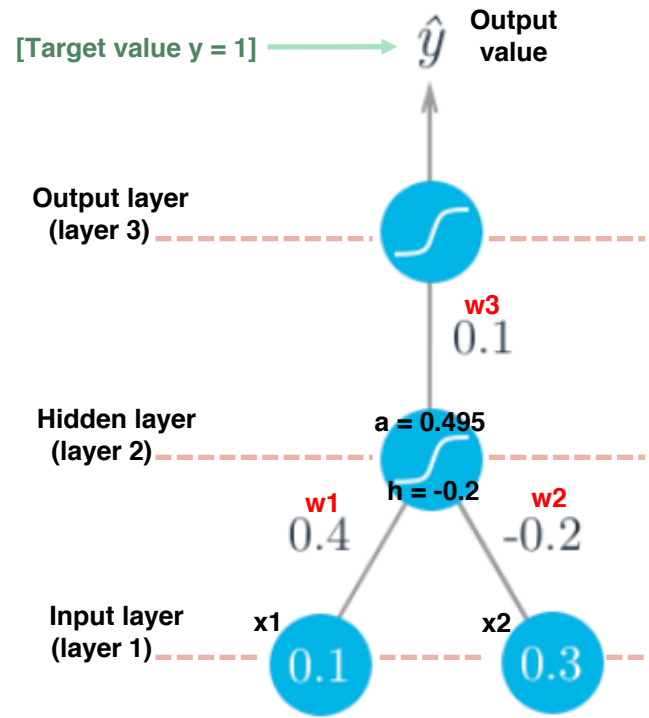
BACK-PROPOGATION: Simple 3-layer example

Assume we're trying to fit some binary data and the target is $y = 1$. We'll start with the forward pass, first calculating the input to the hidden unit

$$h = \sum_i w_i x_i = \overset{x1}{0.1} \times \overset{w1}{0.4} - \overset{x2}{0.2} \times \overset{w2}{0.3} = -0.02$$

and the output of the hidden unit

$$a = f(h) = \text{sigmoid}(-0.02) = 0.495.$$



BACK-PROPOGATION: Simple 3-layer example

Assume we're trying to fit some binary data and the target is $y = 1$. We'll start with the forward pass, first calculating the input to the hidden unit

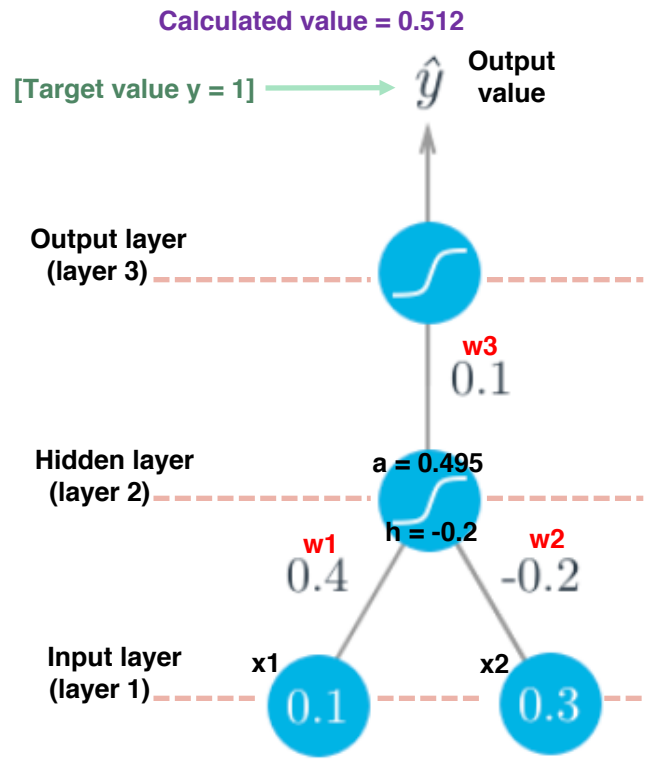
$$h = \sum_i w_i x_i = \overset{x1}{0.1} \times \overset{w1}{0.4} - \overset{x2}{0.2} \times \overset{w2}{0.3} = -0.02$$

and the output of the hidden unit

$$a = f(h) = \text{sigmoid}(-0.02) = 0.495.$$

Using this as the input to the output unit, the output of the network is

$$\hat{y} = f(W \cdot a) = \text{sigmoid}(0.1 \times 0.495) = 0.512.$$



BACK-PROPOGATION

With the network output, we can start the backwards pass to calculate the weight updates for both layers. Using the fact that for the sigmoid function $f'(W \cdot a) = f(W \cdot a)(1 - f(W \cdot a))$, the error term for the output unit is

$$\delta^o = (y - \hat{y}) f'(W \cdot a) = (1 - 0.512) \times 0.512 \times (1 - 0.512) = 0.122.$$

Now we need to calculate the error term for the hidden unit with backpropagation. Here we'll scale the error term from the output unit by the weight W connecting it to the hidden unit. For the hidden unit error term, $\delta_j^h = \sum_k W_{jk} \delta_k^o f'(h_j)$, but since we have one hidden unit and one output unit, this is much simpler.

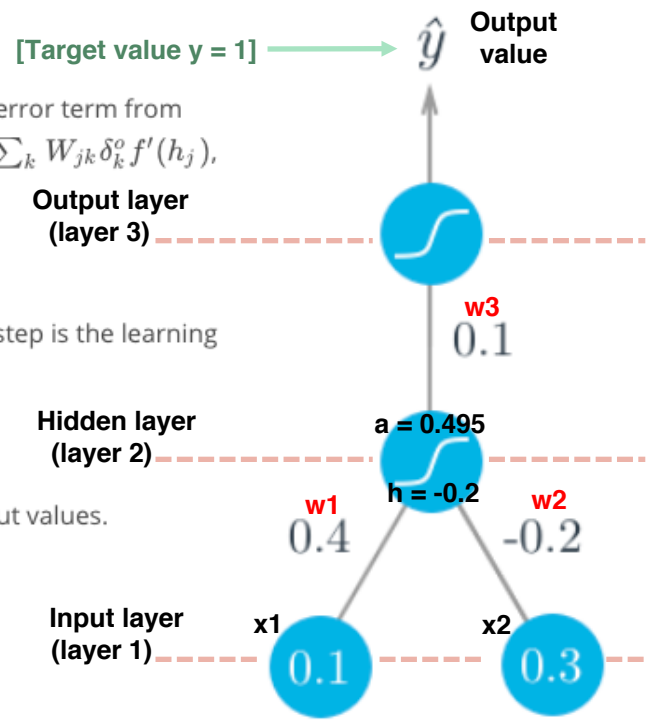
$$\delta^h = W \delta^o f'(h) = 0.1 \times 0.122 \times 0.495 \times (1 - 0.495) = 0.003$$

Now that we have the errors, we can calculate the gradient descent steps. The hidden to output weight step is the learning rate, times the output unit error, times the hidden unit activation value.

$$\Delta W = \eta \delta^o a = 0.5 \times 0.122 \times 0.495 = 0.0302$$

Then, for the input to hidden weights w_i, it's the learning rate times the hidden unit error, times the input values.

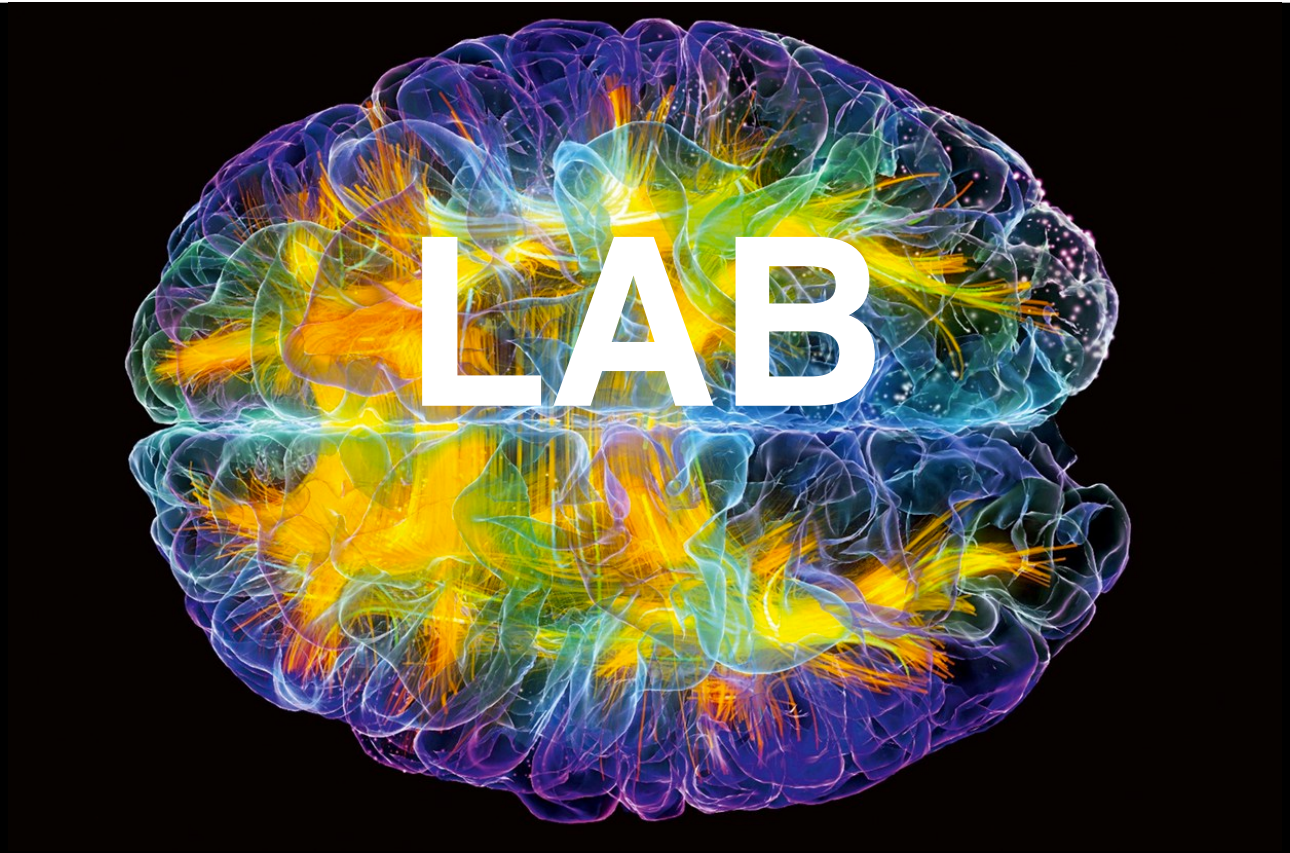
$$\Delta w_i = \eta \delta^h x_i = (0.5 \times 0.003 \times 0.1, 0.5 \times 0.003 \times 0.3) = (0.00015, 0.00045)$$



A word about: Learning Rate

- Exactly what is says!
- (Cats example)
- Can be thought of as "How quickly the model adapts to new information"
- High Learning rate => Algo adapts quickly (but may jump around!)
- Low Learning rate => Algo adapts steadily (will take longer to train)
- Chosen by the user (reasonable default = 0.5)

Perceptron - Back Propagation Example



Pros

- Can approximate almost any type of function
- Easier to run
- Very topical area of machine learning (lots of investment)

Cons

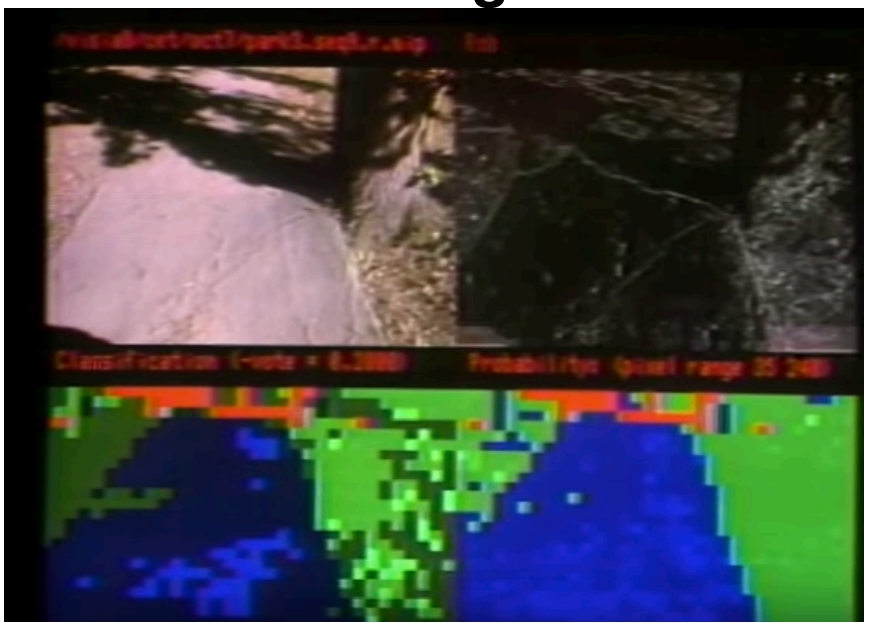
- Requires many training samples to be considered good
- Hard to describe what is happening
- Requires a lot of hardware / computation power
- Slow to train

DATA SCIENCE PART TIME COURSE

WHY ARE NEURAL NETWORKS IN VOGUE?

#topical #trending #rage

NavLab 1986 Carnegie Mellon - 1st self Driving Car



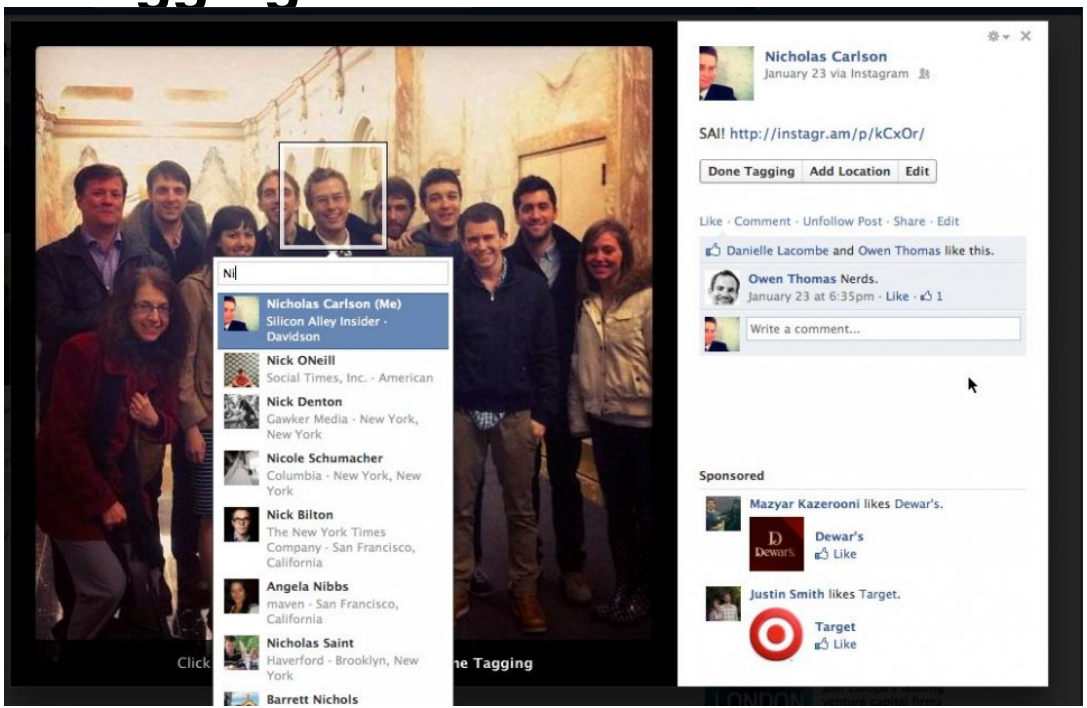
https://www.youtube.com/watch?v=ntIczNQKfjQ

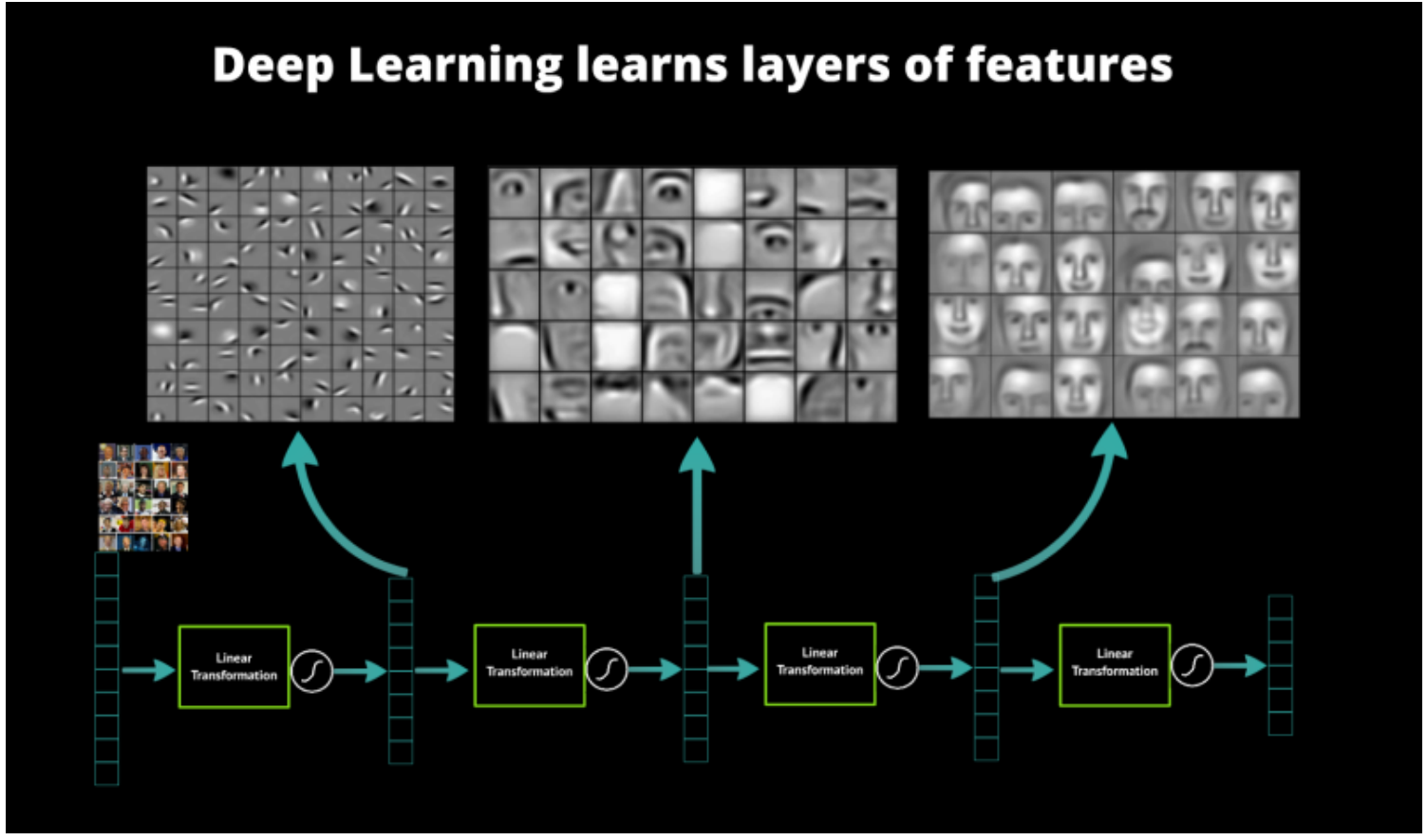
Uber Self Driving Car

Tesla Model 3

Assistants

Facebook Auto Tagging




Deep Learning learns layers of features
Linear Transformation
Linear Transformation
Linear Transformation
Linear Transformation

When is deep learning the right choice of algorithm?

When the input features are dense:

- **Images**
- **Videos**
- **Audio**
- **Text**

When is deep learning the right choice of algorithm?

When the input features are dense:

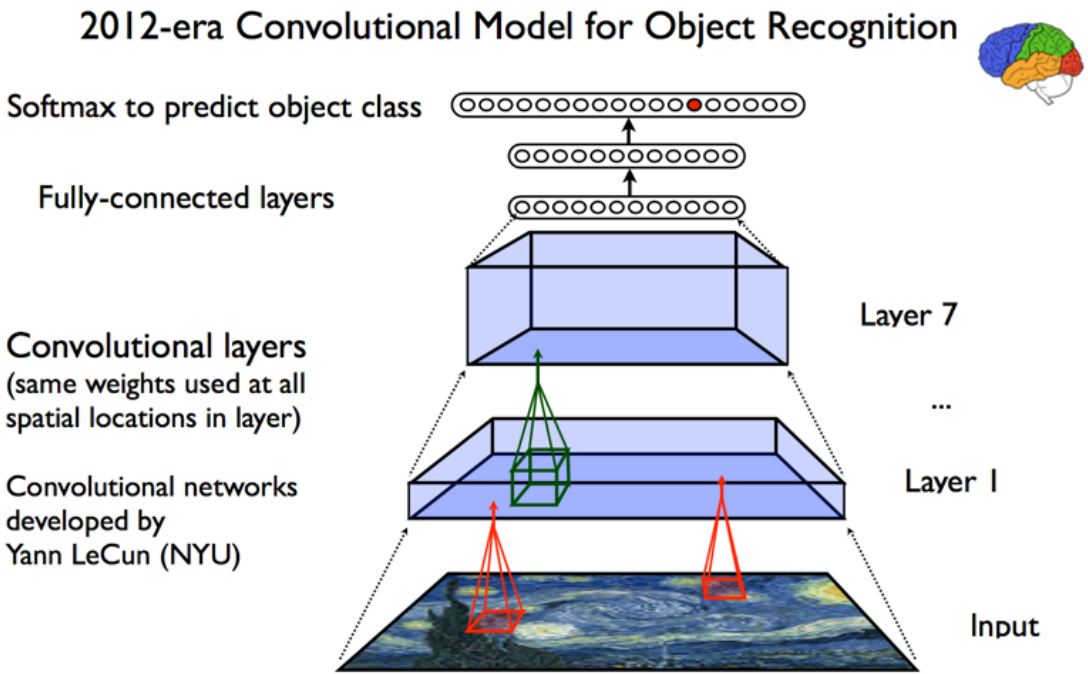
- **Images**
- **Videos**
- **Audio**
- **Text**

Some recent interesting applications:

- **Colorisation of Black and White Images.**
- **Adding Sounds To Silent Movies.**
- **Automatic Machine Translation.**
- **Object Classification in Photographs.**
- **Automatic Handwriting Generation.**
- **Character Text Generation.**
- **Image Caption Generation.**
- **Automatic Game Playing.**

Video on what neural networks see. Helpful for realising how the networks learns to distinguish through transformations.

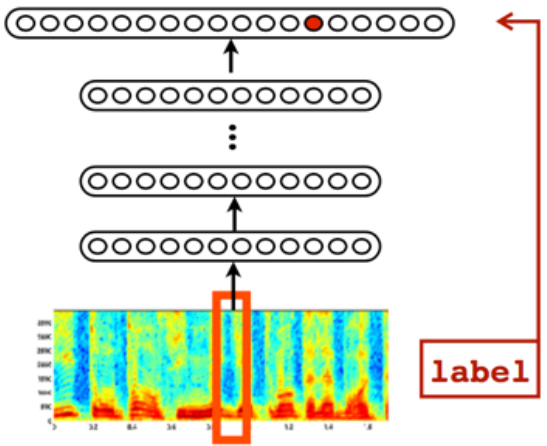
https://aiexperiments.withgoogle.com/what-neural-nets-see



Basic architecture developed by Krizhevsky, Sutskever & Hinton (all now at Google).

Won 2012 ImageNet challenge with 16.4% top-5 error rate

Acoustic Modeling for Speech Recognition



Close collaboration with Google Speech team

Trained in <5 days on cluster of 800 machines

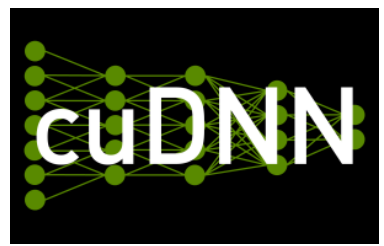
30% reduction in Word Error Rate for English
("biggest single improvement in 20 years of speech research")

Launched in 2012 at time of Jellybean release of Android

http://deepdreamgenerator.com/

TensorFlow™ is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

GPUs for processing



A demo of deep learning being applied in the browser:

http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

DATA SCIENCE

HOMEWORK

Homework

Prepare you project presentations

Extra Reading

Evaluation of deep learning frameworks: https://github.com/zer0n/deepframeworks/blob/master/README.md

Beginner: Fundamentals of Deep Learning - Nikhil Budhema (http://shop.oreilly.com/product/0636920039709.do)

Advanced: Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville (http://www.deeplearningbook.org/)

DAT11SYD - FINAL PROJECT GUIDELINES

- 30-mins per person: 25mins presenting [25m] + 5mins Question time[5m]

- 1 feedback sheet per person, per presentation – be constructive

- Pitch audience: Technical Senior Management
  ➔ Good mix between technical understanding & business acumen

Key points to get across in your presentation:

1. Motivation - Business Problem / Real World Problem description [4m]
2. Your idea / proposed solution (e.g. going to solve using techniques X+Y) [1m]
3. Data Preparation – What is available / what challenges faced? [5m]
4. Methodology – What model/technique you used / what you learned [5m]
5. Validated Results – How well did the model perform on DEV and TEST? [5m]
6. Conclusions – Explain anything unexpected & Sell the successes [5m]
7. Q+A [5m]

DATA SCIENCE PART TIME COURSE