

Machine Learning & Ethics

William Seymour | Human Centred Computing
[william.seymour@cs.ox.ac.uk](mailto:wiliam.seymour@cs.ox.ac.uk)



Hello!

- 3rd year PhD student in Human Centred Computing
- Help teach Imperative Programming and Computer Security modules
- Sideline in competitive security (hacking/lockpicking)
- Everything you see today will be online at
<https://github.com/mcnutty26/uniq>



DEPARTMENT OF
**COMPUTER
SCIENCE**



Overview

- Introduction to Machine Learning (13:30-14:10, Access Grid)
 - ML vs AI vs algorithms
 - Regression
 - Decision trees
 - Random forests
 - Neural networks
- Practical: Building a Classifier (14:20-15:00, Comlab)
 - Primer on R and RStudio
 - Training and testing classifiers
 - Leaderboard competition
- Ethics in Machine Learning (15:10-15:50, Access Grid)
- Practical : Operationalising Ethics (16:00-16:30, Comlab)



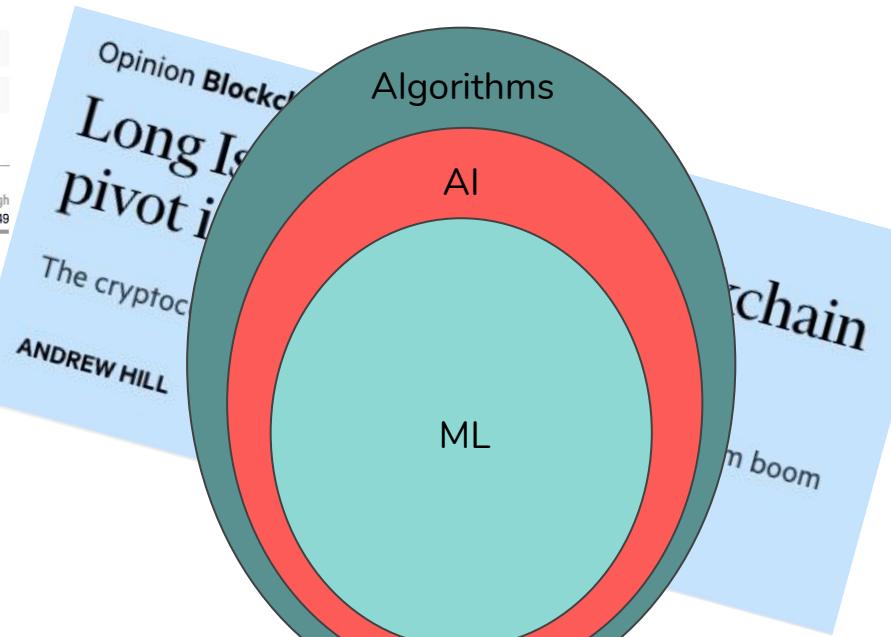
In other words, plan your startup now...

LONG ISLAND ICED TEA (LTEA)

▼ 5.71 USD -1.20 (-17.37%)

11:20:46 AM EST NAS

Prev. Close	6.91	Market Cap (USD)	23.53 M	STOCK	NAS
Open	5.58	Volume (Qty.)	596,193		





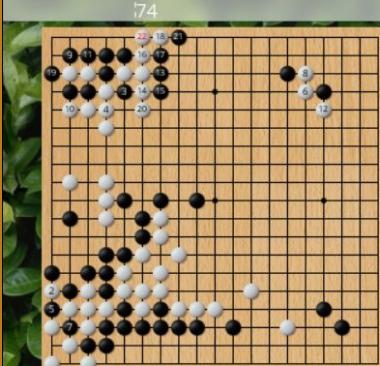
What even is “machine learning”?



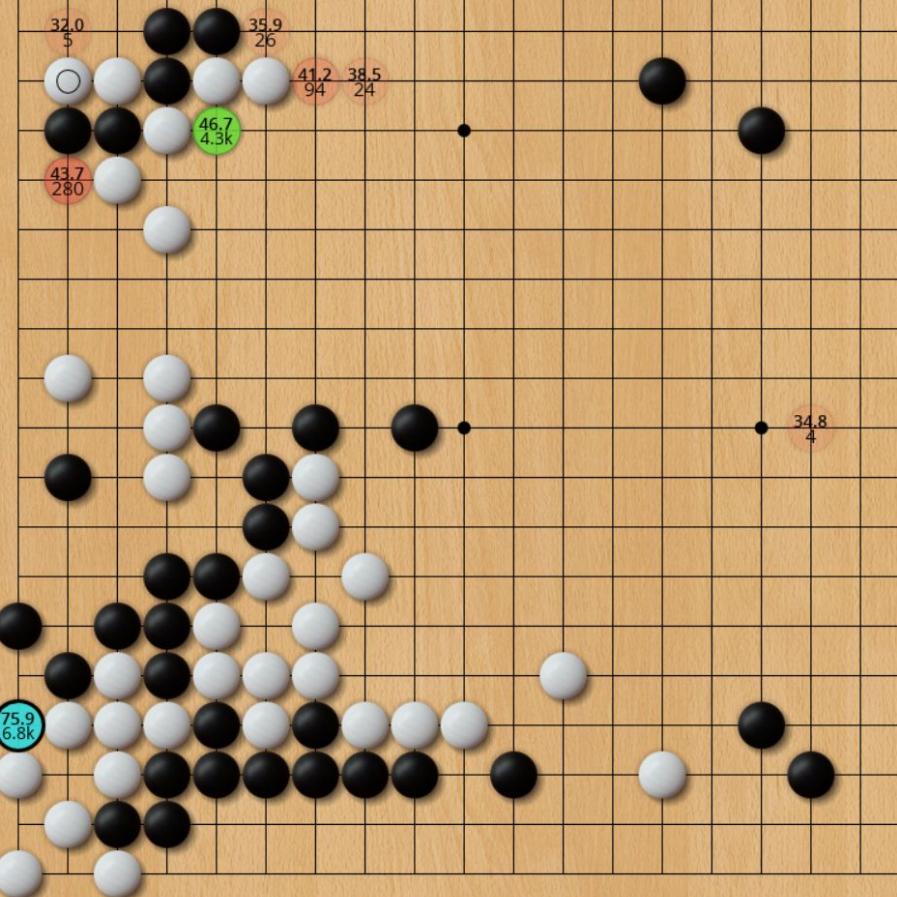
**“Using statistical
methods to have
programs improve
over time”**

2 4
63.8% 36.2%

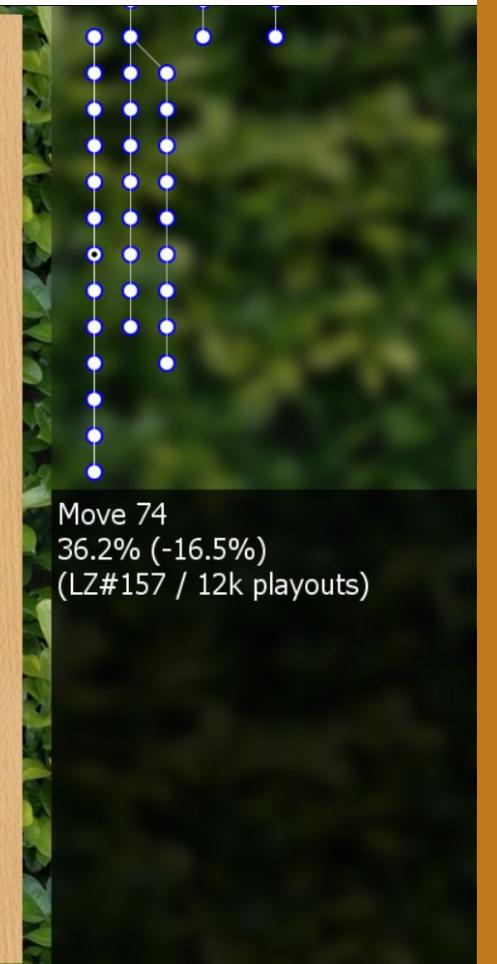
Last move: -16.5%



Pondering off LZ#157



Move 74
36.2% (-16.5%)
(LZ#157 / 12k playouts)



supervised



Photo By Ghislain Bonneau © <http://www.gbj.com>

unsupervised





Accuracy

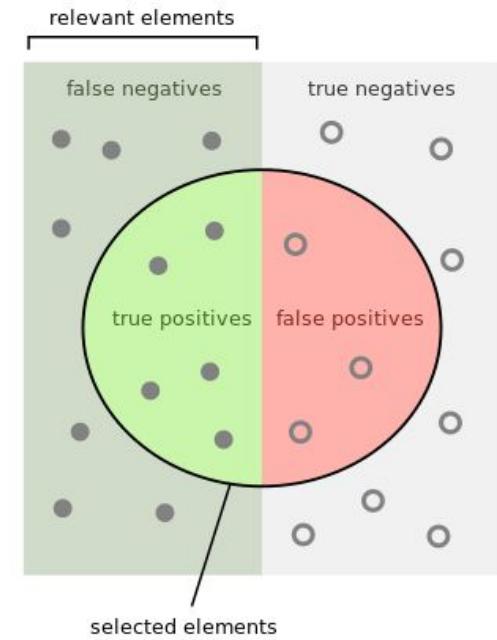
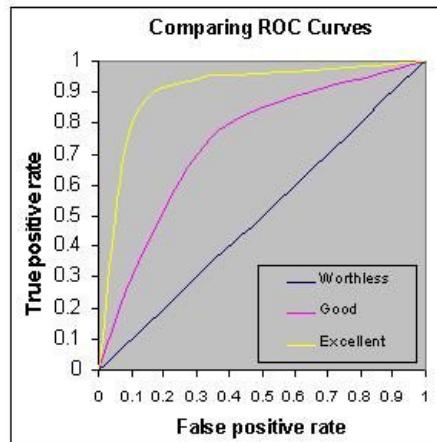


**Accuracy is just a
percentage, right?**



Accuracy and Friends

- Precision is the probability that a (random) result predicted positive is actually positive.
- Recall is the probability that a (random) positive result is predicted positive.
- Other options like ROC curves give us different information about a classifier



How many selected items are relevant?

How many relevant items are selected?

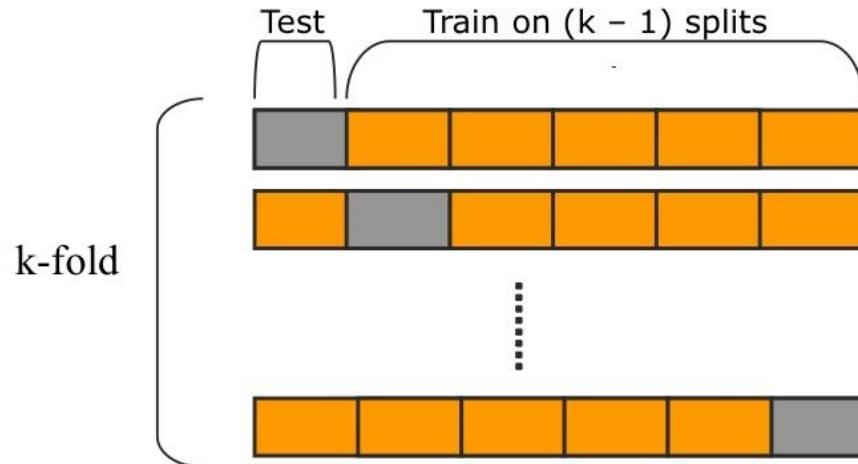
Precision =

Recall =



Training and Testing

- You calculate these metrics using a test data set
- Using the same data to test and train leads to “overfitting”
- Various (better) options like train/test split and k-fold validation



Regression

$$y = \beta X + \varepsilon$$

Statistics

2009

$$y = \beta X + \varepsilon$$

MACHINE
LEARNING

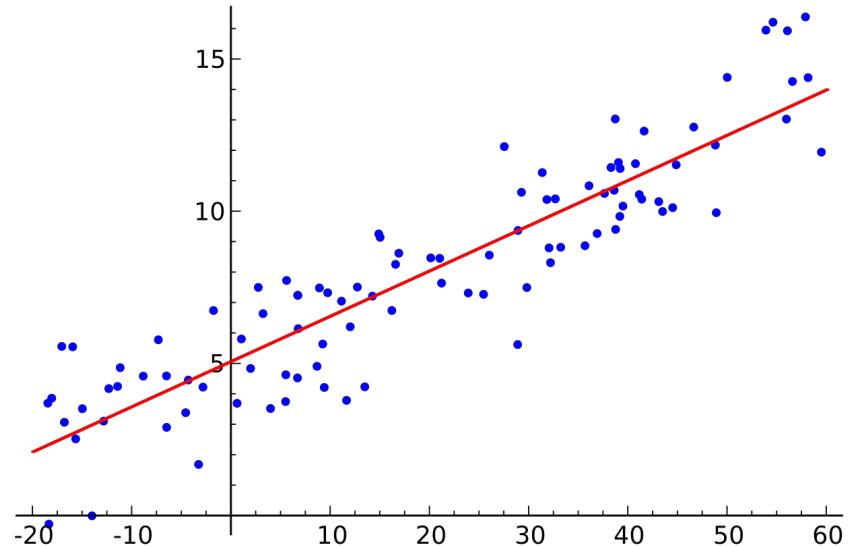
2019



Linear Regression

- Predict the values of numeric data
- Linear regression same as in S1
- $y = \alpha + \beta x + \gamma z + \dots$
- Trying to minimise sum of squared errors when fitting the line
- For a single independent variable:

$$\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$





Linear Regression Example

$$\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Password	Numbers	Crack Time (s)	$(x - \bar{x})$	$(y - \bar{y})$
v3BYUAHbJA	1	1.37		
PMAXimkXNs	0	0.52		
nX37FkRcLL	2	2.10		
wQ5TpMXAOT	1	1.92		
bS9Qqtjyb0	2	2.84		
sajWxrZqd9	1	1.65		
dcLhGSrGJG	0	0.45		
fajY1MLD00	3	2.99		
OQhMcTl8IA	1	1.87		
x4cu3tORHH	2	2.43		

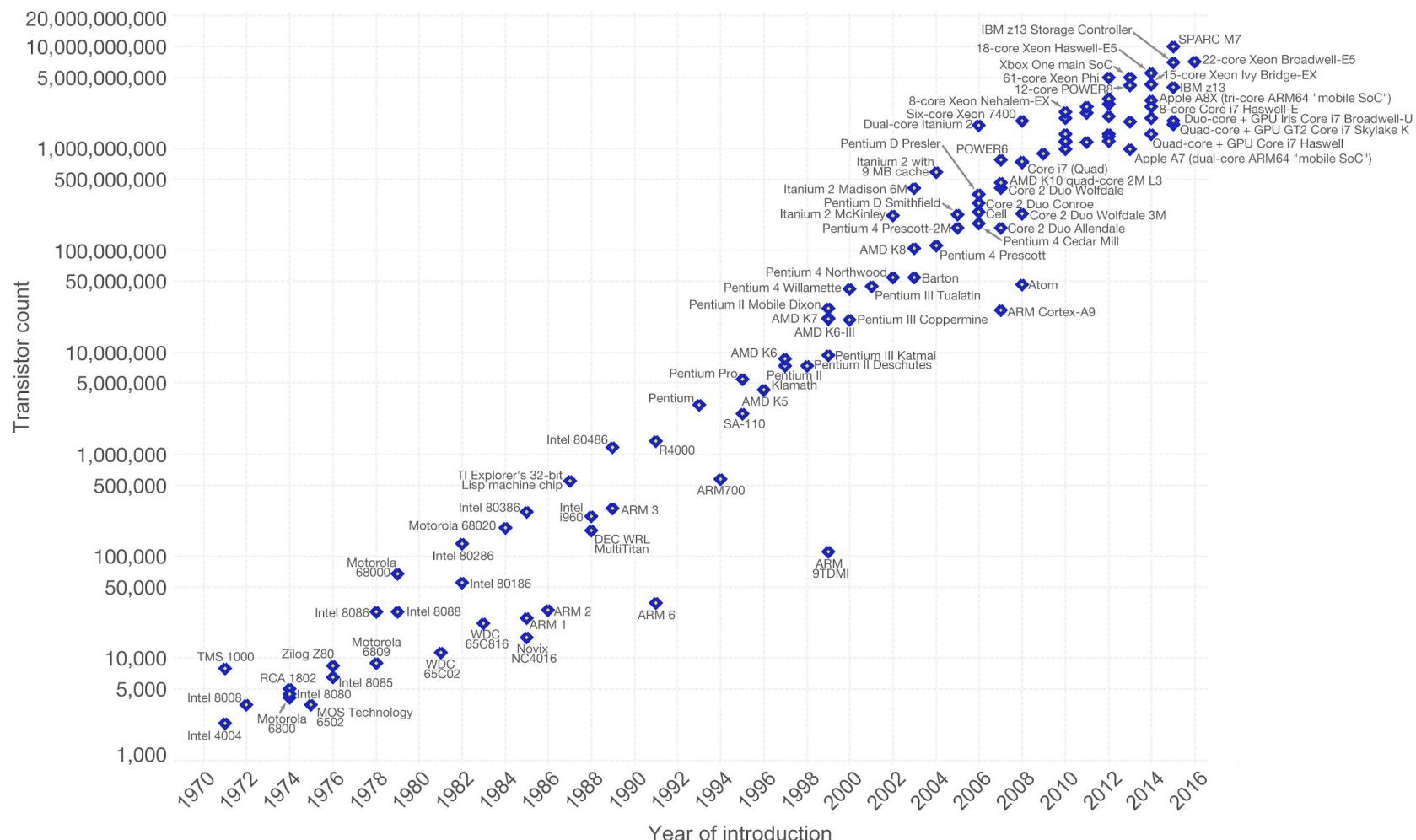


Linear Regression Example

$$\frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

- Average(x) = 1.3
- Average(y) = 1.8
- Sum of x errors = 1
- Sum of y errors = 0.14
- Sq. sum of x errors = 1
- Coefficient = 0.14
- Intercept = 1.66

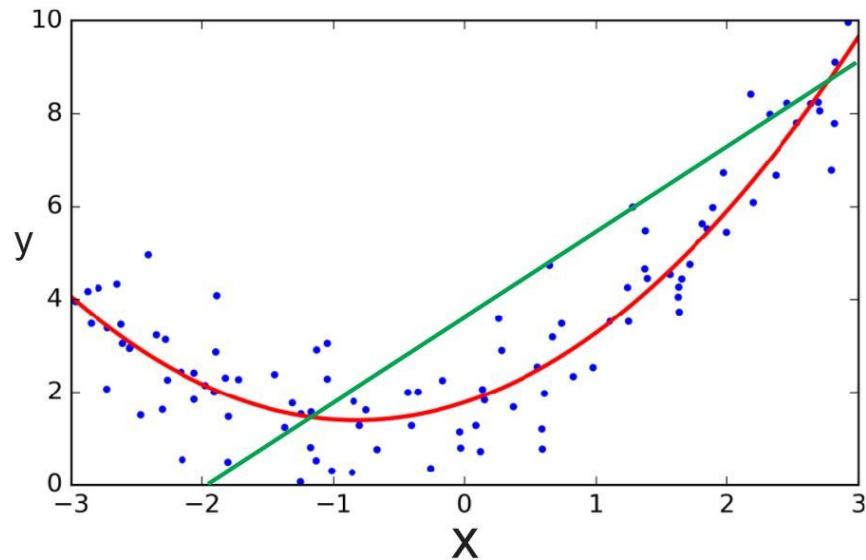
Password	Numbers	Crack Time (s)	$(x - \bar{x})$	$(y - \bar{y})$
v3BYUAHbJA	1	1.37	-0.3	-0.43
PMAXimkXNs	0	0.52	-1.3	-1.28
nX37FkRcLL	2	2.10	0.7	0.3
wQ5TpMXAOT	1	1.92	-0.3	0.12
bS9Qqtjyb0	2	2.84	0.7	1.04
sajWxrZqd9	1	1.65	-0.3	-0.15
dcLhGSrGJG	0	0.45	-0.3	-1.35
fajY1MLD00	3	2.99	1.7	1.19
OQhMcTl8IA	1	1.87	-0.3	0.07
x4cu3tORHH	2	2.43	0.7	0.63

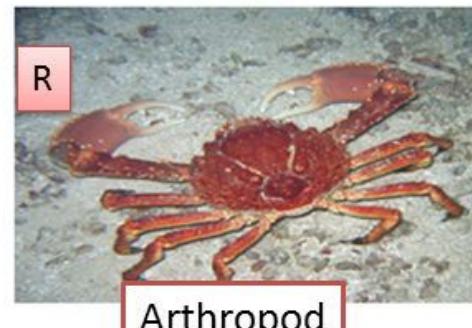
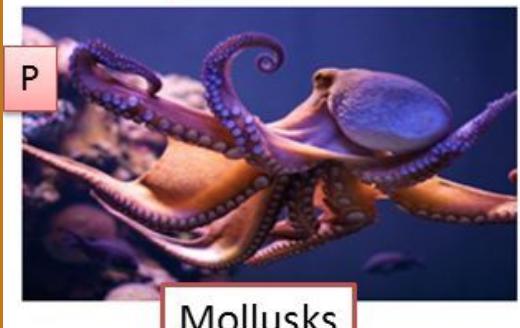
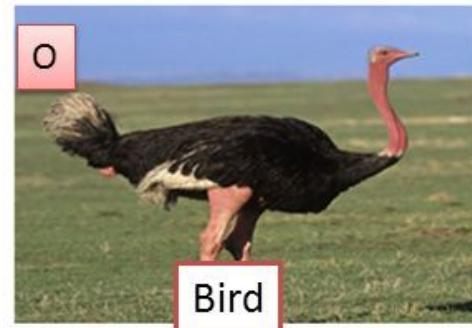
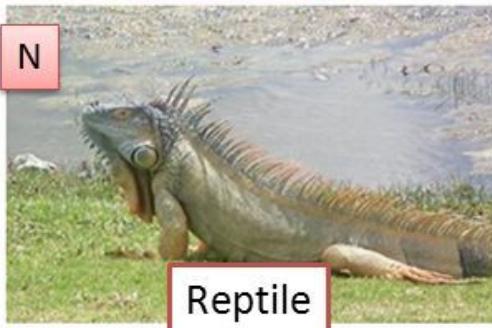




Polynomial and Logistic Regression

- Commonly polynomial & logistic
- Apply principles of linear regression to higher order relationships
- Logistic regression deals with binary dependent variables (true/false)





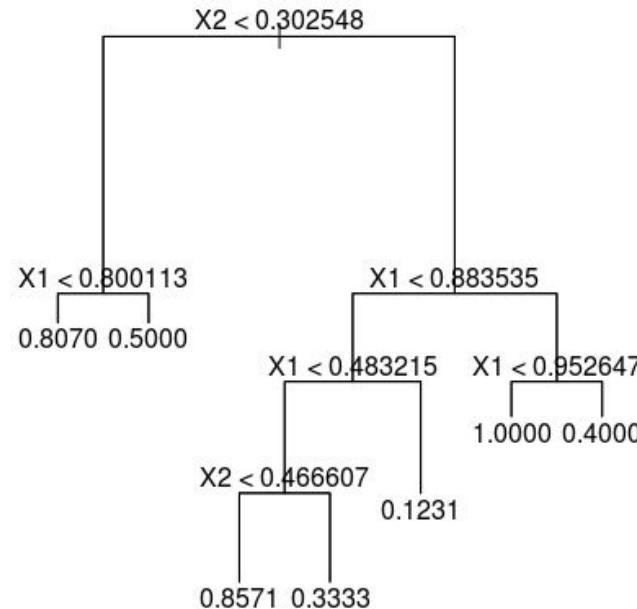
Trees and Forests





Decision Trees

- Match categorical data to classes
- Create an n-ary tree
- Try to minimise incorrect classifications
- Can also split on other functions such as information gain



$$I_G(p) = \sum_{i=1}^J p_i \sum_{k \neq i} p_k = \sum_{i=1}^J p_i(1 - p_i) = \sum_{i=1}^J (p_i - p_i^2) = \sum_{i=1}^J p_i - \sum_{i=1}^J p_i^2 = 1 - \sum_{i=1}^J p_i^2$$



CART for Decision Trees

Split on writing hand?

- $p(I | L)^2 = (1/3)^2 = 0.11$
- $P(\neg I | L)^2 = (2/3)^2 = 0.44$
- $p(I | R)^2 = (3/7)^2 = 0.18$
- $p(\neg I | R)^2 = (4/7)^2 = 0.33$

$$\text{Total } L = 1 - 0.44 - 0.11 = 0.44$$

$$\text{Total } R = 1 - 0.18 - 0.33 = 0.49$$

$$\begin{aligned} \text{Gini Index} &= (3(0.45) + 7(0.49)) / 10 \\ &= 0.48 \end{aligned}$$

Writing Hand	Multilingual	Plays Instrument
Right	No	No
Left	No	No
Right	Yes	Yes
Right	No	Yes
Right	Yes	Yes
Left	No	No
Left	Yes	Yes
Right	No	No
Right	Yes	No
Right	No	No



CART for Decision Trees

Split on multilingual?

- $p(I \mid Y)^2 = (3/4)^2 = 0.56$
- $P(\neg I \mid Y)^2 = (1/4)^2 = 0.06$
- $p(I \mid N)^2 = (1/6)^2 = 0.03$
- $p(\neg I \mid N)^2 = (5/6)^2 = 0.69$

$$\text{Total } Y = 1 - 0.56 - 0.06 = 0.38$$

$$\text{Total } N = 1 - 0.03 - 0.69 = 0.28$$

$$\begin{aligned} \text{Gini Index} &= (4(0.38) + 6(0.28)) / 10 \\ &= 0.32 \end{aligned}$$

Writing Hand	Multilingual	Plays Instrument
Right	No	No
Left	No	No
Right	Yes	Yes
Right	No	Yes
Right	Yes	Yes
Left	No	No
Left	Yes	Yes
Right	No	No
Right	Yes	No
Right	No	No



CART for Decision Trees

Gini index for writing hand was **0.48**

Gini index for multilingual was **0.32**

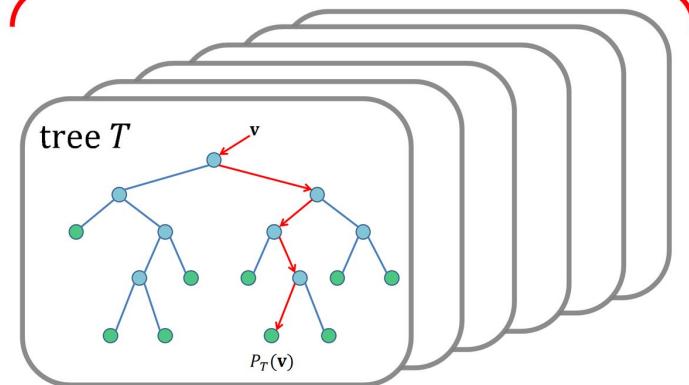
Therefore we should split by writing hand

Note: the highest possible Gini index for binary variables is

$$1 - (0.5)^2 - (0.5)^2 = 0.5$$

**Ok, having one tree is cute.
What if we had a whole *forest* of trees?**

Decision Forest



Random Forests

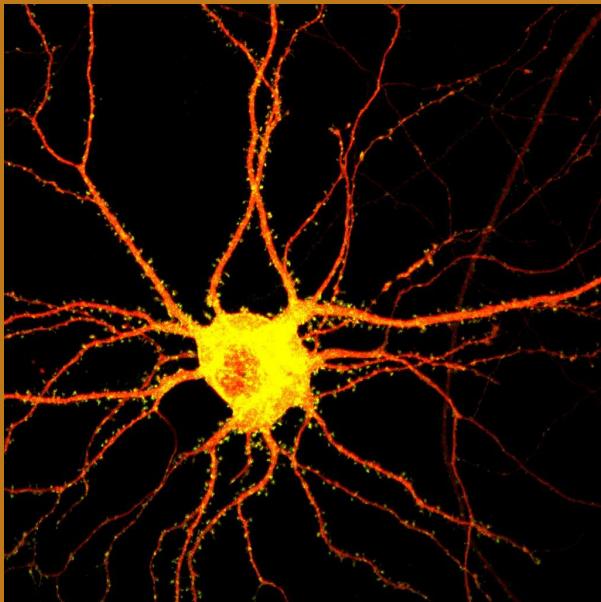
- Ensemble approach to machine learning
- Generate n trees and output the modal class
- Normally uses a technique called ‘bagging’
- Generate new training sets by sampling with replacement
- Helps with the overfitting problem



But, wait...

None of this is
DEEP LEARNING

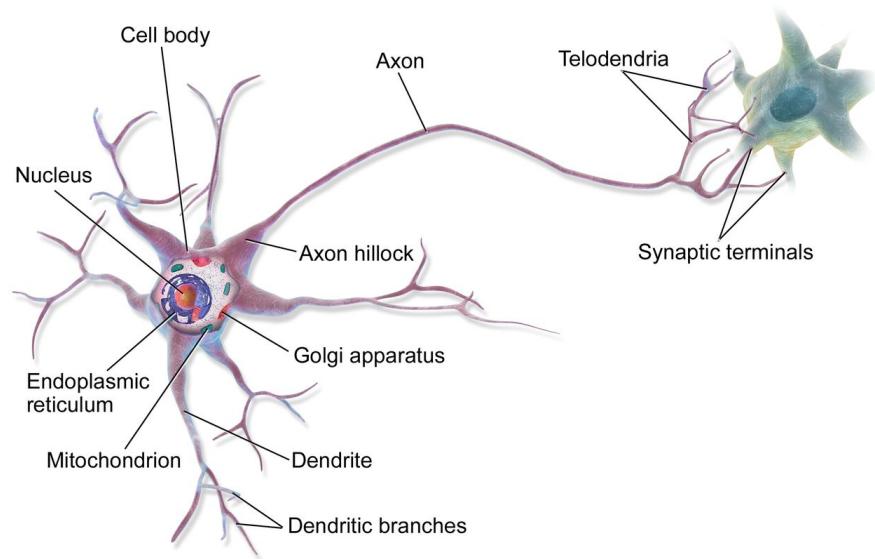
Neural Networks





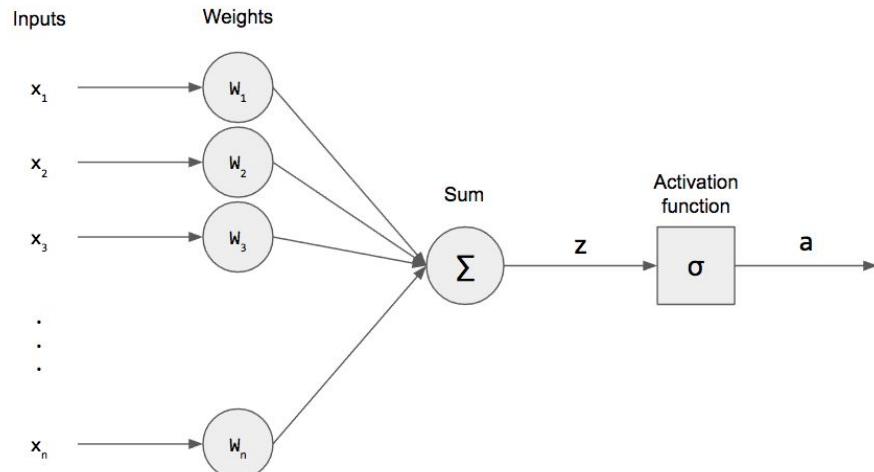
Neuroscience 101

- Basic unit of the brain is the neuron
- ~ 86bn neurons in the brain
- Modern i7 core has 2bn transistors
- About 100tn synapses
- (UK National Debt is ~£1.8tn)
- Has inputs and outputs
- “All or nothing” activation





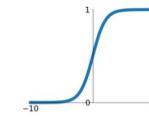
Enter the Perceptron



- Each input is weighted
- Output can vary (typically -1 to 1)
- Common activation functions include:

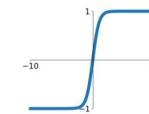
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



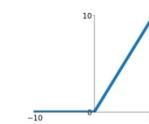
tanh

$$\tanh(x)$$



ReLU

$$\max(0, x)$$

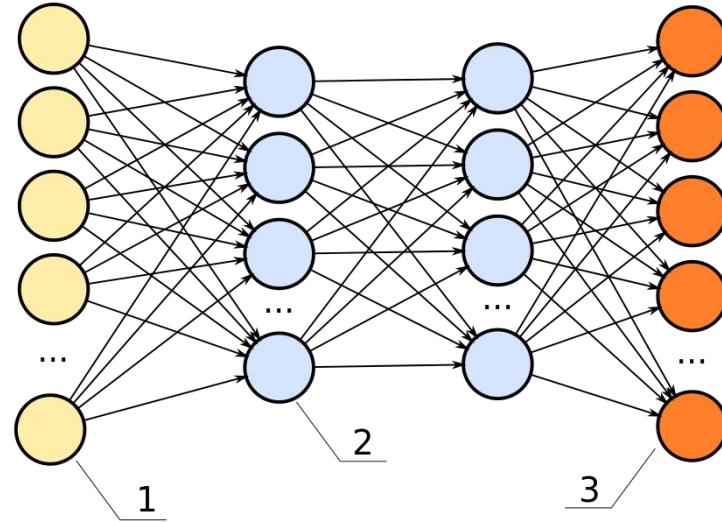


**Ok, having one perceptron is cute.
What if we had a whole *brain* of perceptrons?**



Bigger = better?

1. Input layer
2. Hidden layer(s)
3. Output layer





**How do we figure out the
weights?**

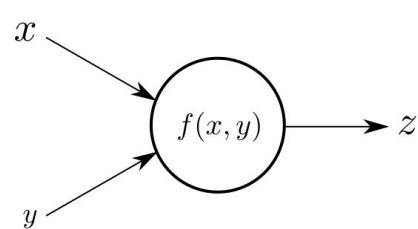


Backpropagation

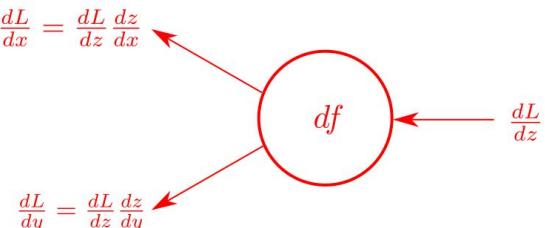
- Calculate the error from the output
- Gradient descent on weights
- How does each weight affect the output?
- Differentiate the output wrt. weights

$$Error = \sum(target - output)^2$$

Forwardpass

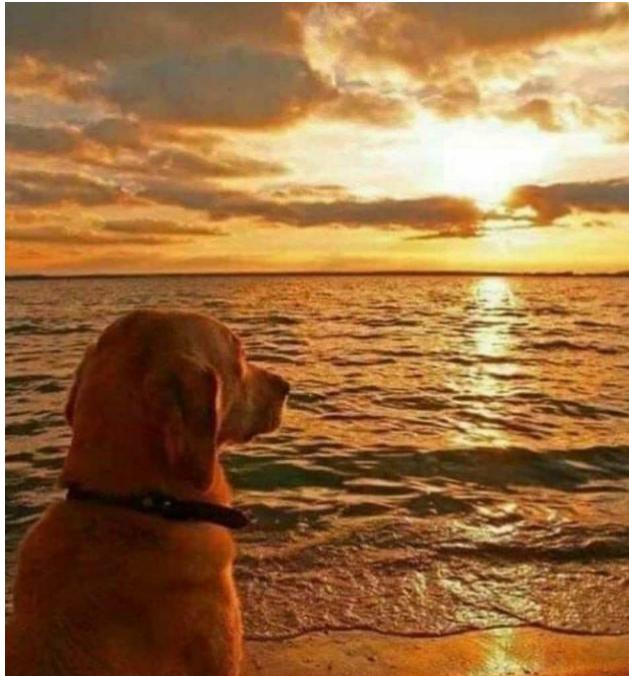


Backwardpass





“Deep” learning



- TLDR; deep = more layers
- NNs have a depth (CAP)
- CAP = 2 theoretically universal
- CAP > 10 “very deep” learning
- $(\cup \circ \square^\circ)^\cup \curvearrowleft \underline{\text{L}} \underline{\text{L}}$



Lab session: Introduction to R



**A disclaimer (and apology)
about using R...**



Loading Packages and Importing Data

```
library("caret")
```

- Load the `caret` package

```
ds = read.csv("filename.csv")
```

- Load the data set in the file
`filename.csv`



Creating Training and Test Data Sets

```
index = createDataPartition(ds$feature, p=0.75, list=FALSE)
```

```
training = ds[index,]
```

```
test = ds[-index,]
```

- Split the dataset `ds` into new datasets containing 75% and 25% of the data
- Make sure that each set has roughly equal proportions of possible `ds$feature` values



Creating and Using a Model

```
model = train(training[,columns], training$target), method = 'type')
```

```
predictions = predict(object=model, test[,columns])
```

- Train a **model** of a certain **type** on the **columns** and **target** feature of the **training** data set

Challenge (for after the break)

Build a Predictive Policing Classifier

Real data from ~7,000 arrests

Predict who will reoffend within 2 years