Introduction

Presenters and Course Goals

Introduction and Goals

Rob Barton, CCIE #6660 (R&S and Security), CCDE 2013::6 is a Distinguished Systems Engineer working in Cisco's Digital Transformation and Innovation group. Rob is a registered Professional Engineer (P. Eng) and has worked in the IT industry for over 22 years, the last 19 of which have been at Cisco. Rob Graduated from the University of British Columbia with a degree in Engineering Physics specializing in computer engineering. Rob is a Cisco Press published author, with titles including QoS, Wireless, IoT, and Machine Learning and Data Analytics. Rob is a regular speaker at global IT the largest IT conferences and has published extensively, co-authoring papers on AI, data analytics, and IoT. Rob’s current areas of work include wireless communications, IoT, and AI/ML in networking systems. Rob also holds several patents in these areas.

Introduction and Goals

Jerome Henry is Principal Engineer in the Enterprise Infrastructure and Solutions Group at Cisco systems. Jerome has more than 15 years experience teaching technical Cisco courses in more than 15 different countries and 4 different languages, to audiences ranging from Bachelor degree students to networking professionals and Cisco internal system engineers. He is certified wireless networking expert (CWNE #45), CCIE Wireless (#24750), CCNP Wireless, developed several Cisco courses focusing on wireless topics and authored several books and video courses on Wireless, IoT and networking. Jerome is also an IEEE member, where he was elevated to the grade of Senior Member in 2013, and also participates to Wi-Fi Alliance working groups. With more than 10000 hours in the classroom, Jerome was awarded the IT Training Award best Instructor silver medal, and CiscoLive Speaker Hall of Fame.

Pearson

Lesson 1: Introduction

1.1 An Introduction to Machine Learning and Data Analytics

AI Is Redefining Our Way of Life

“Alexa, order pizza!”

“Start engine, unlock doors.”

“Where can I find an 12’ ladder?”

Enabling machines to do what we formerly thought only humans could do

But what about AI in the enterprise?

Amazon

Lights-out Warehouses

20% improved operating expenses

Macys

Unparalleled Experience

Virtual concierge answers customer questions

PayPal

Secure Enterprise

Fraud detection algorithms protect customer’s digital transactions

Netflix

Customer Retention

80% subscriber video choices from recommendation engines

ML/AI Application Examples

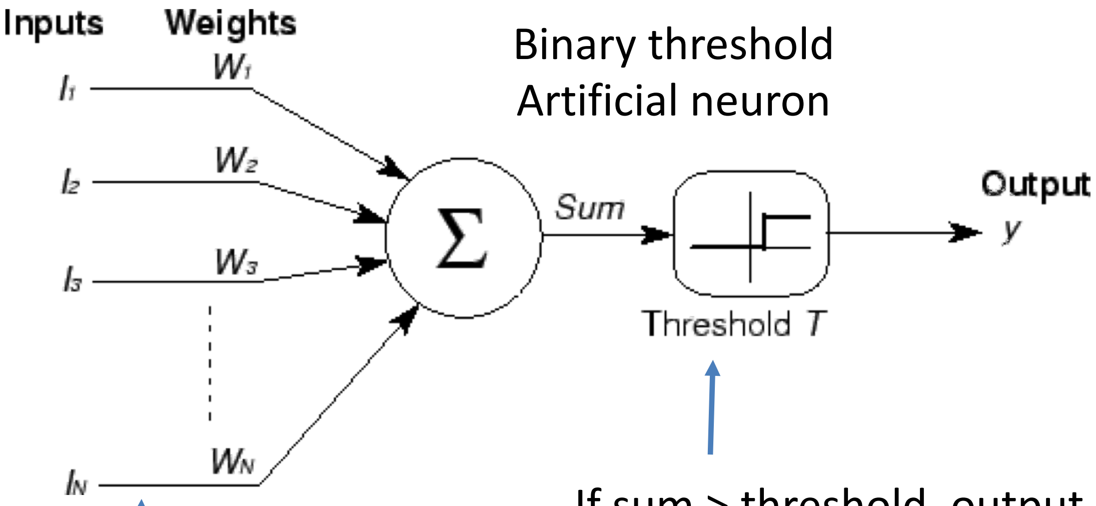
Lip Reading | Deep Blue | IBM Watson | Deep Mind | Play Music | Netflix

Counting people | Recognition | NLP Translation | Self-driving | Networks

And Many more: CRM, Healthcare, Personal Assistants, ….

A Little History . . .

McCullogh & Pitts - 1943



Binary threshold
Artificial neuron

If sum > threshold, output 1
Output 0 otherwise

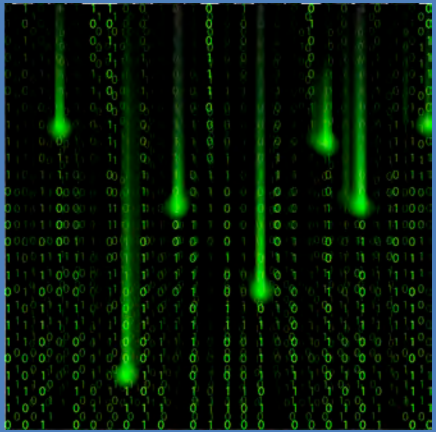
Weighted inputs

In the 1950s and 1960s, Principles of Neurodynamics were examined and Symbolic ML expanded

When I show these shapes to the camera

This IBM 704 computer can say “it’s a triangle”

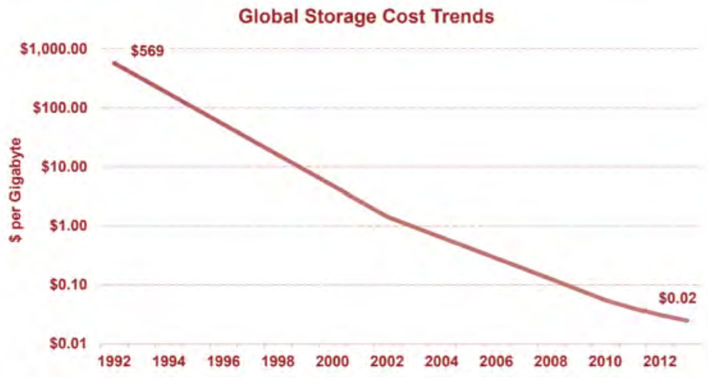
Enter the 1970s – the Winter of Machine Learning



ML/AI Isn’t What We Thought it was Going to be a few years ago . . .

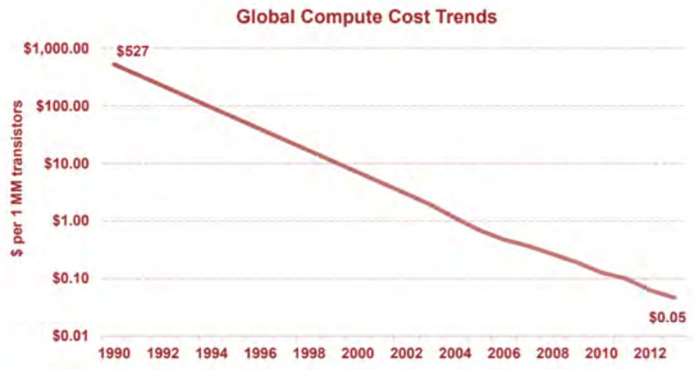
Why Is ML Emerging Now?

Cheap data storage…



The decrease in storage cost has led to the emergence of Big Data

…and cheap processing



Training ML takes a lot of compute power, which has become much cheaper through cloud computing and GPUs

Other key factors include:

- Mathematical advances for training of NNs
- Abundance of data

Sources: Domingos, Ibid.; Mary Meeker, “Internet Trends 2014”, Kleiner Perkins Caufield Byers, 28May2014

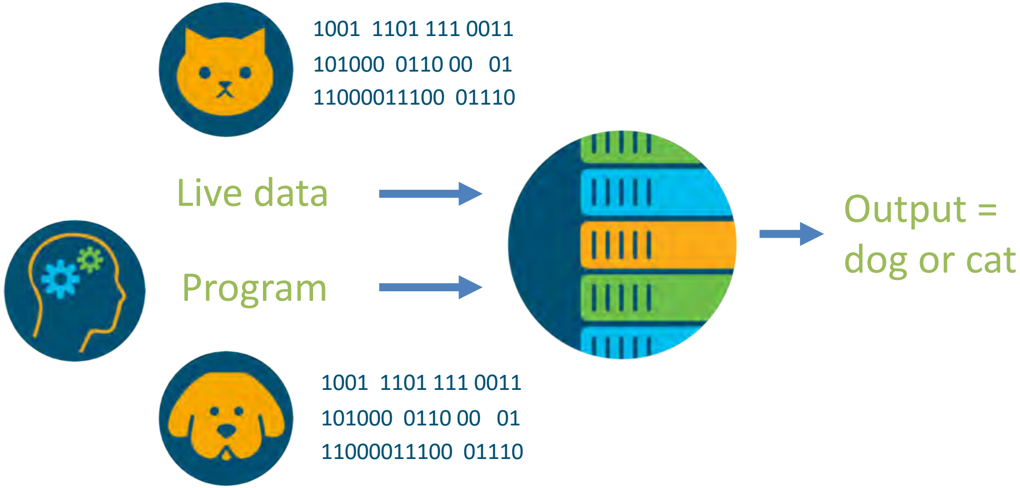
Lesson 1: Machine Learning Overview

1.2 Understanding the Machine Learning Landscape

Artificial Intelligence (AI)

- Artificial Intelligence (AI) is a very generic term which refers to teaching a machine to imitate human behavior

Question: Do “If-then-else” statements explicitly programmed by humans count as AI?



The Magic of Machine Learning

ML uses models to tackle new issues in real time without human intervention

Machine Learning Definitions

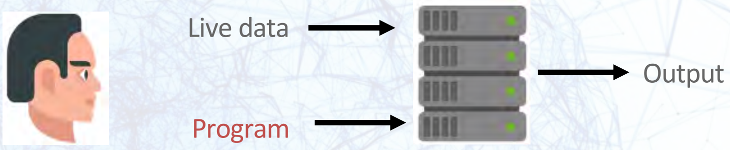
Arthur Samuel (1959)

Field of study that gives computers the ability to learn without being explicitly programmed

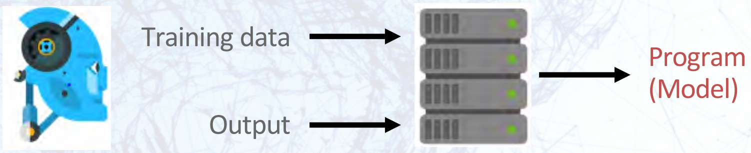
Tom Mitchell (1997)

A computer program is said to learn if its performance at a task T, as measured by a performance P, improves with experience E

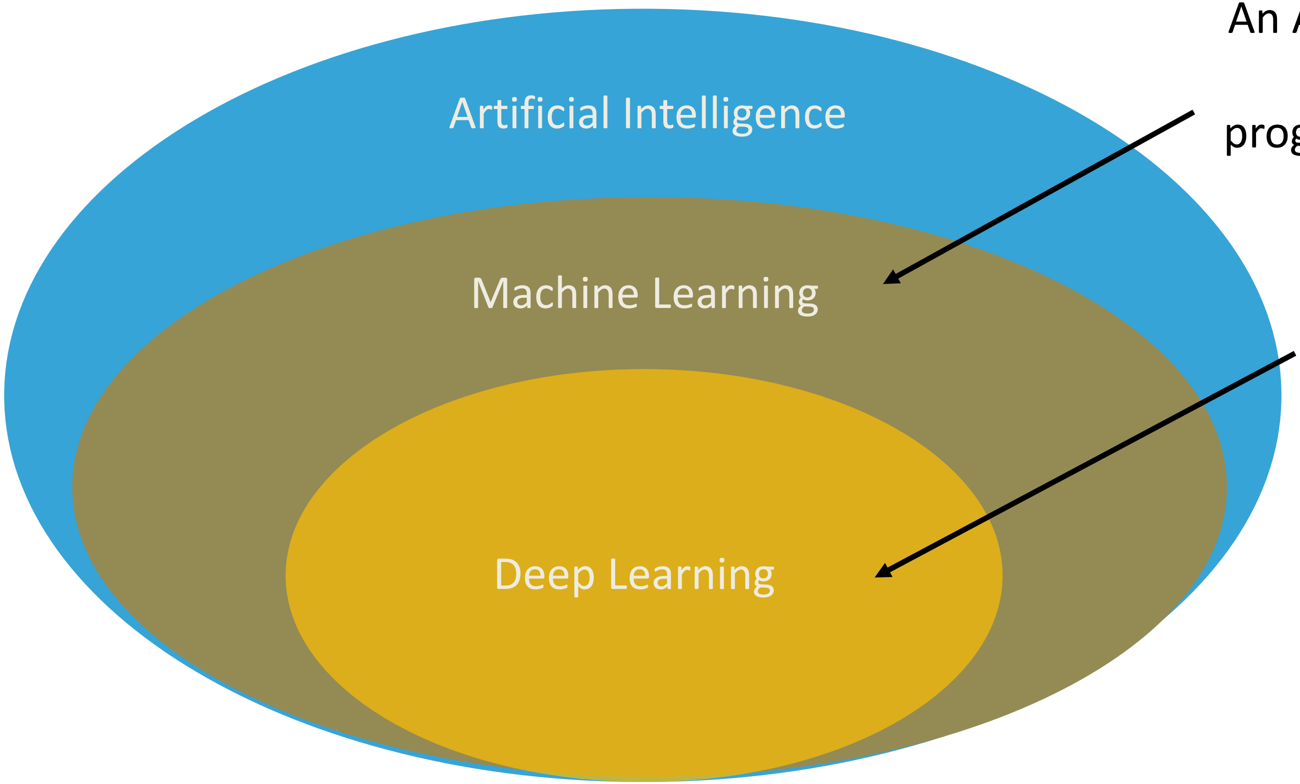
Traditional, Rules-based



Machine Learning / AI



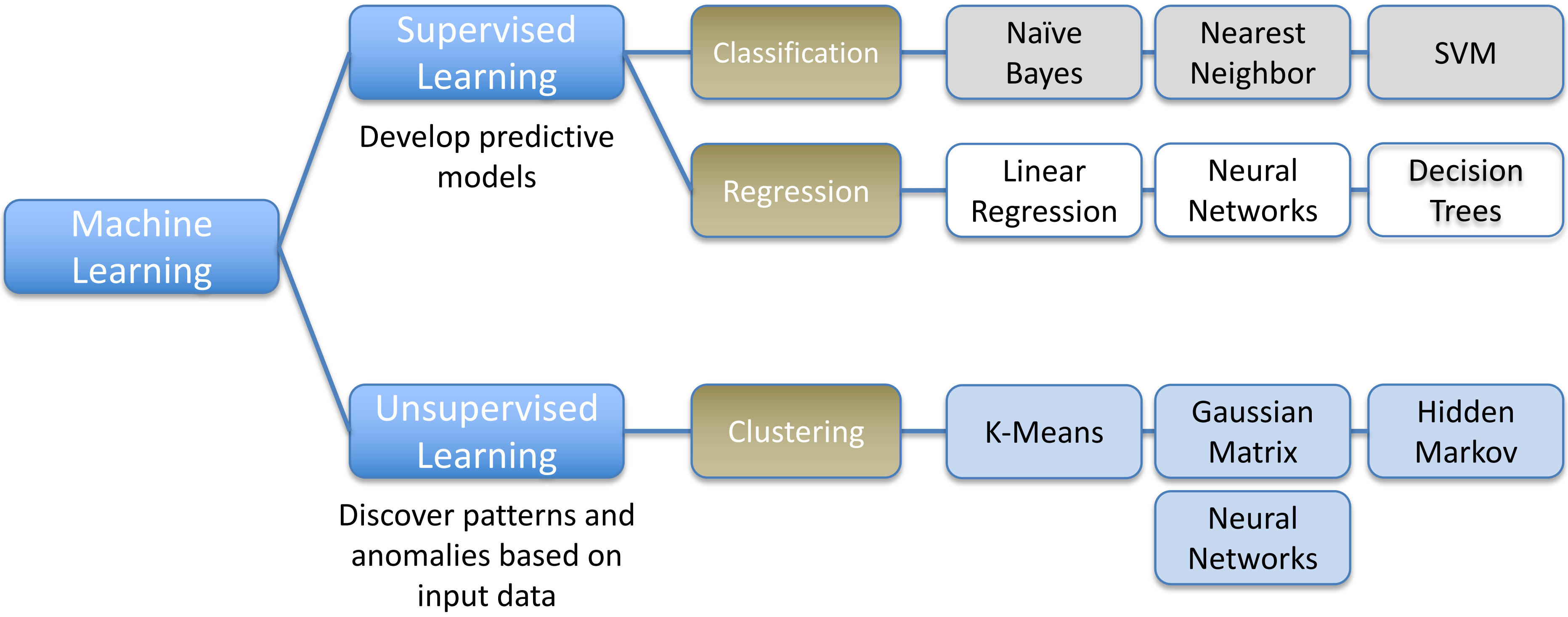
Comparing AI, ML, and Deep Learning



An AI technology where all the rules are not set in the program, but are learned while the program is used

A form of ML that uses Neural Learning Nets

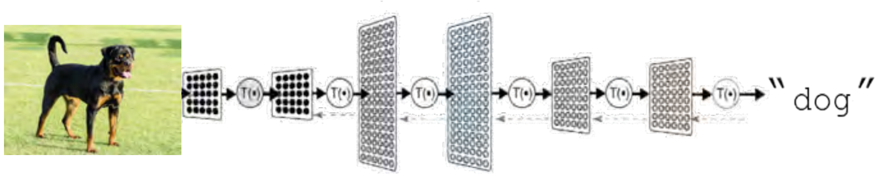
ML Is a Complex Landscape



The Main Methods – Supervised Learning

Supervised Learning

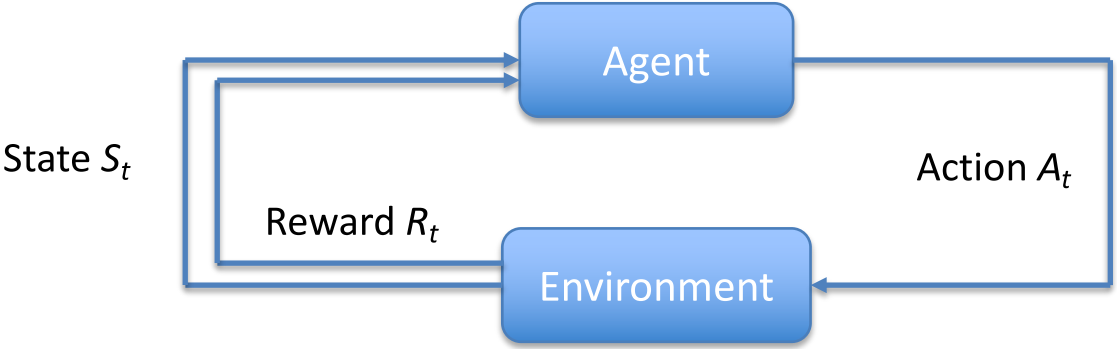
Learning with a labeled training set



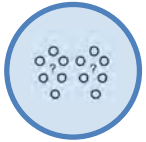
- When you have a lot of historical data and can train a mathematical model to predict future events
- Good at mapping A → B

The Main Methods – Reinforcement Learning

- A semi-supervised learning model
- No training data or correct/incorrect guidance needs to be given.
- Involves behavioural psychology
- In summary, a lot of trial and error

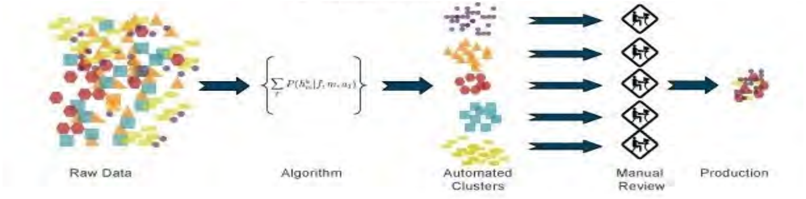


The Main Methods – Unsupervised Learning



Unsupervised Learning

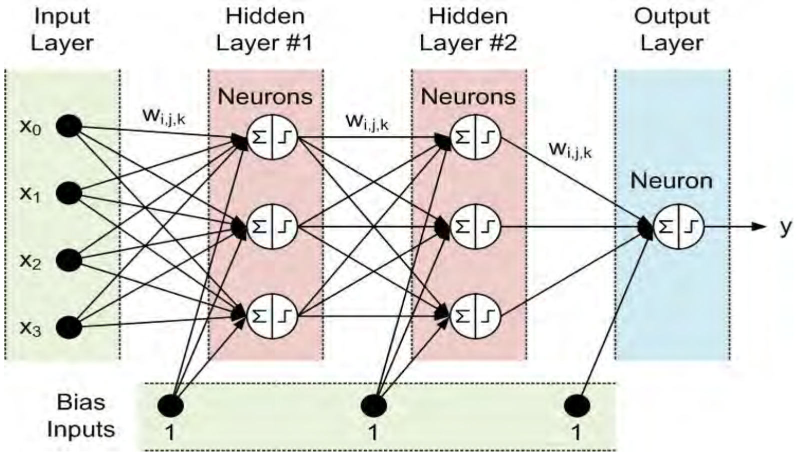
Discovering patterns in unlabeled data



- When you have a lot of data, but you're not sure what the patters are
- Very good a clustering and finding anomalies

Deep Learning (Neural Networks)

- A type of machine learning that mimics the way the brain processes information
- Involves parallel processing of data
- Breaks complex ML jobs into smaller ones
- Works great with hardware acceleration technologies like GPUs

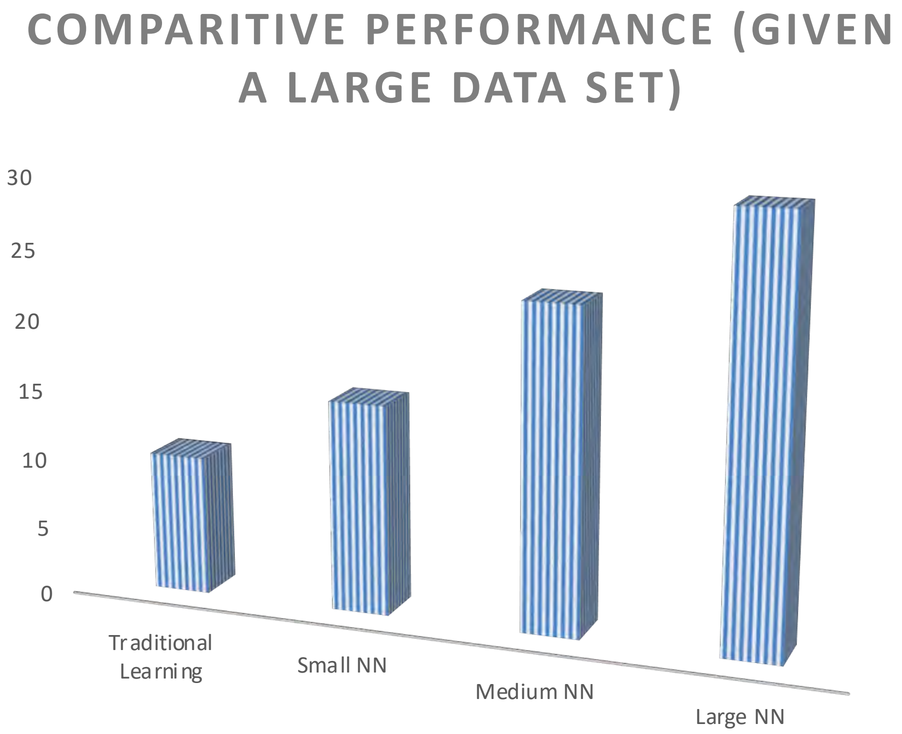


Geoffrey Hinton

The “Godfather of Deep Learning” and Neural Networks

(Photo by Johnny Guatto / University of Toronto)

Performance Comparisons



The Main Methods – Statistical Models

Bayesian Inference

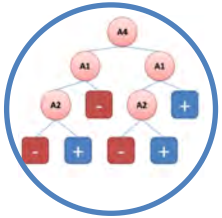
How confident are you in the result?

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- A method of statistical learning – using a small amount of historical data and combining it with new data
- Often used as a “quick and dirty” method of implementing a Machine Learning algorithm
- Naive Bayes

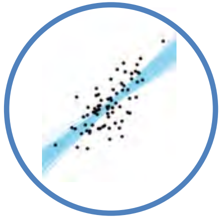
An Algorithm for Every Problem!

- Very Active Communities … Many Algorithms Available
- A tool for every problem!



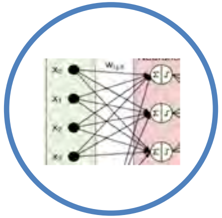
Classification

Clustering



Regression

Inference

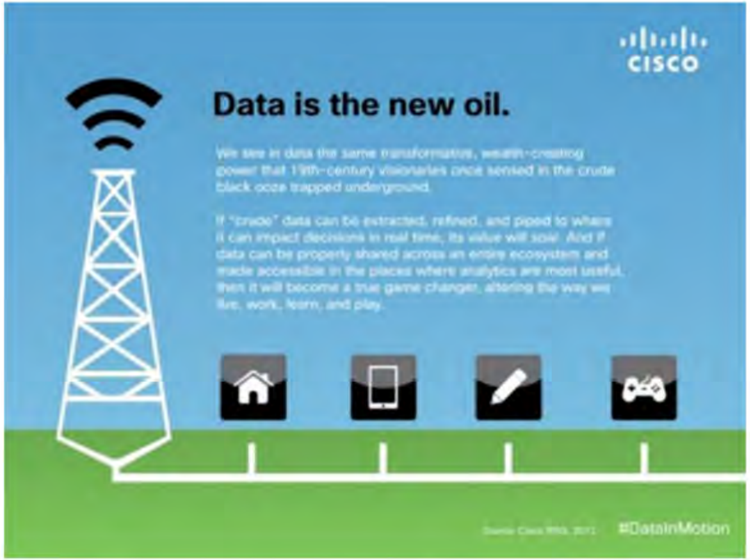


Neural Networks

However . . . .

The methods often have dependence on domain experts

Machine Learning / AI Require Data



- Data has become the most valuable commodity in the world
- Only useful if you can get to it, refine it, move it and then interpret it correctly

Data is the rocket fuel of Machine Learning – Dr. Andrew Ng

Lesson 2: Big Data Analytics 101

2.1 Understanding Data

What is Data?

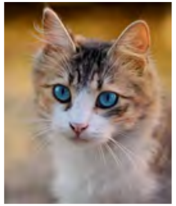
Cost of a Used Car		
Age of Car	**Miles / Km**	**Cost of Car**
1.5	17,000	$23,000
3	42,000	$18,500
10	192,231	$4,700
10	175,023	$5,200

$\underbrace{\text{Age of Car, Miles / Km}}_{A}$ $\underbrace{\text{Cost of Car}}_{B}$

Create a Mapping
A → B

Requires that we know the field of data that we are looking at

Labeling Data

Image	Label
	Cat
	Wolf
	Giraffe

Data must be correctly labeled before it can be analyzed or ML algorithms applied

Storing Data in a Relational Database Model

Customer			
Name	**ID**	**Phone**	**Email**
Irene	14213	408-923-1242	irene@gmail.com
Walter	62342	514-231-2315	walter@yahoo.com
Miyuki	12344	416-231-2341	miyuki@icloud.com

Bank Account	
Account Number	**Balance**
234123	$1,232
423142	$5,231
521231	$50

Transactions	
Credit	**Debit**
$123	
	$611
	$512

Comparing Structured vs. Unstructured Data

Structured Data		Unstructured Data
Numbers, dates, etc.		Images, videos, measurements
Traditional RDBMS		Big Data System
Smaller Storage Req'd		Requires Massive Storage
Displayed in Tables		Not displayed in tables

Lesson 2: Big Data Analytics 101

2.2 Data Messaging Systems

The Data Management Lifecycle

To analyze data, a process is required to get it ready for analysis

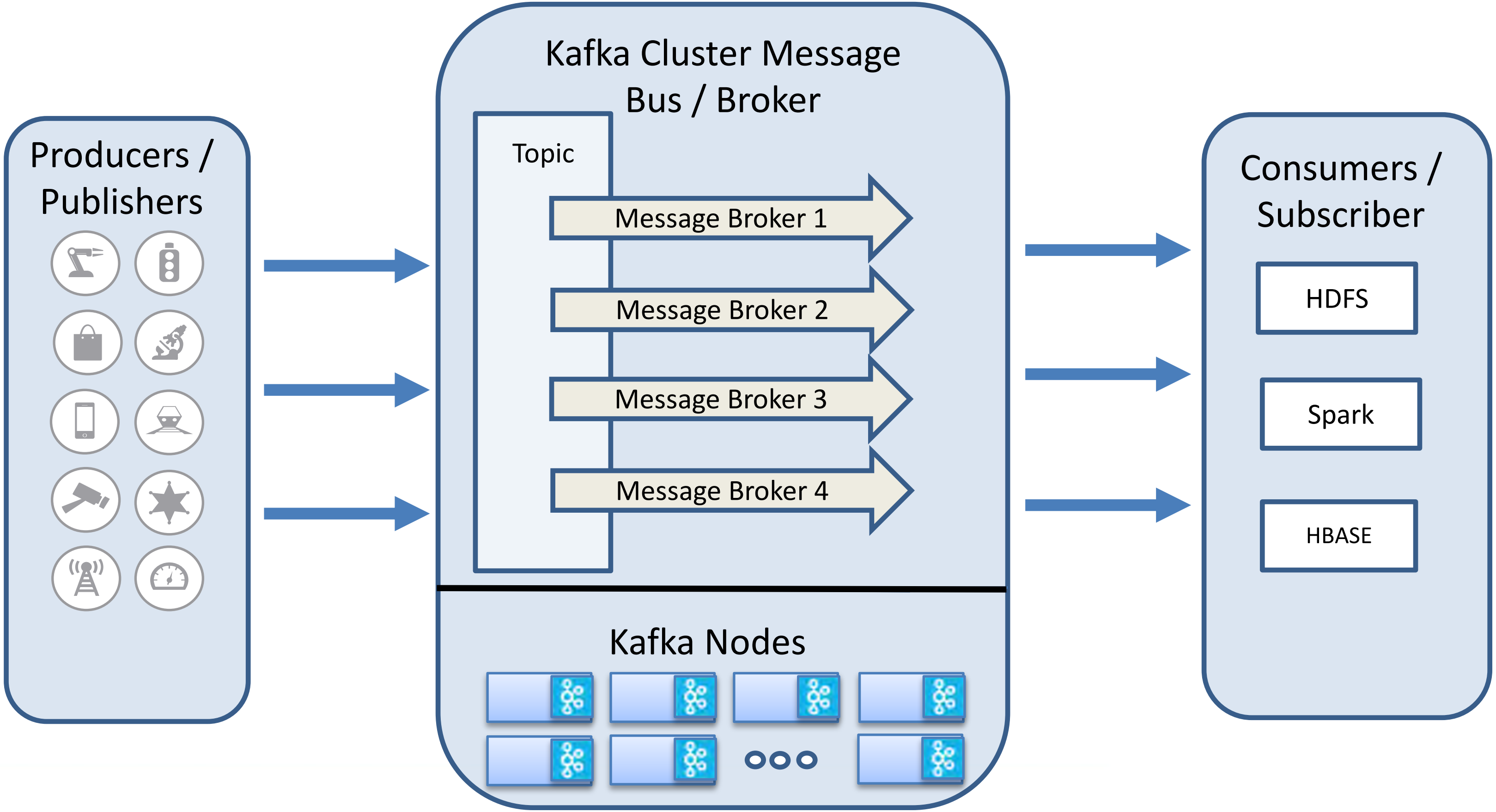
Connecting to Data → Data Preparation / Normalization → Analyze Data (AI/ML, Statistical Models) → Reporting, Visualization or Alerts

Data Messaging Systems

- **Question**: How does data get into the data lake / data warehouse?
- **Answer:** A Data Messaging / Brokering System is used

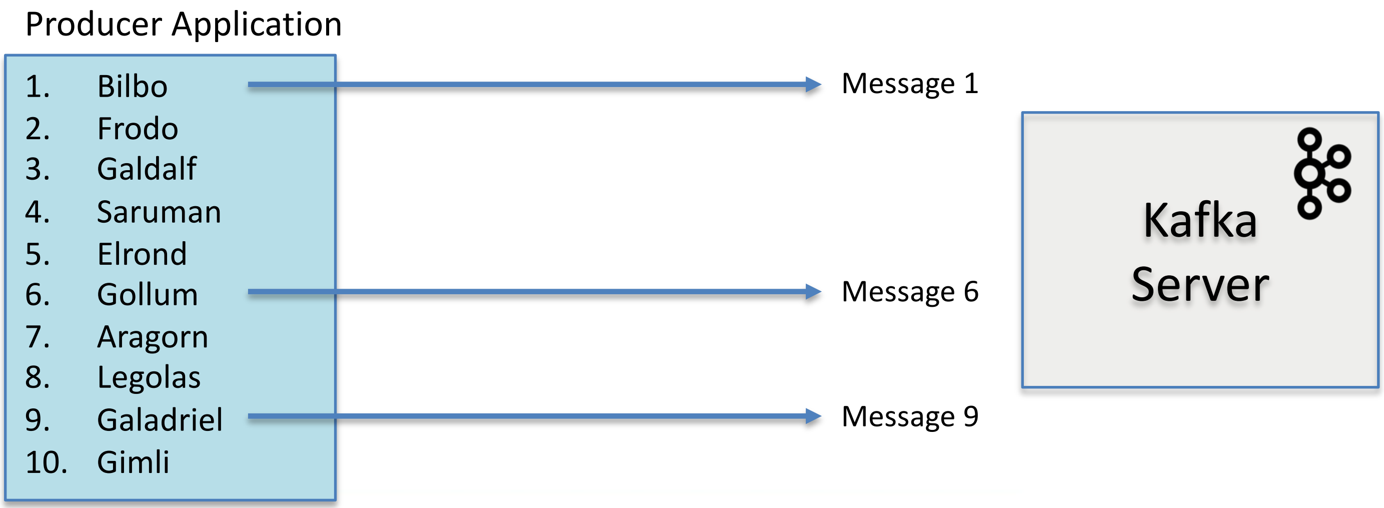
- A Message Broker is a intermediary platform that processes data messages between two devices (a publisher and a subscriber)
- Send data to multiple applications (clients) at the same time
- Queueing and buffering for later delivery

Apache Kafka Data Flow



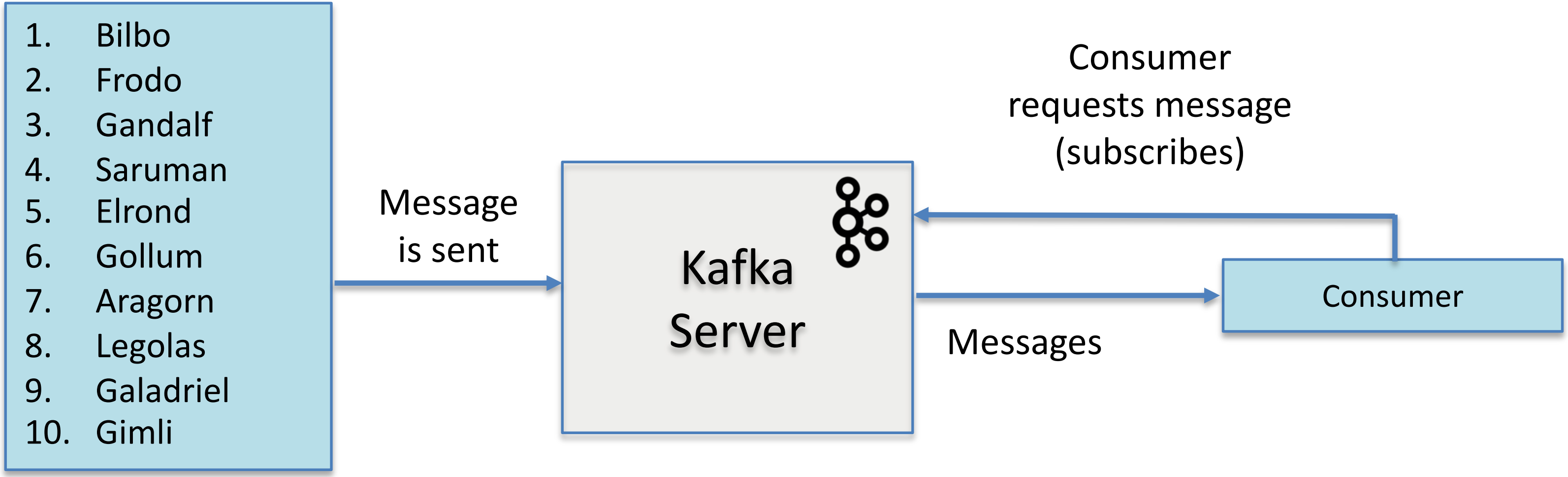
Producers

- Sends the message to broker (Kafka)
- Small to medium size of data – for example, if you want to send a file to Kafka, it will send one line at a time



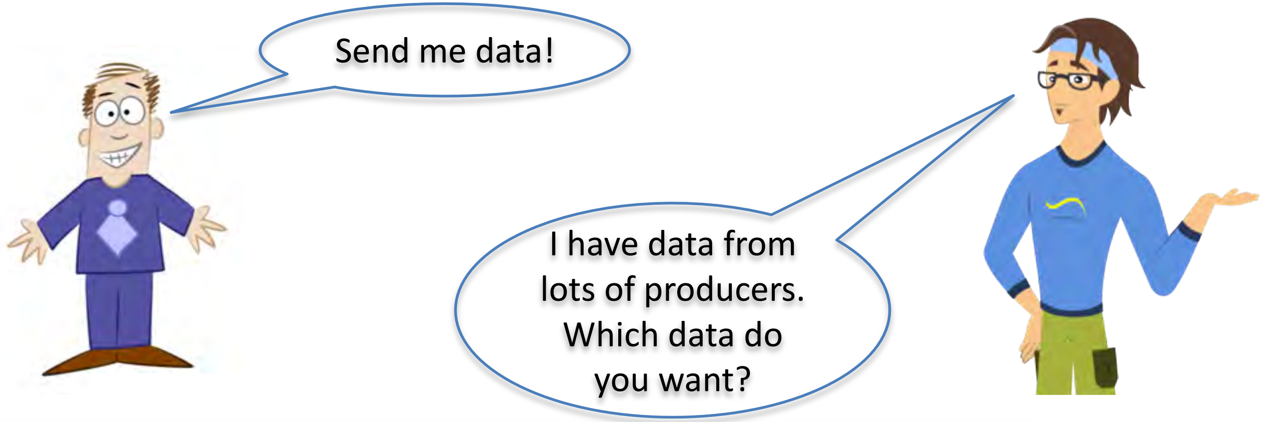
Consumers

- The consumer subscribes to broker
- Anyone can read the data off the broker



Broker Topics

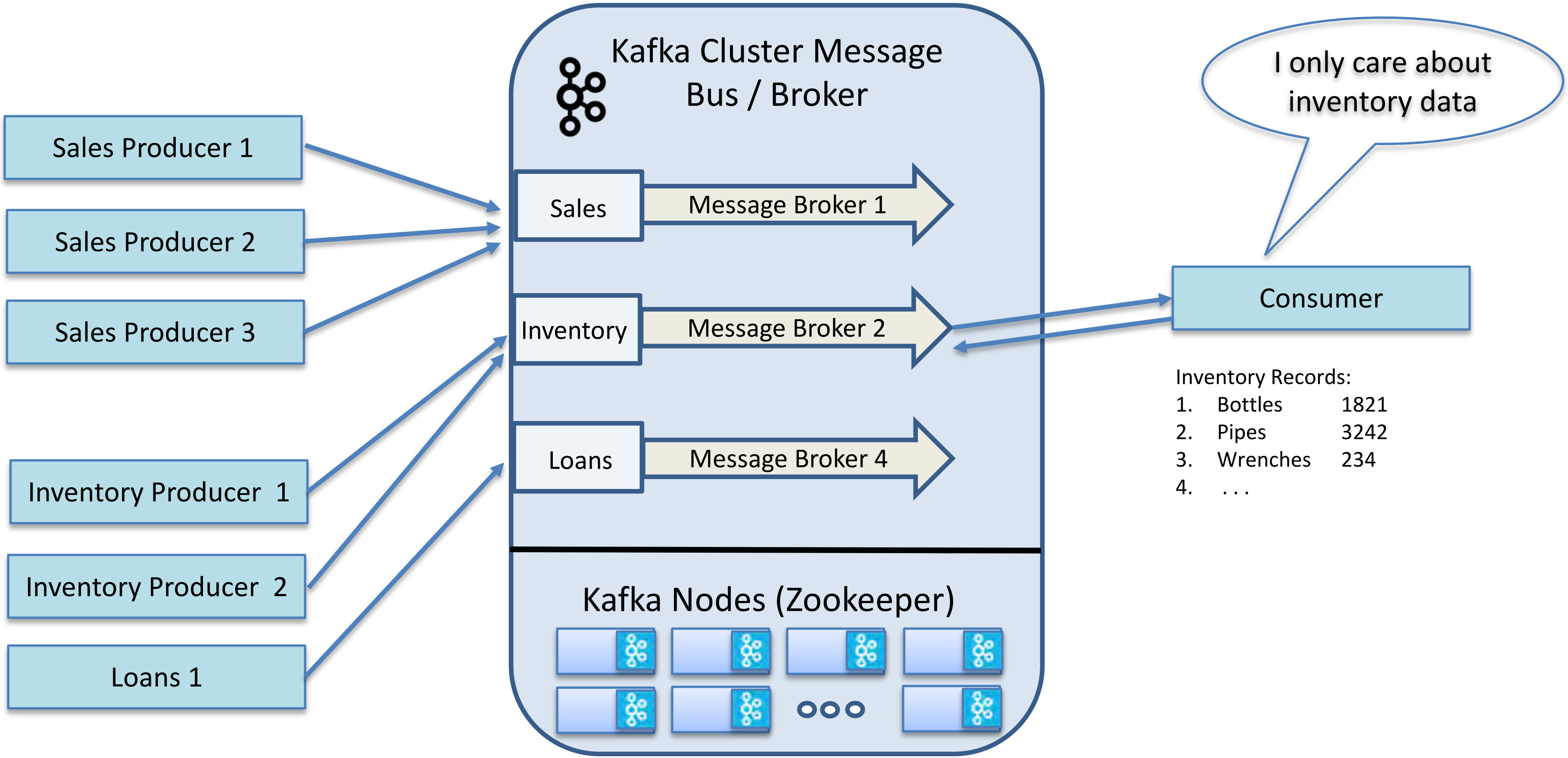
- How does the Consumer know which data to ask for from the Kafka Cluster?
- There might be multiple data from multiple producers.



Broker Topics

- We need to have an identification mechanism
- A Topic is a unique but arbitrary name given to a data set (a Kafka Stream)
- Think of this like "channel" you tune in to receive data message

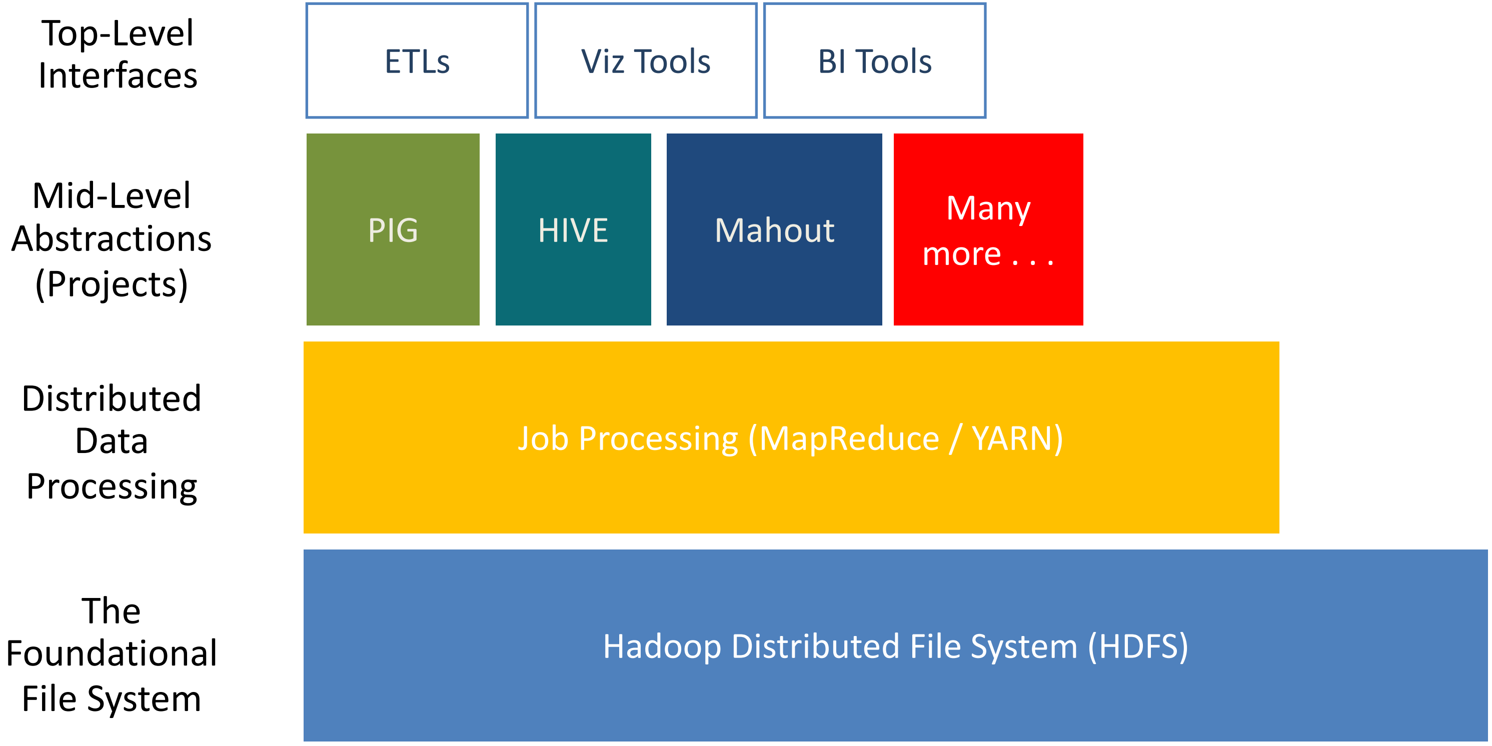
Subscribing to Topics



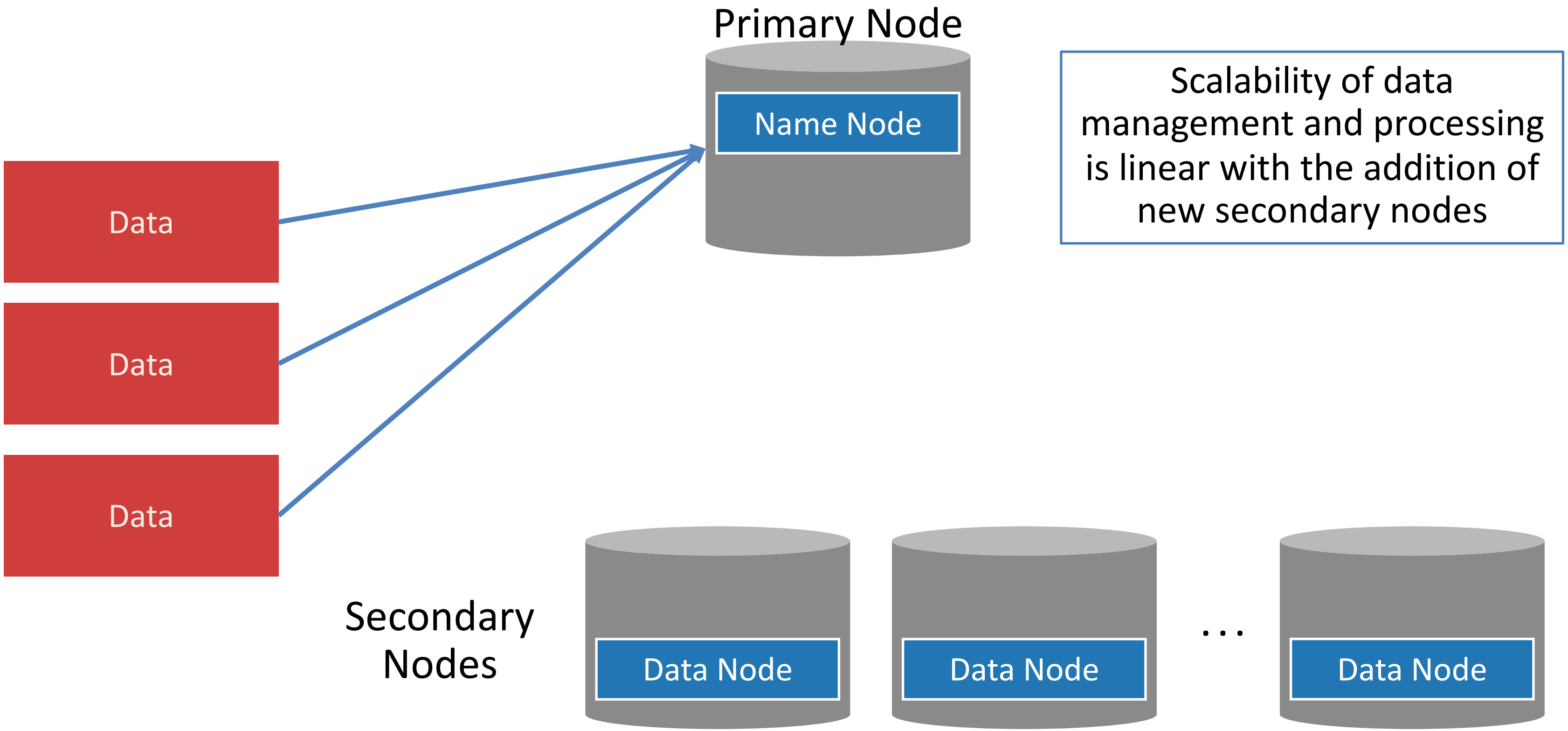
Lesson 2: Big Data Analytics 101

2.3 Principles of Distributed Data Analytics

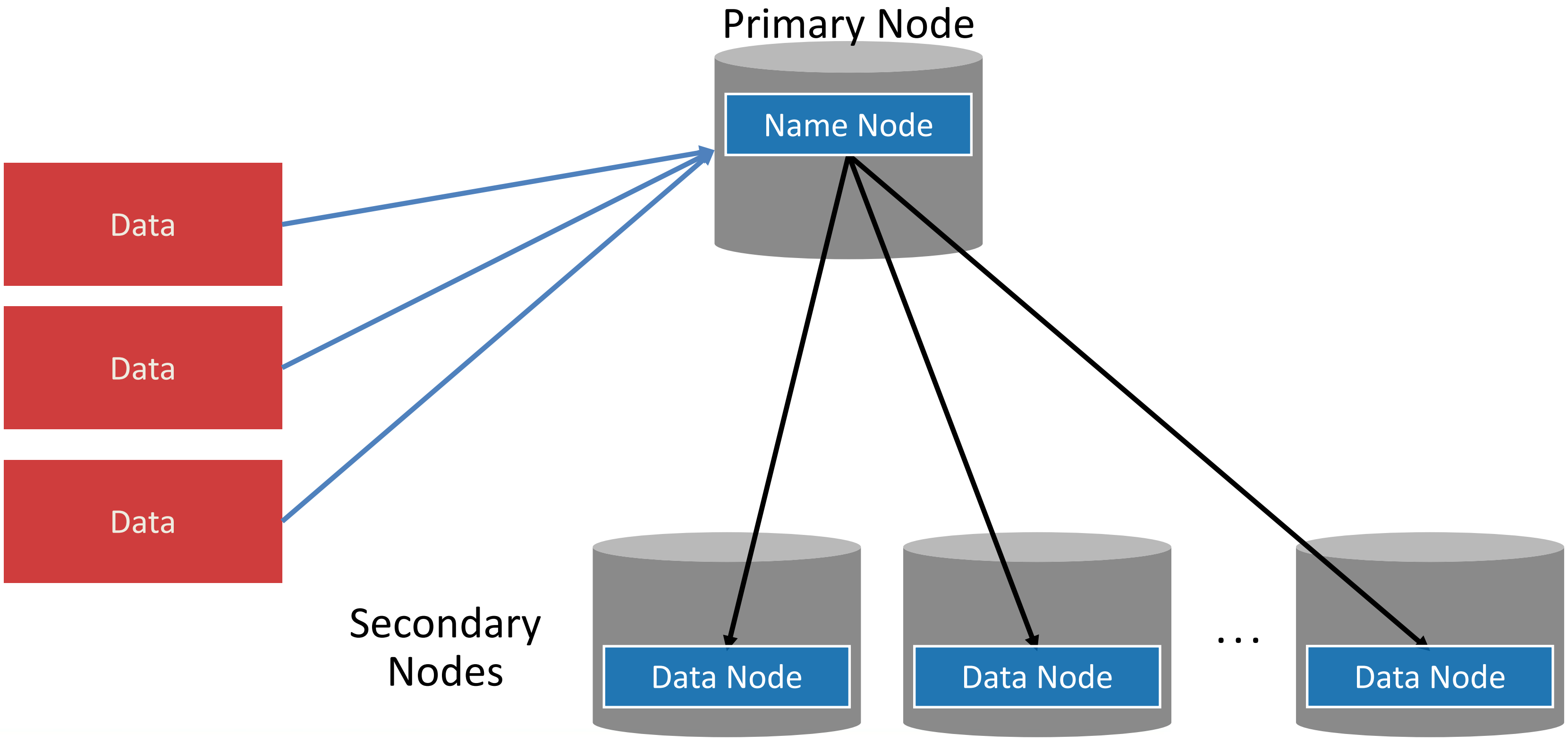
The Hadoop Stack



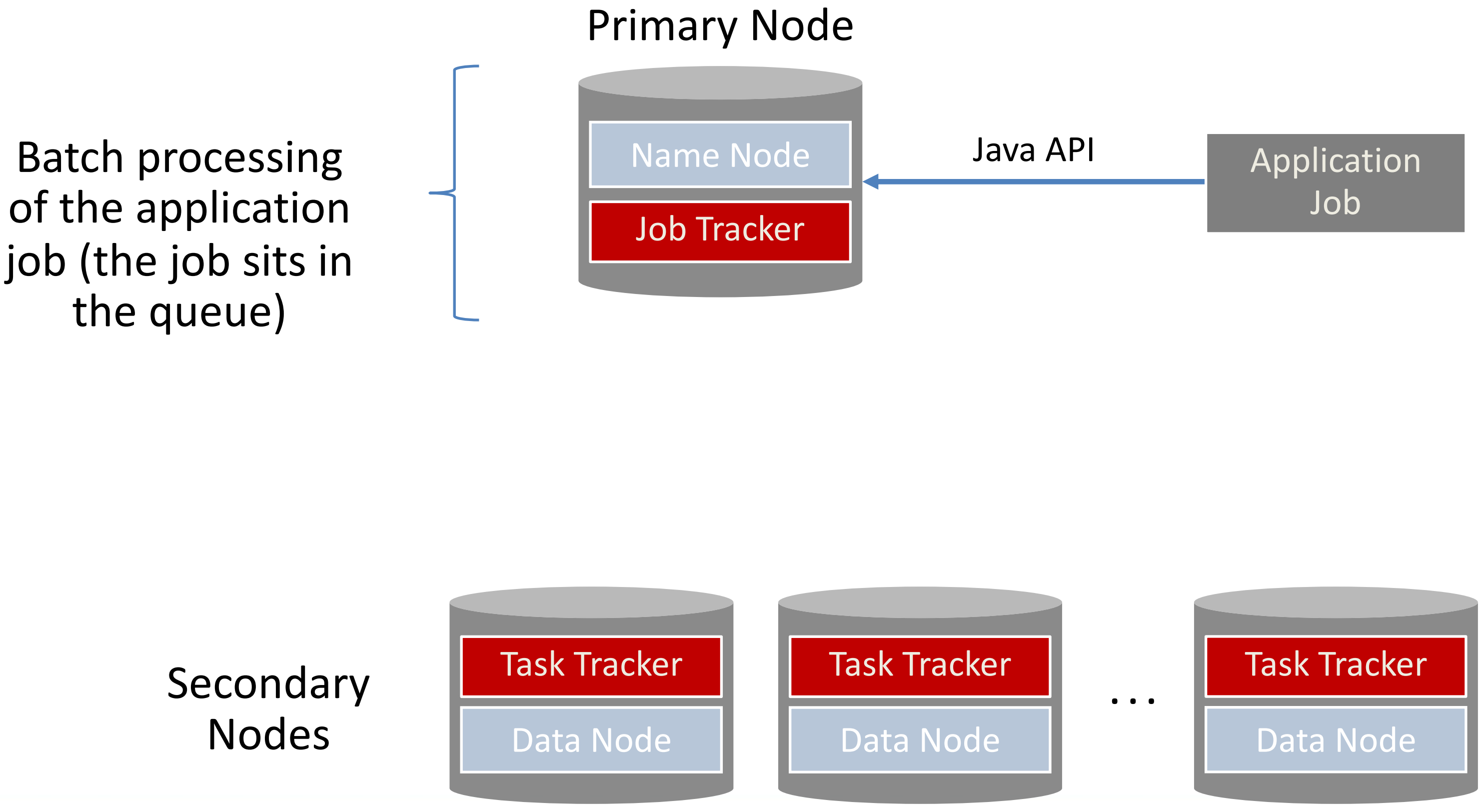
Basics of the Hadoop File System (HDFS)



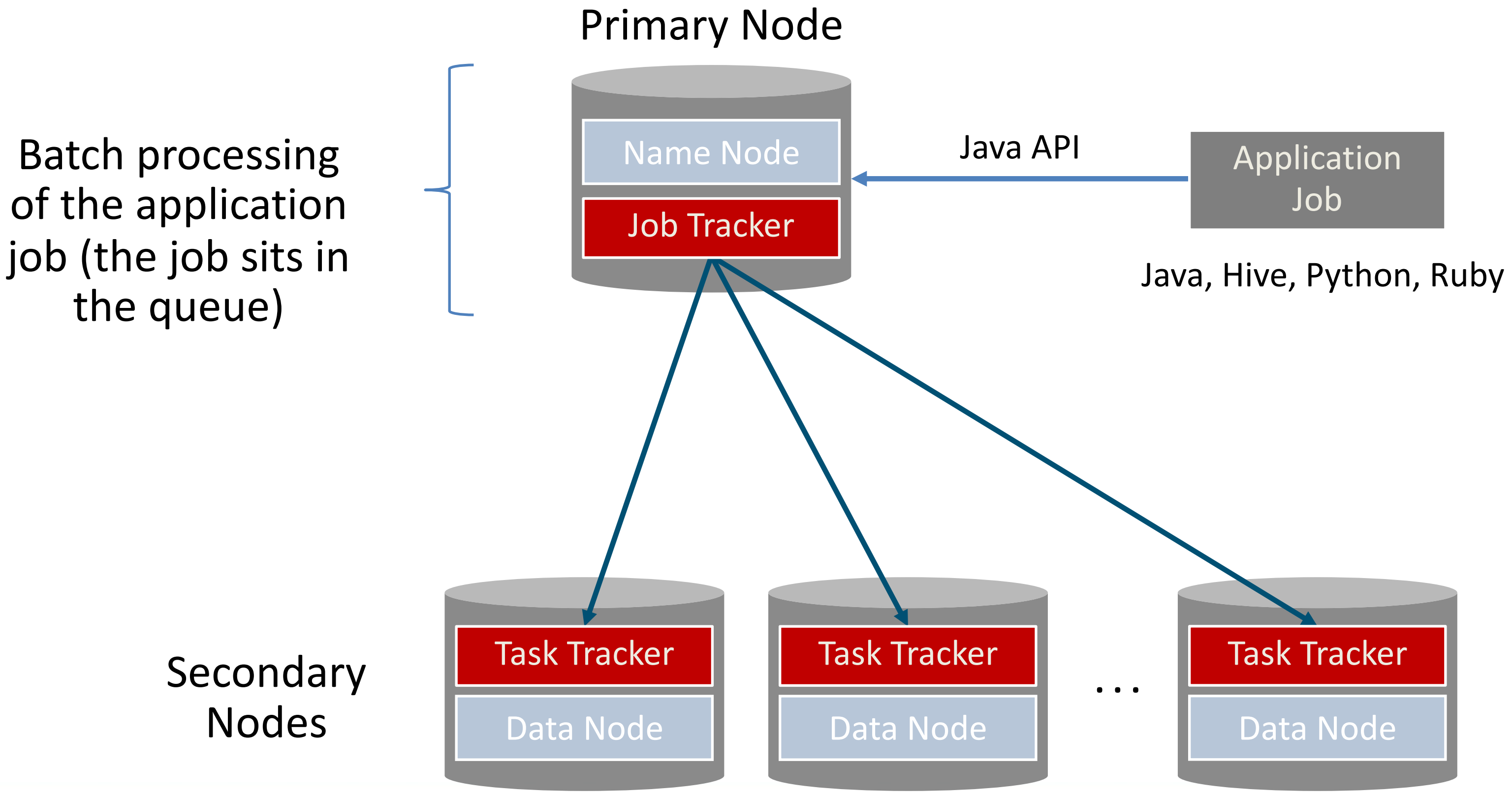
Basics of the Hadoop File System (HDFS)



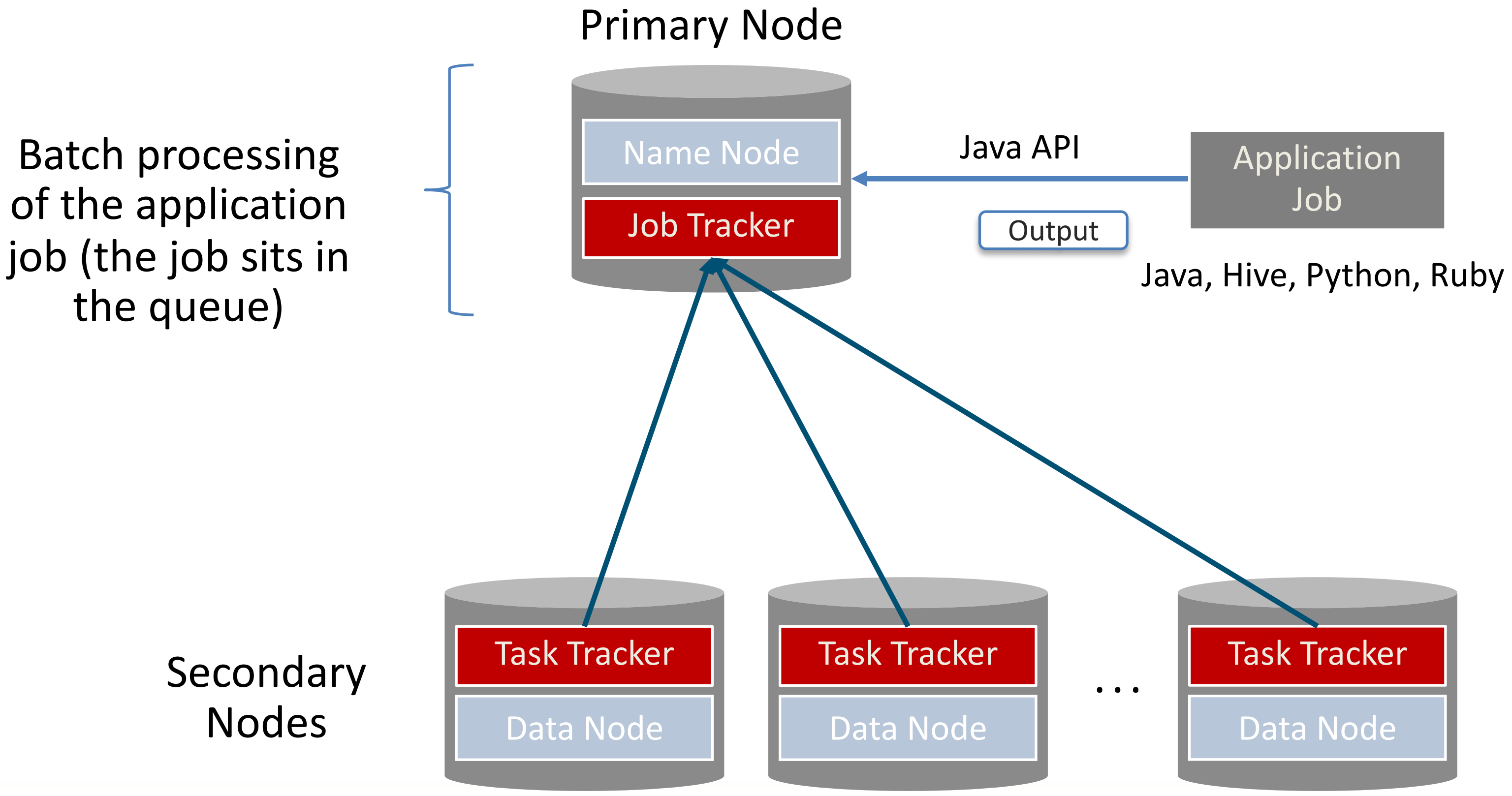
MapReduce: Job Trackers and Task Trackers



MapReduce: Job Trackers and Task Trackers



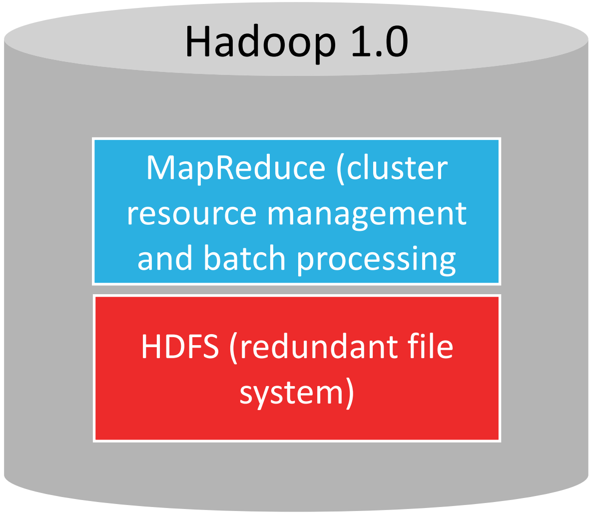
MapReduce: Job Trackers and Task Trackers



Lesson 2: Big Data Analytics 101

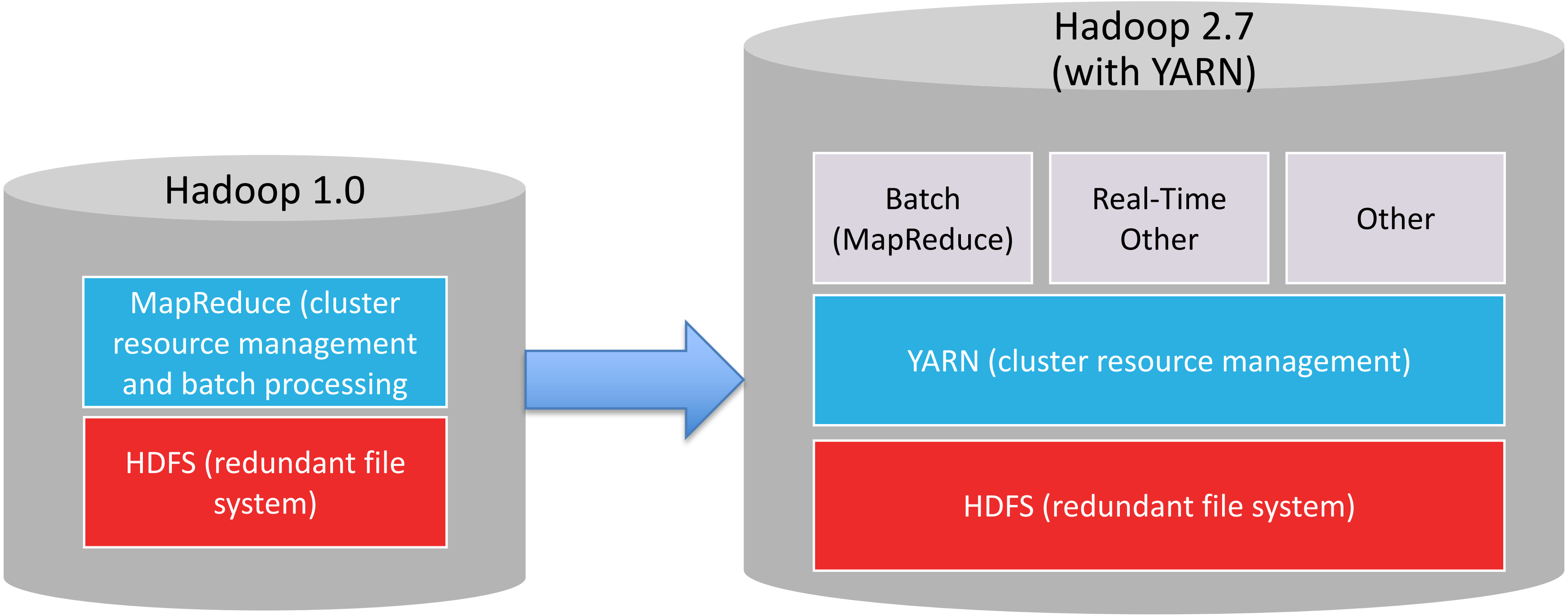
2.4 Exploring YARN

Limitations of Hadoop 1.0

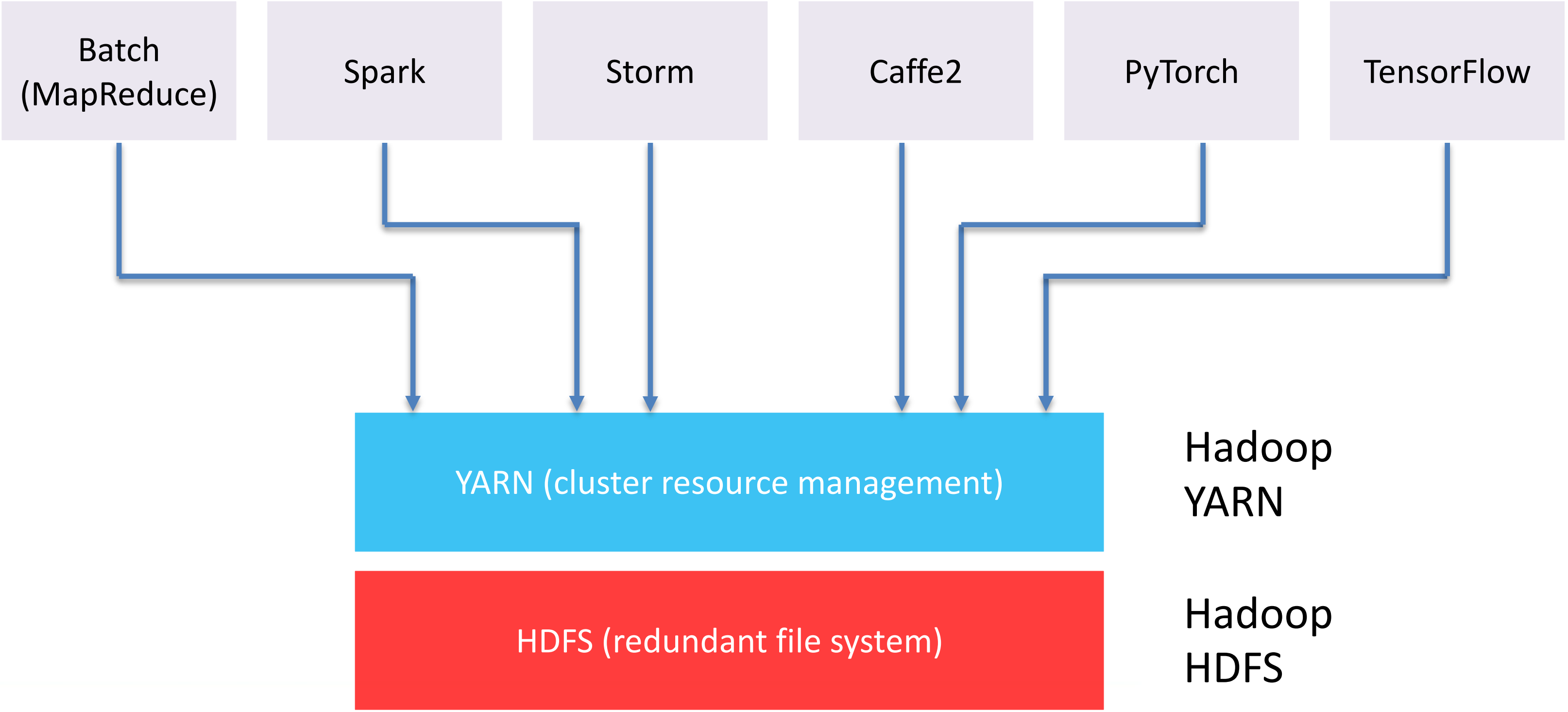


- The implementation of Hadoop 1.0 relied on the MapReduce processing model
- MapReduce manages both the Secondary's resources and job processing for the cluster
  - No ability to use the HDFS data lake for anything else
  - Limited to batch processing
  - Single processing engine

Hadoop 2.0 – YARN (Yet Another Resource Negotiator)



Execution of Multiple Frameworks

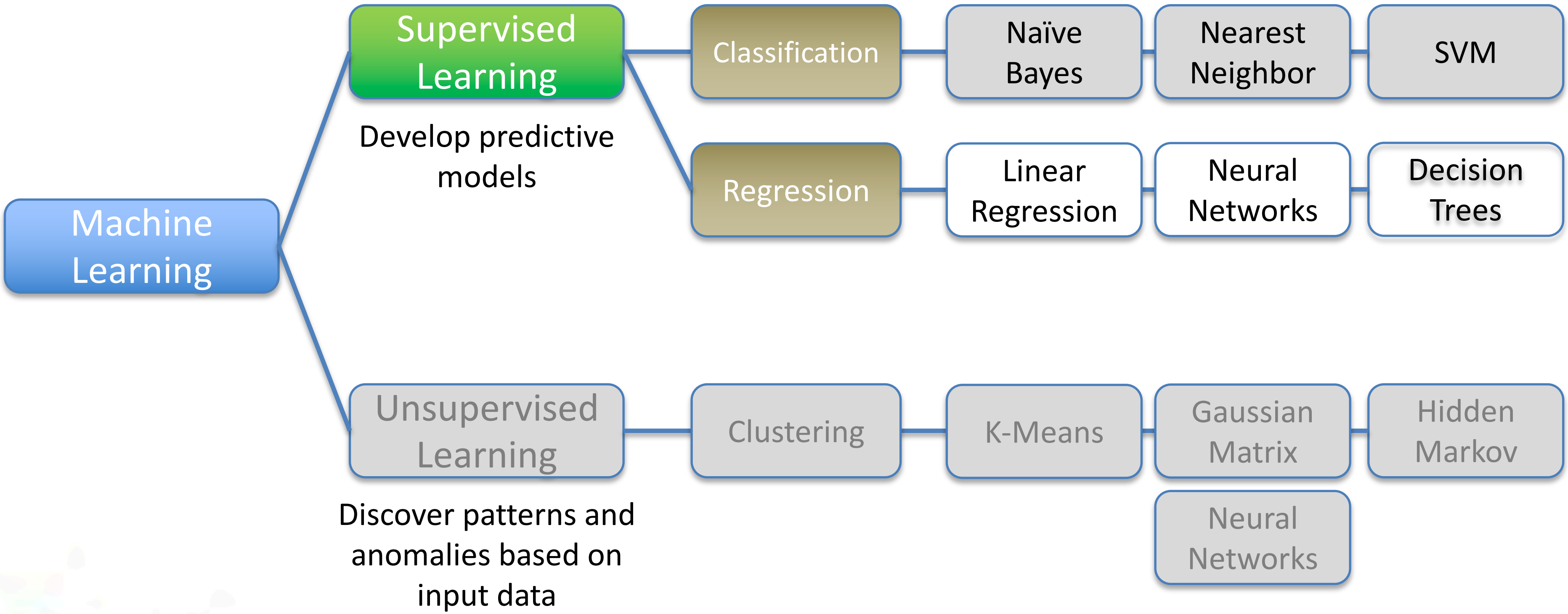


Lesson 3: Machine Learning 101 – Supervised Learning

Lesson 3: Supervised Learning

3.1 Understanding Supervised Learning

ML Is a Complex Landscape



Is the bridge on the verge of collapsing?

> 12,000 sensors

> 360,000 individual values

1.158863642 E+1843926 combinations

How many of them combined result in bridge fault?...

When there is too much data, computing combined individual thresholds is no longer feasible

Pearson

Two Types of Questions

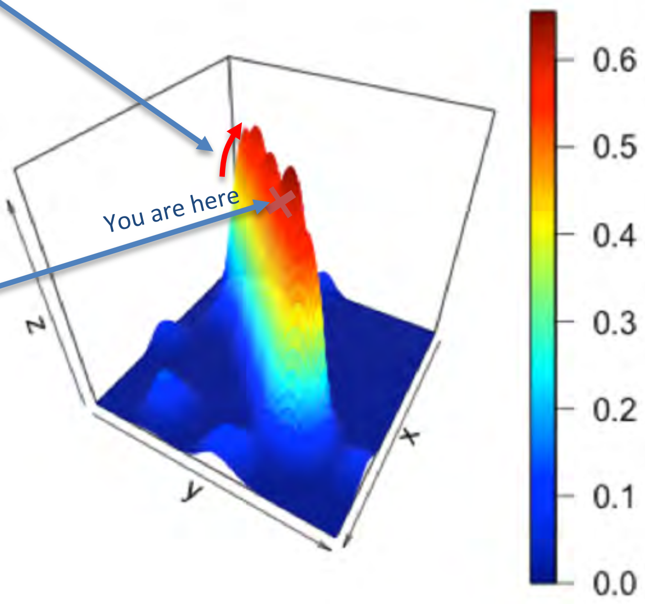
Most of the time, you try to answer two types of questions

Where is this trending toward?

- Is this likely to continue?
- What happens if this continues?
- Etc.

What is happening now?

- Am I at breaking point?
- Are conditions present for x to happen?
- What is the right value for z knowing x and y?
- Etc.



In all cases, you can see the answer if you can graph the data properly

Supervised Learning Provides Both Answers

- You inject data into a learning machine
- You tell the machine what component relates to another
  - E.g. the hardness of water depends on the quantity of calcium and magnesium
  - E.g. this combination of pixels = label "dog"

- Then you ask the machine to describe the best equation for this relationship
  - This is called ***regression***

Lesson 3: Supervised Learning

3.2 Linear Regression

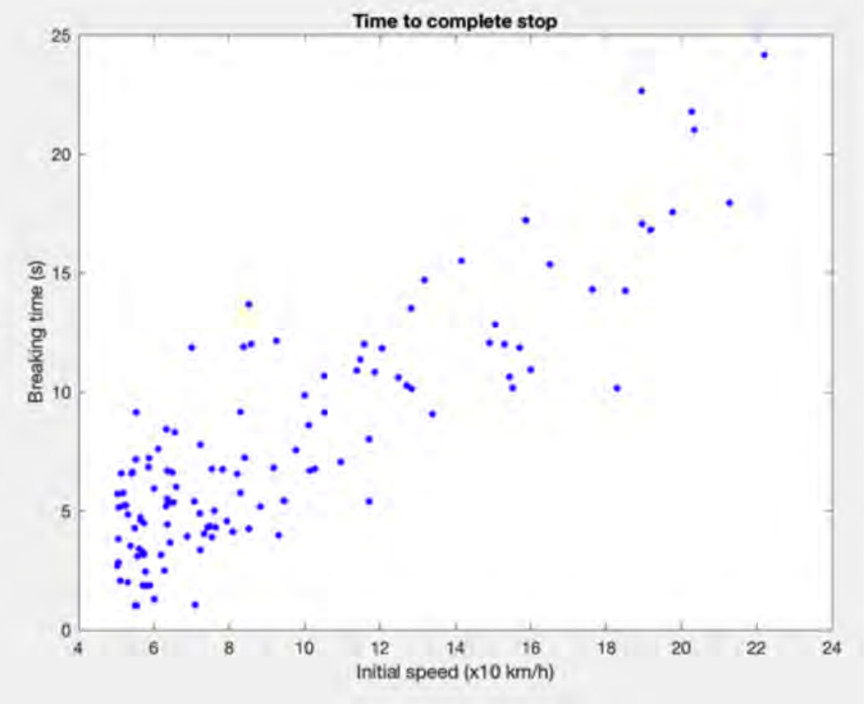
Supervised Learning: When You Know the Answer

You know the right answer, but there is too much data for you to produce the right answer for each case input into the system

- Example: How long does it take for a car to break to a halt?
- Depends on car weight, brake types / states, tires, road, humidity…
- How do you do that?

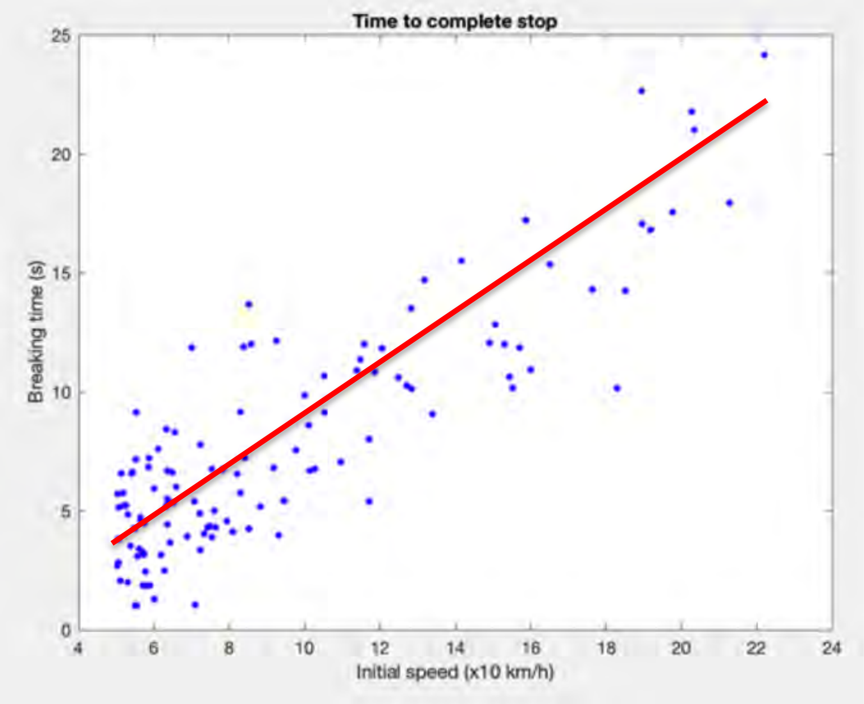
Linear Regression

1. You collect data about a car stopping from different speeds, and you plot these points – to make the process more visual, let's consider the time to stop required, depending on the initial speed.
2. Then you let the machine find the relationship speed and time needed.
3. Next, if you know the speed, you should know how long it will take to stop.



Why Linear Regression

- Visually, it is clear that there is a direct relationship between the speed and the time needed to stop
- The relationship is "linear"*
- You can draw a line, but could a computer draw that line?



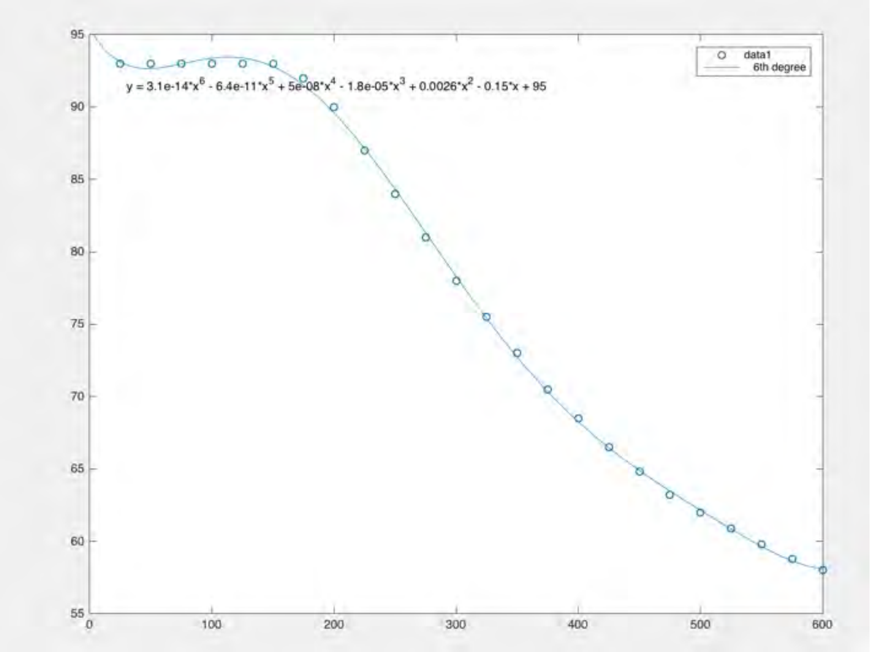
* That line is not "perfect" for all points, but it is the "closest" to all points

Supervised Learning Can Get Complex

Supervised learning can get complex -> non-linear and in many dimensions

$$y = 3.1^{-14}x^6 - 6.4^{-11}x^5 + 5^{-8}x^4 - 1.8^{-5}x^3 + 2.6^{-3}x^2 - 0.15x + 95$$

- The equation of the line you try to find can be complicated
  - It is still regression, but not "linear" anymore
- The variables you use, their exponents, etc. are called "features"
- Feature engineering is the primary role of the data scientist

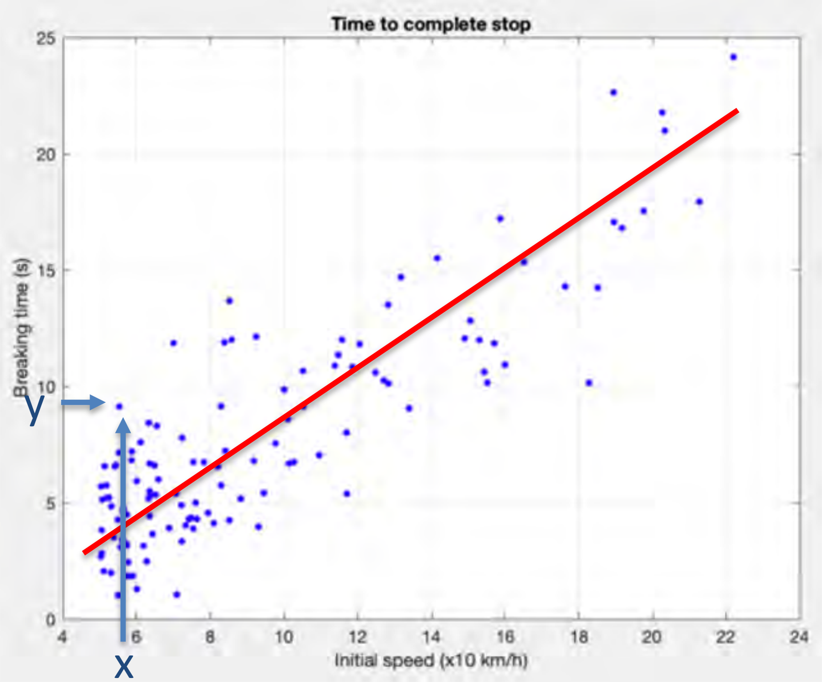


How Linear Regression Works

Each blue point has coordinates (x,y) that you know from your dataset;

x → ← y

```
6.1101,7.592
5.5277,9.1302
8.5186,13.662
7.0032,11.854
5.8598,6.8233
8.3829,11.886
7.4764,4.3483
8.5781,12
6.4862,6.5987
5.0546,3.8166
```



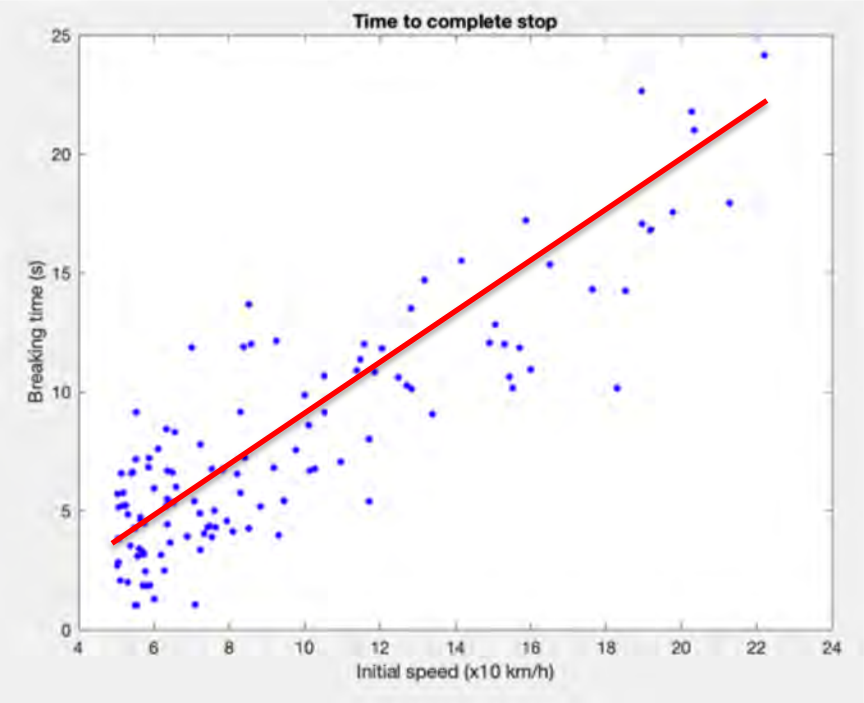
Describing a Line

Let’s do some math.

The red line is an equation

y = ax + b

(in ML, we say $\theta_0 + \theta_1 x$)



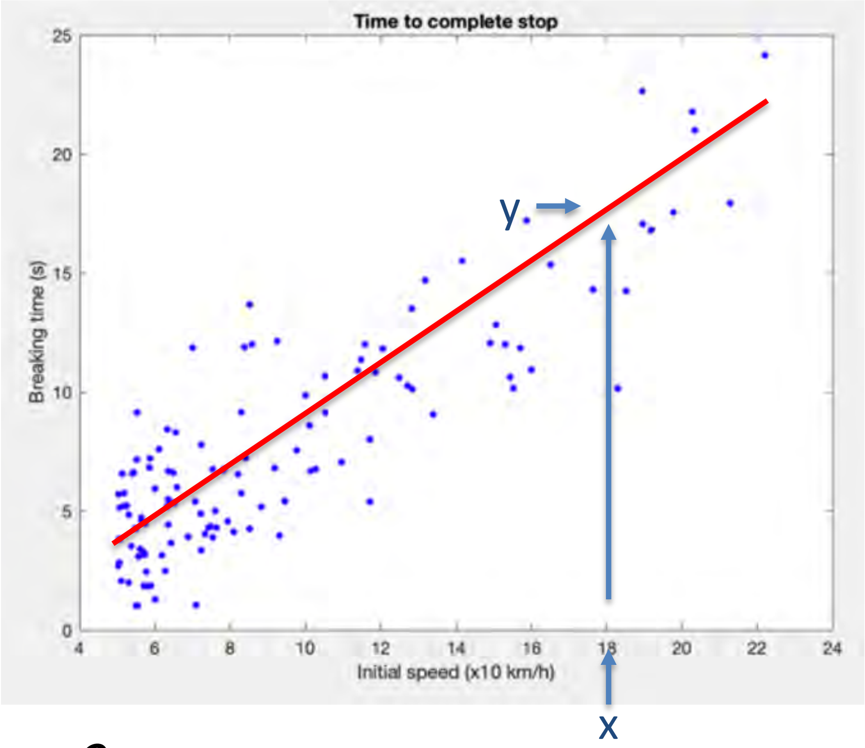
Line Equation in Machine Learning

$$\mathbf{y} = \boldsymbol{\theta_0} + \boldsymbol{\theta_1}x$$

Means that if you take an x value

Then compute $\boldsymbol{\theta_0} + \boldsymbol{\theta_1}x$

You should find the y on the line



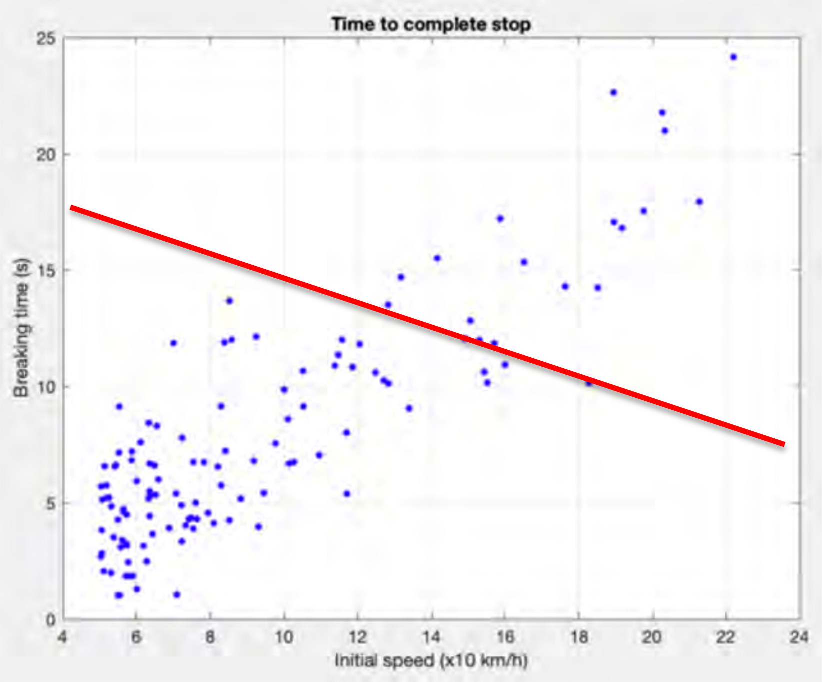
The issue is that you do not know $\boldsymbol{\theta_0}$ or $\boldsymbol{\theta_1}$

Finding the Equation – Random Start

Take two random θ_0 and θ_1

(in most cases, 0 and 0)

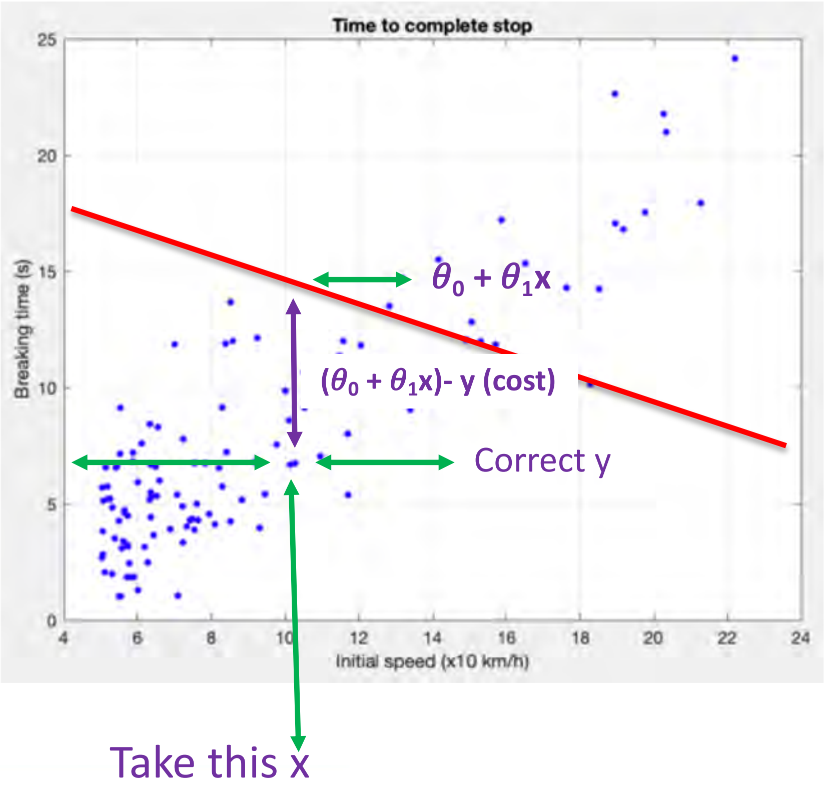
Your line is likely to be very wrong at first (but that's okay)



The Hypothesis Function

Then, for every blue dot you have (x_n, y_n)

Do $\theta_0 + \theta_1 x$ (that's your hypothesis function) and check how far you are from y_n



Compute the Cost of your Line

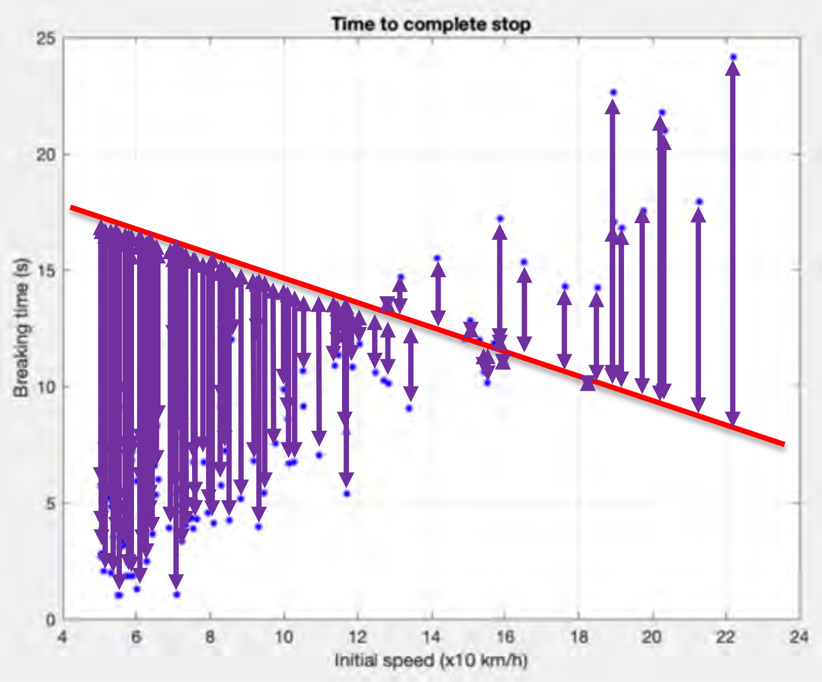
Repeat for each point for which you have (x_n, y_n)

Add all these mini-differences

$(\theta_0 + \theta_1 x) - y$

The total is “how far is your theoretical line from the best line for these points”

(your “total cost”)



The Cost Function

Another way to say it: the Cost Function is:

$$J\ (\theta_0, \theta_1) = \frac{1}{2}m \sum_{i=1}^{m}((\theta_0 + \ \theta_1 x_i) - y_i)^2$$

One of the **most** important equations in all of ML/AI

Repeat the Process

Then?

Change slightly $\boldsymbol{\theta_0}$ and $\boldsymbol{\theta_1}$

and repeat…

Then ask yourself:

Am I closer to the real y
with the new θ_0 and θ_1 ?

Gradient Descent

This changing process is called

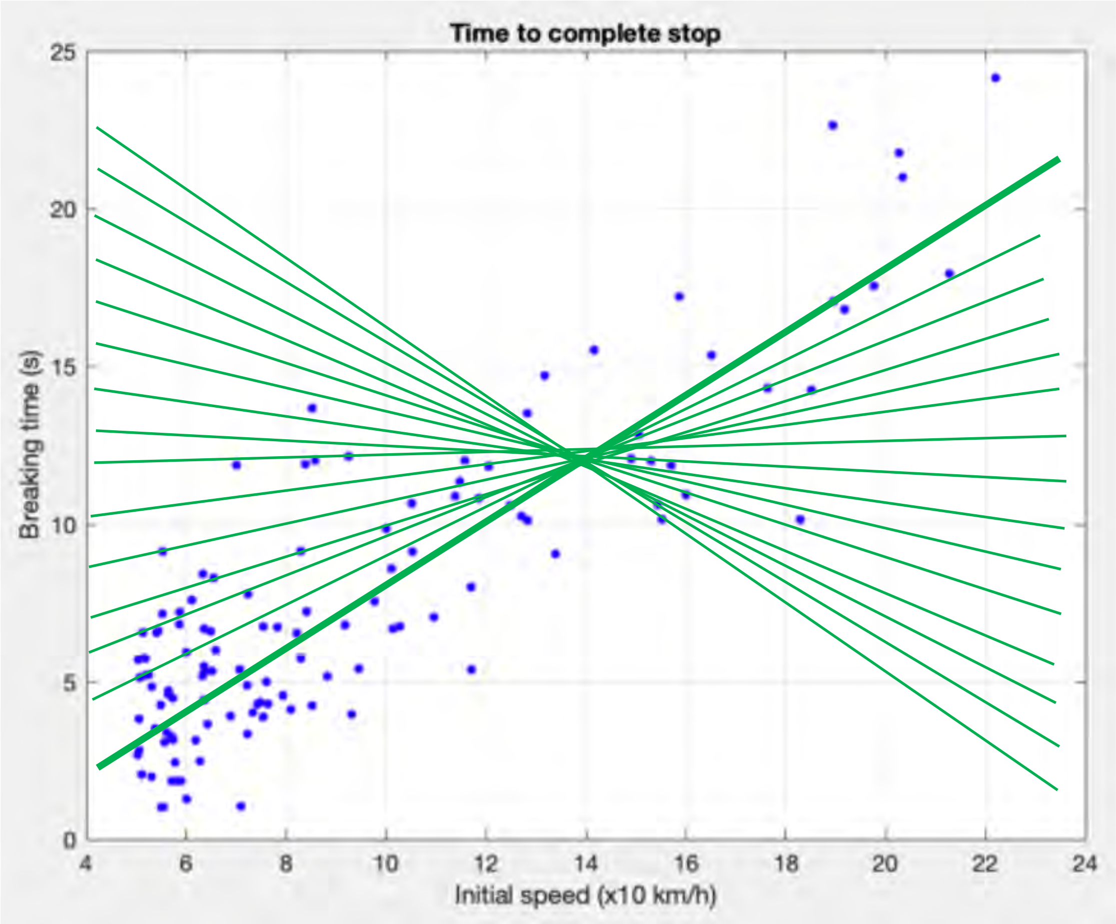
gradient descent

You can call it ***brute-force***

trying all θ_0 and θ_1

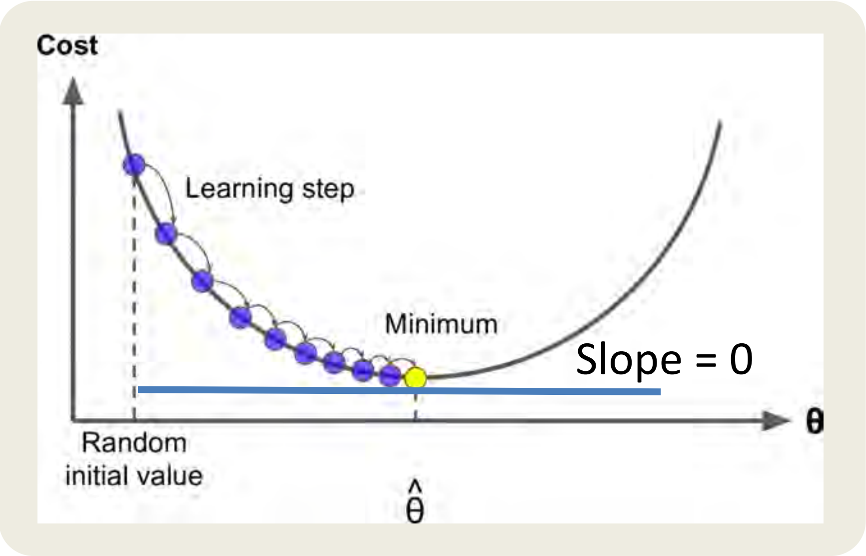
until you find the right values

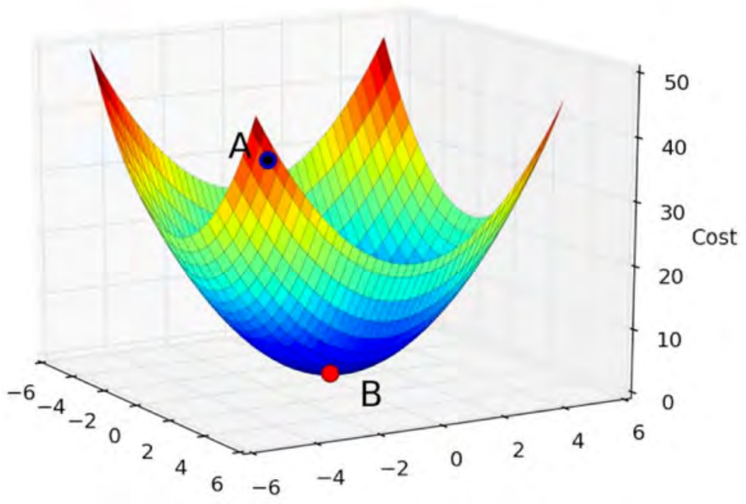
Gradient Descent Illustration



Math with "cost" can help find the best next θ_0 and θ_1

We can use Calculus and derivatives



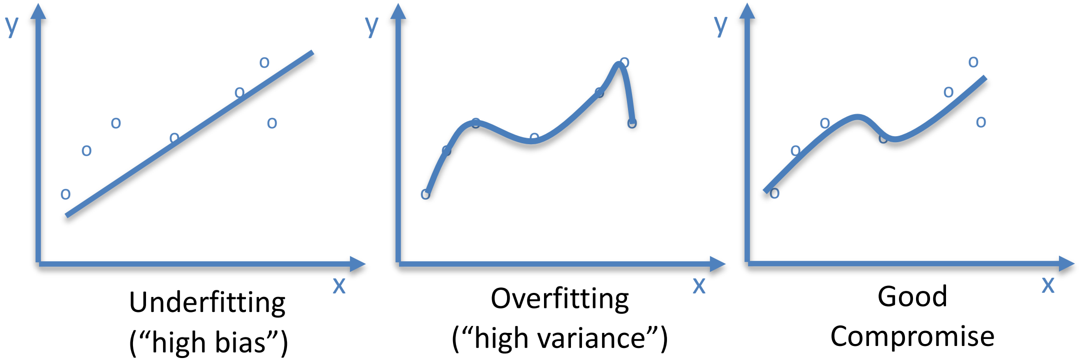


When slope is 0… it means you are at the local minimum of the cost function

Especially useful in larger dimensions

Finding the Right Line

The main challenge in Supervised Learning is to find the right equation… and figure out if the samples represent the full population



Use 2 or 3 Sets

To avoid the high bias / high variance issues, divide your data set into 3 groups:

- Training dataset
  - E.g. 60% of your data, train your model
- Validation dataset
  - E.g. 20% of your data, verify and refine your model
- Test dataset
  - E.g. 20% of your data, confirm your model
  - If the model does not work, rework it!

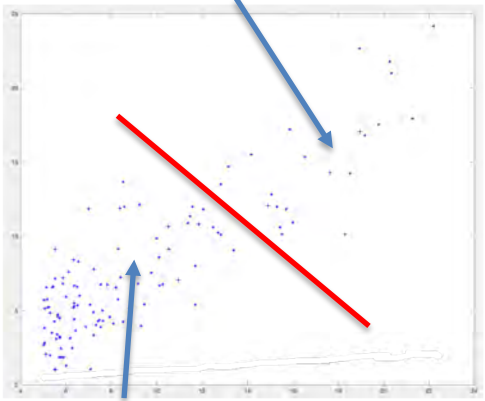
Lesson 3: Supervised Learning

3.3 Logistic Regression

Logistic vs. Linear Regressions

- In some cases, data represents different categories
- Your goal is now to split the data points in groups
    - E.g. finding clumps in a pipe that are both large and hard
    - Organizing data in groups is called ***classification***

x or y is large (more than 14)



x and y are “small” (less than 14)

ML gotcha:

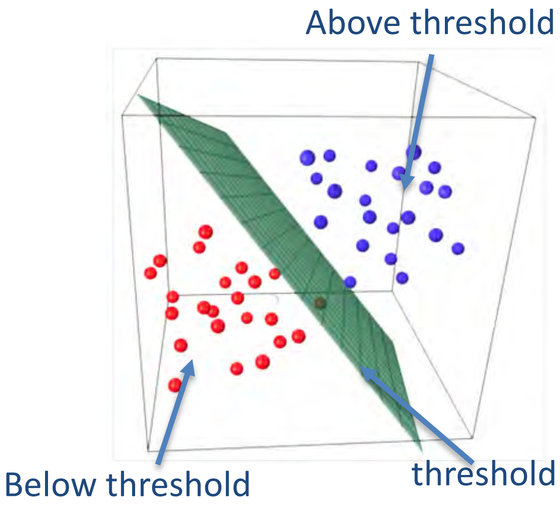
Algorithms that perform classification are called ***logistic regression*** algorithms

Different from data description algorithms (***linear regression***)

Principles of Logistic Regression

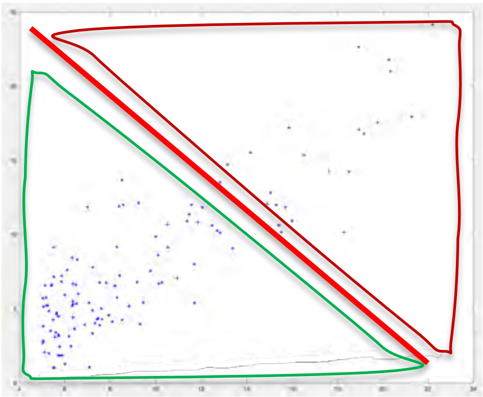
Logistic regression (simple models) uses thresholds:

- Apply an equation to your data, choose it cleverly
  - Choose the right parameters and weights
  - You can get a line of separation
- Data Points which output result below threshold are in one category, above threshold in another category



A Probabilistic approach

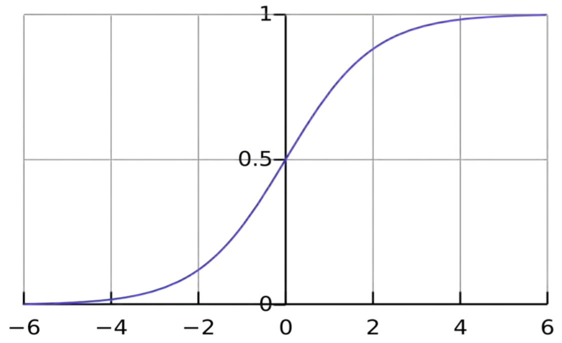
- With classification, you are not trying to compute the right ***y*** for a given ***x***
- You just want the answer “green group or red group”?
  - Sometimes with a likelihood indication
  - This binary logic is great, because you can decide if ***1*** means ***red*** or ***green***



A Probabilistic Approach

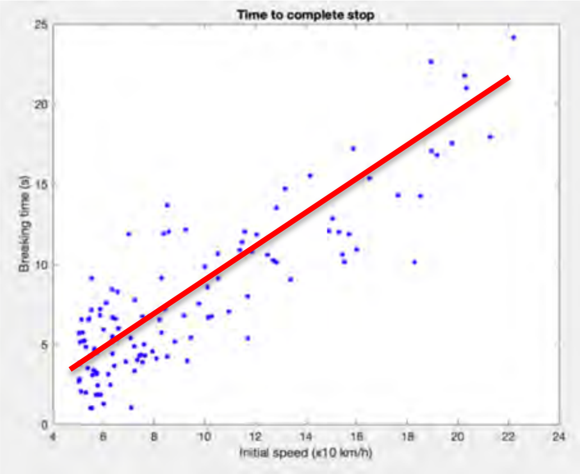
Group membership is a probability

- We can use an equation that leverages this property:
  - equation output is close to 0 -> group A
  - equation output is close to 1 -> group B
- This logic also helps with our general reasoning
  - “With these parameters, can I predict that the next input will be more likely in group A or B?”



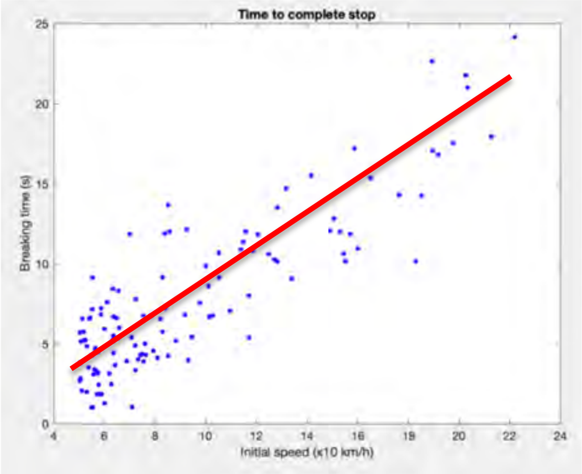
Car Breaking Issue

In our car breaking example, classification is not "how long before stop based on car speed" but "is the car going to hit the tree?" (yes/no)



Car Breaking Issue

- With linear regression, we found the equation of the line
  - (y = $\theta_0 + \theta_1$x) (y = -1 + 0.94x)
  - We could design our groups as:
  - Less than 140 km/h -> less than 12.16 seconds to break -> no hitting
  - More than 140 km/h -> more than 12.16 seconds to break -> hitting*



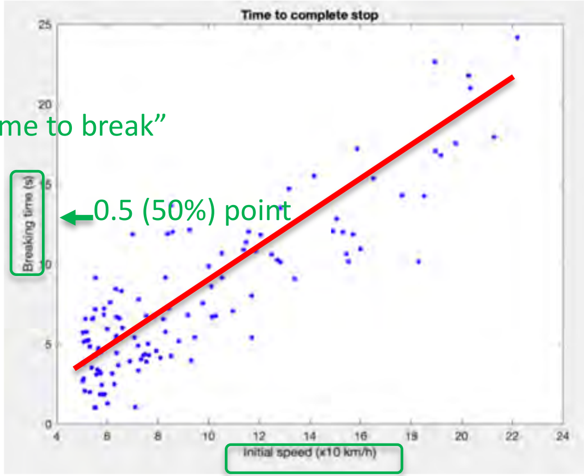
- As this is a probability, we could even say:
- As speed increases, the likelihood of hitting the tree increases, and is more than 0.5 (50%) over 140 km/h

* Note that as speed is in tenth of km/h, the time is in tens of seconds, i.e. 120 = 12 seconds

Probability and Car Break

- We could trade our graph scale accordingly
- If we want ***12.16*** to be our midpoint (0.5, or 50%), then we can just modify our equation:
- $P = \dfrac{-1+0.94x}{24.32}$ ← 2x12.16

Change "time to break" to "likelihood of hitting the tree"

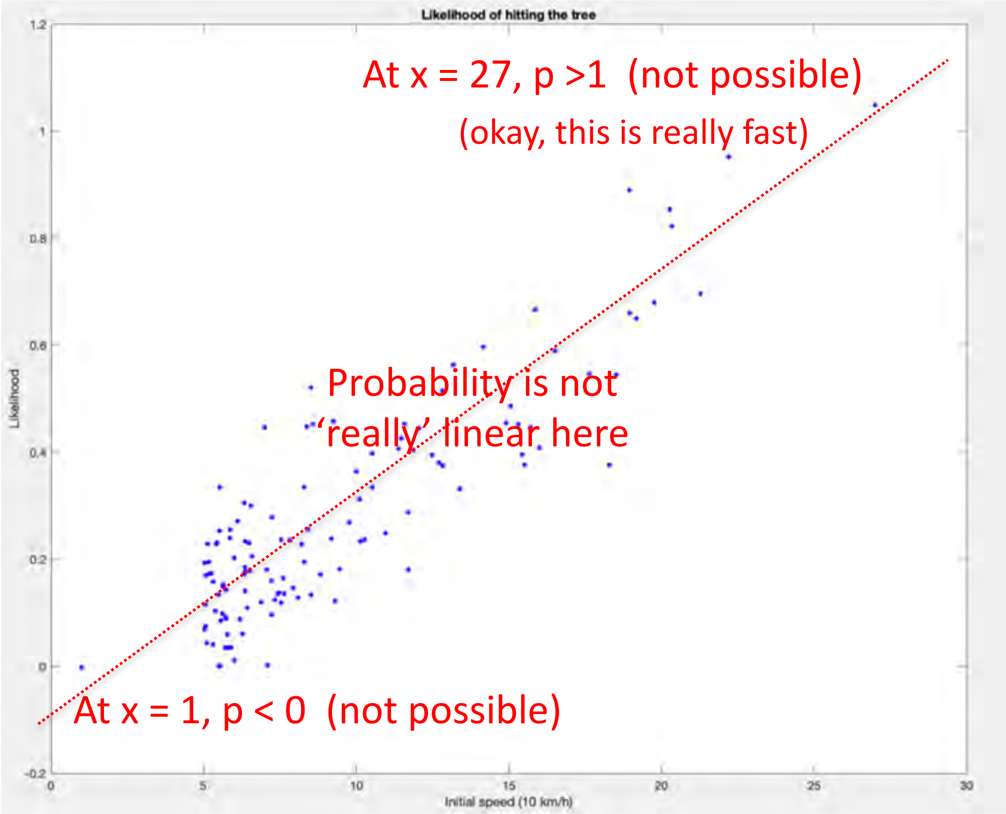


Keep this scale

- At x = 14, p = 0.5 (50%)
- At x = 20, p = 0.73 (73%)
- At x = 10, p = 0.34 (34%)
  - All seems to look good

Limitation of the Linear Model

This “linear” model breaks at the edge:



A Different Equation

- Logistic regression uses a method where:
  - ✓ Output numbers are positive
  - ✓ min is 0 (but not less)
  - ✓ max is 1 (but not more)
- There are a few ways to solve these conditions, but one that works very well:
  - $e^{\text{something}}$ is always positive
    - but can be more than 1
  - $\frac{something}{something+1}$ is always between 0 and |1|

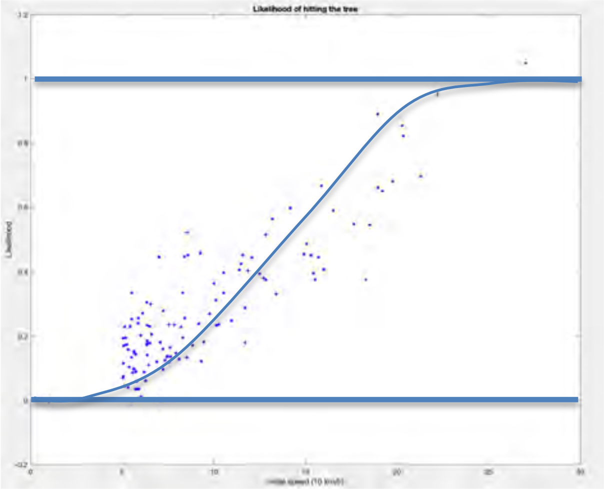
The Logistic Regression Equation

- If we use:

$$p = \frac{e^{\theta_0 + \theta_1 x}}{e^{\theta_0 + \theta_1 x} + 1}$$

- We get the conditions we want!
- It looks complex, but $\theta_0 + \theta_1 x$ is our line equation, and for mathematicians, it is a simple transformation:

$$\ln\left(\frac{p}{1-p}\right) = \theta_0 + \theta_1 x$$



The Sigmoid Function

- This is too long to write!

$$p = \frac{e^{\theta_0 + \theta_1 x}}{e^{\theta_0 + \theta_1 x} + 1}$$

- Let's do a math trick:
- $e^{\theta_0 + \theta_1 x} = (e^{\theta_0 + \theta_1 x})\,(1)$
- $e^{\theta_0 + \theta_1 x} + 1 = (e^{\theta_0 + \theta_1 x})\,(1 + \frac{1}{e^{\theta_0 + \theta_1 x}})$
- And $\frac{1}{e^{\theta_0 + \theta_1 x}} = e^{-(\theta_0 + \theta_1 x)}$

In general, $\frac{1}{a^n}$ is the same as a^{-n}

Still there?

If we take $\frac{e^{\theta_0 + \theta_1 x}}{e^{\theta_0 + \theta_1 x}}$ out,

we can write:

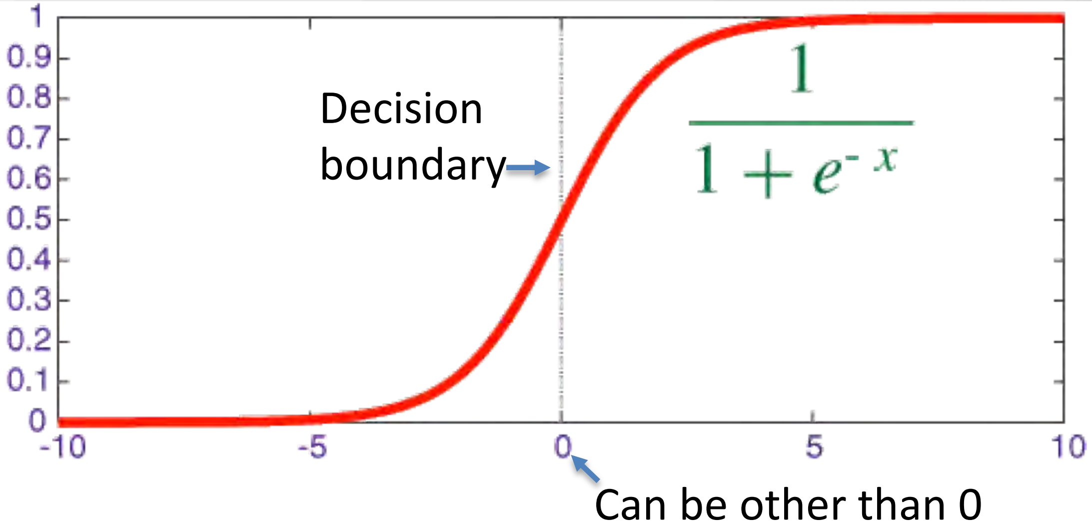
$$p = \left(\frac{e^{\theta_0 + \theta_1 x}}{e^{\theta_0 + \theta_1 x}}\right)\left(\frac{1}{1+e^{-(\theta_0 + \theta_1 x)}}\right)$$

$$p = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

The Sigmoid (or Logistic) Function

It was worth it!

- The sigmoid function is the most common function for logistic regression:
  - Gets your data into 2 groups
  - Tells you the probability of belonging to a group or the other
  - Directly relates to your “linear” equation

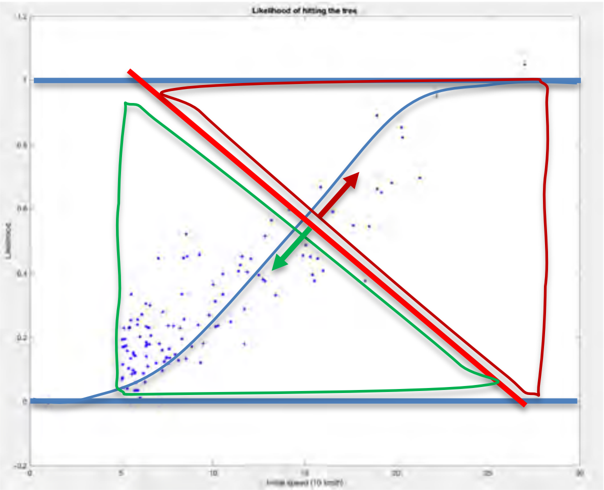


Thresholds and Cost

You can “manually” map “> 0.5 = group A” from our result

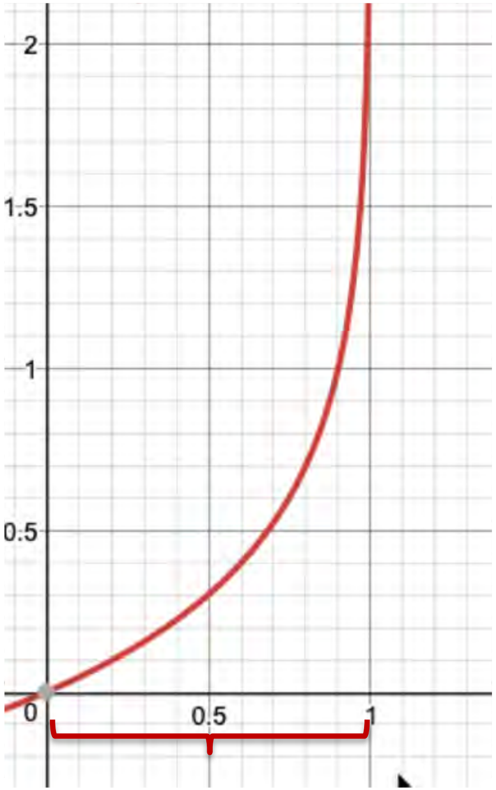
You can also directly use the probability value output

- If you train your model and want a cost, you may want to push points to one side or the other:
  - A point in the green area has a super high **red** cost
  - A point in the red area has a super high **green** cost



Logistic Regression Cost Function

- A common way to adapt the cost function to the logistic regression work is to use:
  - (remember: $p = \frac{1}{1+e^{-(\theta_0 + \theta_1 x)}}$)
  - -log(p) (cost lower for p=1)
  - -log(1-p) (cost lower for p=0)
  - Compute both for each p, and find the lowest cost



Lesson 3: Supervised Learning

3.4 Support Vector Machines

Decision Boundary

The "threshold" between groups is the decision boundary

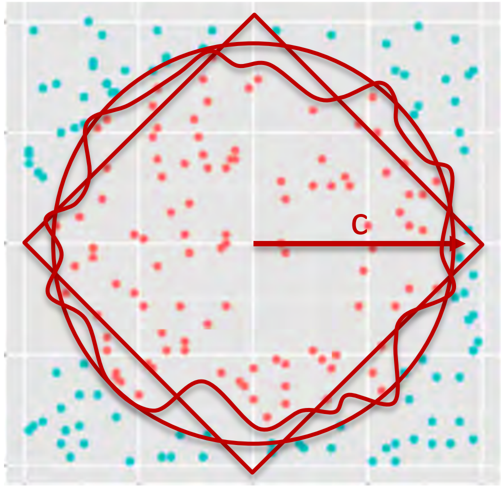
- It can be a simple straight line or a more complex shape
- Then its equation is going to change accordingly

$$x^2 + y^2 = c$$

$$|x| + |y| = c$$

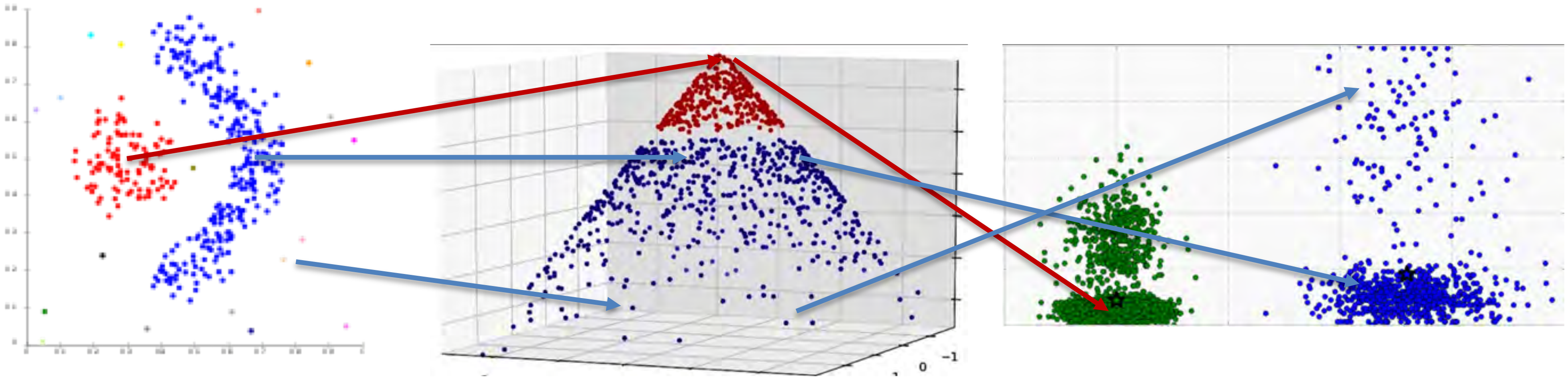
$$x_1^a + x_2^b + \dots + x_n^p = c$$

- The same 'best fit challenge' (as in linear regression) is present



Graph As You Need

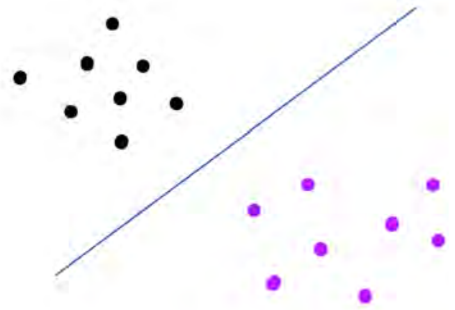
- The decision boundary line type depends on the features and how you graph the data
  - Then your decision boundary equation will determine if a point is “more likely in group A or in group B”



When Sigmoid Does Not Work for You

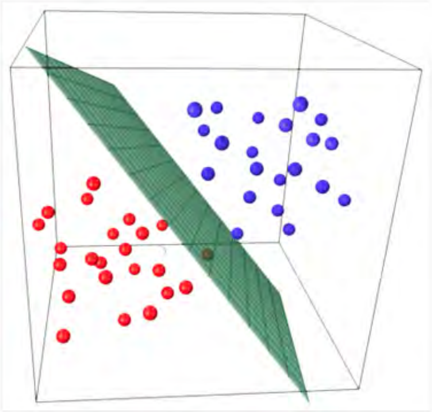
For many graphs, logistic regression is not the best choice

- All you need is find the groups (and their members)
- The graph structure may not work well for sigmoid... and you don’t want to change the graph:
  - Data shows what you want
  - You have more than 2 groups
  - …



Looking for the Separation Line

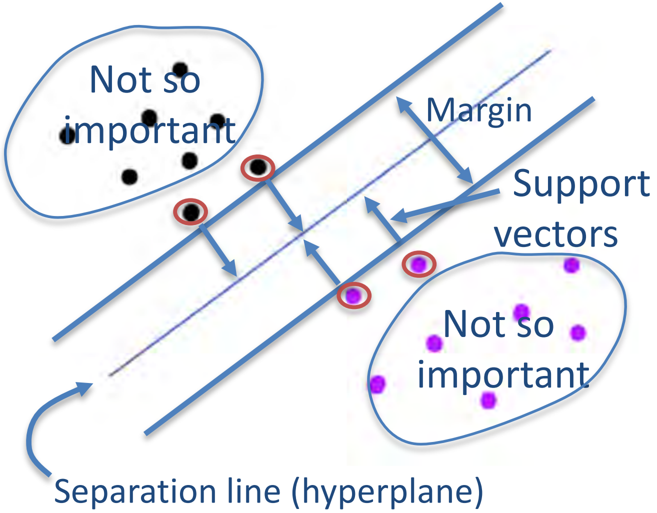
- There may be multiple possible separation lines
- Support Vector Machines (SVM) helps you find the “best” line
  - If you have more than 2 dimensions, the “line” becomes “a plane”, or “a hyperplane”
- The ***best*** line is the one farthest from all points



SVM Terminology

SVM helps you find the middle line

- Once you determine the groups, consider the points at the edge
- Then finding the line that is farthest from these support vectors is a ***constrained optimization problem***
  - Farthest means maximizing the ***margin***
- SVM solves it with a formula called ***Lagrange multipliers***

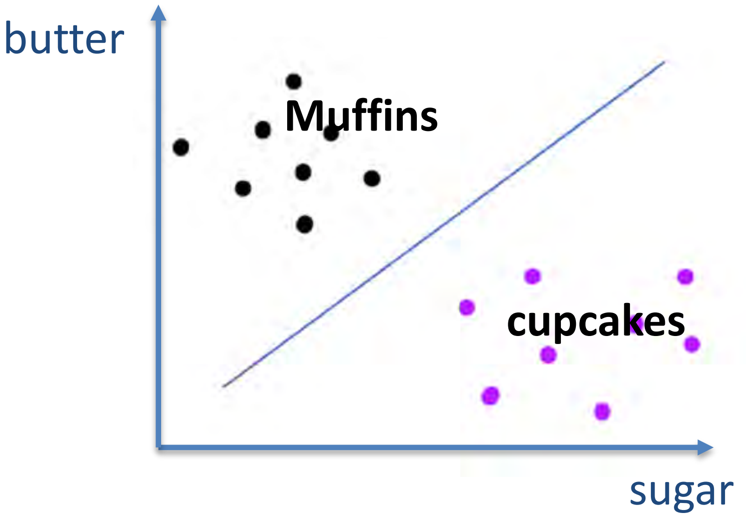


Finding Max Distance

- Remember that this is your training set
  - You know which point is in which group
- E.g muffins vs cupcake
  - Muffins: lots of flour, quite some milk, quite some sugar, low butter
  - Cupcake: quite some flour, some milk, some sugar and quite some butter

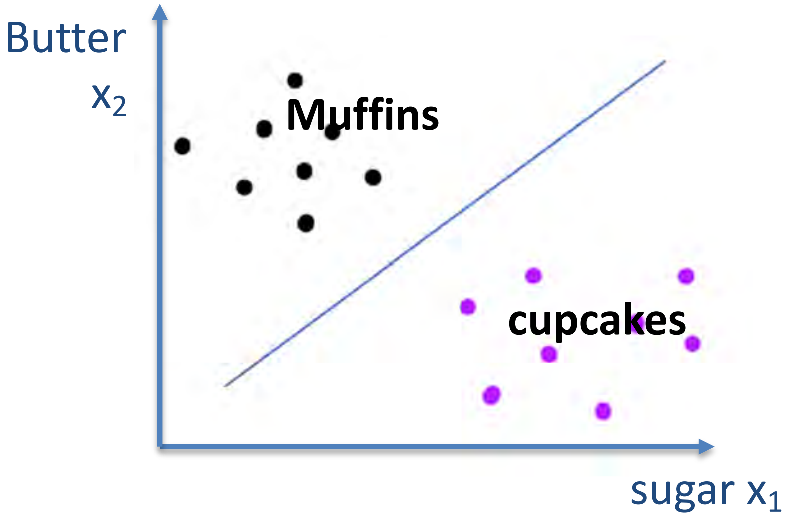
Finding Max Distance

- sugar = x_1, butter = x_2, flour = x_3, milk = x_4
- To simplify the graph, let's look at butter vs sugar:
  - Small x_1, large x_2 = muffin
  - Large x_1, small x_2 = cupcake
- Each dimension has a weight
  - High w_1, low w_2 -> cupcake
  - Low w_1, large w_2 -> muffin



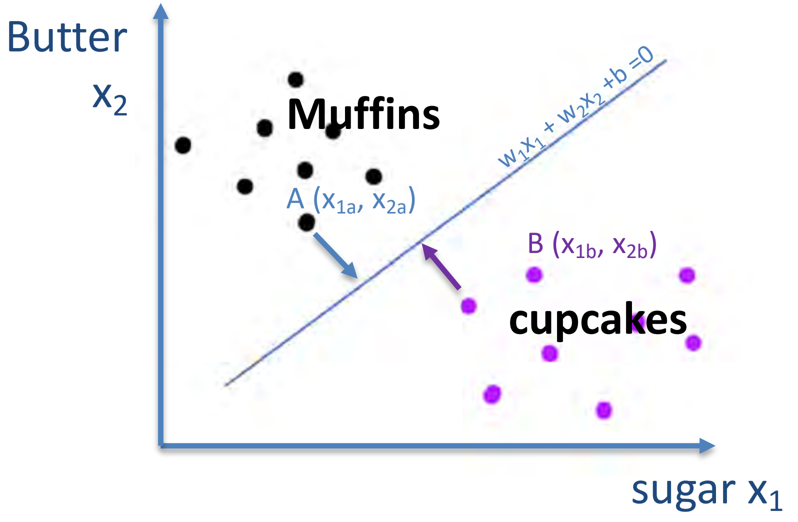
Finding Max Distance

- We use a similar idea of cost as in linear regression (with a variant)
- Start with a line:
  - It is slightly different from linear regression, instead of ax+b = y, you need wx+b = 0
    - W is your weight, here: $w_1x_1 + w_2x_2 + b = 0$
    - (the goal is to find the w's, positive result will mean one group, negative results will mean the other group)



Finding Max Distance (Max Margin)

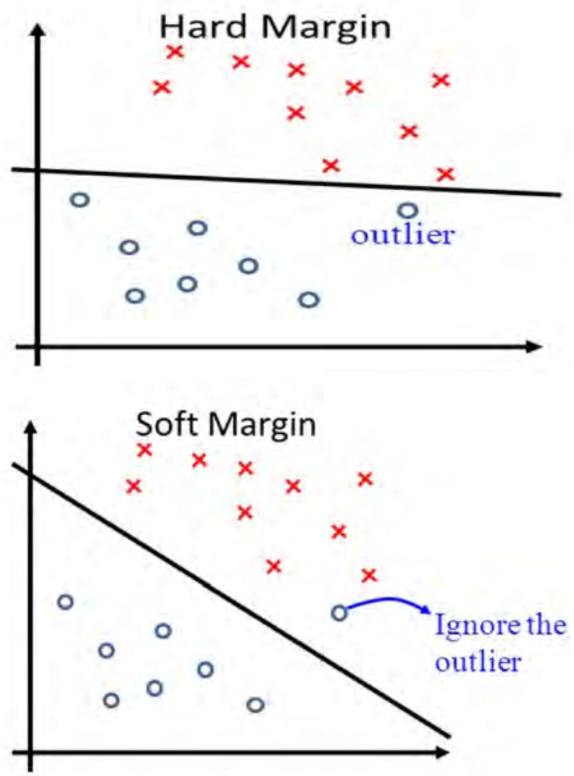
- Then for each point, compute your distance to the line:
  - A shortcut in linear algebra (vector dot product) helps us:
  - $d_A = w_1x_{1a} + w_2x_{2a} + b$
  - $d_B = w_1x_{1b} + w_2x_{2b} + b$
  - When the arrow goes one direction, result will be positive, negative in the other direction
  - Run the computation through your training set… and your goal is to find the maximum distance (max margin)



Soft Margin

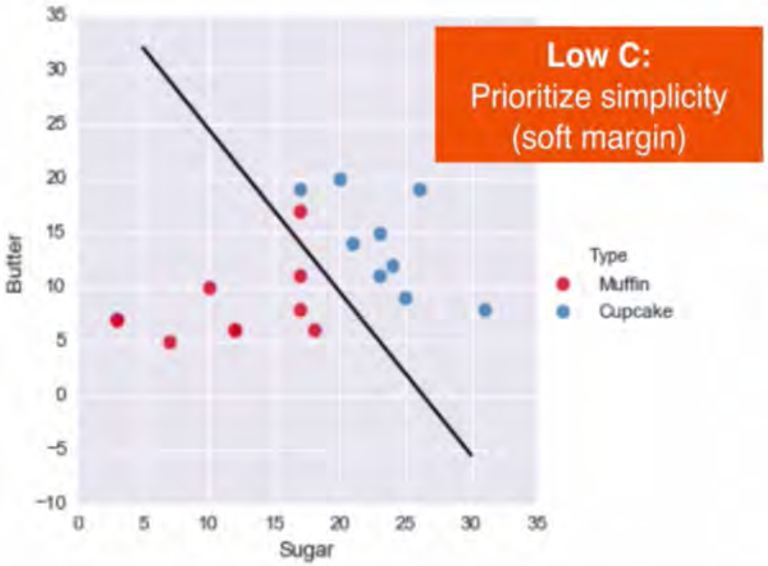
Most “modern” SVM ML tools allow you to compute the margin… and be soft with it

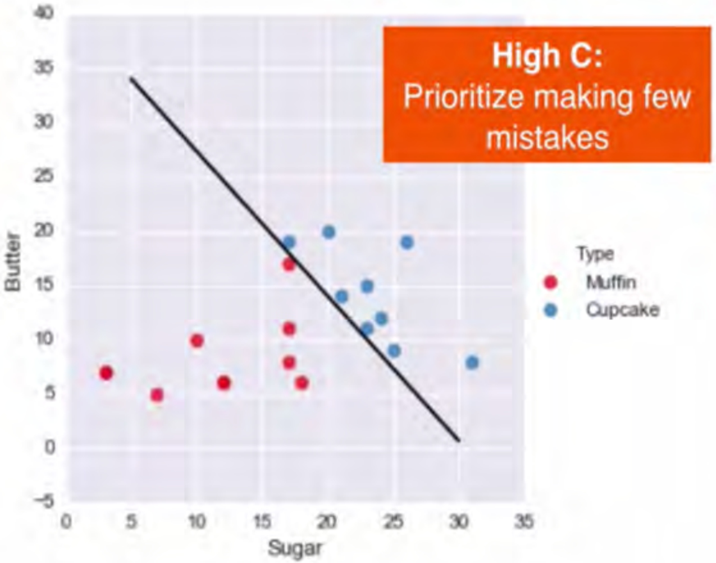
- Useful when your training set has “normal” outliers
- This “softness” is often called the ***C parameter***



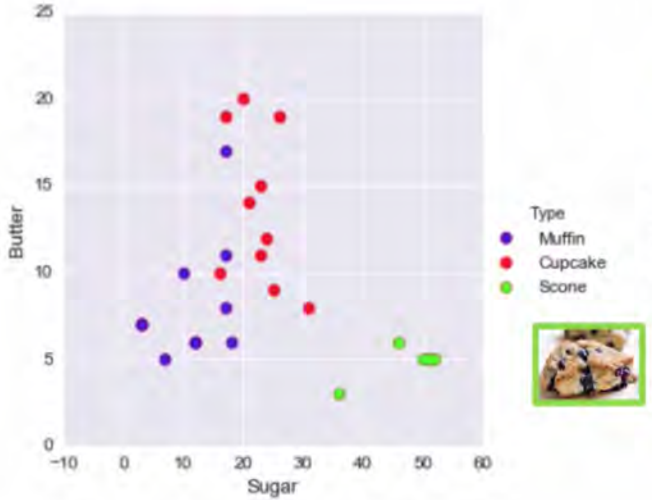
Soft Margin in Action

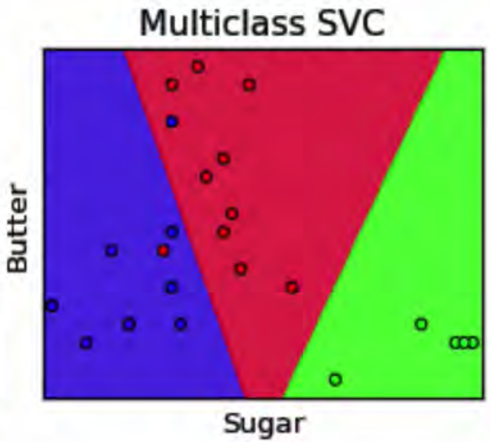
Determining the right softness is still your job

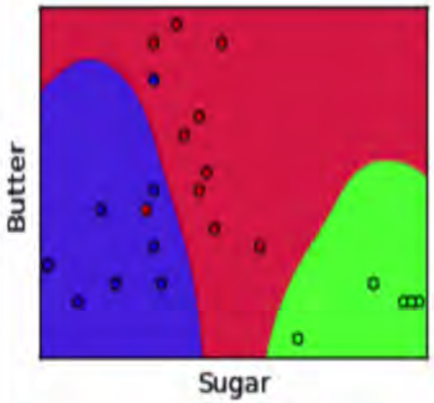




Multiple Classes







You can determine more than one line

- Process is fairly similar
- And there are math projections to twist lines and dimensions

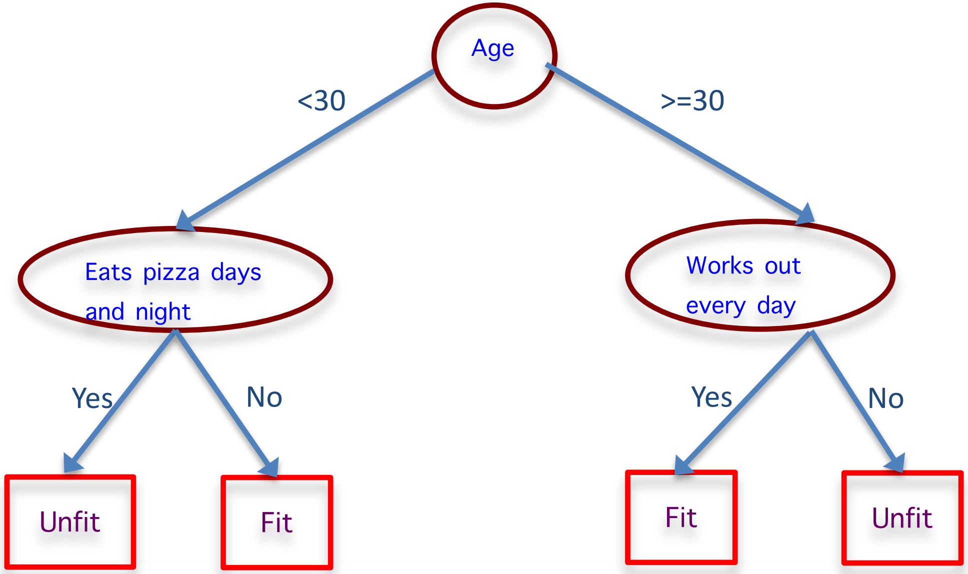
Lesson 3: Supervised Learning

3.5 Random Forests

Decision Trees

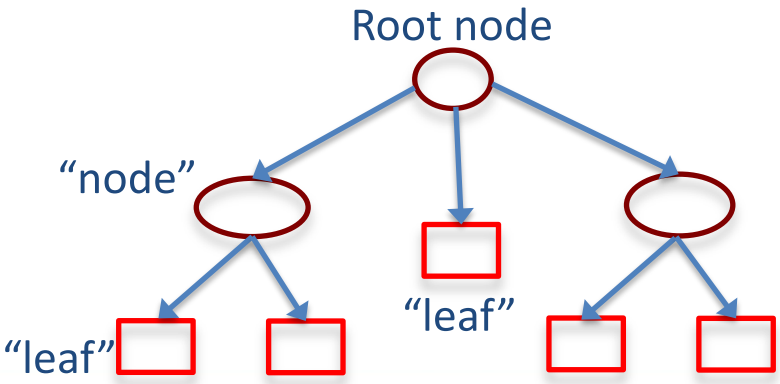
You probably know them… common on paper and in ML

- Each step asks a question, classifies the answer then branches out depending on that answer
- Answer can be number, yes/no (0/1), name/category.



Decision Trees

- Decision trees can represent categories or numbers
- When data becomes large, ML can help build the tree structure (computes relationship between data points)
  - If variables are continuous numbers, it is a (linear) ***regression algorithm***
  - If variables are categories, it is a ***classification algorithm***



How to Grow a Tree

- Multiple ways
- Suppose you want to test the potential of a customer buying a computer
  - Pick one variable (e.g. credit)
  - Then check if there is a good match

Fair rating	
Yes	No
5	2

Excellent rating	
Yes	No
3	3

credit_rating	buys_computer
fair →	no
excellent	no
fair →	yes
fair →	yes
fair →	yes
excellent	no
excellent	yes
fair →	no
fair →	yes
fair →	yes
excellent	yes
excellent	yes
fair	yes
excellent	no

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31-40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31-40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31-40	medium	no	excellent	yes
31-40	high	yes	fair	yes
>40	medium	no	excellent	no

How to Grow a Tree

Repeat with each other variable

Age

<=30

Buys computer	
Yes	No
3	2

31-40

Buys computer	
Yes	No
4	0

>40

Buys computer	
Yes	No
3	2

Income

low

Buys computer	
Yes	No
3	1

Medium

Buys computer	
Yes	No
4	2

high

Buys computer	
Yes	No
2	2

Student

Yes

Buys computer	
Yes	No
6	1

No

Buys computer	
Yes	No
3	3

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31-40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31-40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31-40	medium	no	excellent	yes
31-40	high	yes	fair	yes
>40	medium	no	excellent	no

Impure Trees

- None of our parameters are a perfect match to predict “buy computer”
  - They are called ***impure***
- Choosing the best of ***impure*** can be done in multiple ways, a popular one is computing the ***Gini impurity***
- It’s the same concept of “distance” or “cost” we saw before, applied to probabilities

“Cost” Ported to Probability

- Gini impurity = 1 – (probability of yes)^2 – (probability of no)^2

Student

Yes

Buys computer	
Yes	No
6	1

No

Buys computer	
Yes	No
3	3

$$\text{Gini} = 1 - \left(\frac{6}{6+1}\right)^2 - \left(\frac{1}{6+1}\right)^2 = 0.245$$

$$\text{Gini} = 1 - \left(\frac{3}{3+3}\right)^2 - \left(\frac{3}{3+3}\right)^2 = 0.5$$

- Repeat for all leaves, then pick the “least impure” as your root node

Why Does Gini Work?

- Gini impurity = 1 – (probability of yes)^2 – (probability of no)^2

Buys computer	
Yes	No
6	1

$$\text{Gini} = 1 - \left(\frac{6}{6+1}\right)^2 - \left(\frac{1}{6+1}\right)^2 = 0.245$$

- In a perfect ("pure") world, there is a perfect match between the leaf results and "buy computers":

Everyone buys computers

1 – (probability of yes)^2 – (probability of no)^2

1 – this is 1 – this is 0

1 – 1 – 0 = 0 (pure)

No one buys computers

1 – (probability of yes)^2 – (probability of no)^2

1 – this is 0 – this is 1

1 – 0 – 1 = 0 (pure)

The Gini Impurity Graph

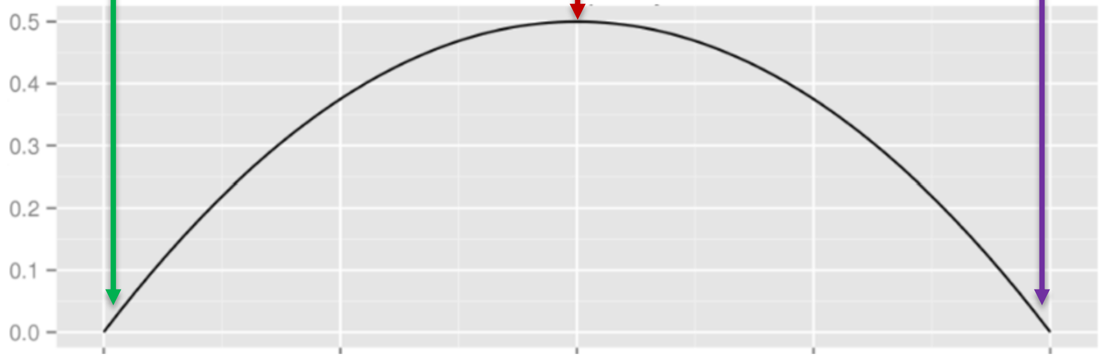
Gini impurity measures how random your match is

- Very impure = completely random -> works 50% of the time, no more

Gini = $1 - (\frac{2}{2})^2 - (\frac{0}{2})^2 = 0.5$

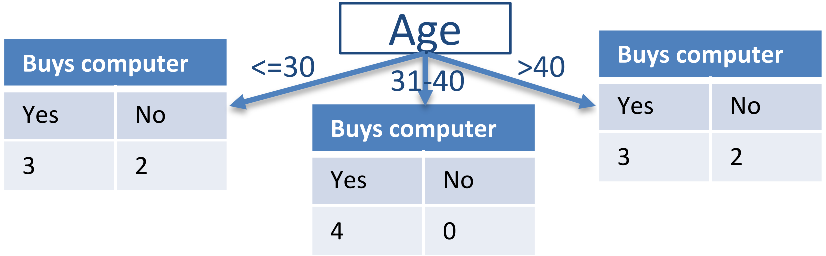
Gini = $1 - (\frac{1}{2})^2 - (\frac{1}{2})^2 = 0.5$

Gini = $1 - (\frac{0}{2})^2 - (\frac{2}{2})^2 = 0.5$



Completing the Tree

- In our case, the lowest impurity parameter ends up being 'Age"



- Once you have a root node, take each branch separately and re-apply the process:

<=30 = 5 | **31-40 = 4** | **> 40 = 5**

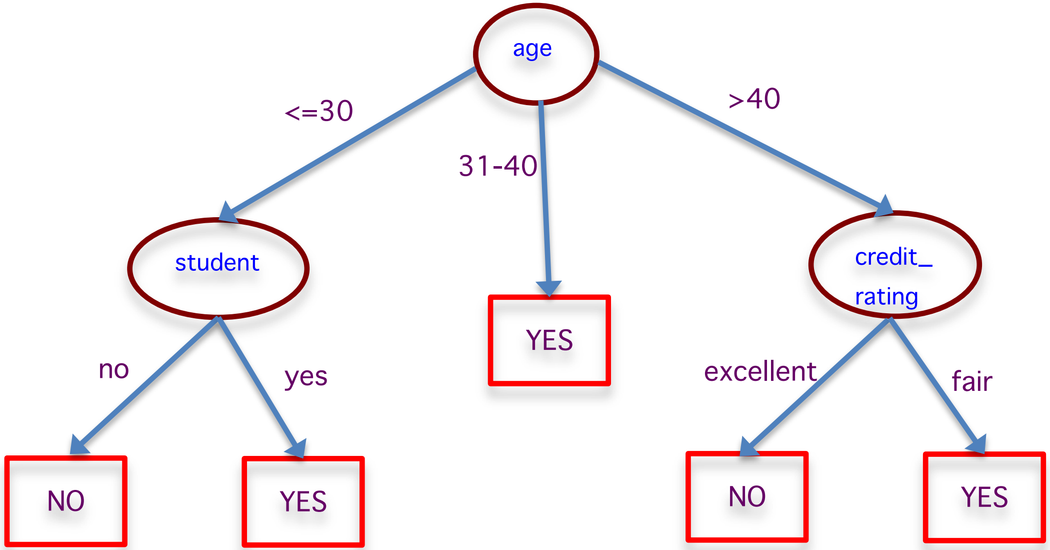
Out of these, student/non student? Income? Credit?

Student(1)	high (2)	Excellent (2)
Non-std(4)	Medium (2)	Fair (3)
	Low (1)	

Then repeat Gini impurity to build your next node

Completing the Tree

You grew your first tree!



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31-40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31-40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31-40	medium	no	excellent	yes
31-40	high	yes	fair	yes
>40	medium	no	excellent	no

Random Forests

- Forests are made of trees…
- Trees are limited, in that they work well with the data used to create them, but adapt poorly to other data sets
- The goal of the random forest is to create multiple trees, that together will get better accuracy
  - Better adaptation to new data
  - Better prediction

Random Forests Principles

To create a random forest, create a bootstrap data set

- Take random variable subset (same or smaller variable size)

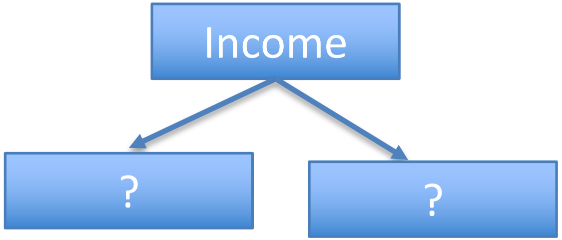
- Take sample data (can take the same sample more than once)

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31-40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31-40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31-40	medium	no	excellent	yes
31-40	high	yes	fair	yes
>40	medium	no	excellent	no

Appears twice

>40	medium	no	fair	yes
<=30	high	no	excellent	no
31-40	medium	no	excellent	yes
<=30	high	no	excellent	no

Random Forests Principles



- Ignore income for the rest of the work (already used)
- Then select 2 new random variables as candidates for each node
  - Then build the tree that way

>40	medium	no	fair	yes
<=30	high	no	excellent	no
31-40	medium	no	excellent	yes
<=30	high	no	excellent	no

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31-40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31-40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31-40	medium	no	excellent	yes
31-40	high	yes	fair	yes
>40	medium	no	excellent	no

Random Forests Principles

Then, build trees

- But only take a (random) subset of variables (columns) at each step

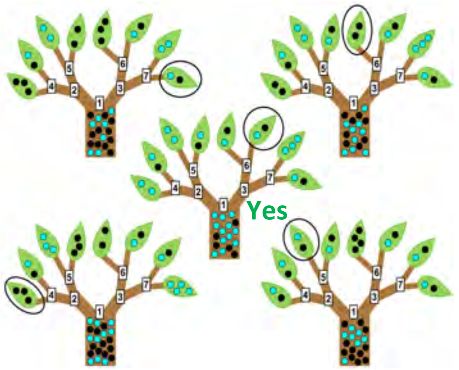
Suppose “income” worked to create the root node

>40	medium	no	fair	yes
<=30	high	no	excellent	no
31-40	medium	no	excellent	yes
<=30	high	no	excellent	no

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31-40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31-40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31-40	medium	no	excellent	yes
31-40	high	yes	fair	yes
>40	medium	no	excellent	no

Building More Trees

- Once you have a tree, restart from the beginning!
  - Take your subset
  - Pick 2 variables randomly
  - Build the first node
  - Repeat down the tree
- In the end, you build hundreds of trees



Testing Your Forest

- Creating a **B**ootstrap and using **agg**regates to build the forest is called **bagging**
- Once the forest is built, take data in your training set that you did ***not*** use for the bootstrap, and test it against the forest
    - Called ***out of bag data***
    - This allows you to test the efficiency of your forest
        - As you know your “buy computer” value, you can test the percentage of trees that get the answer “right” -> this is the forest efficiency

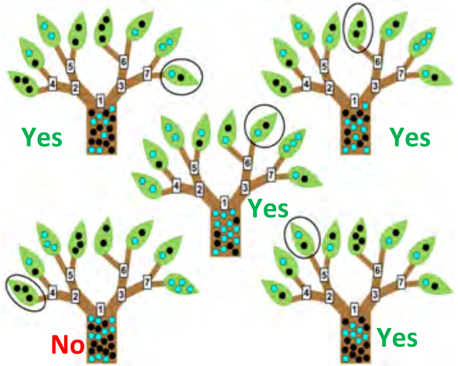
What to Do with All These Trees

When new data comes in (a new potential computer buyer in our example):

- Collect the data:

31-40	medium	no	fair

- Then run the data through each tree:
    - Buy computers?
- The max votes wins!
    - No = 1 , Yes = 4 -> **Yes**



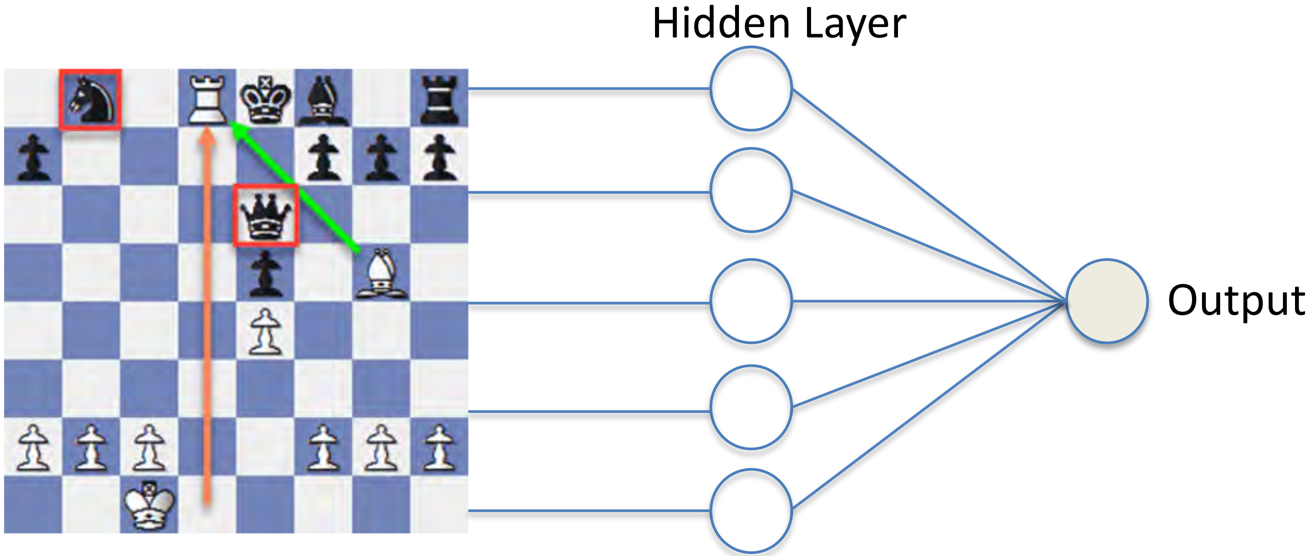
Lesson 4

4.1 Reinforcement Learning

RL vs. Supervised Learning

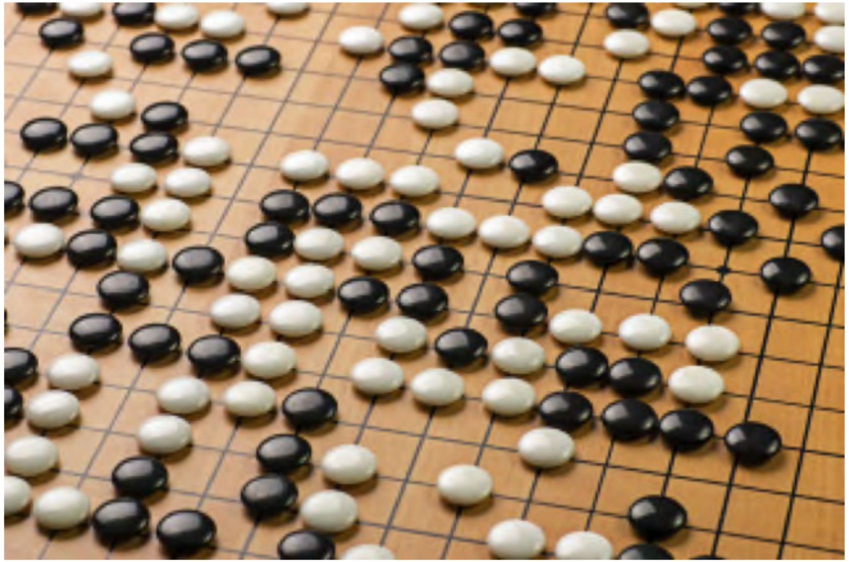
Supervised leaning is great at mapping an input to an output (e.g. find a function that performs A → B mapping)

- Requires lots of training data
- Good at classification and regression



RL vs. Supervised Learning

- In 2016 AlphaGo (Deepmind) defeated the world champion Go master, Lee Sedol
- AlphaGo used innovative moves not seen before (can't do this with supervised learning)



Understanding Reinforcement Learning

How did you learn learn to ride a bike?
- Was it supervised learning?
- Was it unsupervised learning?

No! Trial and error! We learn from experience.

A Feedback / Trial and Error Type of ML

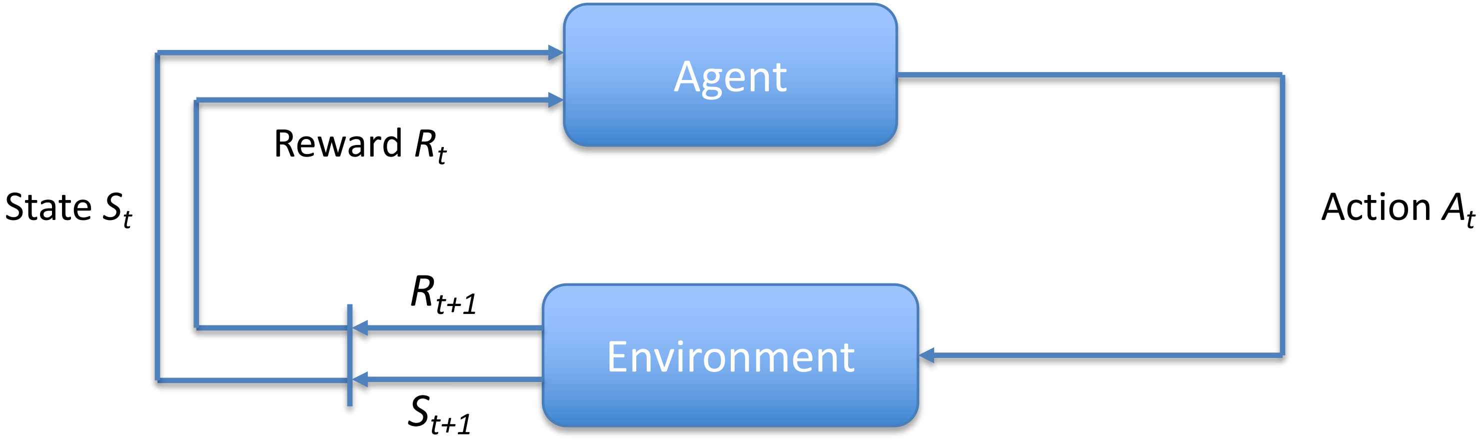
- How do you learn from minimal feedback?
- 2 types of feedback: negative and positive (reward / punishment)
- No external training data or correct/incorrect guidance needs to be given
- Inspired by behavioural psychology (Pavlov's dog experiment)
- A relationship is formed between the input stimulus (bell) and reward (food)

Key Concepts of Reinforcement Learning

- Learn about the environment through rewards and punishment
- When you get a reward or a punishment, your interaction changes and the environment itself can change
- Also involves trial and error (you don’t know the reward / punishment before you try different things)
- RL is also focused on delayed rewards
- Very often this involves a sequence of actions to achieve a reward (e.g. paying chess involves a sequence of moves)
  - Need to learn association between a sequence of actions and a final reward
  - Over time, you are learning a policy
  - Do all this in a stochastic world – there is always an element of randomness and almost infinite branching options to most decisions (how do you train a computer for this?)

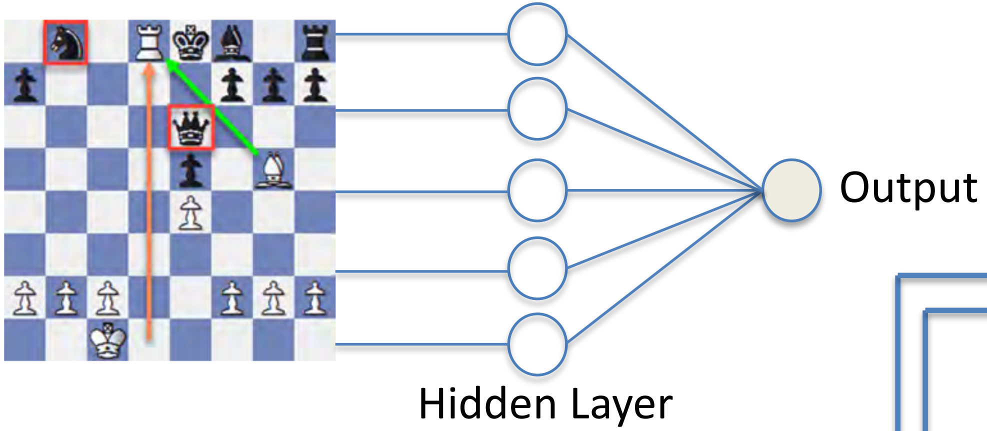
Reinforcement Framework

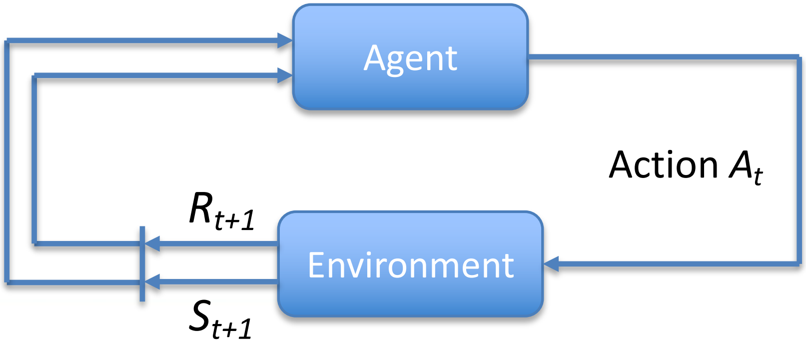
- An agent senses the state of the environment
- Takes an action on the environment
- A reward is given (e.g. -1, 0, +1)
- The state of the environment has changed – the agent gets the update



Policy Networks

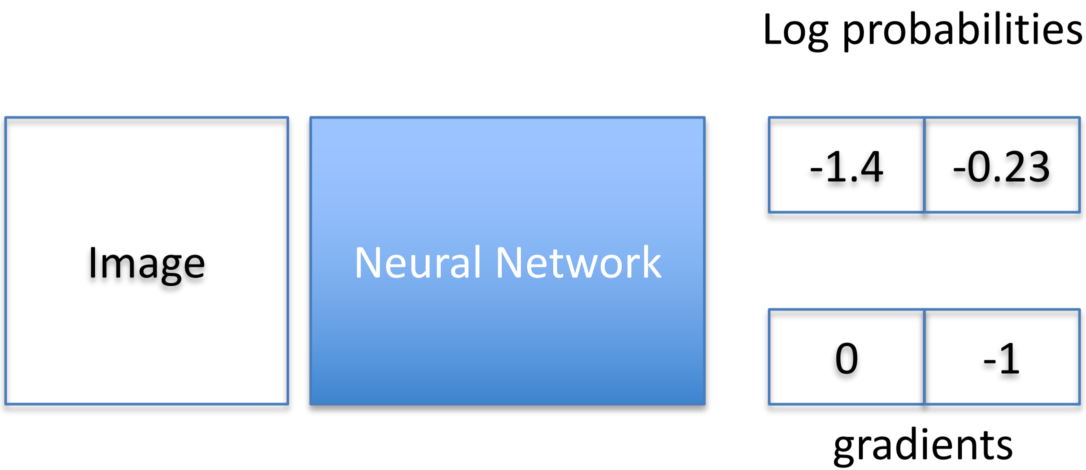
- Begin with a completely random network
- Feed the network a frame from the game engine and it produces an output action (-1, 0, +1) and it sends it back to the game engine
- Loop continues





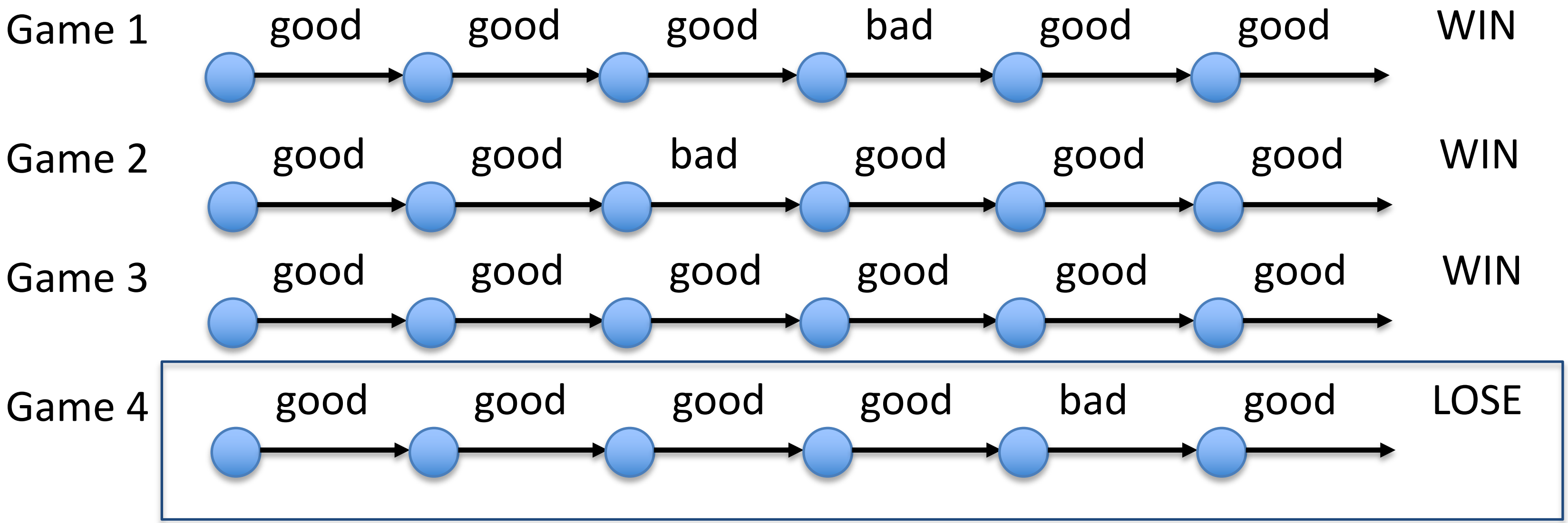
Policy Neural Networks

- Check out Andrej Karpathy's blog on Pong from Pixels



The Credit Assignment Problem

- Sparse rewards don’t reward individual good moves

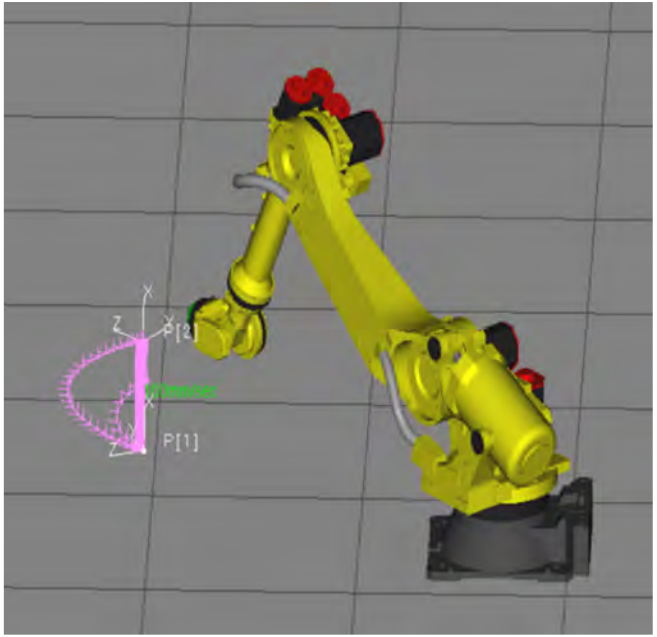


Reward Shaping

- The purpose of reward shaping is to guide your policy to some sort of desired outcome.
- E.g. in chess, you could give your agent a reward every time it takes an opponents piece, or puts your opponent in a well-known difficult position.
- These extra rewards will guide your policy to the desired training outcome
- The downside – it's a custom process for each type of environment
    - Like learning the rules of a game

Challenge of Sparse Rewards with Robotics

- Robots are physically capable of moving in very complex ways, but programming them to do even simple tasks is extremely complicated

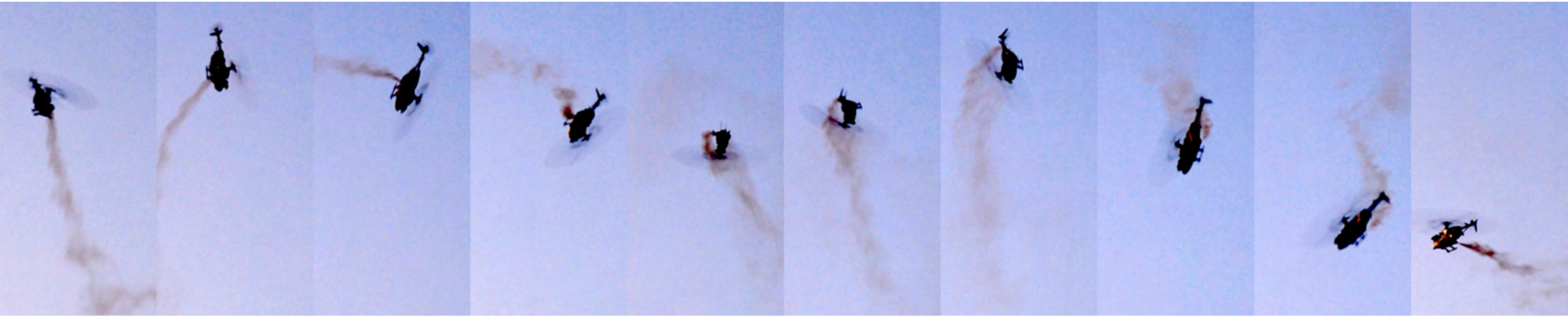


- Robotics is more of a software challenge than a mechanical /hardware one
- Sparse Rewards would only give an award if the robotic arm brought you a cup of tea.
- It might have done 99 things right, but if one was wrong, the cup drops and a negative reward is given.

RL Remains Very Promising

- RL involves careful engineering – balancing reward shaping with desired outcome.
- Autonomous vehicles, robotics, game playing, many more

Stanford University Autonomous helicopter aerobics demonstration



Lesson 5: Unsupervised Learning

Lesson 5: Unsupervised Learning

5.1 Understanding Unsupervised Learning

Unsupervised Learning

- You **do not** know the right answer, and there is too much data for you to guess
  - Example: you discover a geyser
  - It erupts at intervals… you sense that the eruption strength relates to the rainfall, but you don't know how
  - You want to predict the characteristics of the next eruption…
  - **How do you do that?**

Unsupervised Learning

1. You collect data about the eruptions (e.g. height, duration, and the amount of rainfall over the previous 3 days)
2. Then you plot
3. Then you ask the system: can you tell me if there are patterns?

Raw Data

Unsupervised Learning

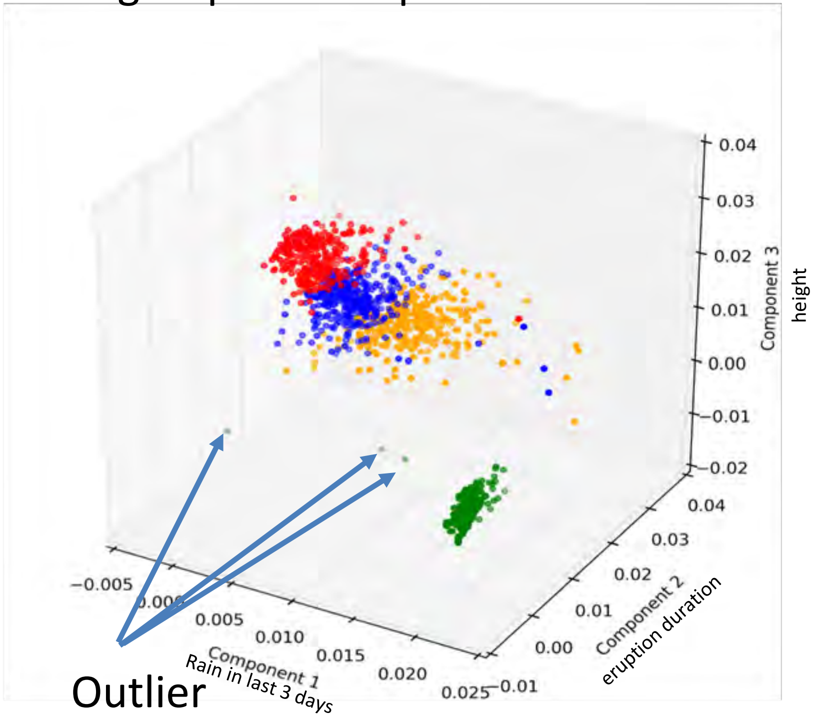
Your graph will group eruptions that have similar characteristics.

- In math, this is simply grouping points that are close to one another

And will spot the outliers

- Those are the abnormal eruptions

4 groups = 4 eruption families



Lesson 5: Unsupervised Learning

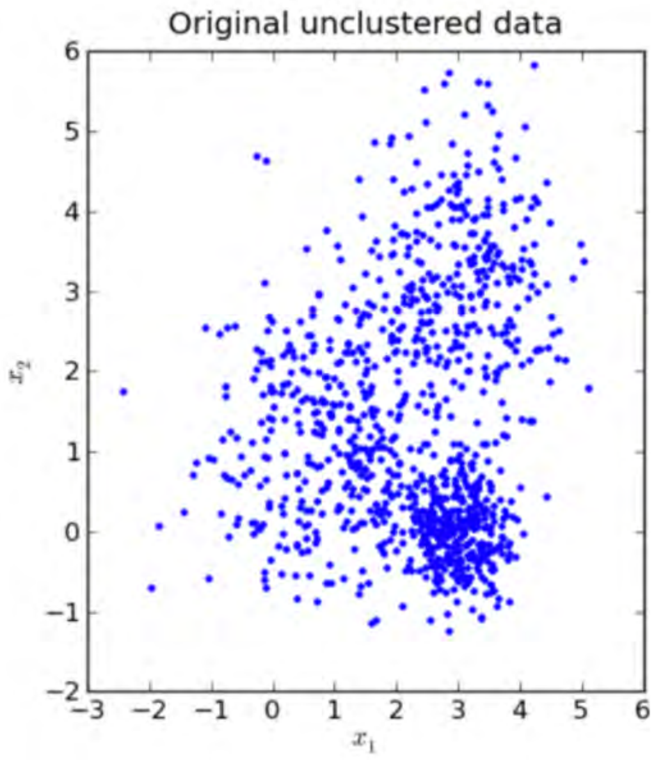
5.2 Explaining the K-Means Algorithm

Unsupervised Learning

The math can take may forms, but a common form is **K-means**

K-means is based on the idea of groups

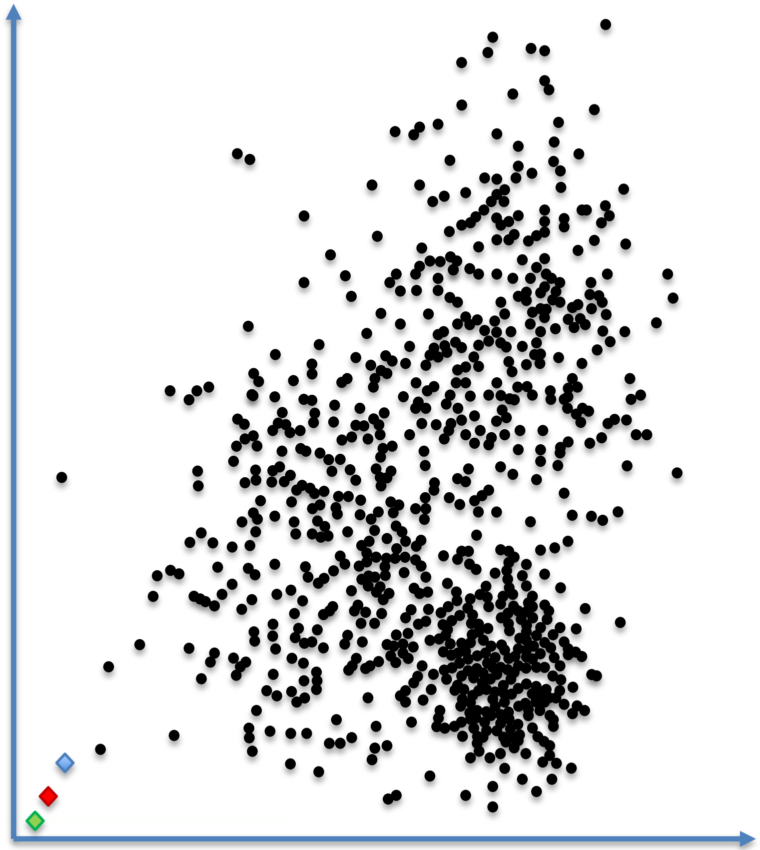
You choose a group number, then compute the best group membership for each point on the graph



K-means Example: 3 Clusters

Generate 3 random points

- We call them centroids (μ_1, μ_2, μ_3)
- Each has a position on the graph:
  - μ_1 ($x_{\mu 1}$, $y_{\mu 1}$), μ_2 ($x_{\mu 2}$, $y_{\mu 2}$), μ_3 ($x_{\mu 3}$, $y_{\mu 3}$)

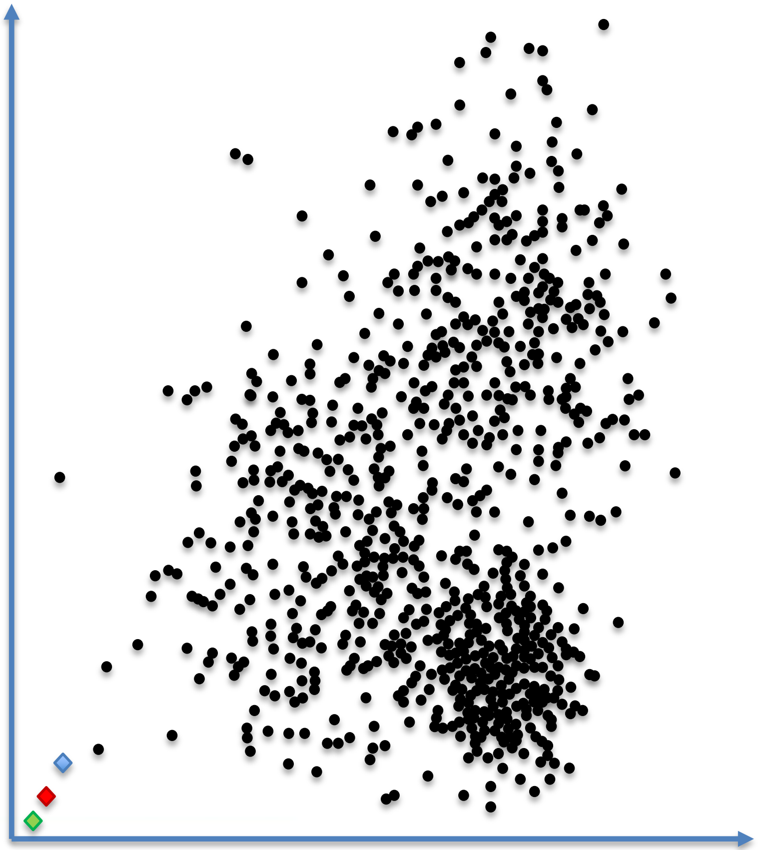


K-means Example: 3 Clusters

Compute each point distance to each centroid

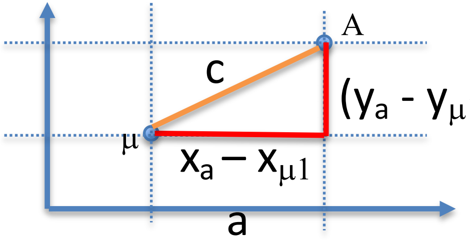
This can be as simple as, for each point A (x_a, y_a), compute:

$$\mathrm{sqrt}[(x_a - x_{\mu 1})^2 + (y_a - y_{\mu 1})^2]$$

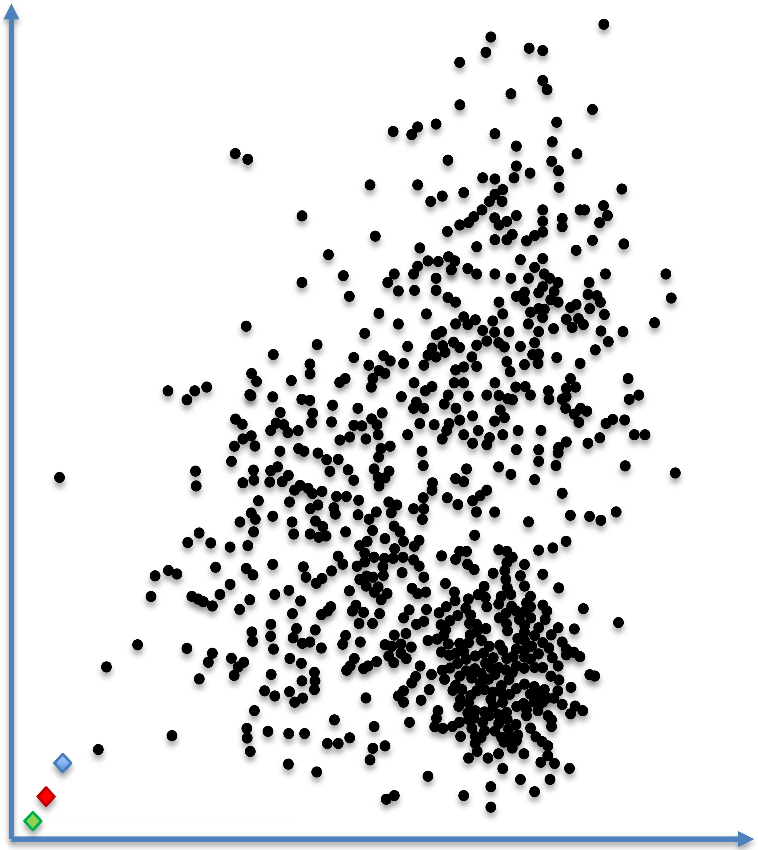


K-means Example: 3 Clusters

$\mathrm{sqrt}[(x_a - x_{\mu 1})^2 + (y_a - y_{\mu 1})^2]$



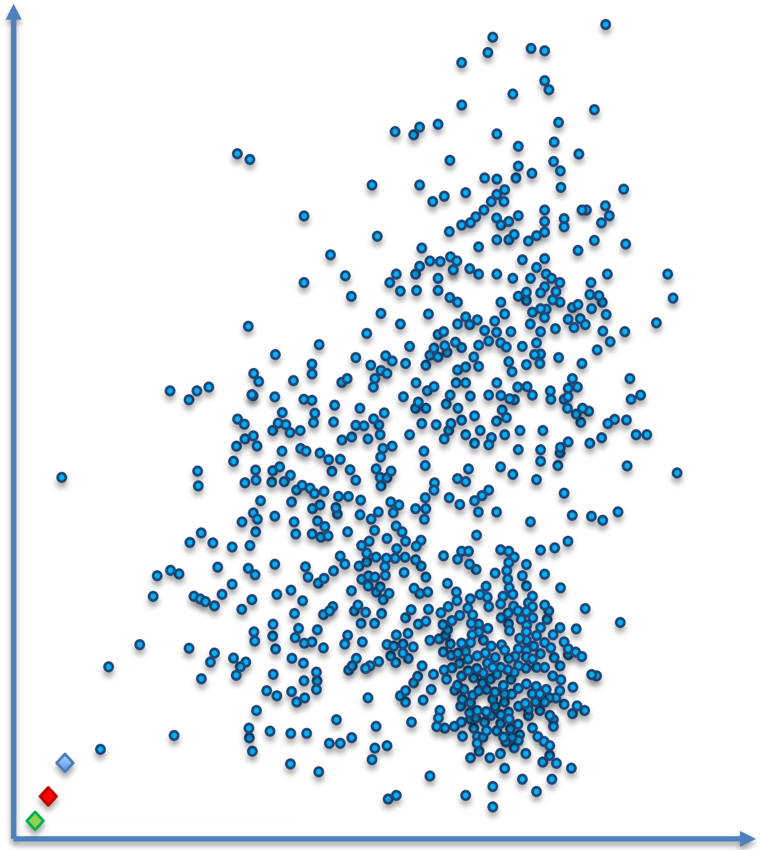
$a^2+b^2=c^2$



K-means Example: 3 Clusters

Compute each point distance to each centroid

Then associate each point to the closest centroid

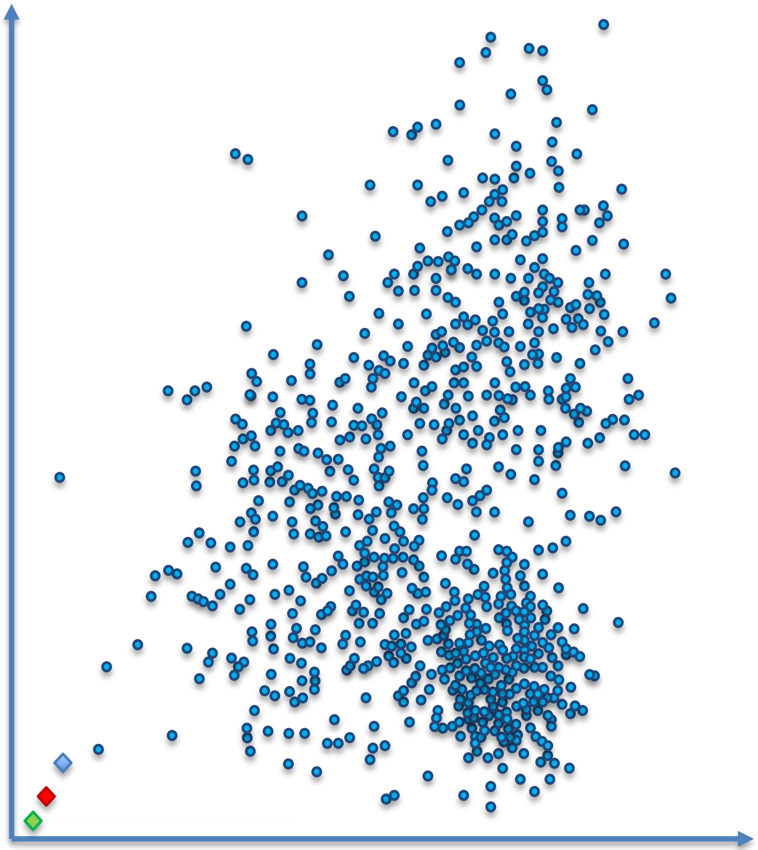


K-means Example: 3 Clusters

Then compute the center of the cluster
(all points belonging to each centroid)

Here, all belong to blue!

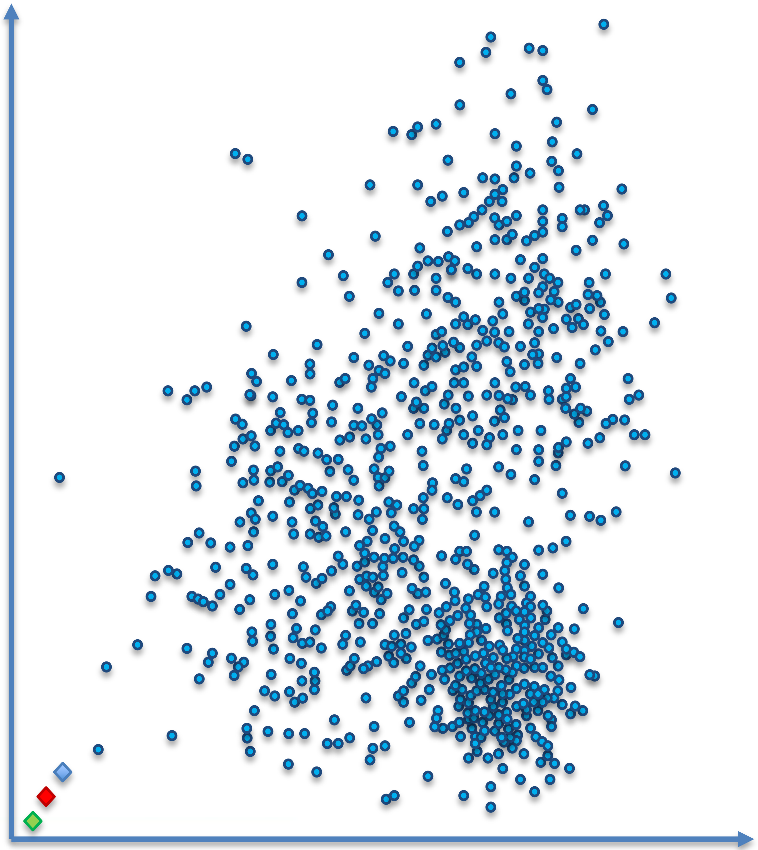
center is $\left(\dfrac{all\ x}{number\ of\ points}, \dfrac{all\ y}{number\ of\ points}\right)$



K-means Example: 3 Clusters

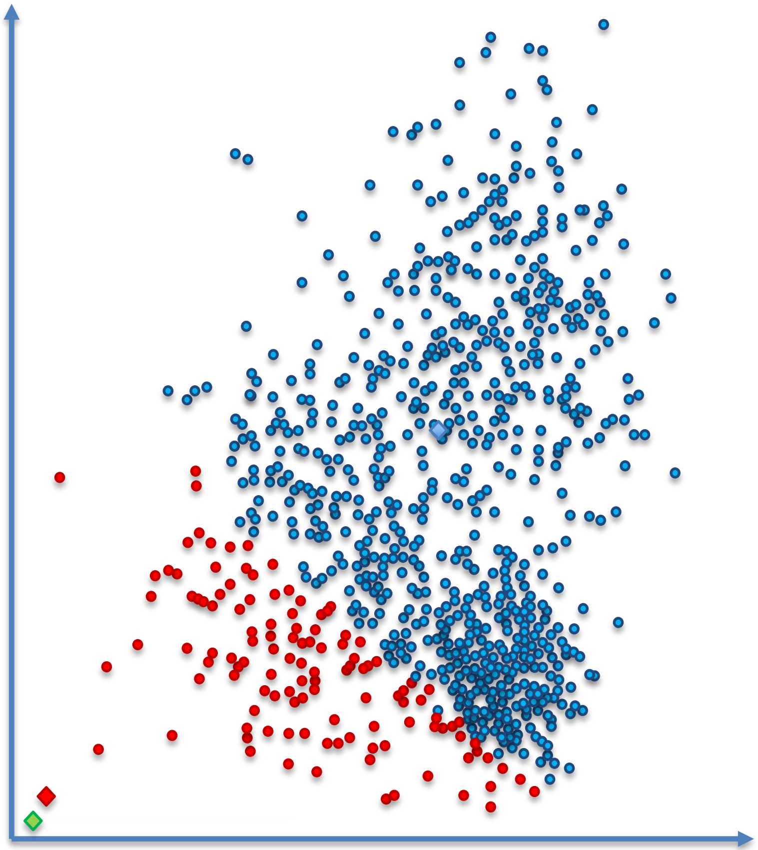
Move your centroids there

(here **red** and **green** do not move as they have no member)



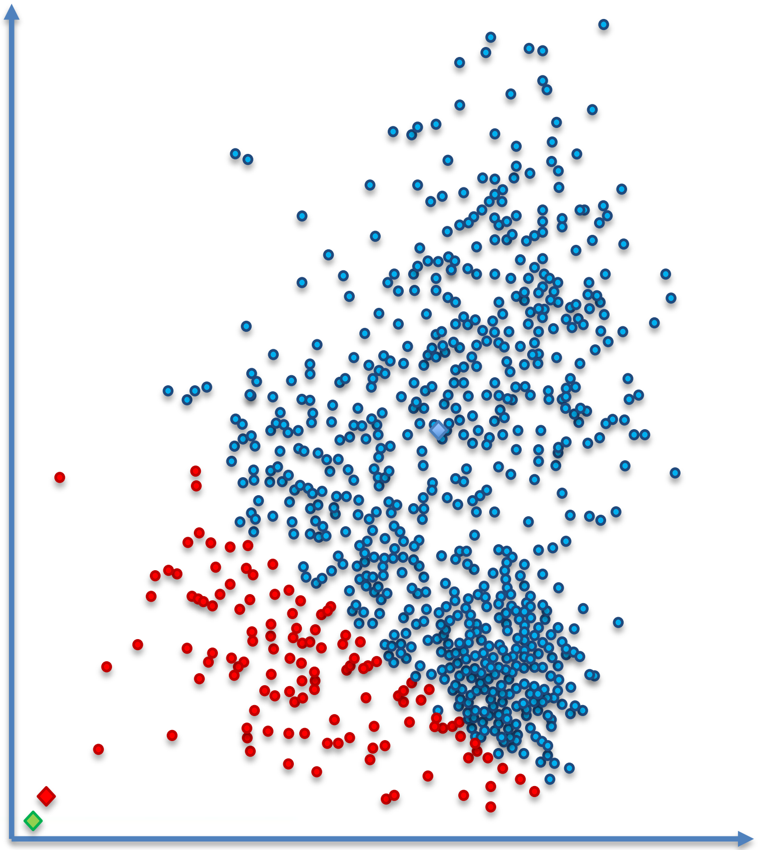
K-means Example: 3 Clusters

Then repeat (compute each point distance to each centroid, associate to closest centroid)



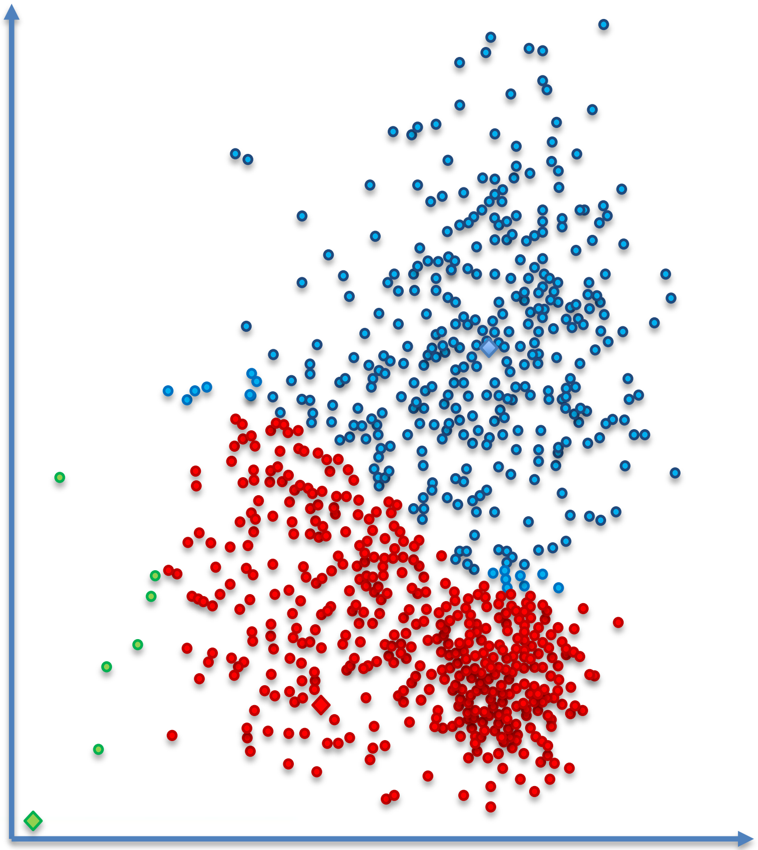
K-means Example: 3 Clusters

Move again each centroid to the cluster center



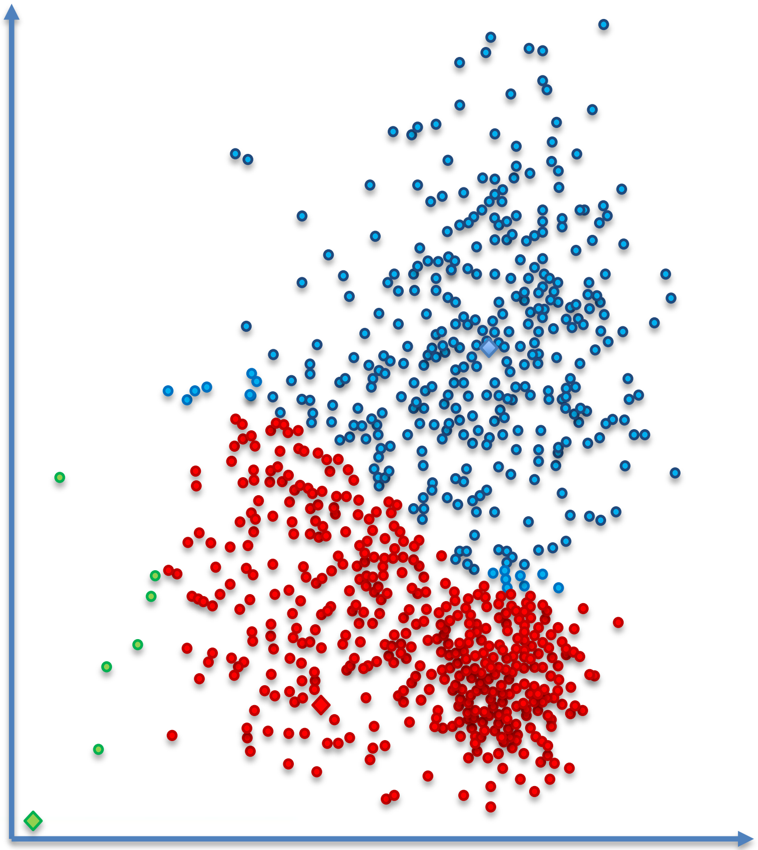
K-means Example: 3 Clusters

Then repeat (compute each point distance to each centroid, associate to closest centroid)



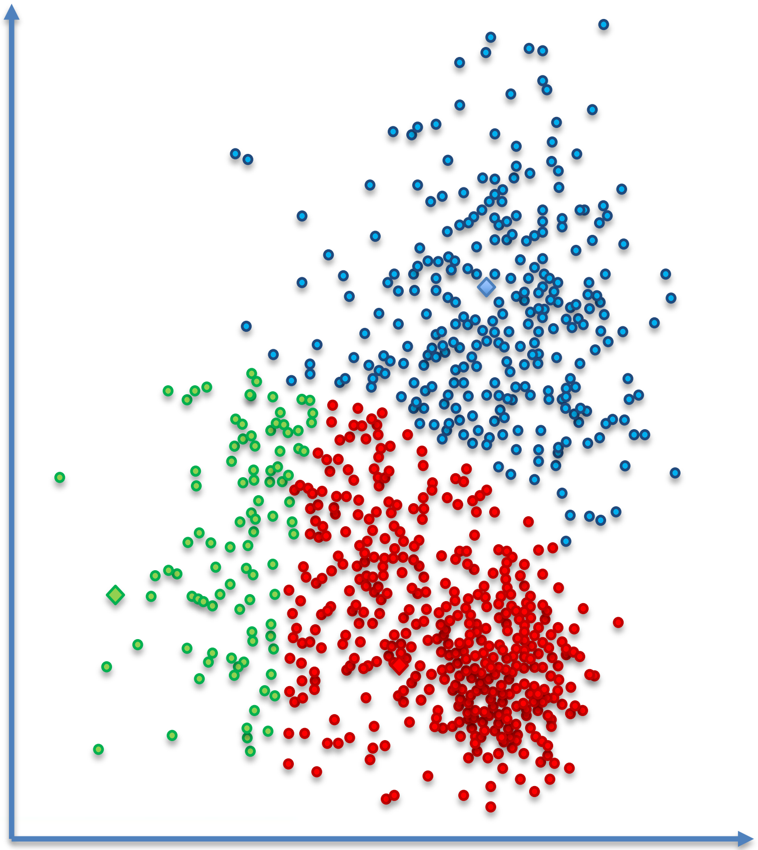
K-means Example: 3 Clusters

Move again each centroid to the cluster center



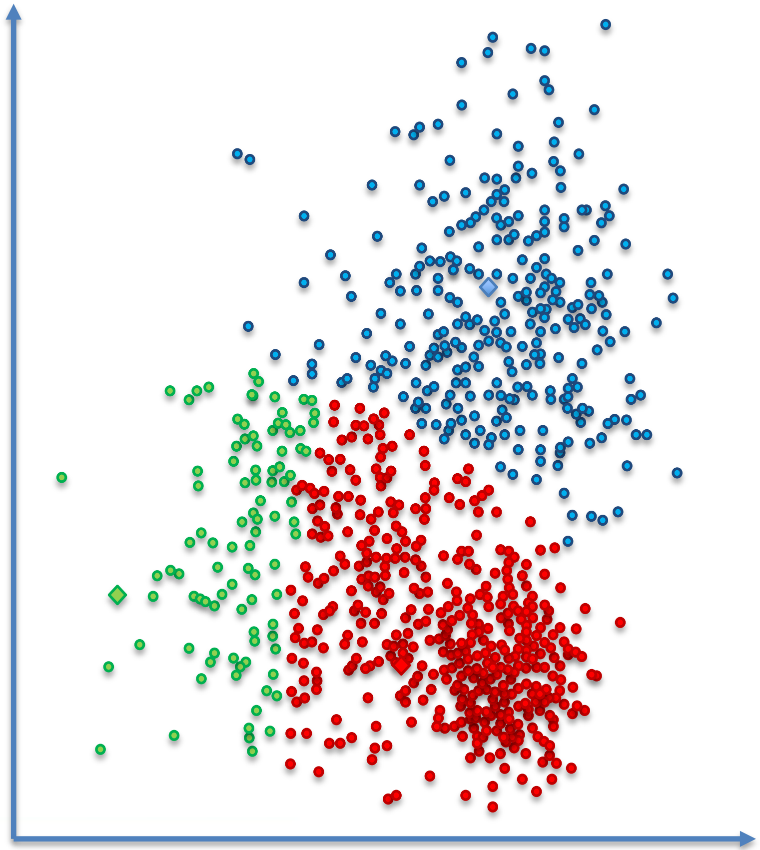
K-means Example: 3 Clusters

Then repeat (compute each point distance to each centroid, associate to closest centroid)



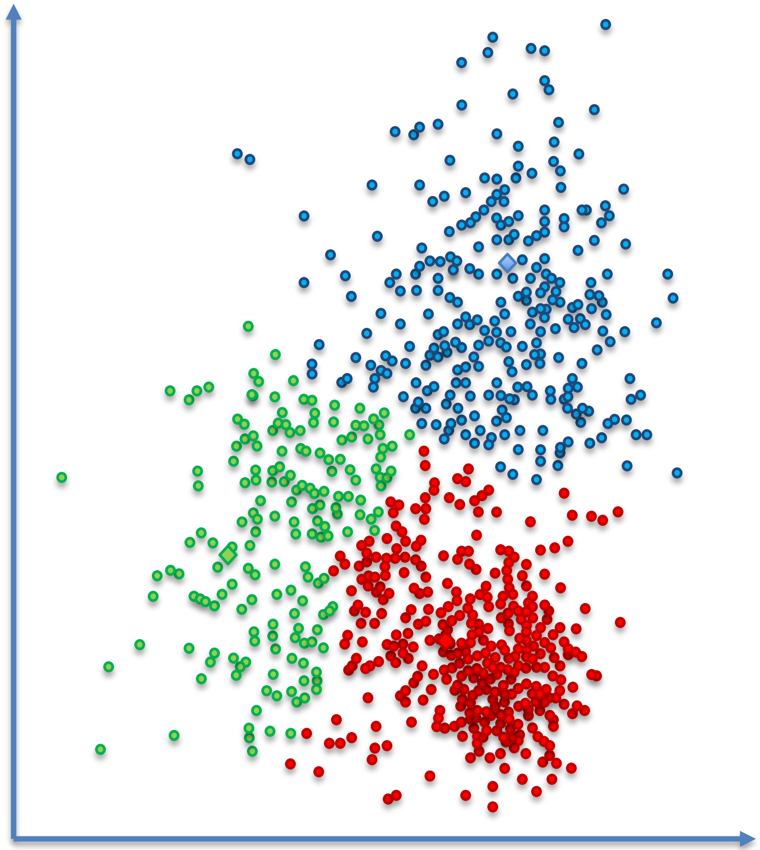
K-means Example: 3 Clusters

Move again each centroid to the cluster center



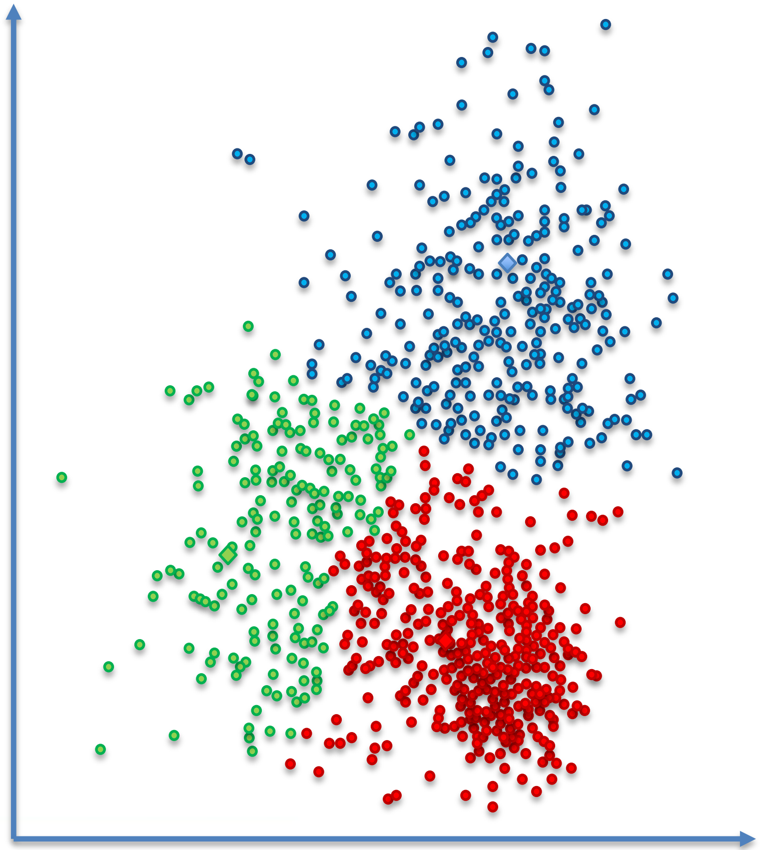
K-means Example: 3 Clusters

Then repeat (compute each point distance to each centroid, associate to closest centroid)



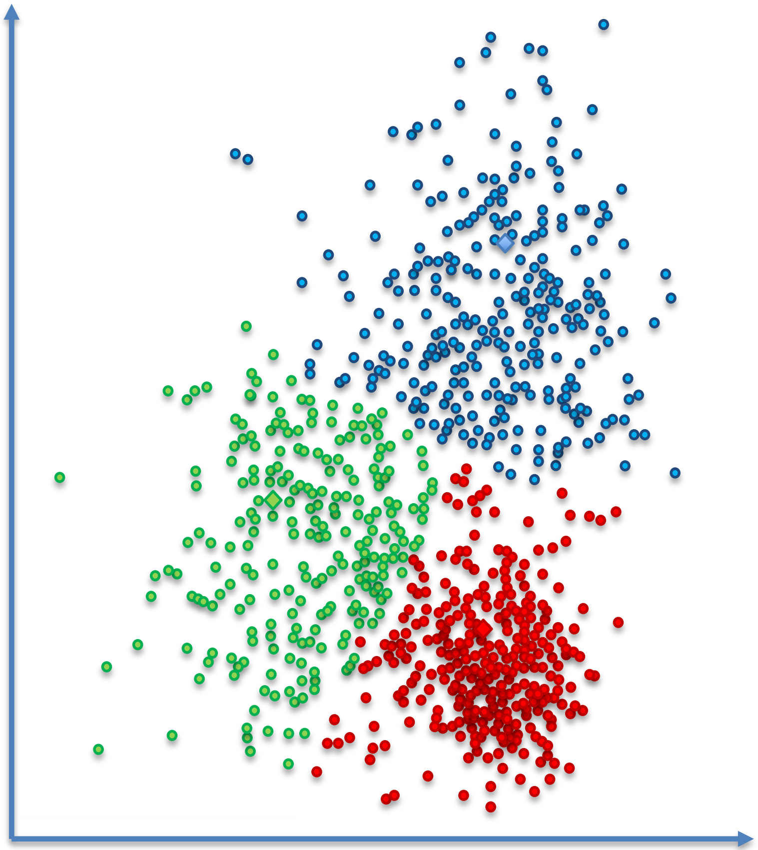
K-means Example: 3 Clusters

Move again each centroid to the cluster center



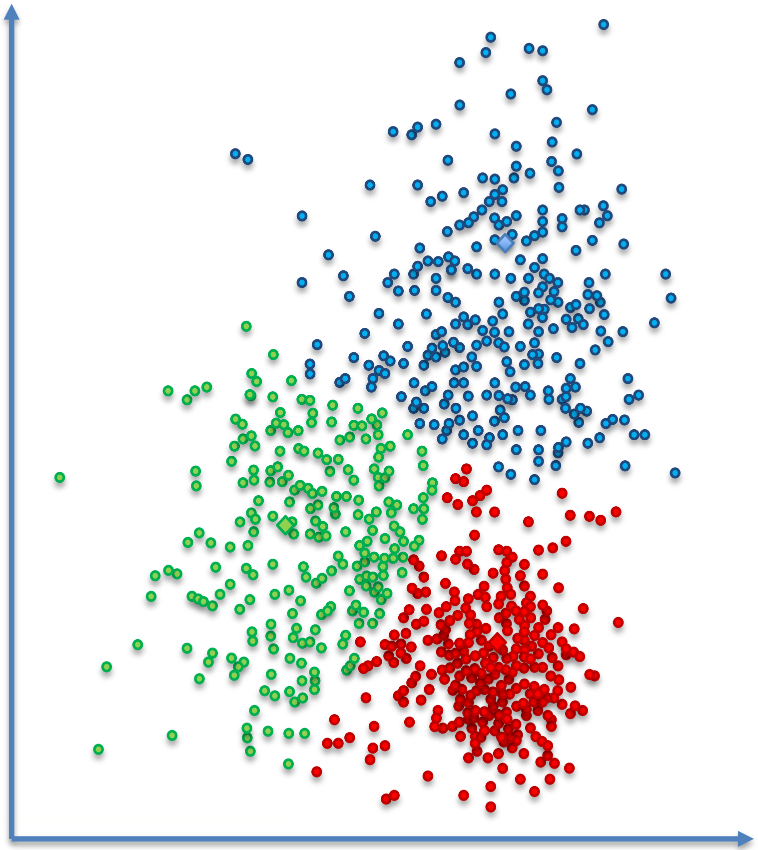
K-means Example: 3 Clusters

Then repeat (compute each point distance to each centroid, associate to closest centroid)



K-means Example: 3 Clusters

Move again each centroid to the cluster center

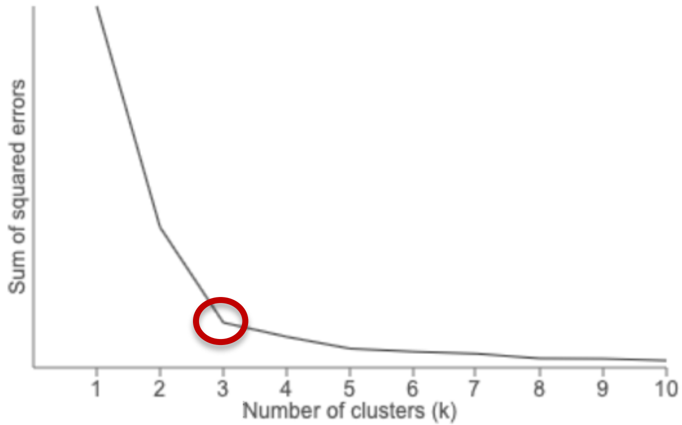


K-means Challenges

A difficulty with K-means is that you need to choose K

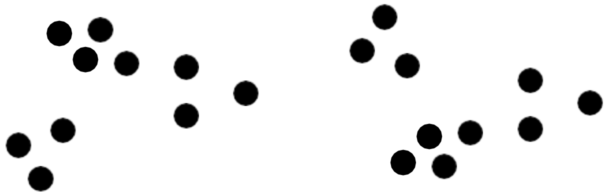
If you are unsure, you can use the ***elbow method***:

- Use several K values, and for each K, compute the Sum of Squared Errors (SSE) – this is the distance of each point to its centroid
- When K=1, SSE is high
- When K = number of points, SSE = 0
- You are looking for the number of “Ks” where adding one more K does not add much benefits

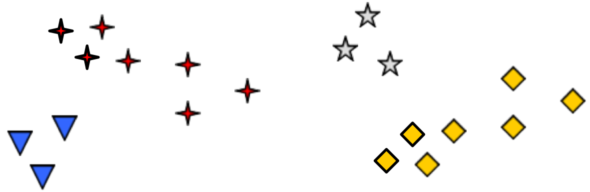


K-means Challenges

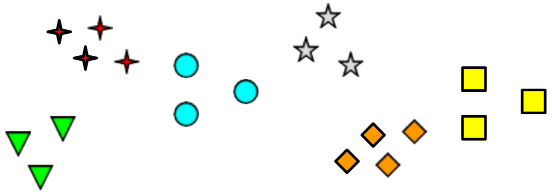
"Cluster" is not a well-defined concept – sometimes, the elbow method is not sufficient



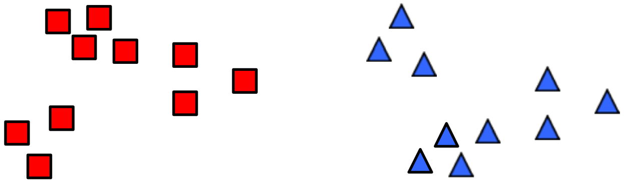
How many clusters?



Four Clusters



Six Clusters

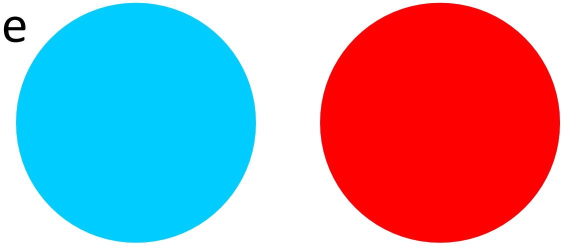


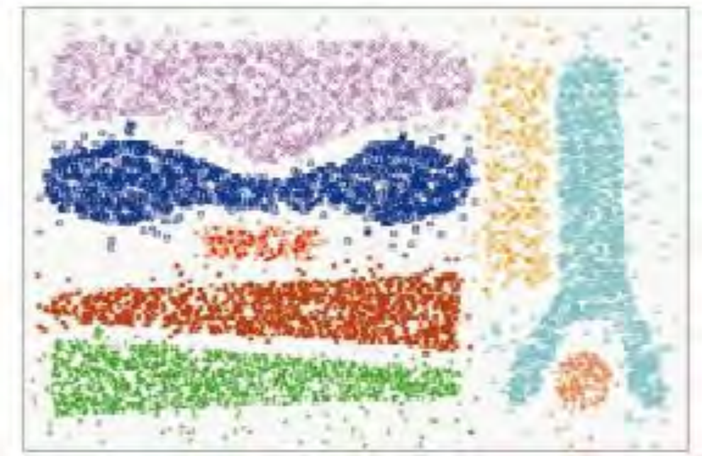
Two Clusters

K-means Challenges

When clusters are well separated, you can usually decide

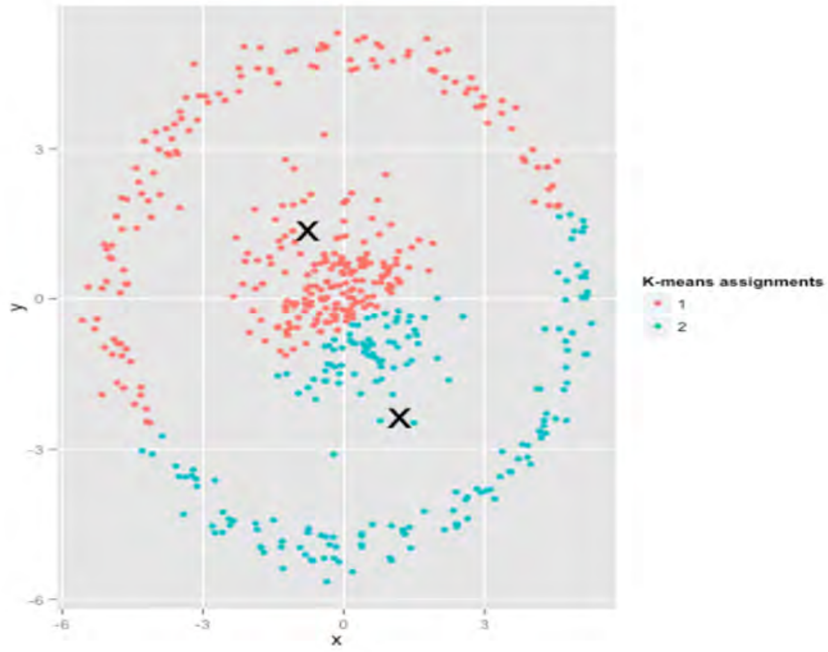
Even when their shape is surprising

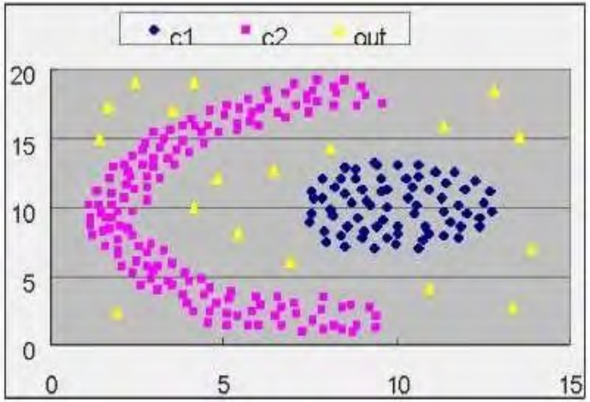




K-means Challenges

However, K means also fails on certain shapes





Lesson 5: Unsupervised Learning

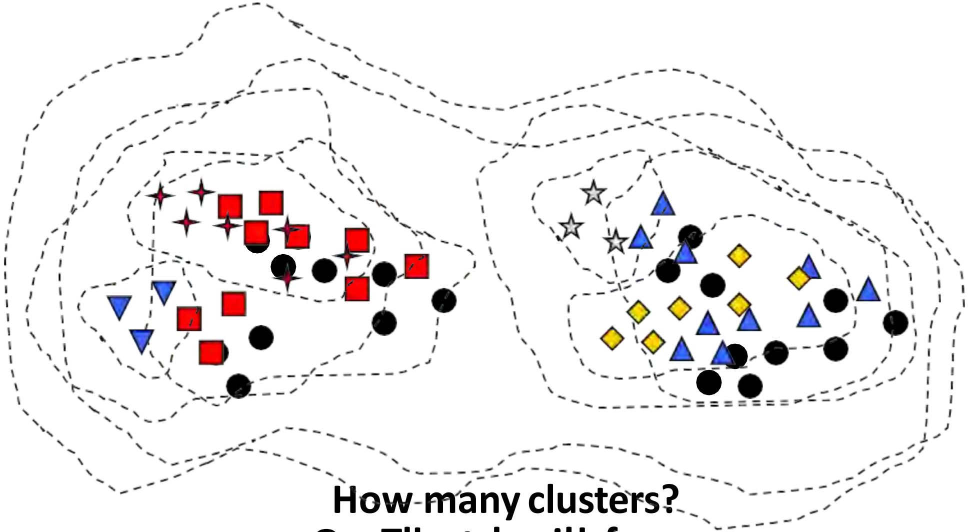
5.3 An Overview of Other Common Algorithms

Pearson

Hierarchical Clustering

A process to create clusters within clusters

- Works similarly as K-means, but towards the opposite direction
- Create n clusters, then identify neighbor clusters and merge them
- Repeat until there is a single cluster



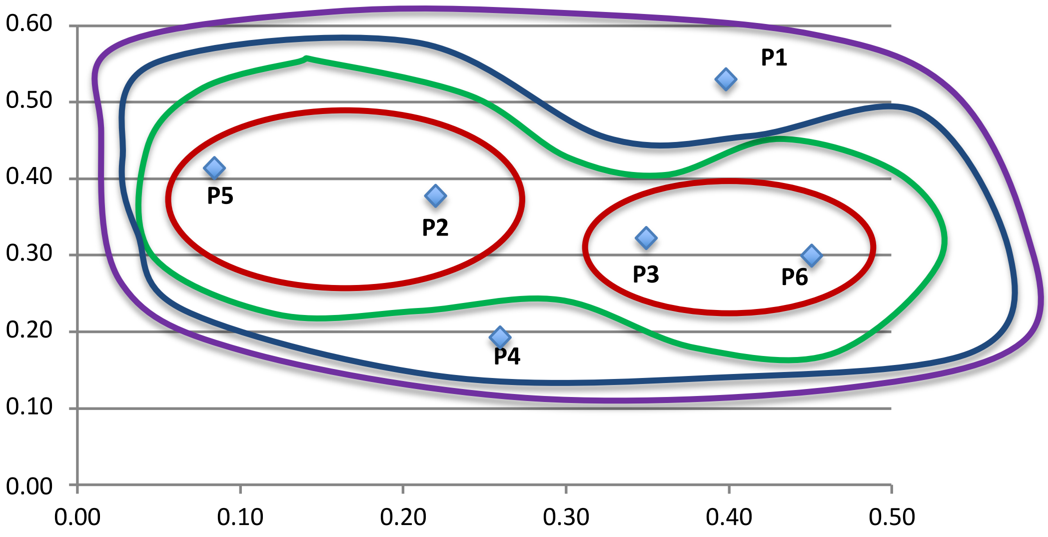
How many clusters?

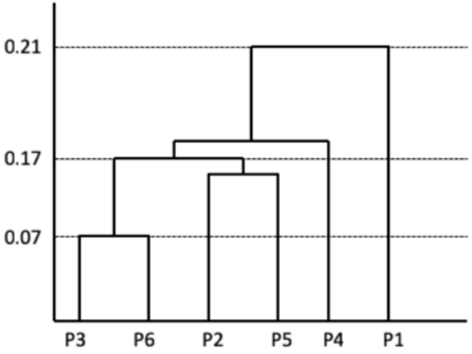
One cluster with two sub-clusters.

Three level of hierarchical clusters.

Hierarchical Clustering Illustration

The result can also be represented as a dendrogram, showing the relationships and distances between clusters

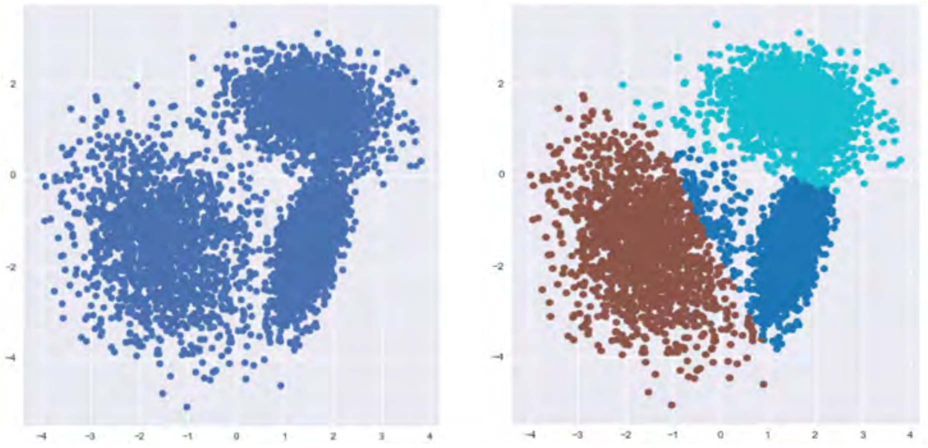




Suppose we found the following 6 centroids

DBSCAN

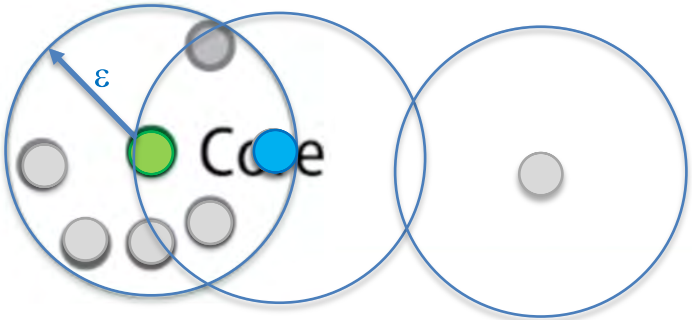
- Density Based Spatial Clustering of Applications with Noise
  - Add “Hierarchical” to make it HDBSCAN
- Does not need you to input the number of clusters
- Good with “noisy” or “messy” data
- Allows “noise” or “outliers” to exist outside of the clusters



How DBSCAN Works

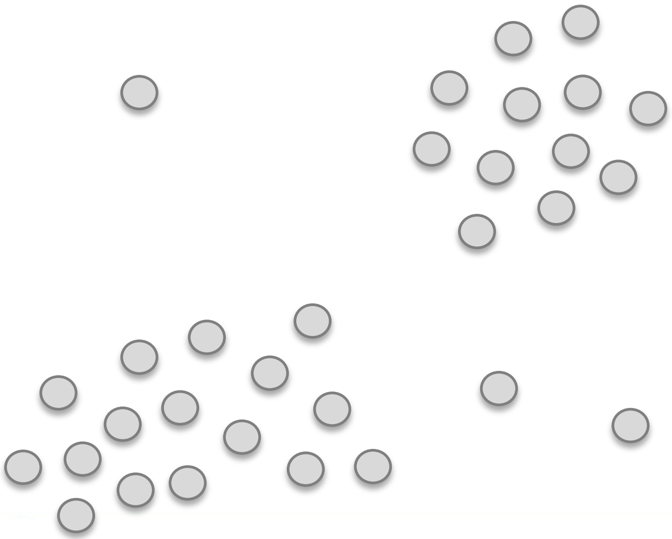
- You need to define some parameters:
  - How many points (minimum) in a cluster (MinPts)
  - The "density", i.e. the minimum radius of a cluster (Epsilon, ε)
- DBSCAN understands 3 types of points:
  - Core points have MinPts or more neighbors within a radius of ε
  - Border points have a core neighbor, but less than MinPts in their ε radius
  - Noise points are neither core nor border

MinPts = 4



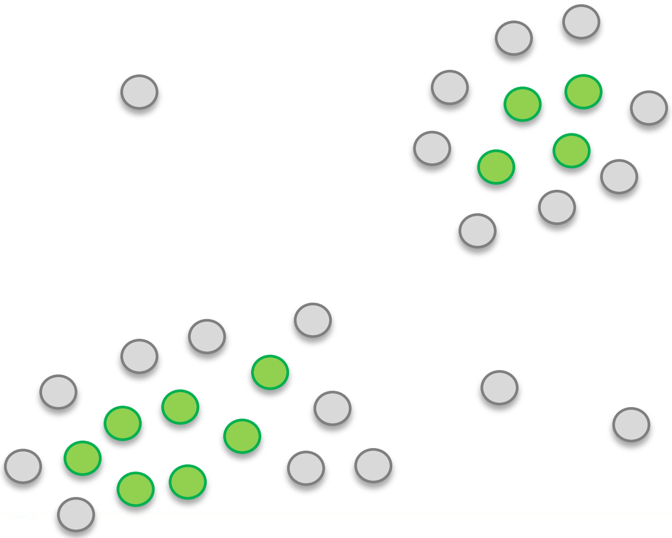
Running DBSCAN

DBSCAN first computes each point's ε, then counts neighbors to determine which points are core



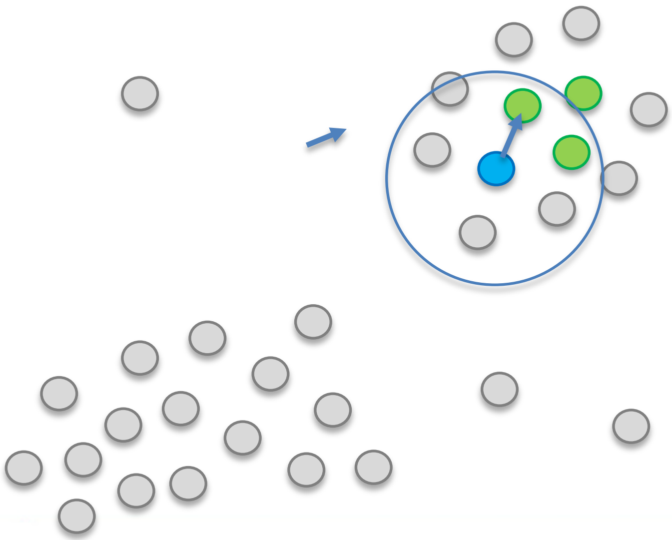
Running DBSCAN

DBSCAN first computes each point's ε, then counts neighbors to determine which points are core



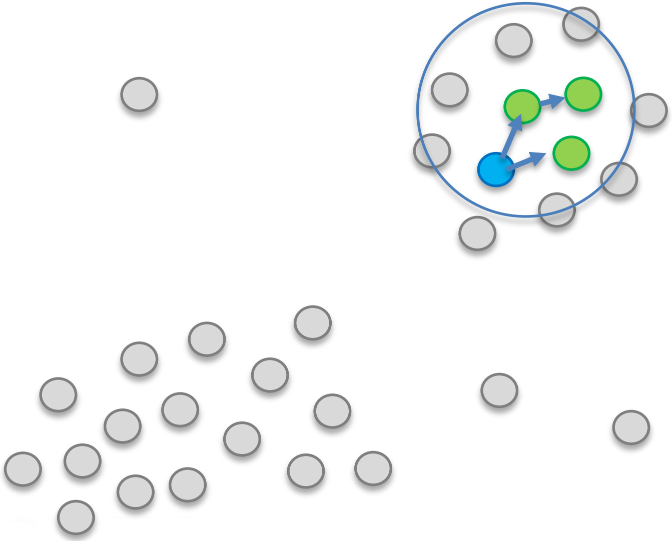
Running DBSCAN

- DBSCAN first computes each point's ε, then counts neighbors to determine which points are core
- Then DBSCAN picks a (random) core point, and selects all its core neighbors



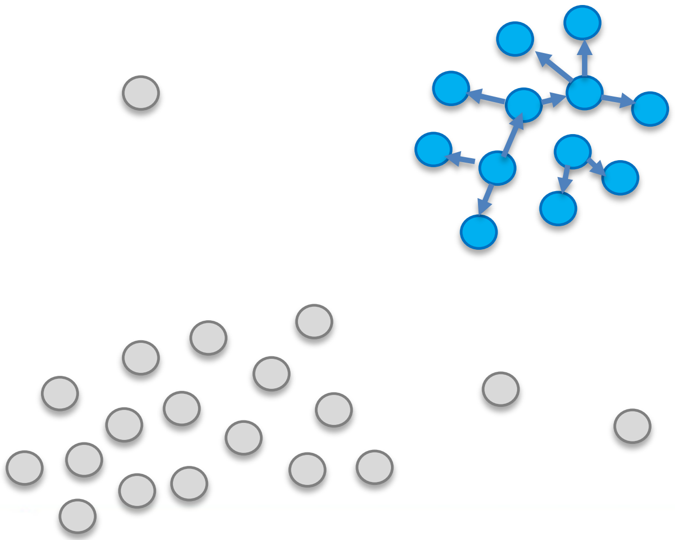
Running DBSCAN

- Repeat until all cores in the cluster are found and linked
- Then find all neighbors (within ε) of all cores



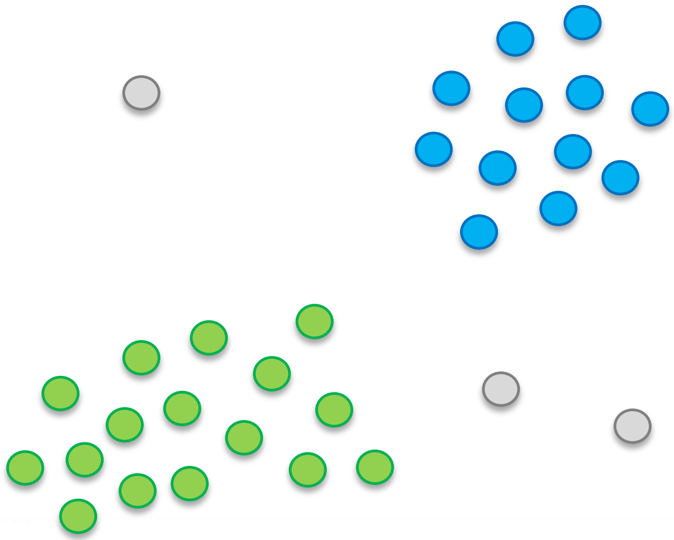
Running DBSCAN

- Repeat until all cores in the cluster are found and linked
- Then find all neighbors (within ε) of all cores
- This is your first cluster



Running DBSCAN

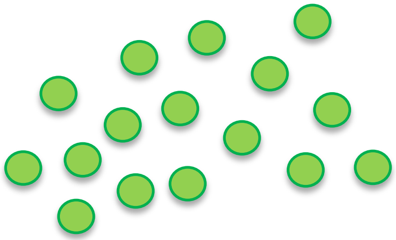
- Repeat until all cores are explored -> all clusters are formed
  - The cluster count has been automatically determined
  - Outliers are allowed outside of clusters

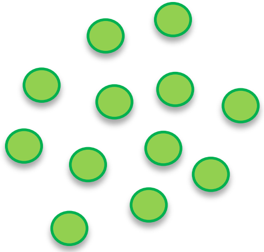


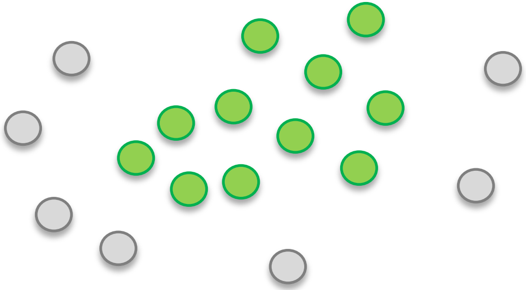
DBSCAN and Beyond

DBSCAN solves some limitations of K-Means, but is sensitive to parameters:

- ε too large -> gigantic cluster
- ε too small -> sparse cluster is seen as noise







DBSCAN and Beyond

- There are many other methods (Gaussian Mixtures Models, Gaussian expectation-minimization, k-harmonic, fuzzy k-means, and many others)
- Start with K-means, think of how you want to graph your data, and explore beyond if it fails…

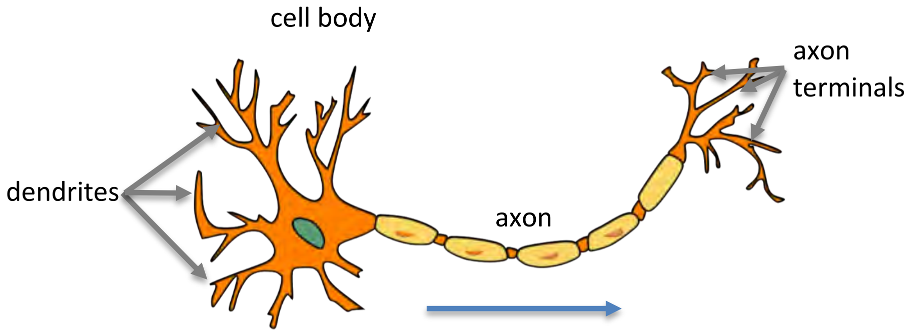
Lesson 6: Deep Learning and Artificial Neural Networks

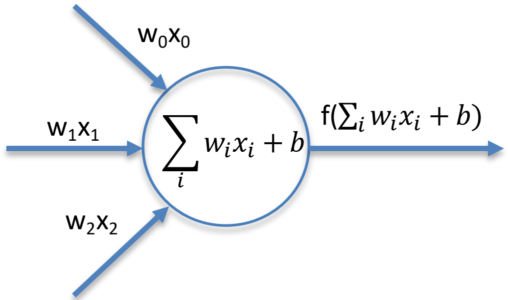
Lesson 6: Deep Learning and ANNs

6.1 Understanding Deep Learning and Neural Networks

Neural Networks

Just like a neuron gets input (electricity) from one or more dendrites, and fires output (electricity in axon) if the input gets beyond a threshold

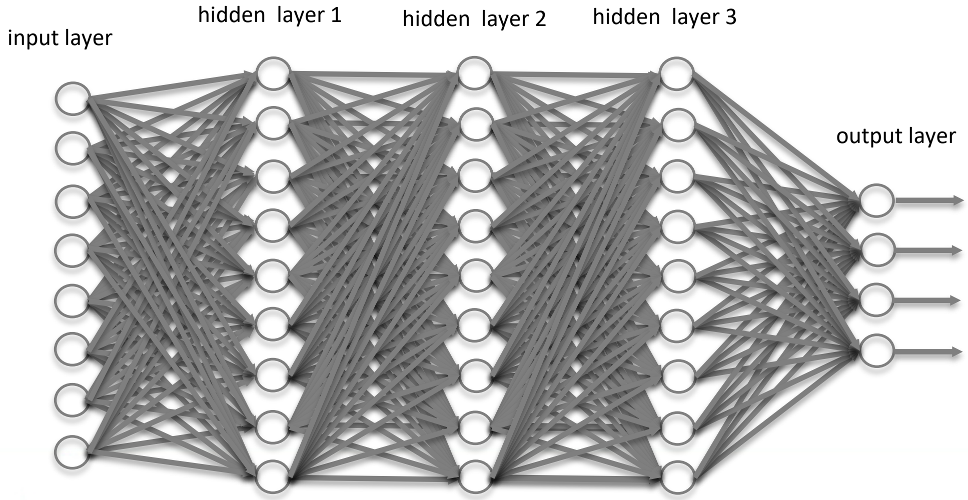




A neural network unit gets inputs, and outputs 1 if the combined input is beyond a given threshold

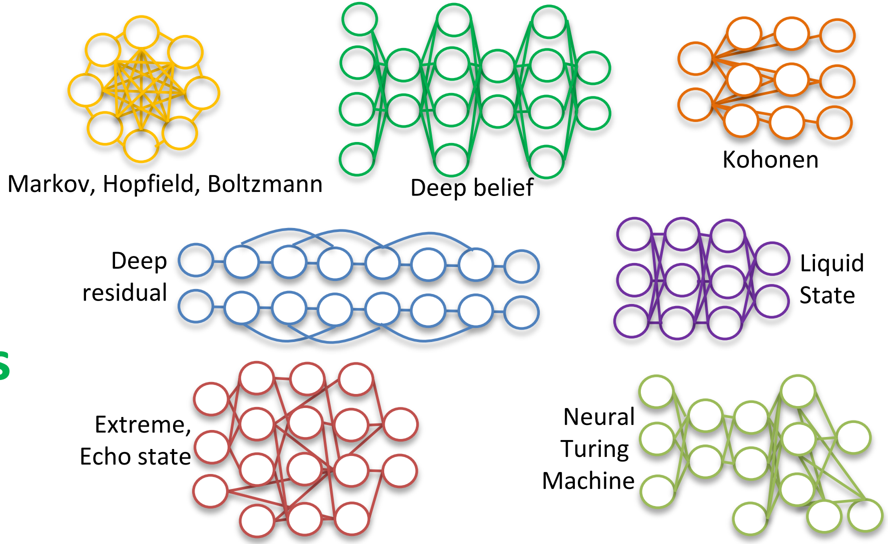
Neural Networks

- Also called Artificial Neural Networks (ANNs)
- Rows of units (neurons) are layered (if multiple layers are used, it is called ***deep learning***)



Neural Networks

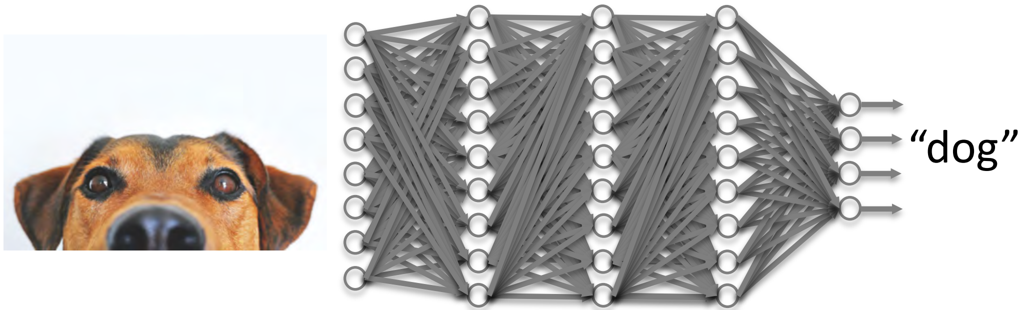
- The way you connect the units can vary immensely
- And this is what makes this **family very rich**
- **Tons of possible applications** depending on what data you are looking at, and what you try to find
- Evolution networks help learn new structures

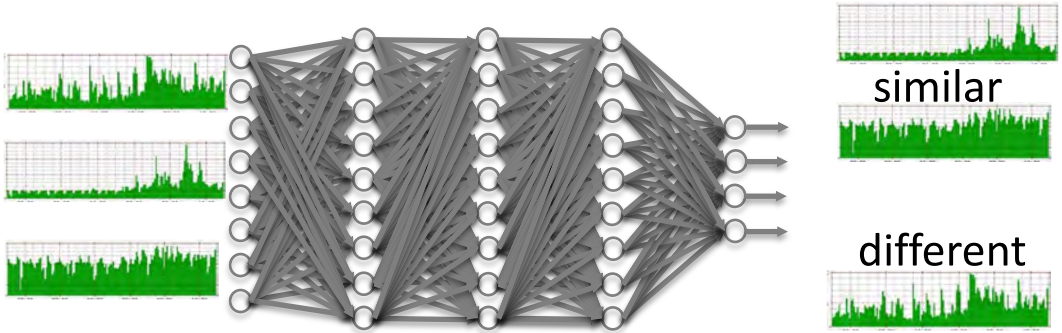


Try a few combinations in parallel, mate the best pairs, introduce small (2%) mutations

Supervised or Unsupervised?

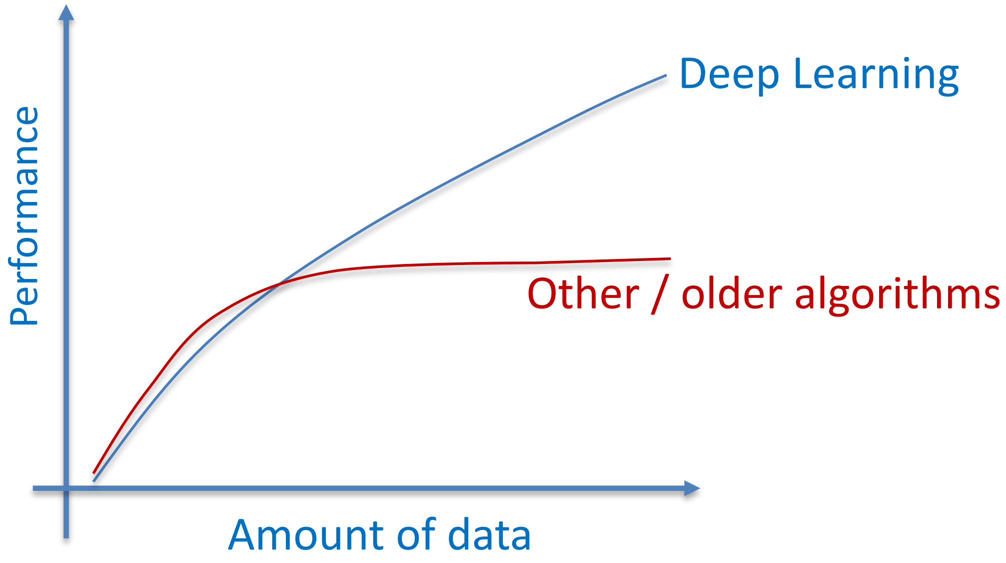
Neural Networks can be either!





Why Is Deep Learning So Important?

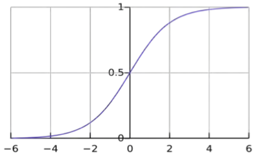
Since the mid-1990s, DL networks have started to outperform other algorithms in pattern learning tasks



Deep Learning vs. Standard Regression

input

feature extraction

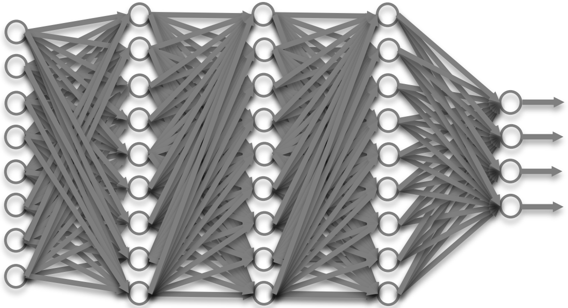


classification

car
not car

output

input



feature extraction AND classification

car
not car

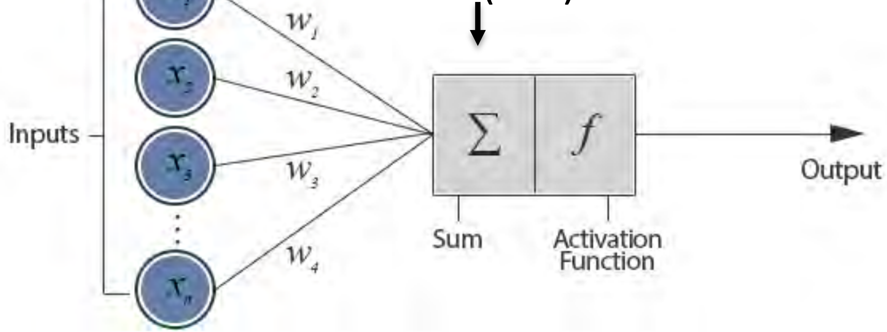
output

Lesson 6: Deep Learning and ANNs

6.2 Deep Learning Process Example

What Does a Single Neuron Do?

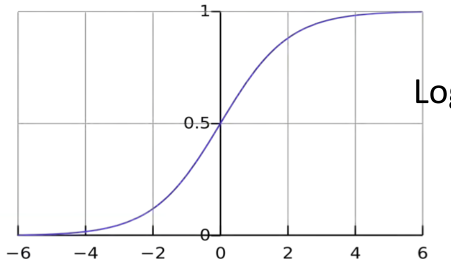
Inner (dot) Product



- The magic of DL happens in the way you train the weights on the neurons
- Uses a mechanism called ***back propagation***

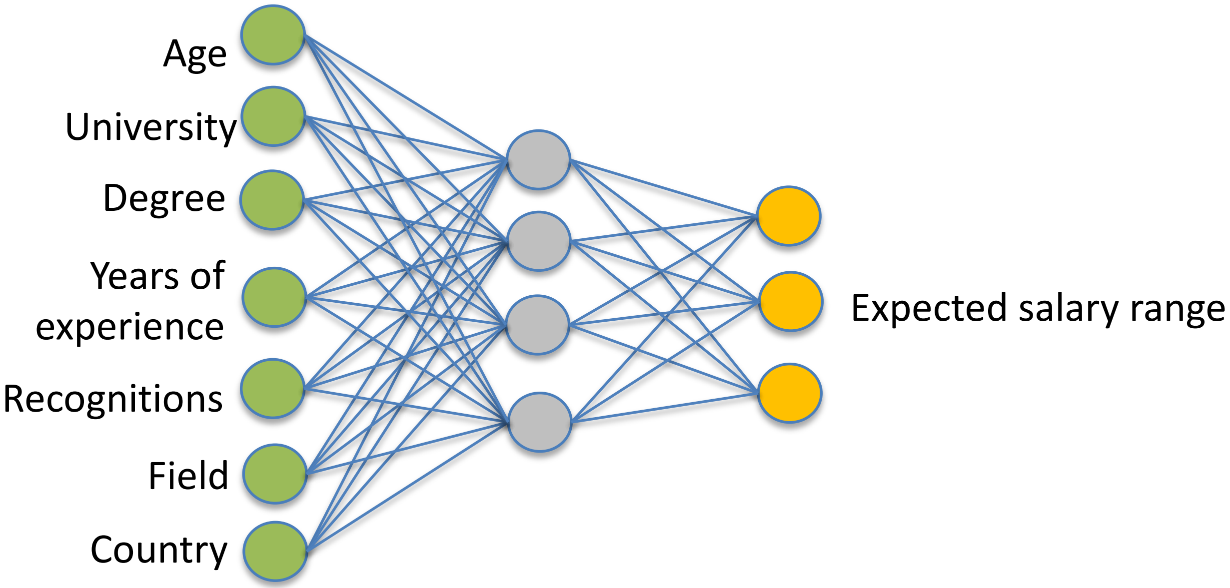
$$\boldsymbol{W}^T\boldsymbol{x} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \sum_{i=1}^{n} w_i x_i = w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

$$f(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}) = \sigma(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}) = \frac{1}{1 + e^{-(\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b})}}$$

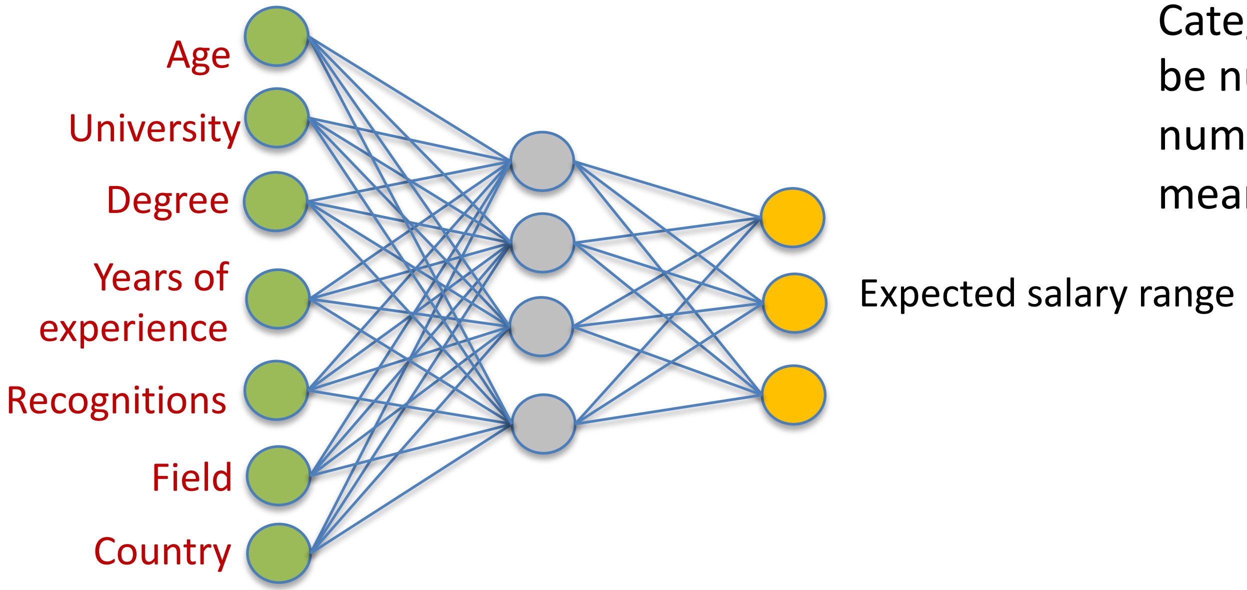


Logistic / sigmoid function

Deep Learning Example

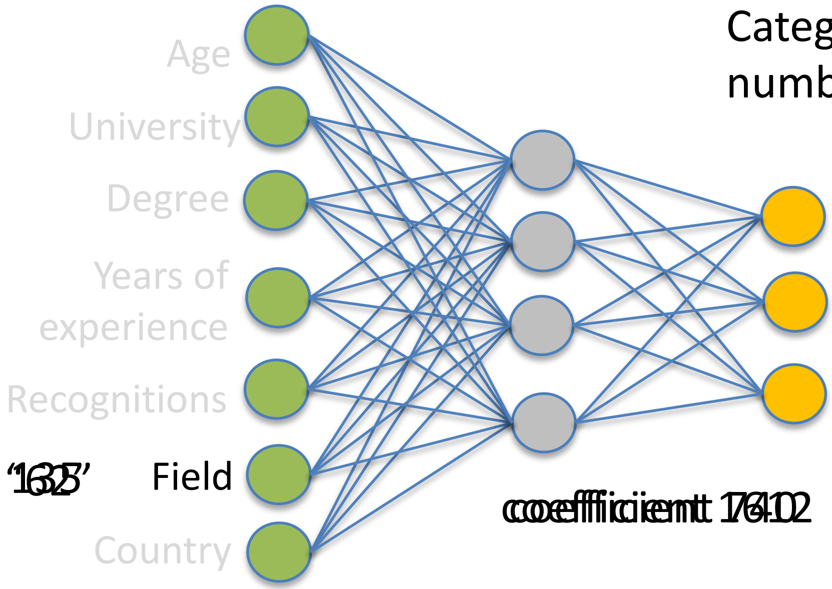


Labels are Numbers



Categories will likely be numbers… but the number itself is not meaningful

Numbers Need to Be Stable



Categories will likely be numbers… but the number itself is not meaningful

100 000 / 120 000

Expected salary range

Each Category Has Its Weight

important

irrelevant

counts somewhat

important

irrelevant

"62"

important

Age

University

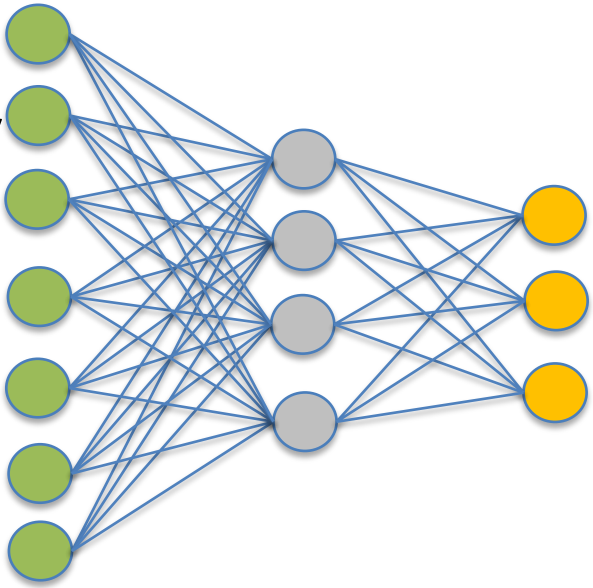
Degree

Years of experience

Recognitions

Field

Country

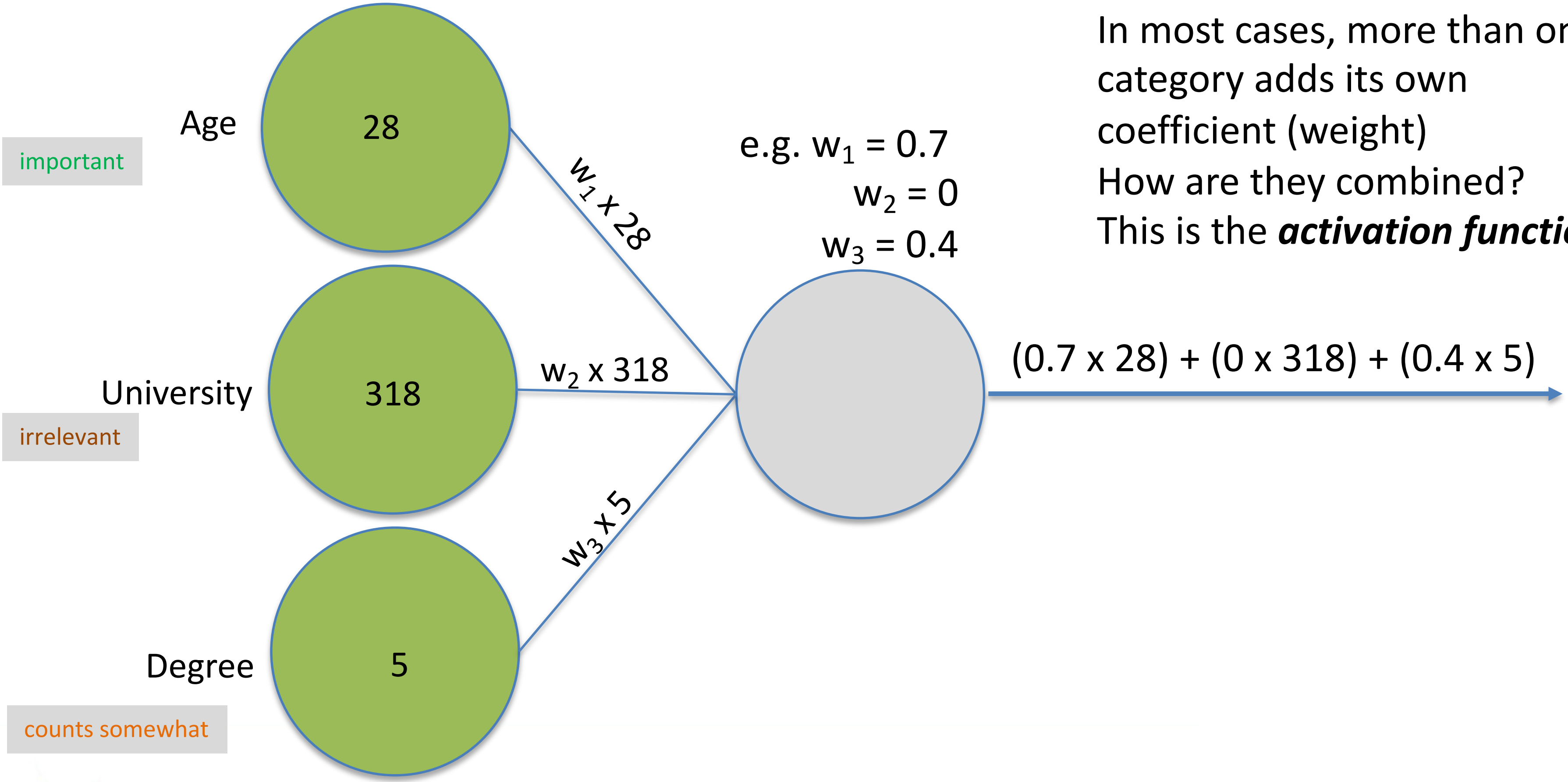


Expected salary range

In most cases, more than one category adds its own coefficient (weight)

How are they combined?

Perceptron Simple Model



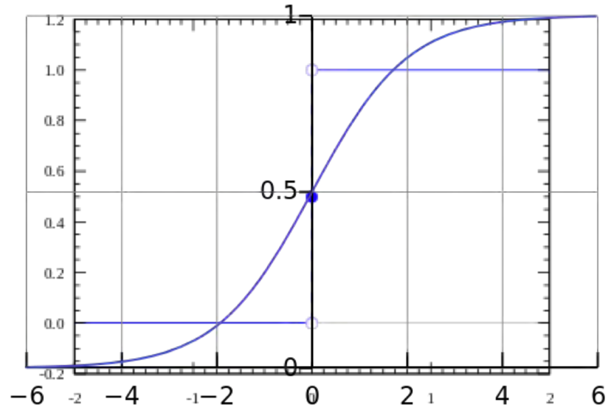
In most cases, more than one category adds its own coefficient (weight)

How are they combined?

This is the ***activation function***

A Better Activation Function

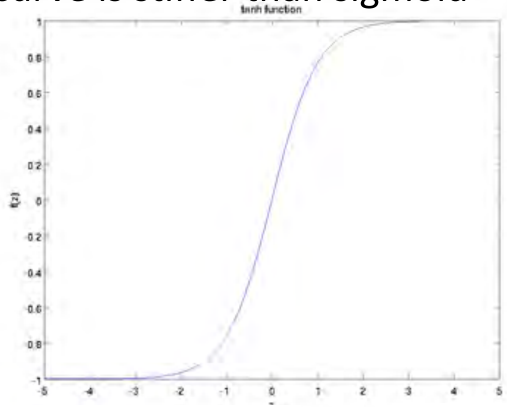
- A simple weight technique is too simple
  - We also need to decide ***if*** that neuron should fire or not
  - e.g. “if years of experience > 5, then use it”
- One technique is to combine the weight with a **step function**
  - If value is over threshold, use that neuron (and its weights)
- You also know the sigmoid function, and it is a better choice than strict step
  - You can fire or not (more than 0 or less than 0), but also use the result to smoothen the weight!

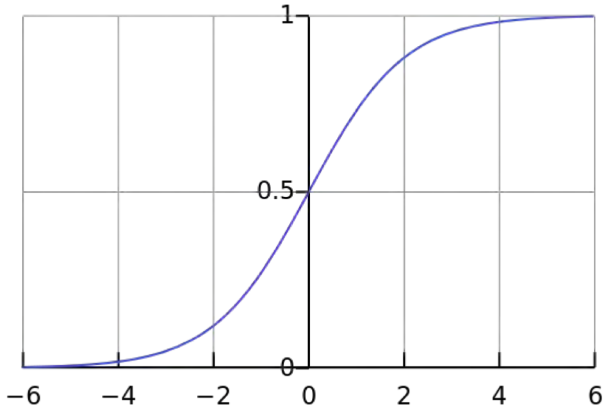


simple step function
sigmoid function

A Better Activation Function

- A variant of Sigmoid is the tanh function
  - curve is stiffer than sigmoid



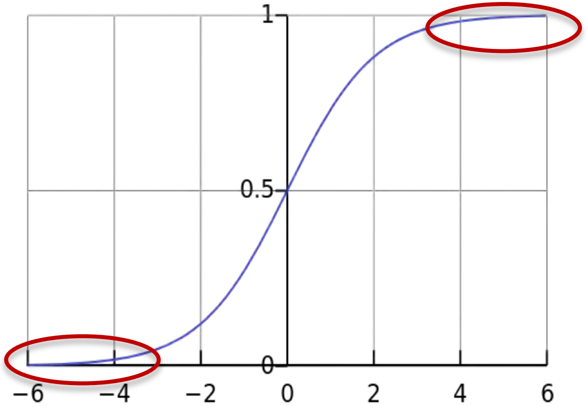


sigmoid function

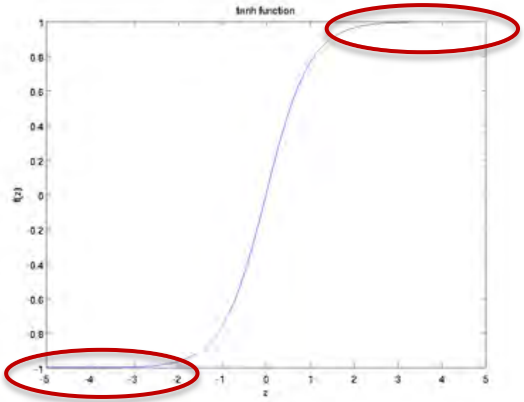
$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1 \qquad tanh(x) = 2\,sigmoid(2x) - 1$$

  - "was" used to make the system more sensitive to mid-values (around 0)

A Better Activation Function



sigmoid



tanh

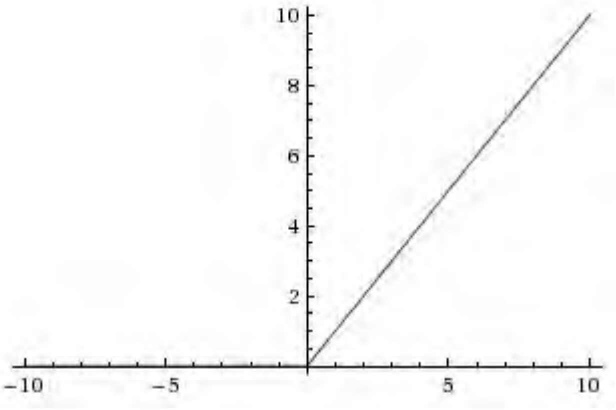
- Sigmoid has limitations at the end of the curve
  - Changes in the input do not affect the output so much
  - tanh is even worse, resulting in what is called the “dead neuron”
    - Use tanh with caution
    - sigmoid is still widely used

A Better Activation Function

Rectified Linear Unit (ReLU) is also very common:

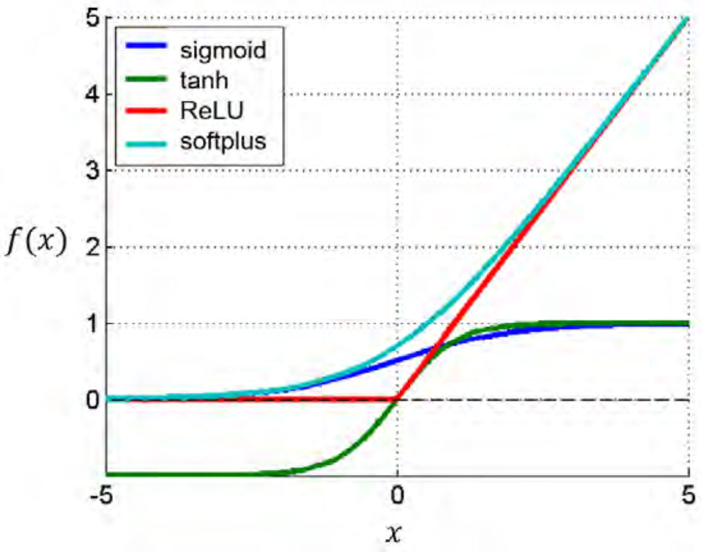
- Anything below 0 is just 0
    - Great to avoid “negative weights”
    - Things ‘count’ or ‘not’
- Anything above 0 is linear
    - Great to avoid “dead neuron” issue
    - a neuron at 0 does not fire, making the system faster

- However… there is a risk of infinite weight on one side, and “gradient is 0, does not descent anymore” in cost computation on the other side



ReLU function

Which Activation Function Should You Use?

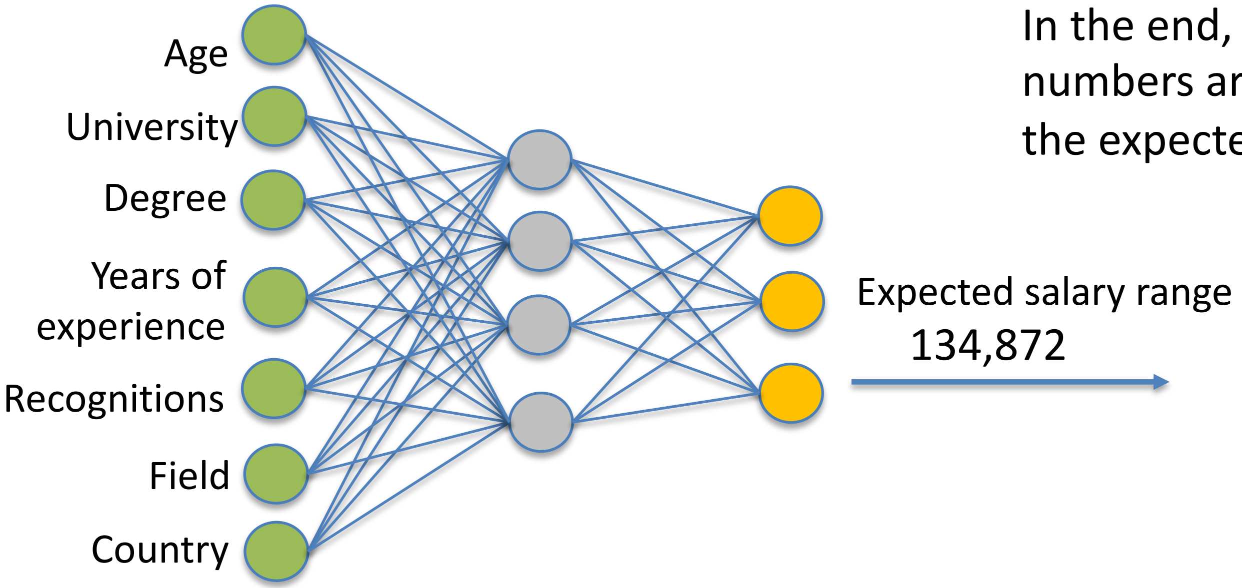


As of today:

- **Sigmoid functions** and their combinations usually work better for classification techniques ex. Binary Classification 0s and 1s.
- **Tanh functions** are not advised or implemented because of the dead neuron problem.
- **ReLU** is a widely used activation function and yields better results compared to Sigmoid and Tanh (especially for images).
- **Leaky ReLU** is a solution for a dead neuron problem during the ReLU function in the hidden layers.

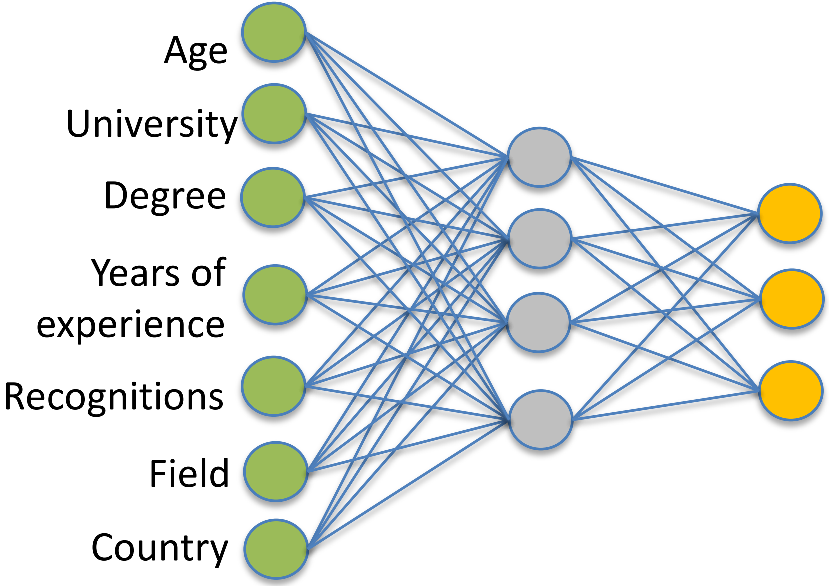
There are other activation functions (softmax, selu, linear, identity, soft-plus, hard sigmoid etc.)

Deep Learning Example – Not an Image!



In the end, all combined numbers are compared to the expected result

Deep Learning Example – Not an Image!



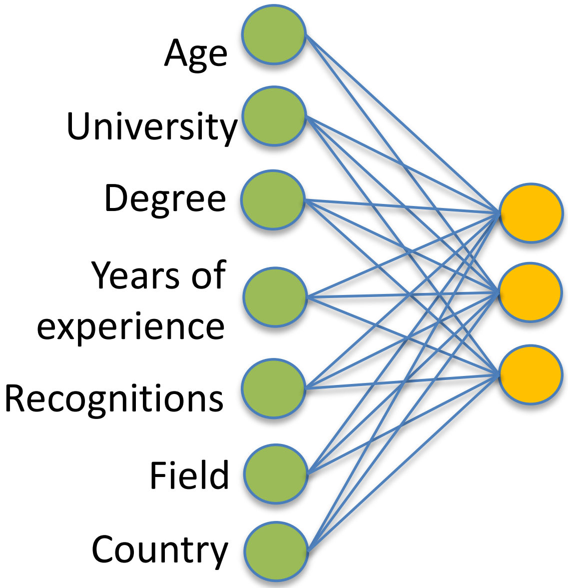
Expected salary
134,872

Should be 100,000!

If the output is wrong, the system needs to go back and revise its weights

This is called ***backward propagation***

The Purpose of Hidden Layers



If there was only one output (one layer), as there is only one possible coefficient per unit, the system would be too rigid

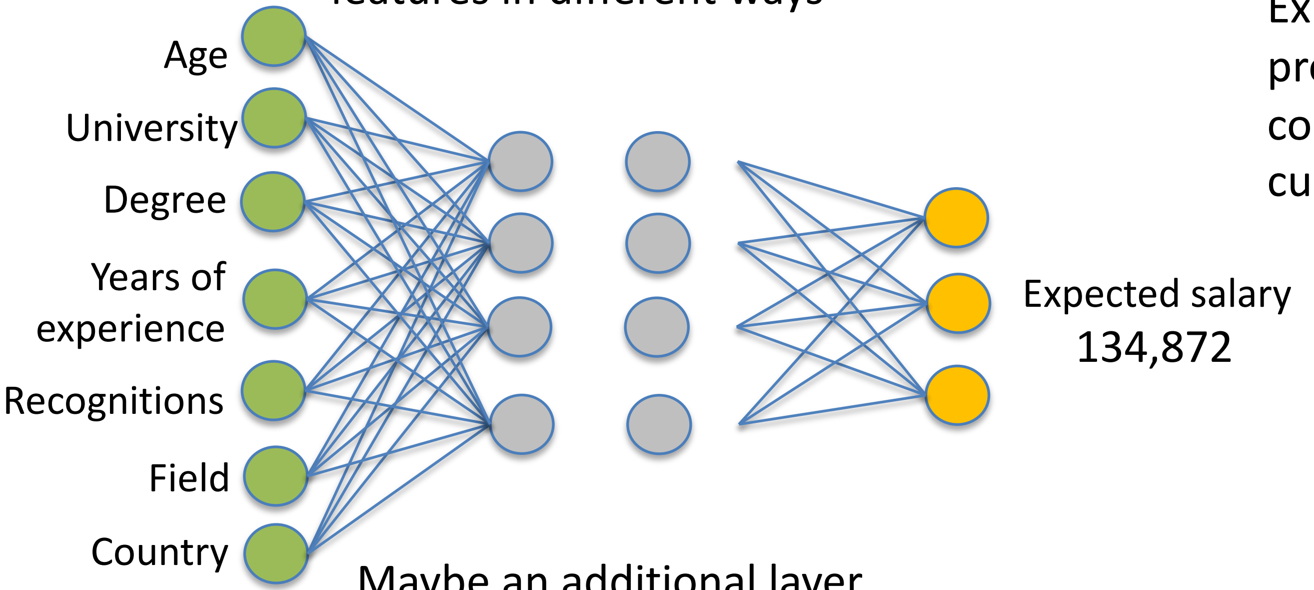
Age A and University K -> 100K

Age B and University M -> 100K

Can’t be!
Age has high weight, university smaller weight

Hidden Layers Help Refine Accuracy

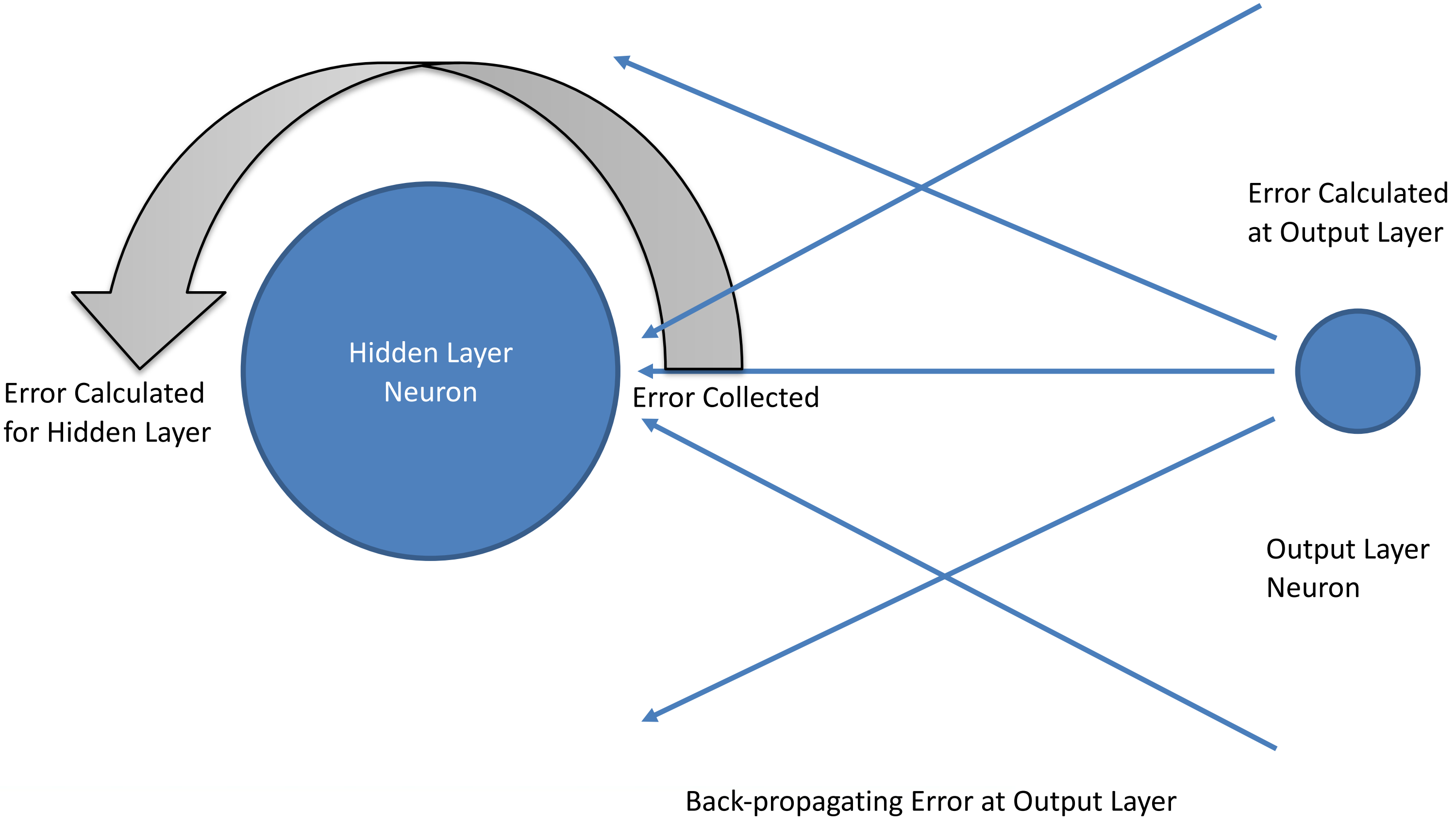
Each unit helps combine
features in different ways



The science of Layers is new
Experimentation on
prediction accuracy vs
computing costs is the
current best method

Maybe an additional layer
could help refine how these
different combinations could
further combine!

Back-propagating Error



Back Propagation Step-by-Step

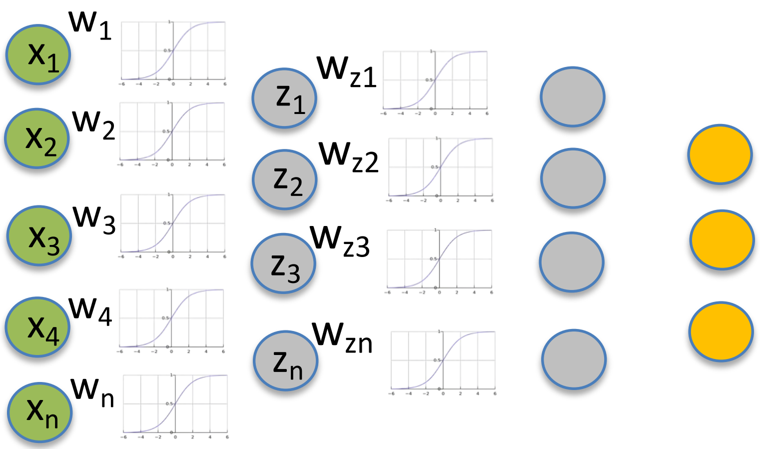
First, compute your forward propagation values:

Take your initial values (x) and random weights (w)

Apply your sigmoid to each unit, combine the output -> you get the x values of the next layer

Select new (random) weights for this Layer, and repeat…

Until you get to your output layer



Back Propagation Step-by-Step

- Because these are derivatives, they tell you if you should change your weight up or down
- The whole process is really “brute force”, from the end result to the input
- Once you have those error numbers, you change your weights a bit (in the direction given by the derivative)
- Then you recompute the output
- Then you compute the error
- Then you back-propagate again
- And repeat again…
- This is the training process for a neural network…
- Until your weights match your results

Lesson 6: Deep Learning and ANNs

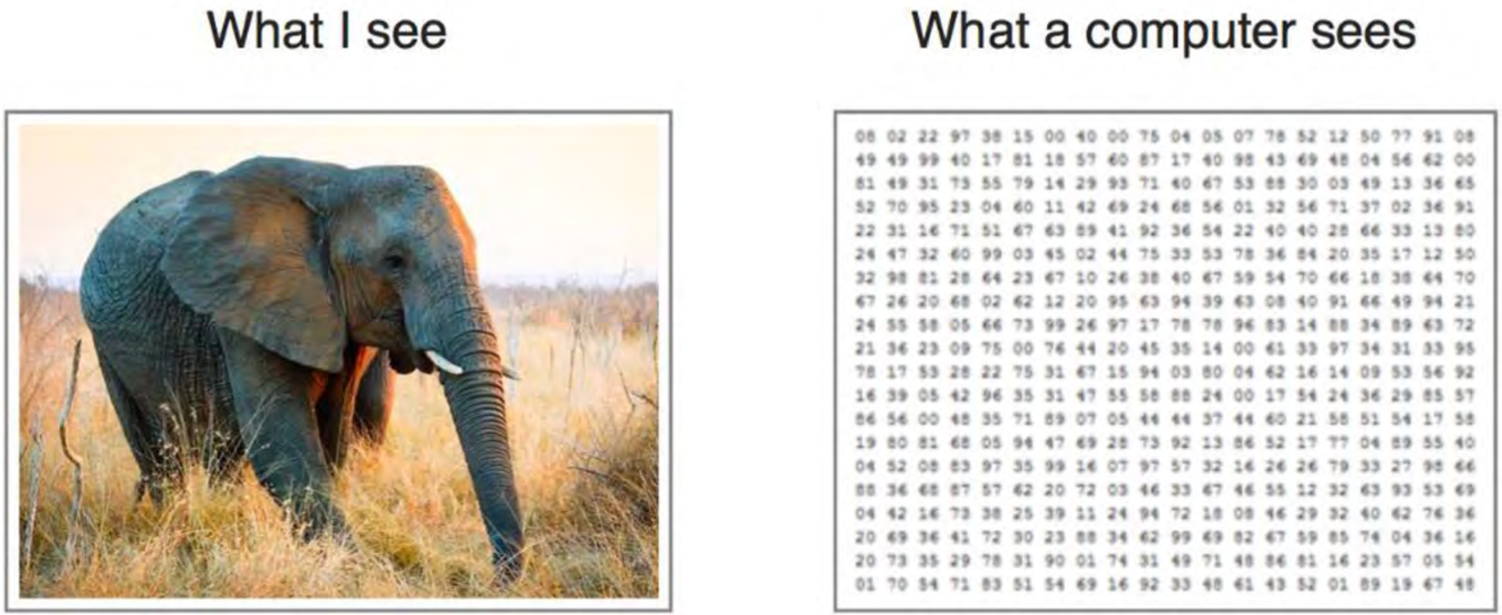
6.3 Applications of Deep Learning for Video Analytics and Natural Language Processing

Neural Networks: Different Types

- Artificial Neural Networks (ANN): generic term for neural networks
- Convolutional Neural Network (CNN): Hidden layers are designed to process the input in a way that optimizes for signal and image processing / recognition.
- Recurrent Neural Network (RNN): Neural networks with "loops" that are optimized for speech recognition, language modeling, translation.
- Generative Adversarial Networks (GAN): Two neural networks are pitted against one another to improve both. A generator (counterfeiter) will create content that the discriminator (police) will try to detect.

Convolutional Neural Networks

Example: Video Analytics



Source: https://cdn-images-1.medium.com/max/1000/1*cot55wd6gdoJlovlCw0AAQ.png

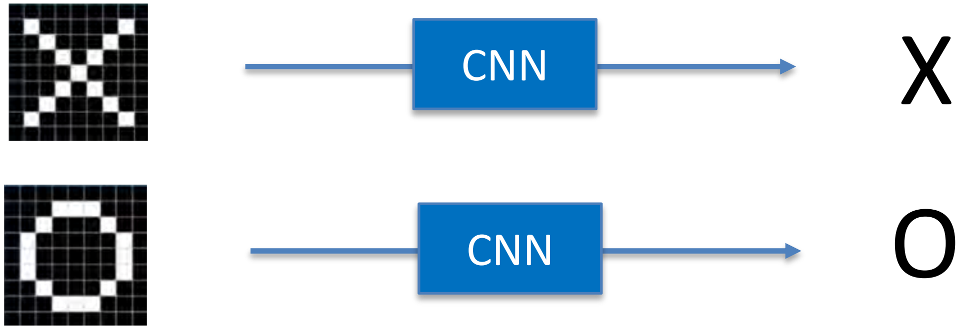
Why CNN?

At a scale of 4K, 8K, using a strict pixel per pixel equation, then comparison against a database of all possible combinations, cannot work anymore

CNN improves the system by thinking in *zones*, and how each zone helps the other toward a known image

Is It an X or an O? Building Our CNN

Our goal:



But also:

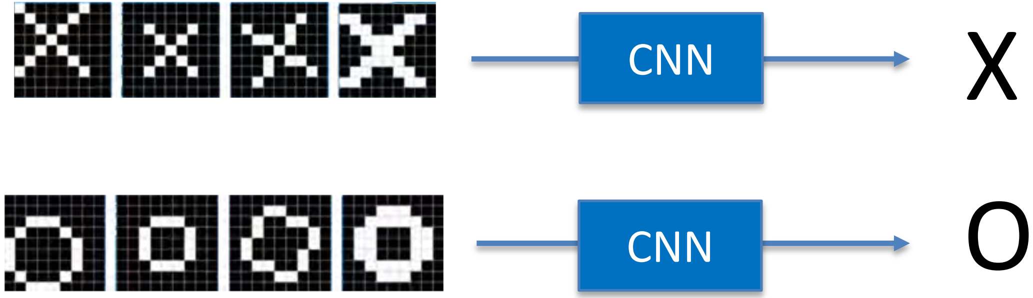
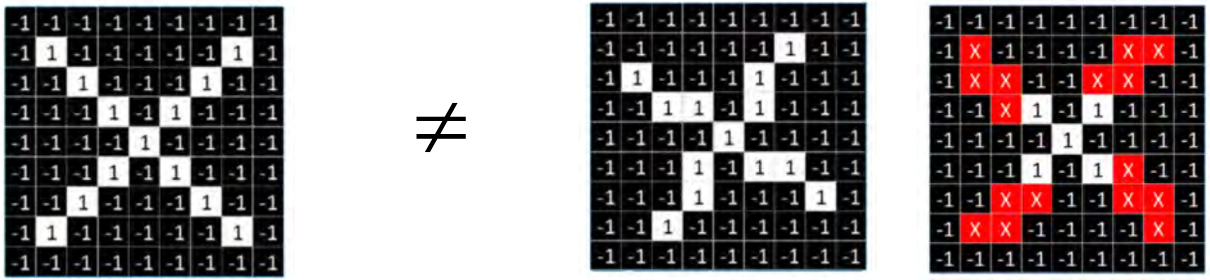
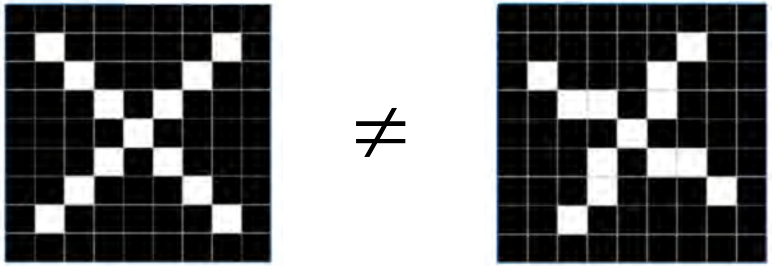


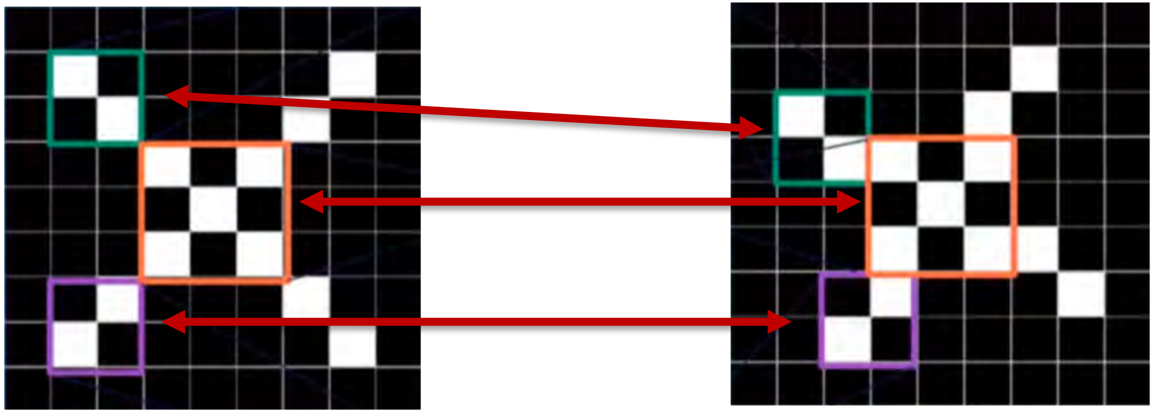
Image credits Brandon Rohrer http://brohrer.github.io

It Is More Than a Pixel Problem



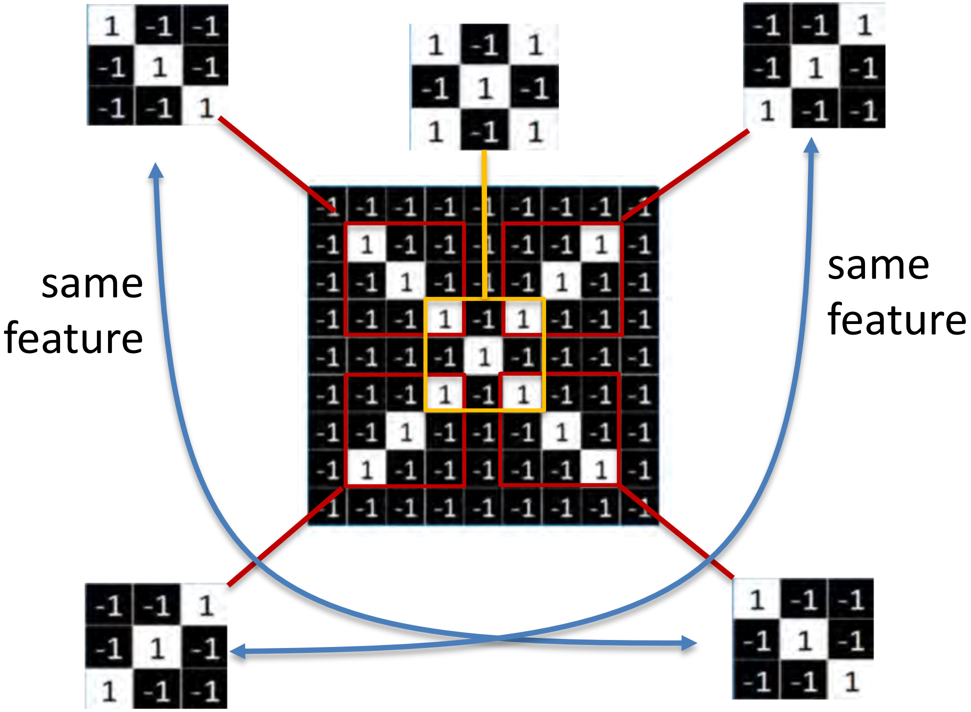


This Is Why CNNs Work by zone-match



The accuracy of your model depends on the size of each zone
We'll soon see that CNN also try to reduce the number of zones

Start by Computing Zones



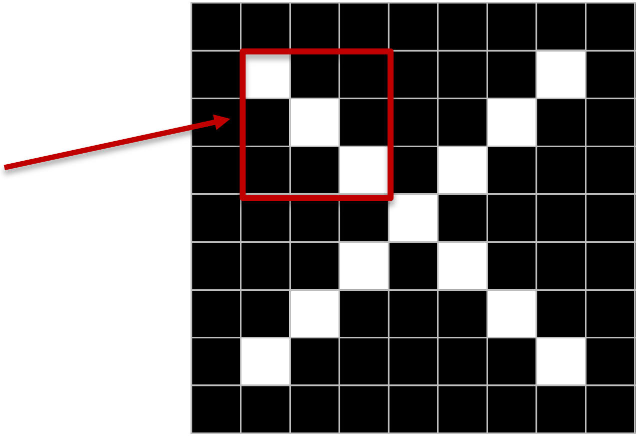
Pixel-by-pixel comparison allows you to find similar pixels, then check if their neighbors are also the same

This allows you to build from a reference image a set of “features”

Compare Each Zone to Each Feature

How similar is this zone to this feature? (we see that they are the same, we want the computer to see that too)

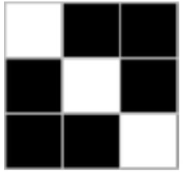
zone



1	-1	-1
-1	1	-1
-1	-1	1

feature

Start by encoding the zone you are testing



zone

=

1	-1	-1
-1	1	-1
-1	-1	1

encoded zone

Compare Each Zone to Each Feature

Then compute feature vs. zone differences with the following recipe:

1	-1	-1
-1	1	-1
-1	-1	1

feature

1	-1	-1
-1	1	-1
-1	-1	1

zone

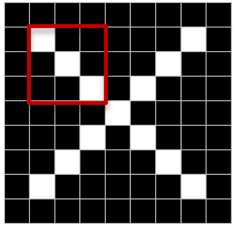
1 x 1 =1
-1 x -1 =1

1	1	1
1	1	1
1	1	1

1. Multiply them pixel by pixel

2. Add all values and divide by the number of values (pixels):

Here (perfect match): $\frac{1+1+1+1+1+1+1+1+1}{9} = 1$



Note! This is a simplified B&W example. In real life, you may compare pixel per pixel, with the color match

Compare Each Zone to Each Feature

Let’s try another (less perfect matching) zone:

1	-1	-1
-1	1	-1
-1	-1	1

feature

1	-1	1
-1	1	-1
1	-1	1

zone

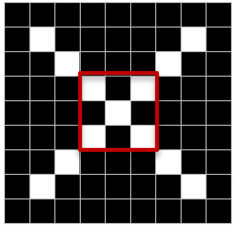
1 x 1 =1
-1 x 1 =-1

1	1	-1
1	1	1
-1	1	1

1. Multiply them pixel by pixel

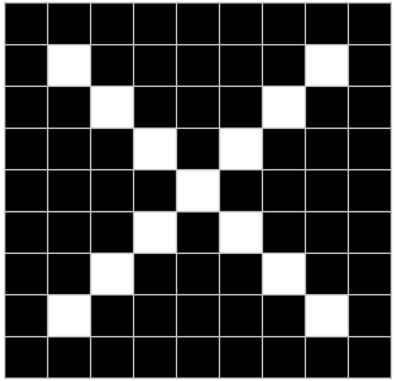
2. Add all values and divide by the number of values (pixels):

Here (partial match): $\frac{1+1-1+1+1+1-1+1+1}{9} = 0.55$



Compare Each Zone to Each Feature

In the end, for each feature, you can build a matching map



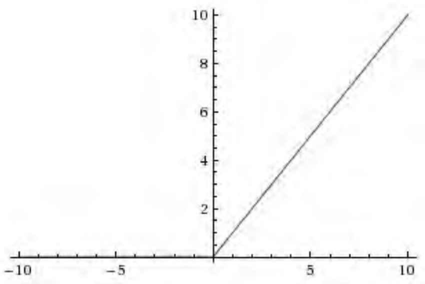
-1	-1	1
-1	1	-1
1	-1	-1

0.33	-0.11	0.55	0.33	0.11	-0.11	0.77
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33

This comparison is called the ***convolution*** Layer
(in math, ***convolution*** studies how a shape is
modified by another)

Cleaning Up the Result

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Do you remember ReLU?
CNN also uses it to remove the negative results (i.e. if match is **0**, then there is no match at all, no need for **match is -0.87**)… and of course, positive results stay the same

Reducing the Image Size

The issue with convolution is the computing cost (and the cost of remembering each feature match result)

To reduce the size of each feature match result, CNN can use *pooling* techniques / layers

Take a match result map and decide of a stride (how many pixels you want to jump across)

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Take your zone and simplify it to the max value (max pooling) or the average value (average pooling)

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

Repeating the Sequence

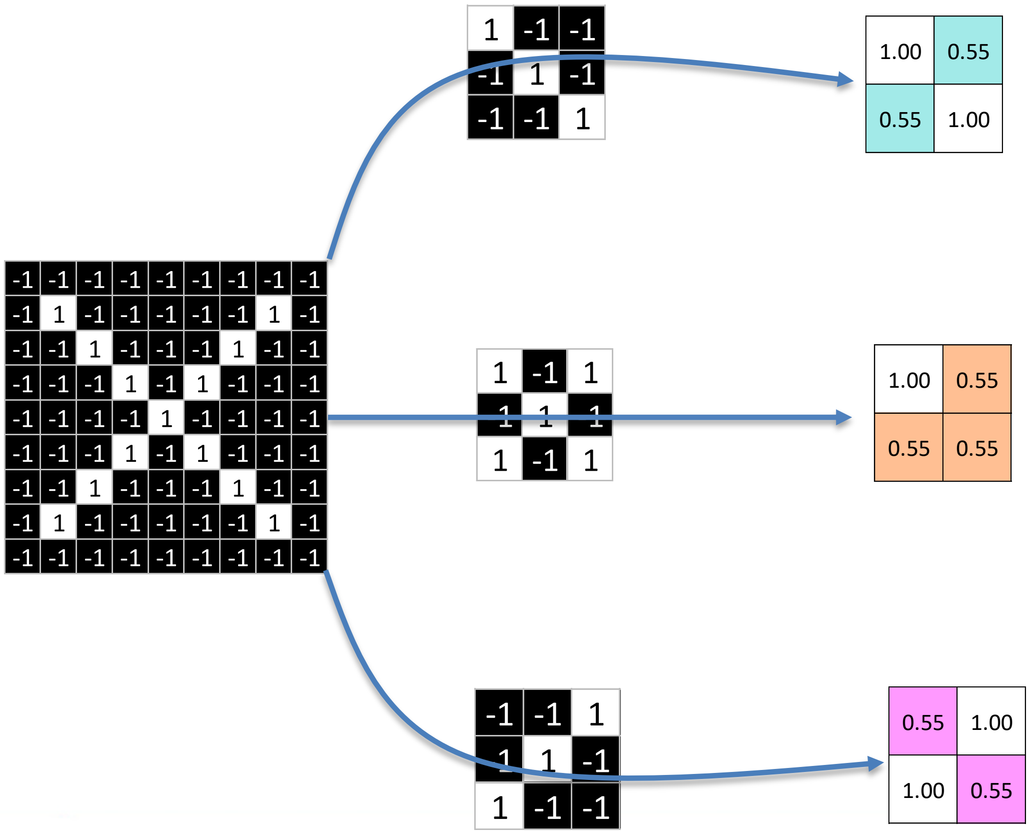
CNN can use these features repeatedly (deep networks) to simplify and loosen the image

Experimenting is key to decide of the repeat cycle

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

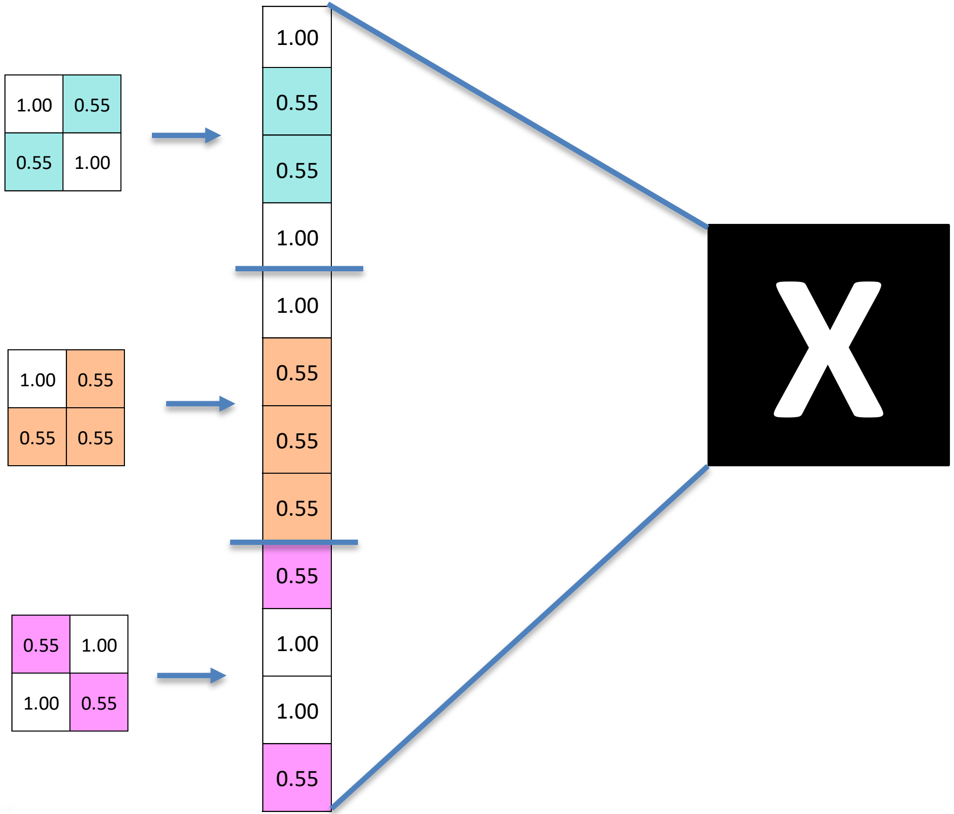
1.00	0.55
0.55	1.00

Sequence Output



In the end, compared (filtered) to each feature, the image results in a set of ordered match values

Applying Weights (Fully Connected Layer)



The combination of all these values is used as a set of learned weights to represent the target image (**X**)

Repeat the same process to learn **O** and other images

A CNN Is Still a NN

CNNs still uses back propagation and gradient descent to learn the correct weights
The weights are not the feature matching map, but each number in the map is associated to a weight -> the back propagation learns the right weight

What is special in CNN is the combination of convolution as the activation function, ReLU to clean and pooling to reduce the output, applied several times

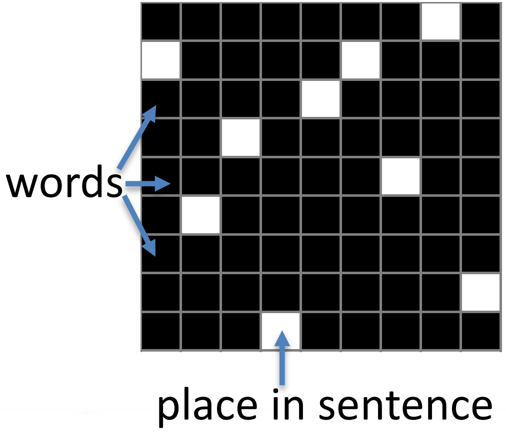
The rest of the process is ***standard NN***

Not Just for Images

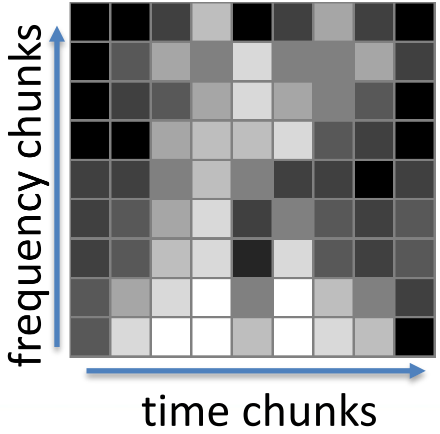
CNNs work well for any “image-like” structure:

- A sequence can be graphed
- There is a logical connection between an element and the next
- Order matters (you cannot easily change the order without changing the meaning)

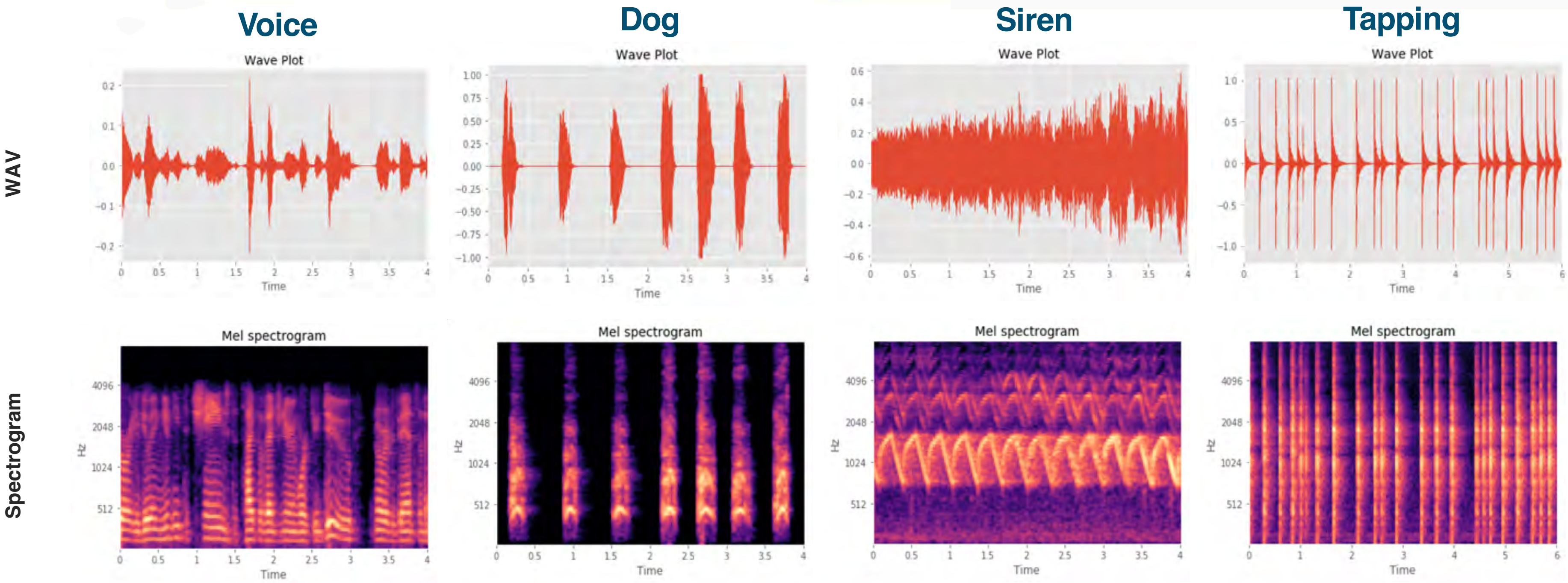
e.g. bag-of-words:



e.g. speech/sound recognition



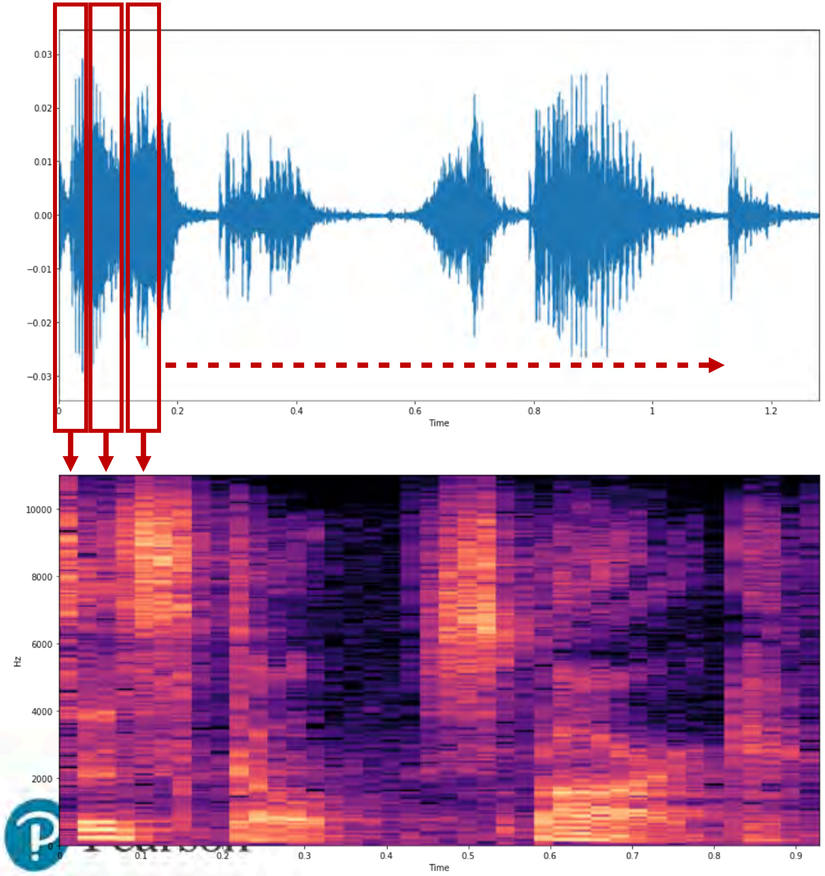
From Signals to Images



Filtering noise from conference calls in near real time
Voices and "noise" have a distinct "images" that can be detected.

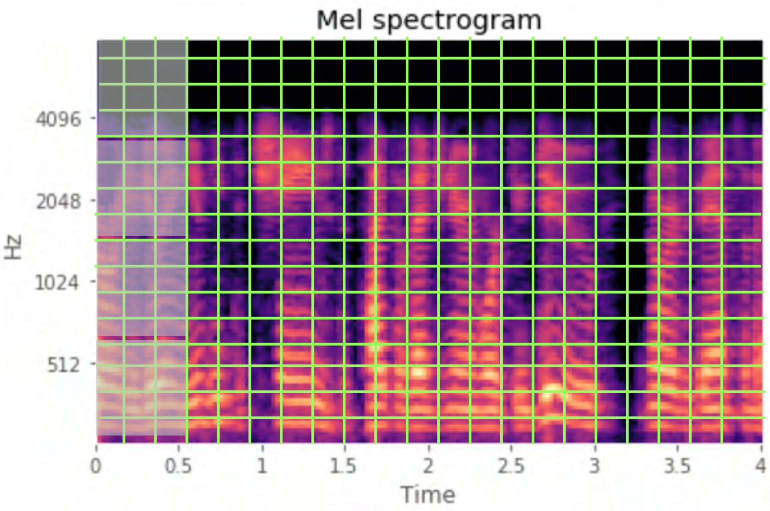
Pearson

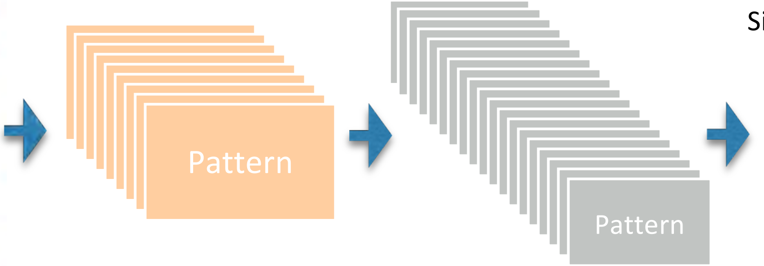
MEL Spectrogram



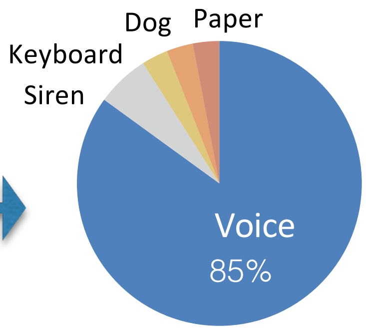
- *We can convert this into an image analysis problem:*
- *Convert the .wav plot into an image:*
- *Take the spectrum of a time slice of the signal*
- *Process it to generate a visual frequency spectrum → **Spectrogram***
- *Continue these steps…*

Analyzing Spectrogram with a CNN





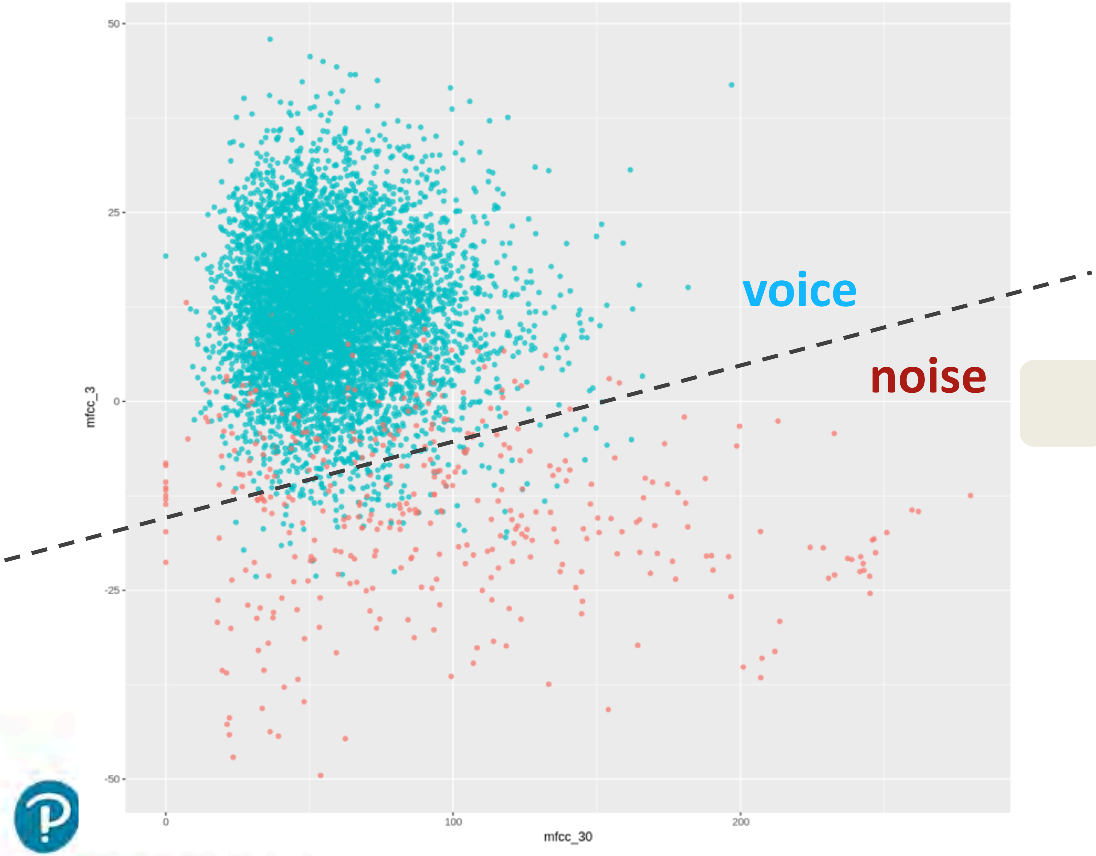
Various layers of a CNN are used to determine features / predict likelihood of voice vs. noise patterns



Likelihood

Pro	Con
Accurate	Intensive computation

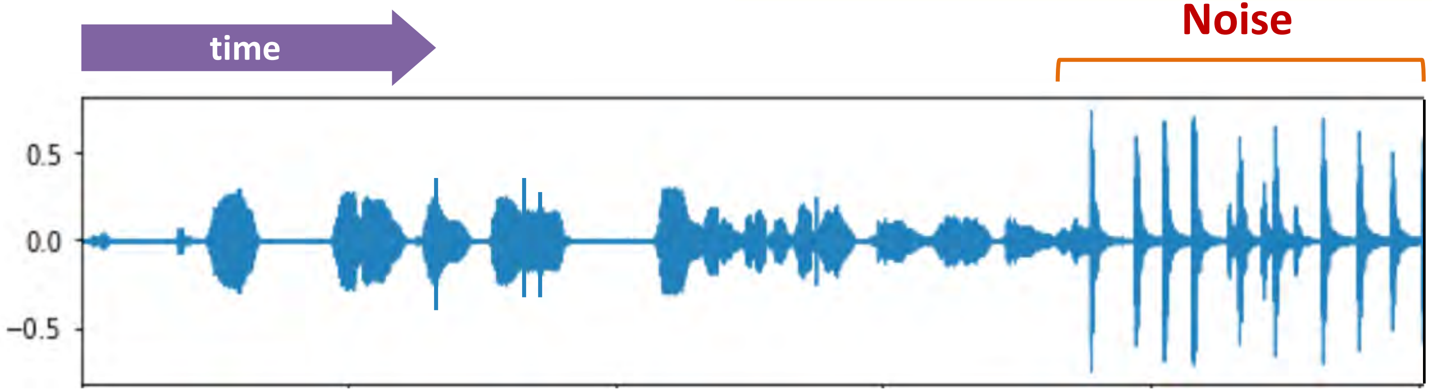
Initial Model: Logistic Regression



- Logistic regression →Linear decision boundary
- Looks at energy levels in different frequencies
- Ignores temporal dynamics

Pro	Con
Lightweight	Less Accurate

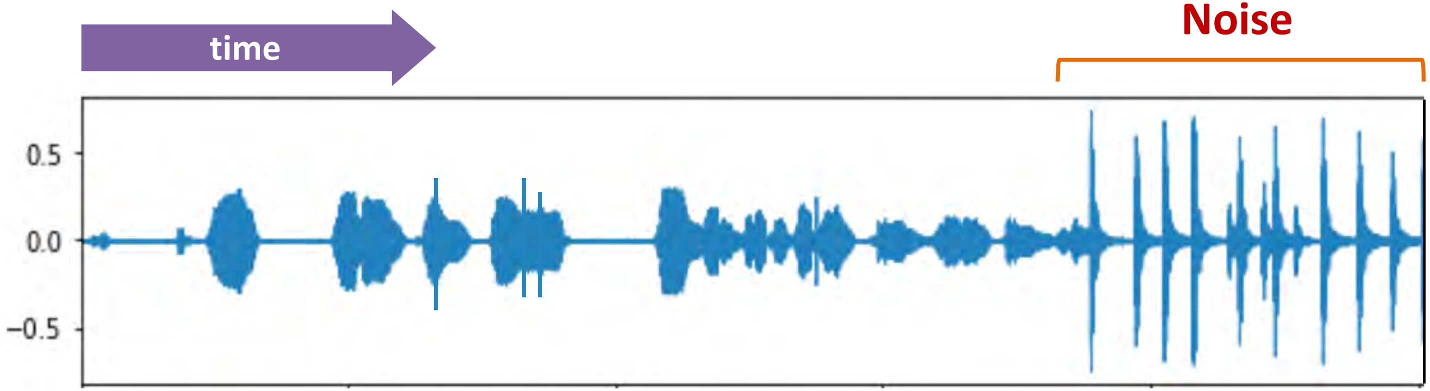
Noise Prediction in Real Time



Time Frame		1	2	3	4	5	6	7	8	9	10
LR Model	Noise	No	No	Yes	Yes	No	No	No	Yes	Yes	Yes
	CPU (%)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

false alerts

Noise Prediction in Real Time



Model cascading

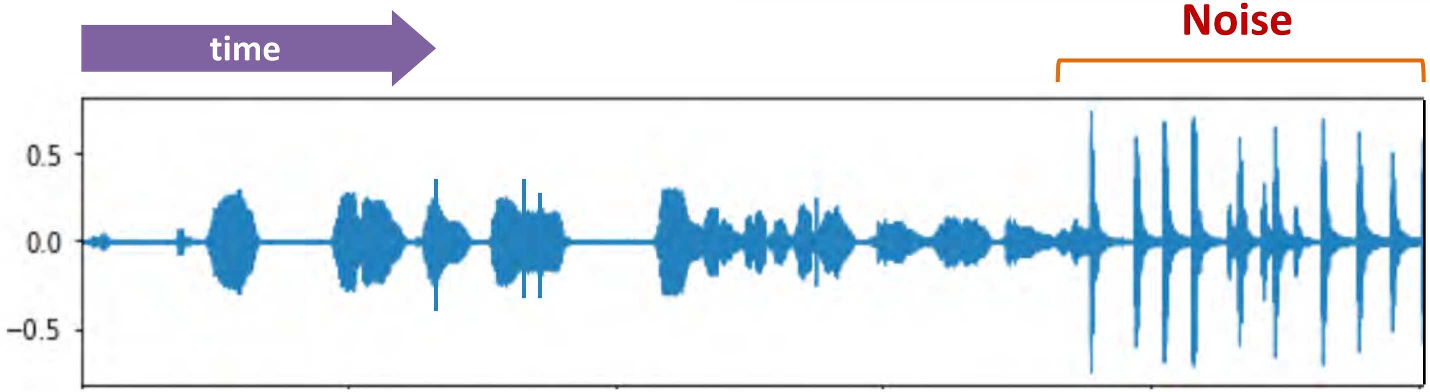
Time Frame		1	2	3	4	5	6	7	8	9	10
LR Model	Noise	No	No	Yes	Yes	No	No	No	Yes	Yes	Yes
	CPU (%)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
CNN Model	Noise	INACTIVE		No	No	INACTIVE			Yes	Yes	Yes
	CPU (%)			2	2				2	2	2

Don't initiate noise suppression

Initiate noise suppression

Pearson

Noise Prediction in Real Time



Time Frame		1	2	3	4	5	6	7	8	9	10
LR Model	Noise	No	No	Yes	Yes	No	No	No	Yes	Yes	Yes
	CPU (%)	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
CNN Model	Noise	INACTIVE		No	No	INACTIVE			Yes	Yes	Yes
	CPU (%)			2	2				2	2	2
Overall CPU (%)		**0.1**	**0.1**	**2.1**	**2.1**	**0.1**	**0.1**	**0.1**	**2.1**	**2.1**	**2.1**

Pearson

Lesson 7: Dimensionality Reduction and PCA

7.1 Dimensionality

Dimensionality

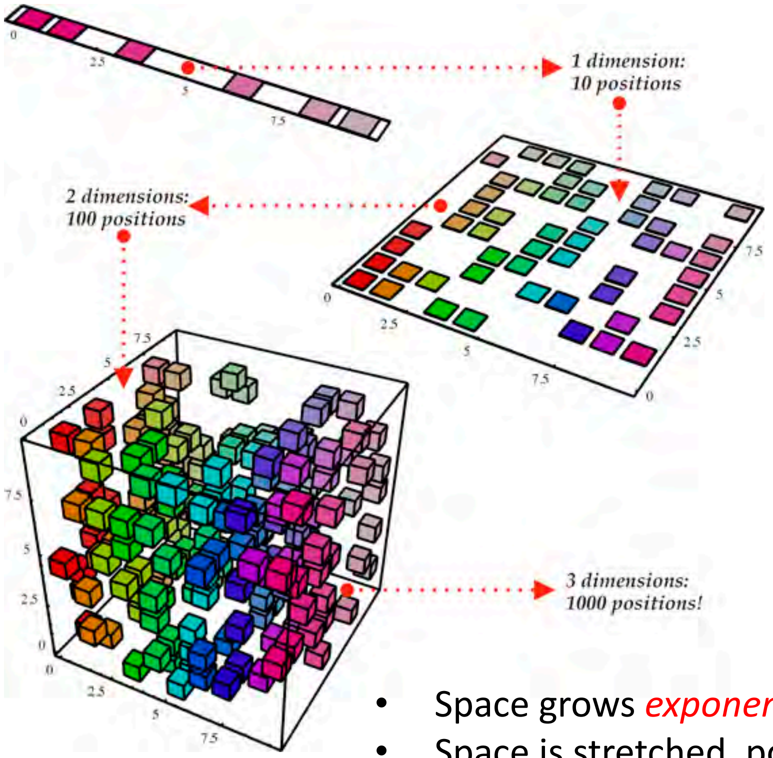
- Machine Learning is good at understanding the structure of high dimensional spaces
- What is a dimension?
  - A “Feature” of the ML model

- Example: you are a manufacturer of smart phones
- You start to find a systemic problem in the device
  - Complex supply chain with hundreds of base components
  - Massive code base
  - Where does the problem lie?
  - There are possibly 1000s of variables (features) involved!

Important and unseen relationships frequently live in high-dimensional spaces

The Curse Of Dimensionality

- There are an *exponential* number of variations to this problem
  - The possible tensor space is stretched, points become extremely spaced out and the computational cost grows.
- Classical Solution: Hope for a smooth enough target function, or make it smooth by handcrafting good features
- This is sub-optimal and often doesn't work!



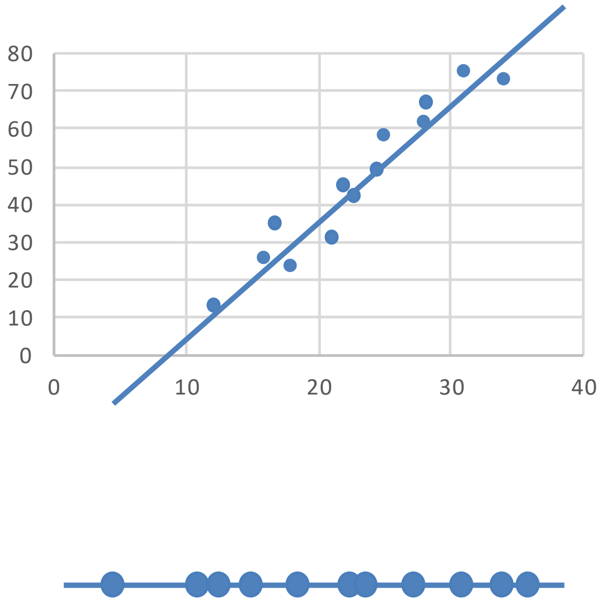
- Space grows *exponentially*
- Space is stretched, points become *equidistant*

Principal Components Analysis: Description

- PCA is a Dimensionality Reduction method
- A method of analyzing data sets to summarize main characteristics
- PCA is a descriptive / exploratory method
- PCA has the goals of:
  1. Reducing the original variables into a lower number or non-correlated (orthogonal) synthesized variables
  2. Visualizing the correlations
  3. Visualizing the proximities

Principal Components Analysis Main Idea

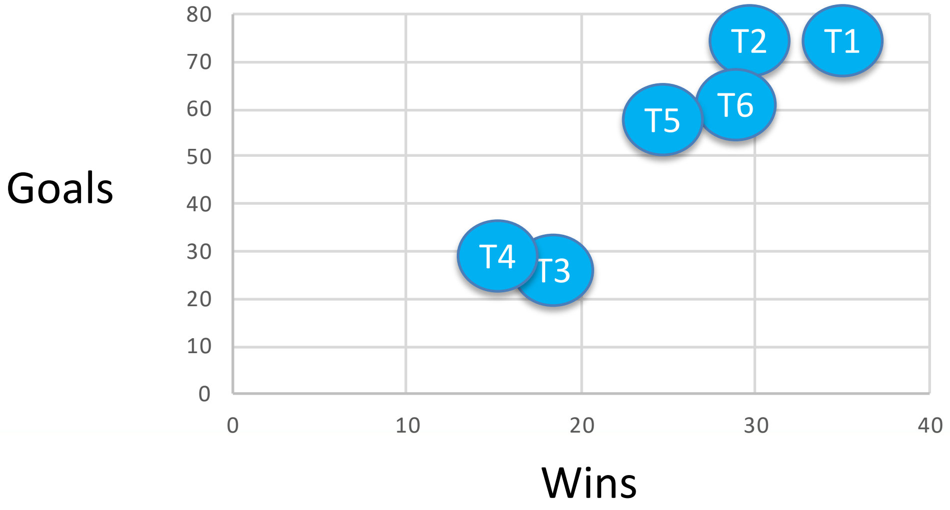
- What we will end up with is a change in the variable space – move from $n \rightarrow k$ dimensions (called eigenvectors)
- The method involves "projection" of a higher dimension into a lower number
- The change in variable space will result in a loss of information – the key is to minimize the loss of information as much as possible



Reduced from 2D to 1D

Example of PCA

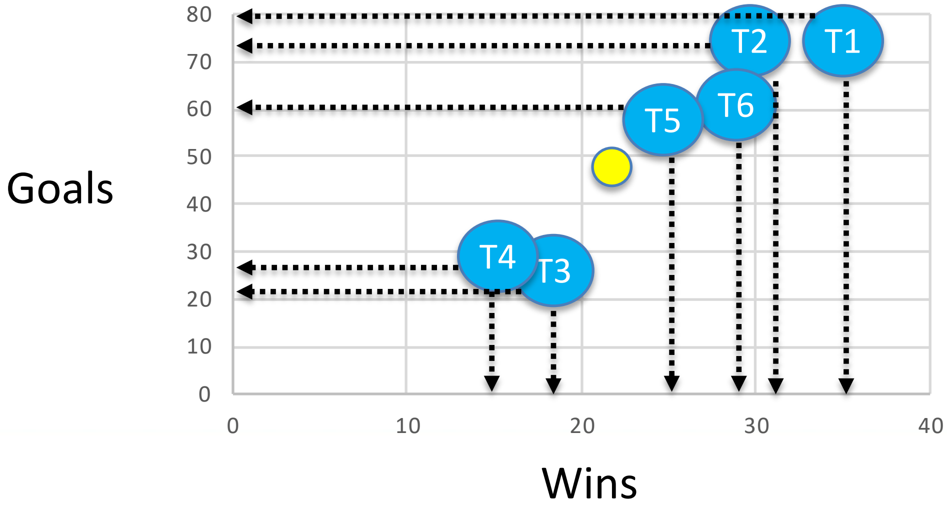
	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6
Wins	34	31	18	16	25	28
Goals	73	75	23	25	58	61



Wins vs. Goals in 2-Dimensions

Example of PCA

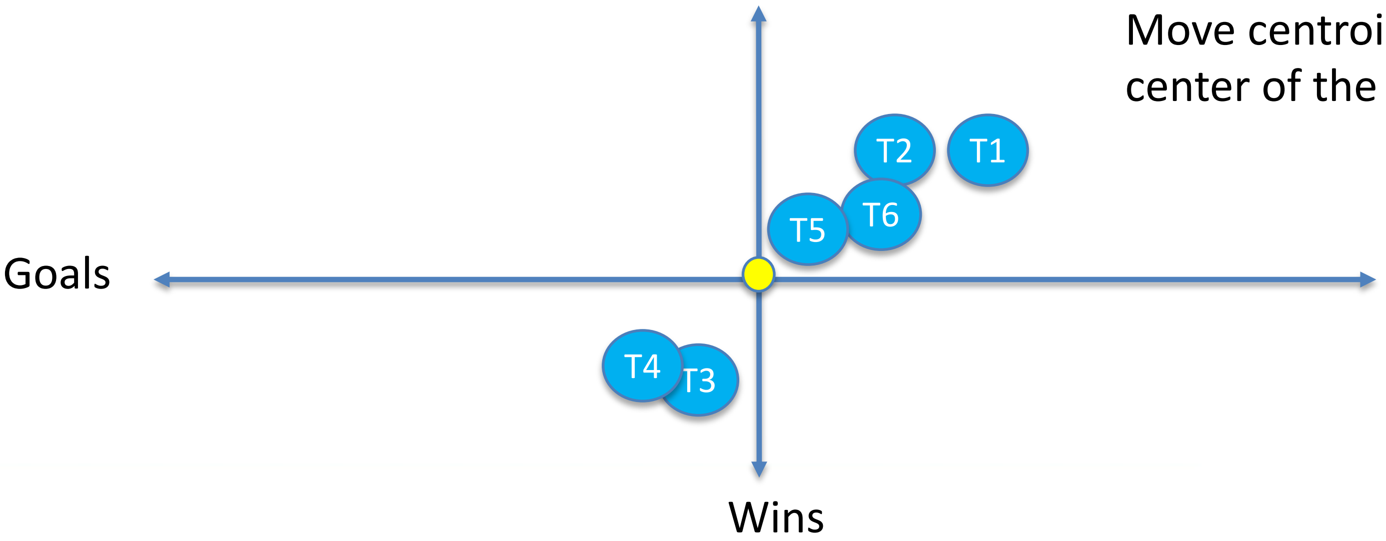
	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6
Wins	34	31	18	16	25	28
Goals	73	75	23	25	58	61



Wins vs. Goals in 2-Dimensions

Example of PCA – Move Centroid to the Center

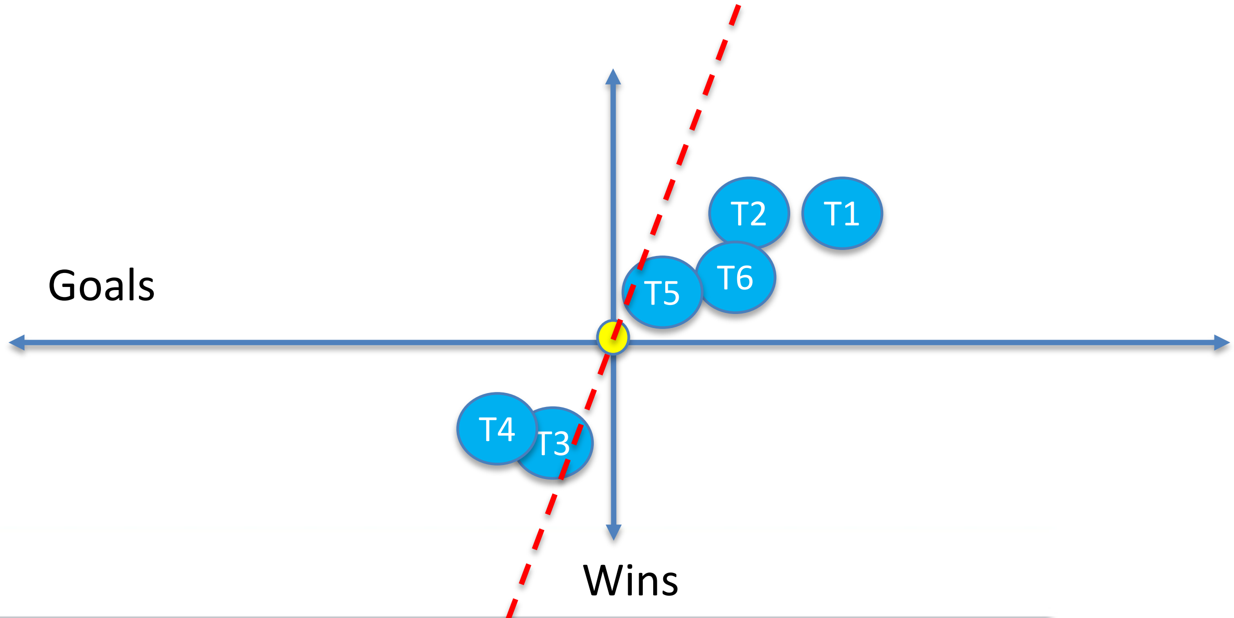
	Team 1	Team 2	Team 3	Team 4	Team 5	Team 6
Wins	34	31	18	16	25	28
Goals	73	75	23	25	58	61



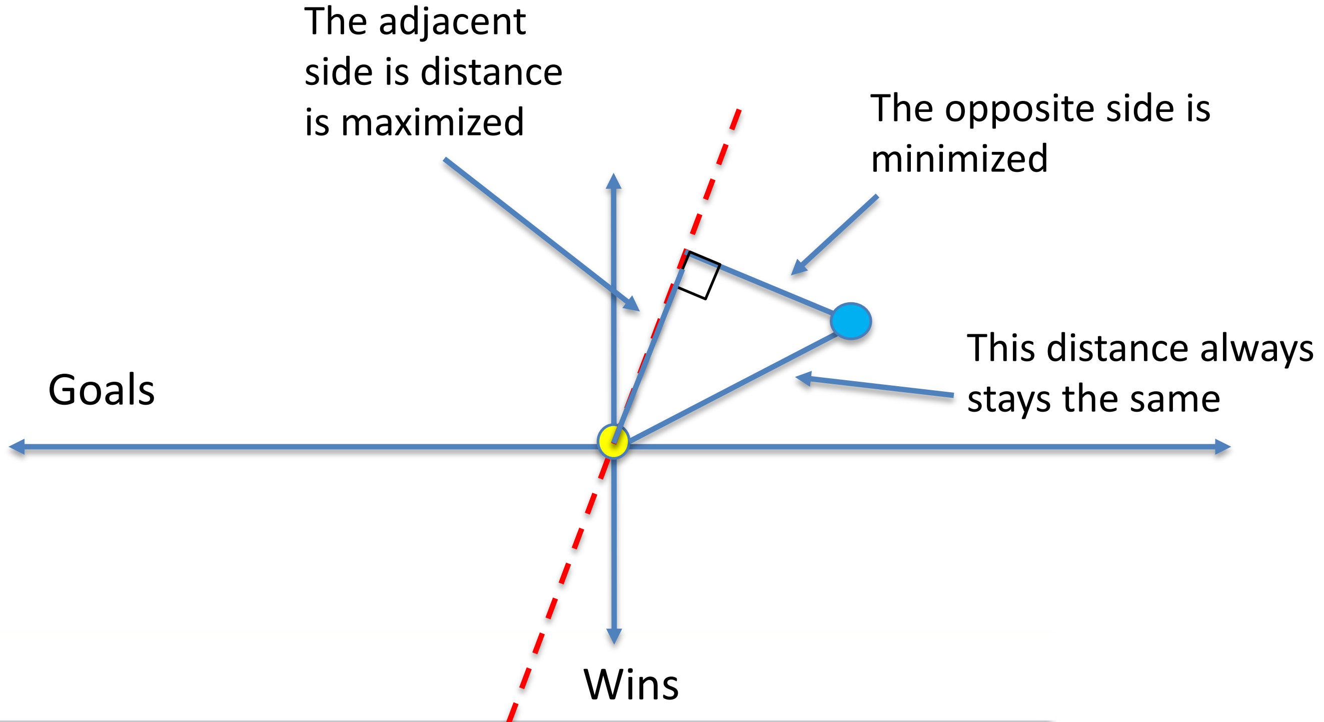
Move centroid to the center of the graph

Example of PCA

Begin to recursively test a line through the data to find the best possible fit



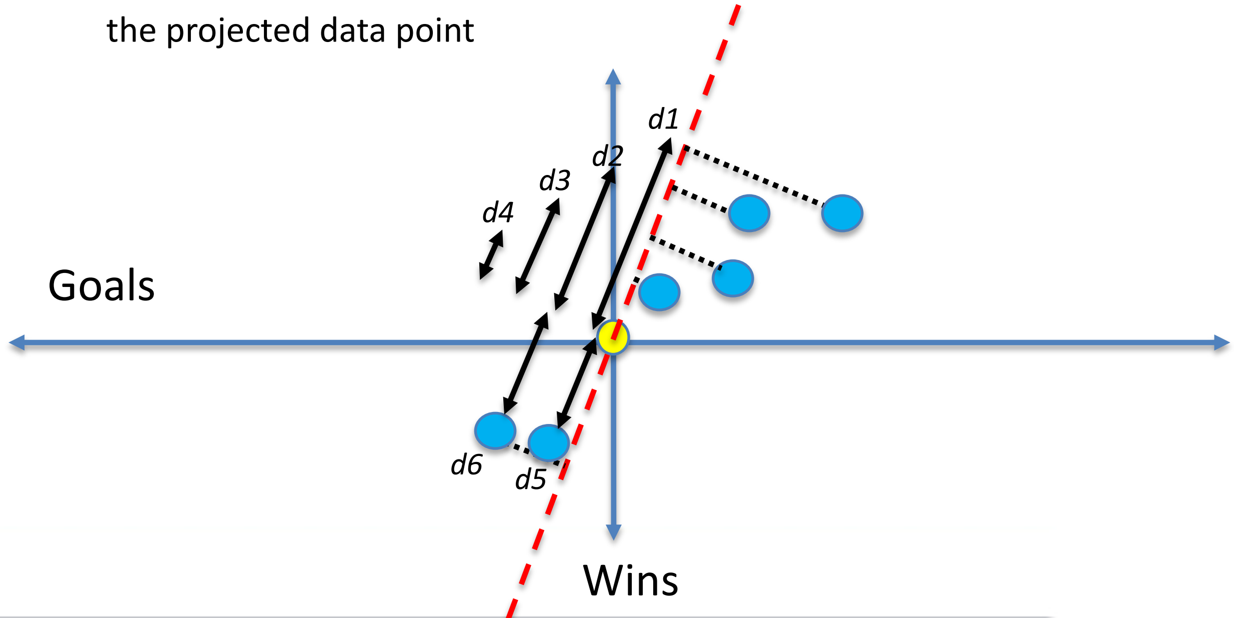
Projecting Data Points to the Line



Sum of Squares of the Distance

Sum of Square Distance = $d1^2 + d2^2 + d3^2 + d4^2 + d5^2 + d6^2$

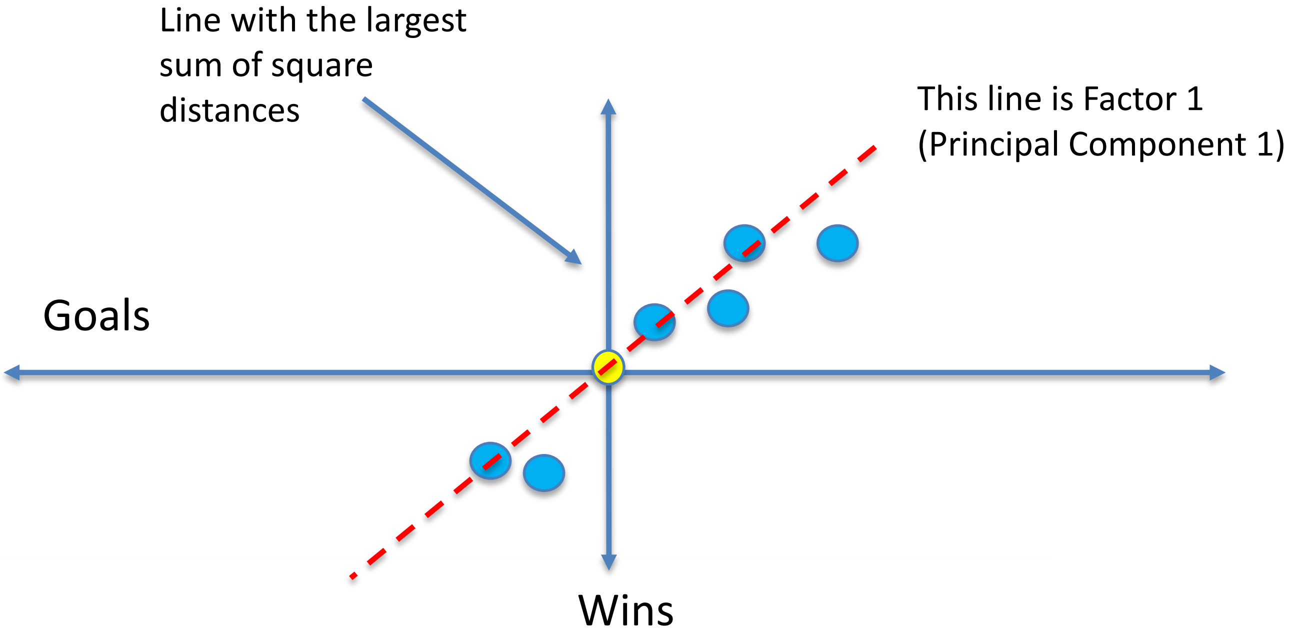
PCA measures this line from
the projected data point



Example of PCA

Sum of Square Distance = $d1^2 + d2^2 + d3^2 + d4^2 + d5^2 + d6^2$

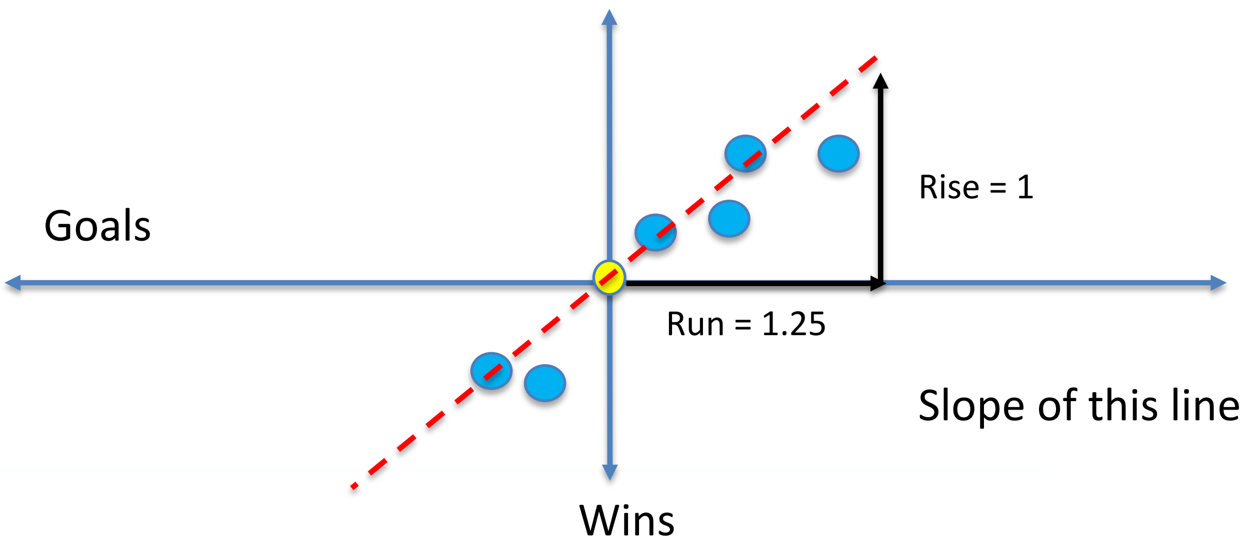
The largest value is known as the ***eigenvalue***



Example of PCA – Sum of Square Distances

Another way to think of this:

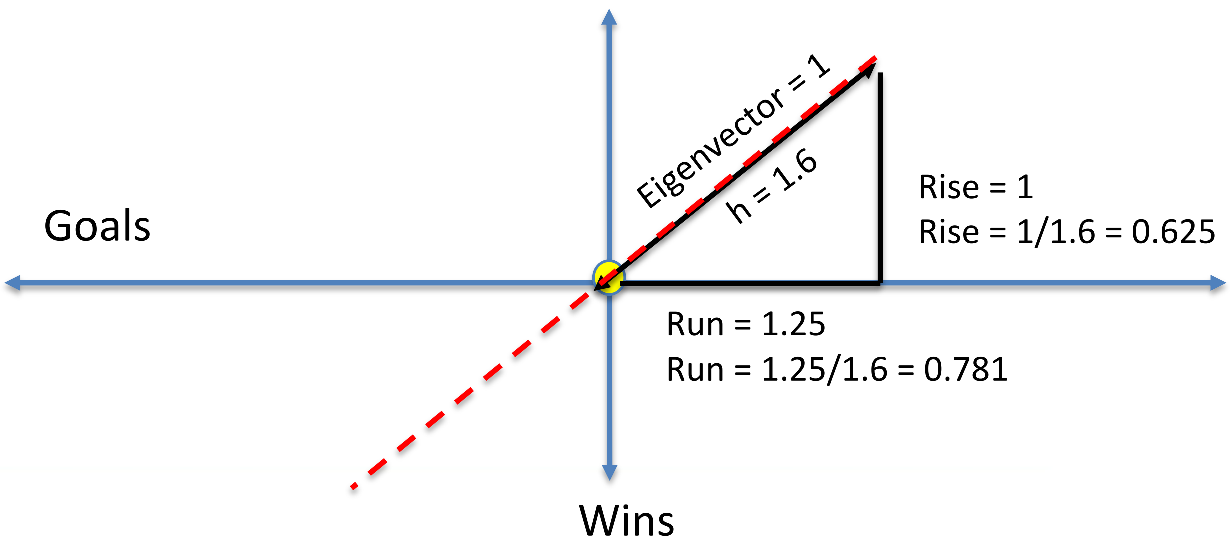
To make Factor 1 (PC1), for every 1.25 goals, you get one 1 win!



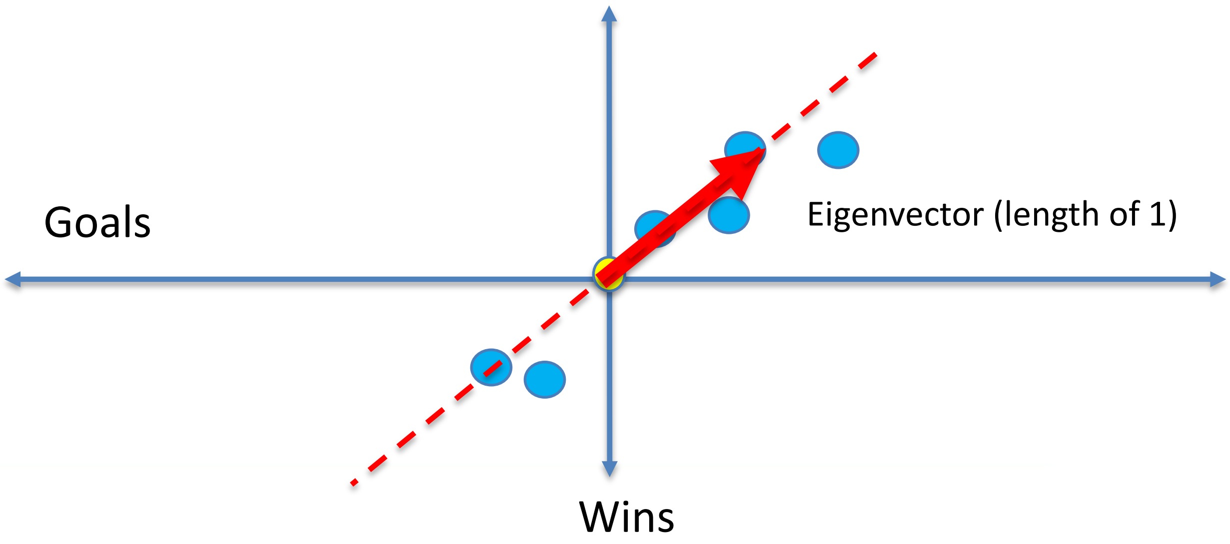
Slope of this line is 0.8

Example of PCA – Eigenvalues

Length of hypotenuse line is calculated as $a^2 = b^2 + c^2 = 1.6$
In PCA, the vector is scaled to 1 (called an **eigenvector**)



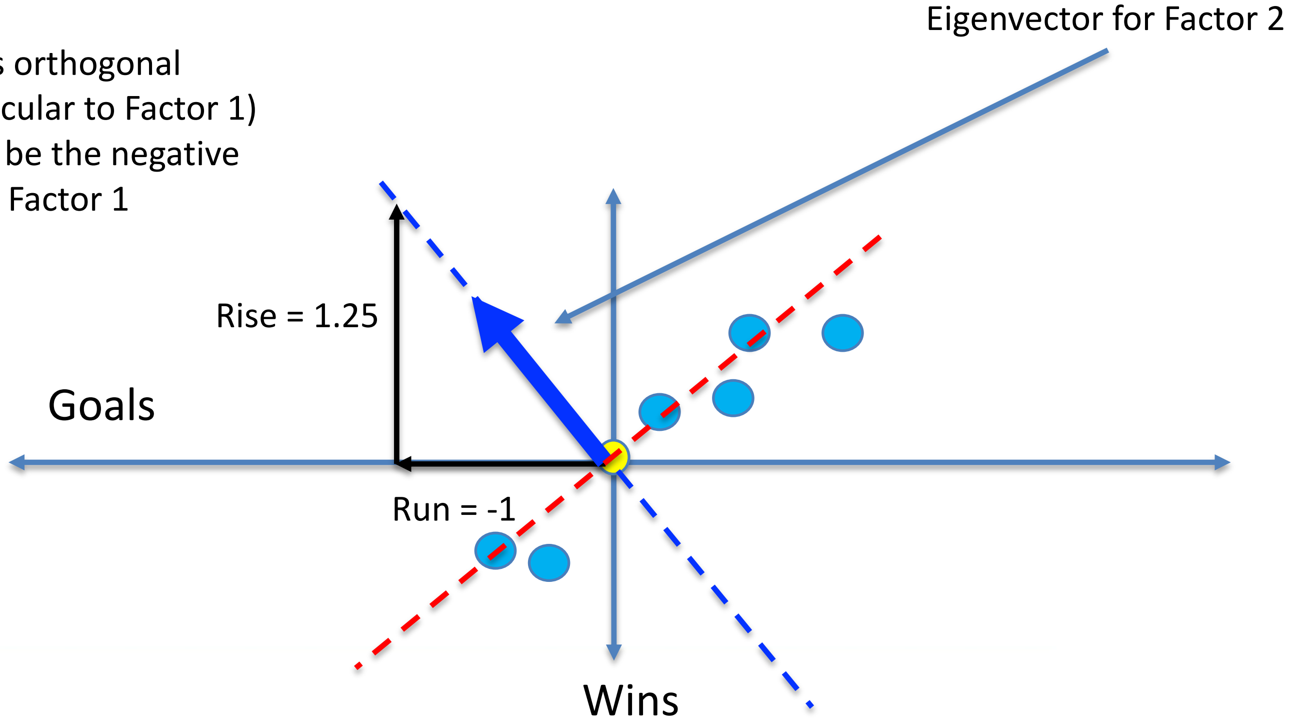
Example of PCA – Sum of Square Distances



Example of PCA – Sum of Square Distances

Now that we now Factor 1, let's figure out Factor 2

- Factor 2 is orthogonal (perpendicular to Factor 1)
- Slope will be the negative inverse of Factor 1



Example of PCA – Variation

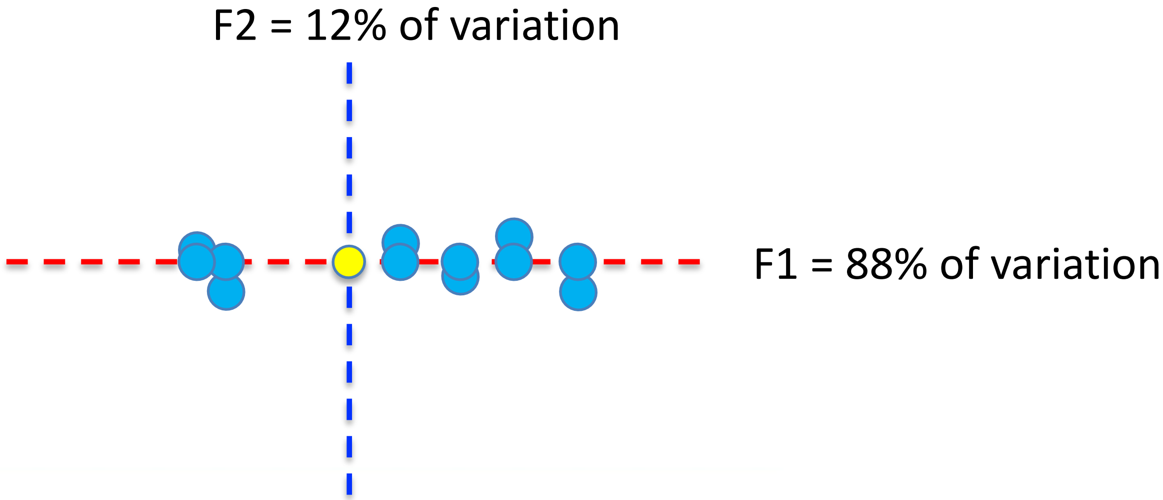
Variation is a measurement of how much each PC is impacting the correlation

Variation for F1 = $\dfrac{\text{Eigenvalue Factor 1}}{\text{n-1}}$

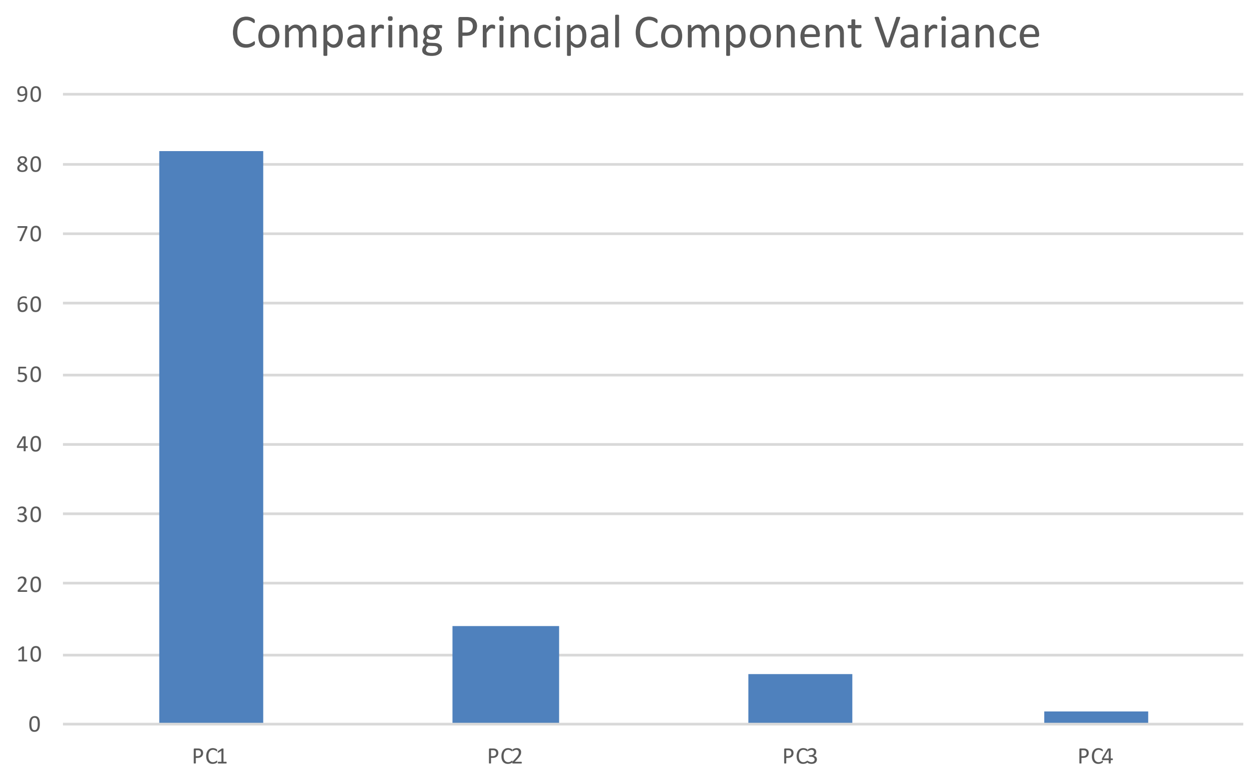
Variation for F2 = $\dfrac{\text{Eigenvalue Factor 2}}{\text{n-1}}$

Variation for Factor 1 = 23

Variation for Factor 2 = 3



Evaluating the Principal Components



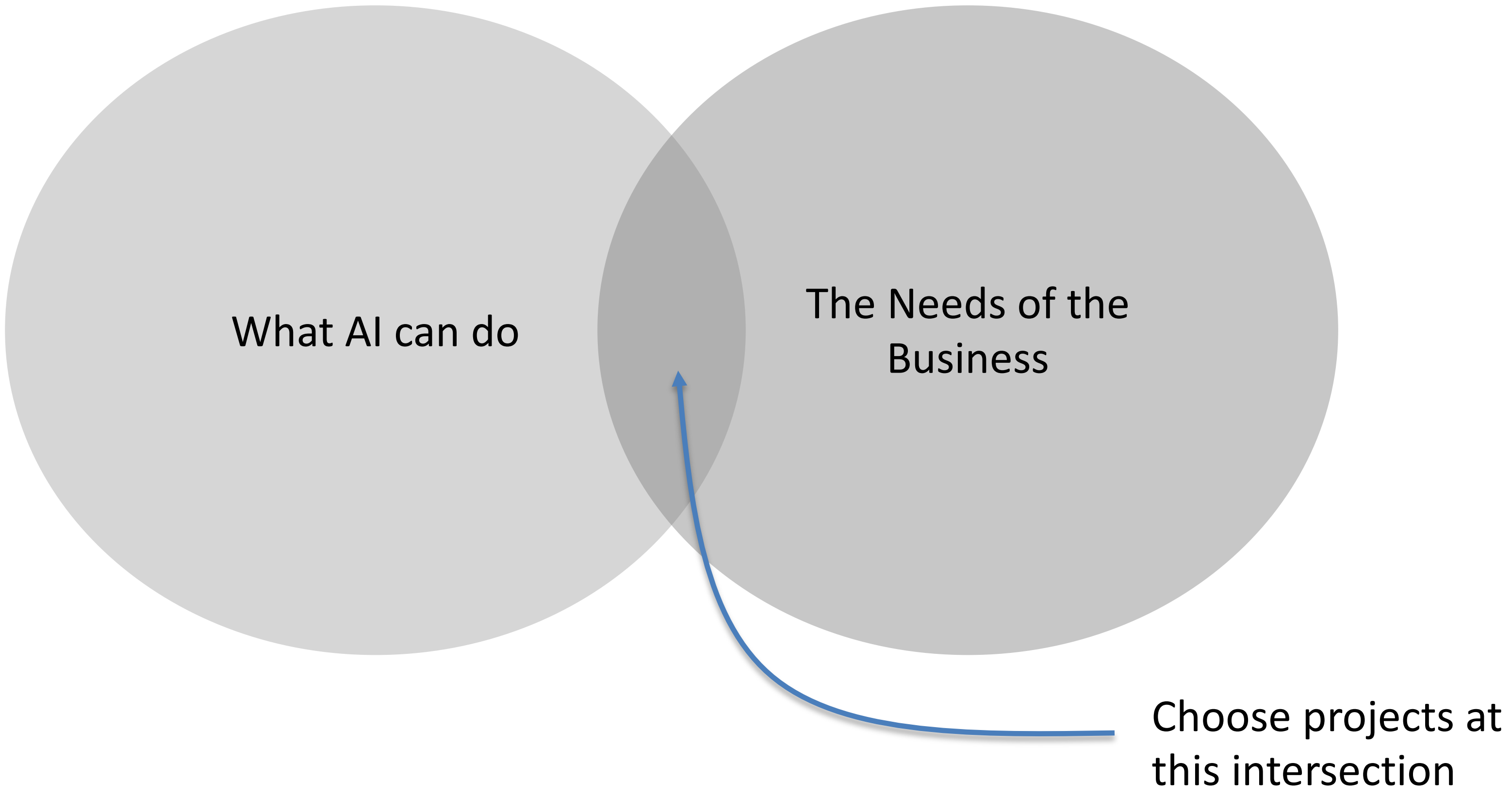
Lesson 8

8.1 Starting an AI/ML Project

Choosing a Machine Learning Project

- Involves multiple parties – both the Machine Learning experts, as well as domain experts
  - Say you wanted to predict customer churn in a retail store
  - Requires both ML experts as well as people that understand business dynamics
  - Often a brainstorming session with both types of team members will reveal good candidate AI/ML projects

Finding the Right Project



What Machines Can and Can’t Do (today)

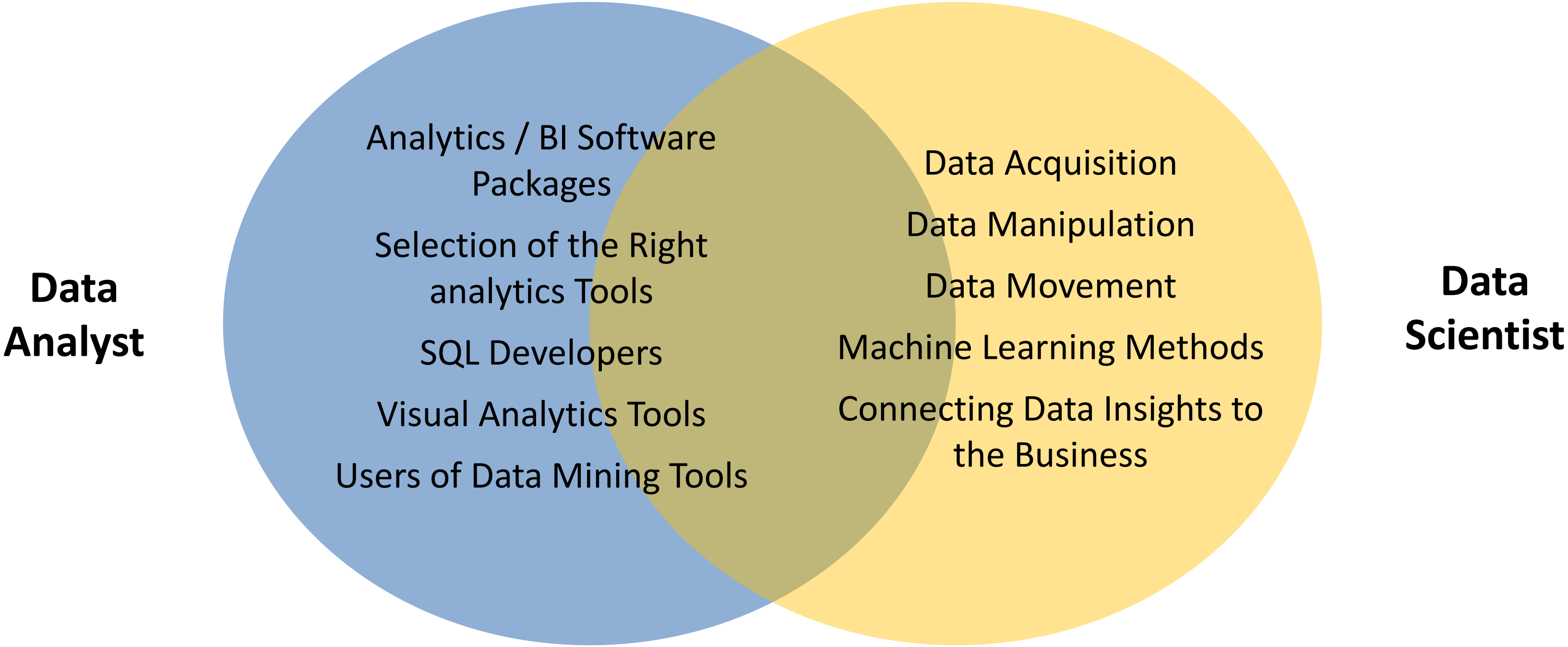
- Machines can see, hear, talk, and … they can learn!
- Machines can produce outcomes they have not been explicitly programmed for (a huge paradigm shift from traditional Computer Programming)
- But . . . they do not have common sense, no true thinking (this is still the realm of science fiction)
- Really good a things that take humans 1 second to think about

Understanding what ML algorithms can and cannot do is one of the key to success in data analytics

What Can Humans do in Less than 1s?

- Machines can:
  - Translate from one language to another
  - Predict if it is going to rain or be sunny
  - Position targeted ads to a person visiting a web site
  - Identify objects in an image
  - Identify fraudulent credit card transactions
  - Actions in driving a car
- Machines Can’t:
  - Typically anything that takes deeper reasoning, creativity, intuition, or a small amount of data

Working with Different Teams



Machine Learning Engineers Design the "A → B" Mapping Algorithms

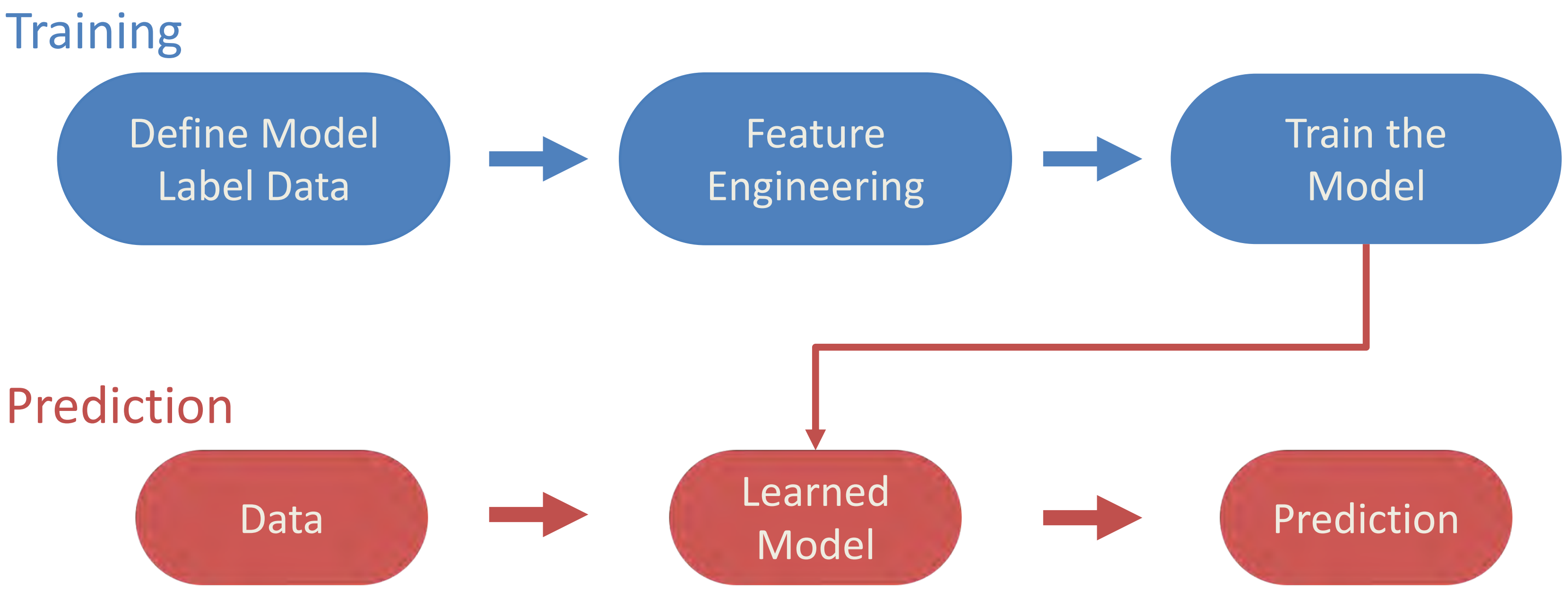
Involving Cross-Functional Teams

Application Infrastructure	Data Warehouse/Lake	AI/ML	Viz Package
AWS	Spark	TensorFlow	Grafana
Azure	HDFS / Hadoop	PyTorch	Prometheus
GCP	SAP Hana	Caffe2	ELK Stack
Premise	NoSQL	mxnet	TICK

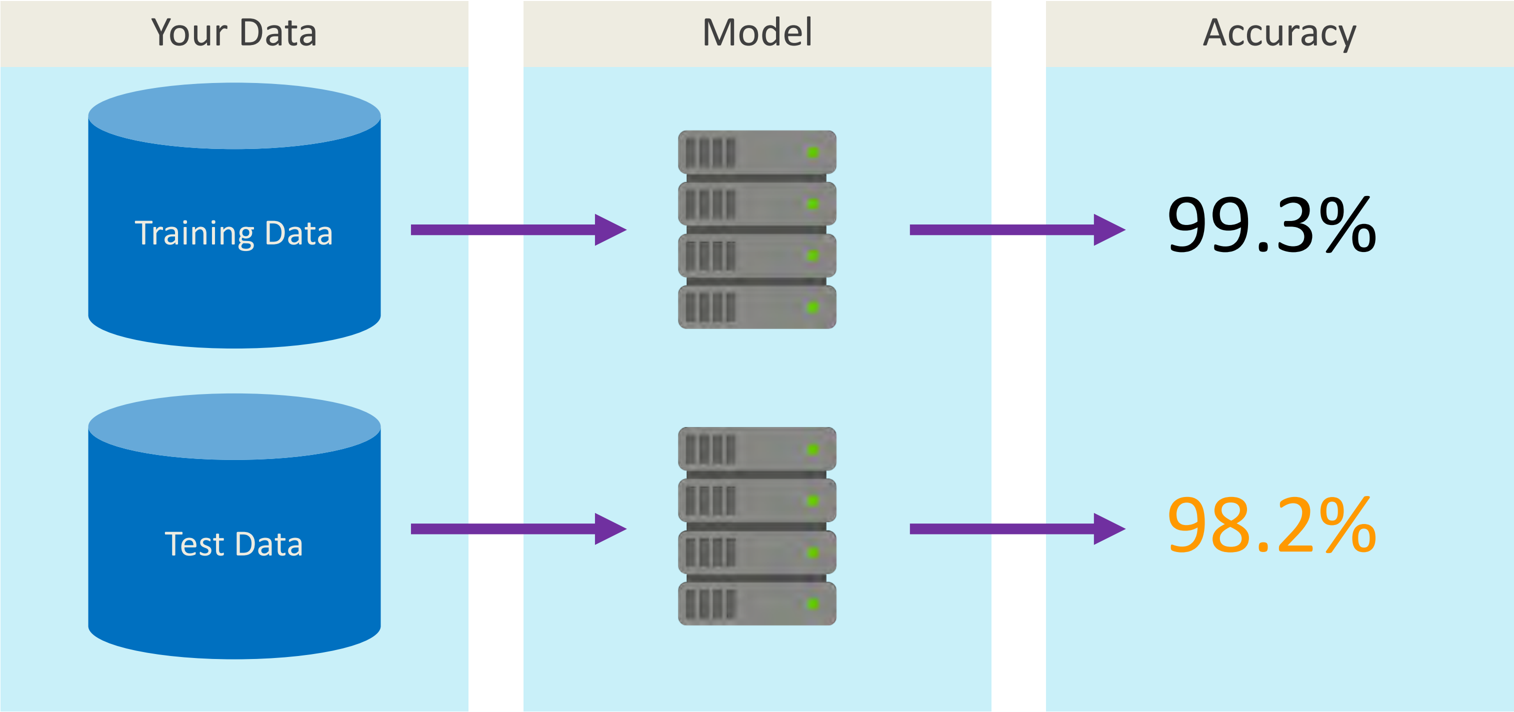
Lesson 8

8.2 An AI/ML Workflow

High Level AI/ML Workflow



Validating Your Machine Learning Model

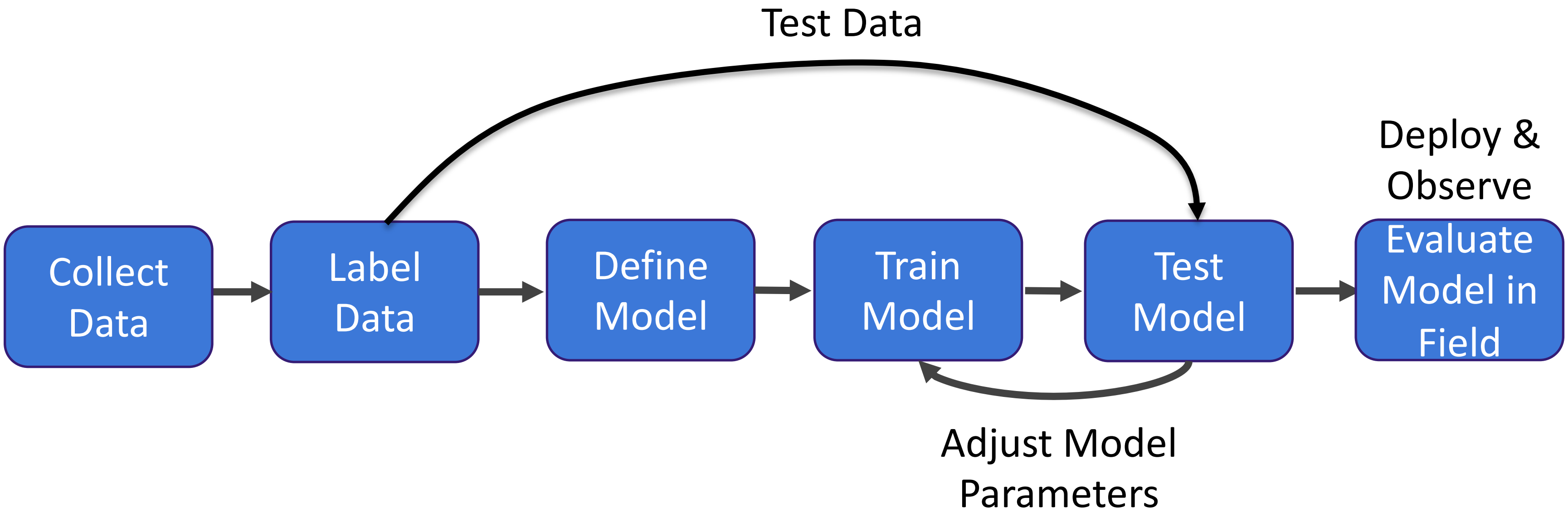


MNIST – Handwritten Digit Data Set

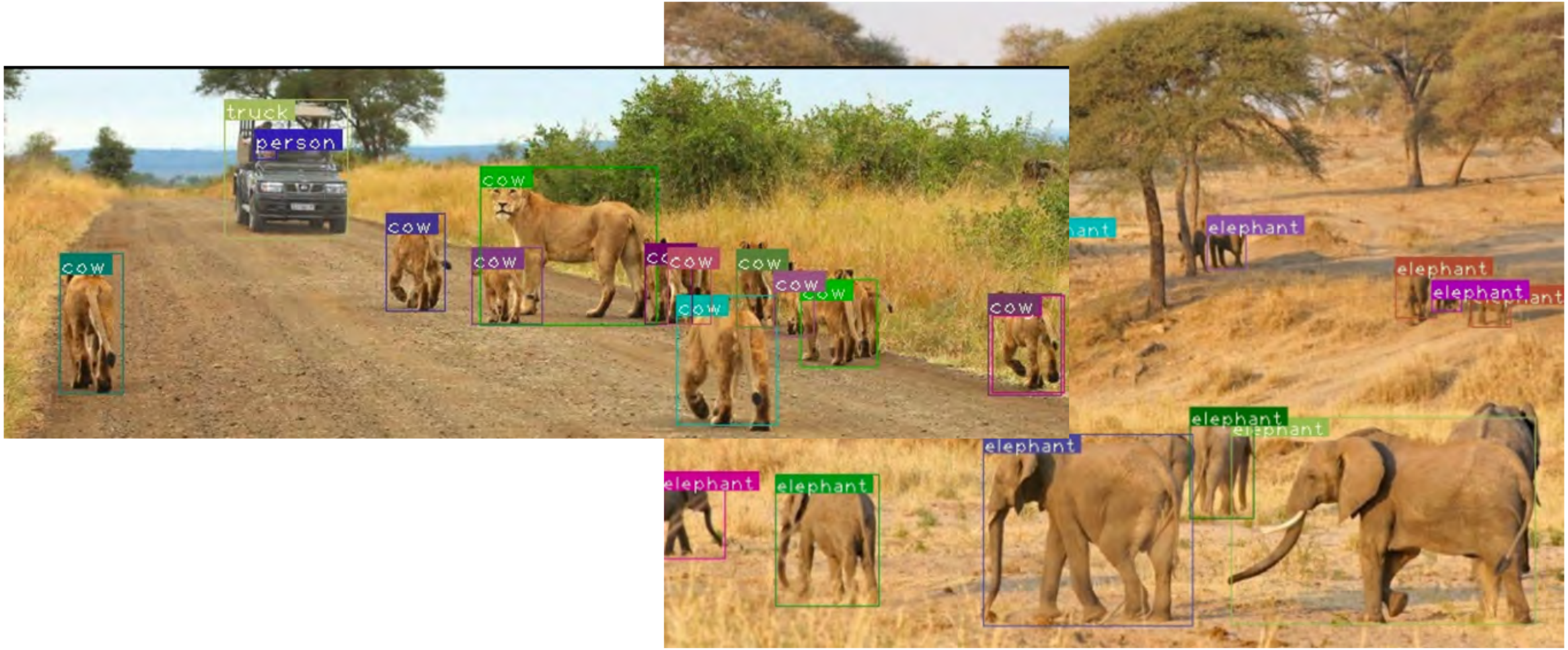
http://yann.lecun.com/exdb/mnist/

- Training set of 60,000 examples (6000 per digit)
- Each character is a 28x28 pixel box
- Test set of 10,000 examples (1000 per digit)

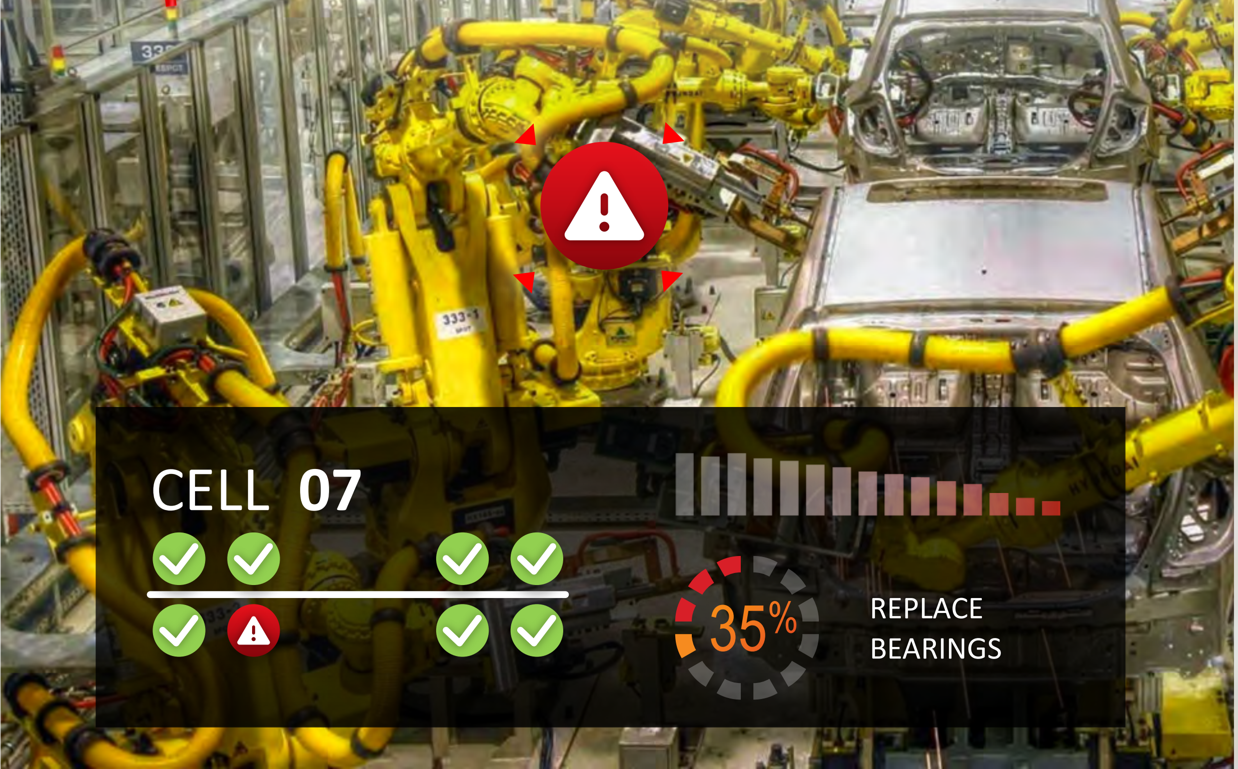
A Typical Supervised Learning ML Work Flow



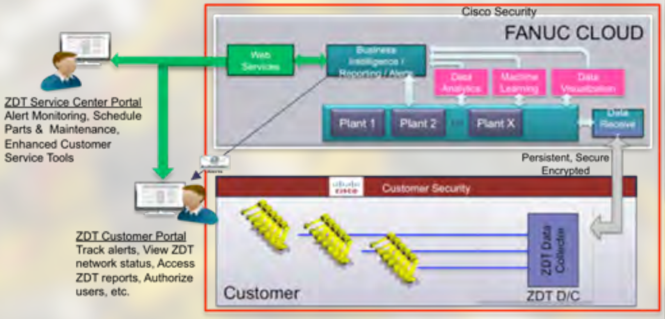
The Model May Need Tweaking Once Deployed



Having Data is Critical, but Progress can be made even with smaller data sets



- Data Collection
- Edge Compute
- Predictive Analytics
- Proactive Part Replacement



The Emergence of Large Data Sets and ML

- Open data sets have been a crucial factor in the success of ML
  - http://image-net.org/
  - https://github.com/awesomedata/awesome-public-datasets

- Allows for direct comparison of learning and inference algorithms
- The result is an improvement of error rates
  - Video analytics (facial recognition)
  - NLP/Voice recognition

https://catalog.data.gov/dataset

Search Data.Gov

DATA.GOV

DATA TOPICS IMPACT APPLICATIONS DEVELOPERS CONTACT

DATA CATALOG

/ Datasets Organizations ?

Search datasets...

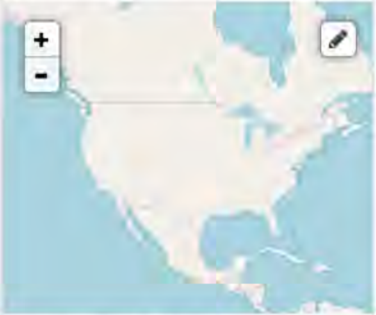
Order by:

Popular

Datasets ordered by Popular

Filter by location Clear

Enter location...



Map tiles & Data by OpenStreetMap, under CC BY SA.

301,343 datasets found

Pittsburgh Wards Map 2527 recent views

County

Allegheny County / City of Pittsburgh / Western PA Regional Data Center — Allows users to look up City of Pittsburgh Wards

HTML

National Student Loan Data System 1977 recent views

Federal

Department of Education — The National Student Loan Data System (NSLDS) is the national database of information about loans and grants awarded to students under Title IV of the Higher...

XLSX XLS XLS XLS XLS XLS 11 more in dataset

Lesson 9: An Introduction to Machine Learning Software Tools

Lesson 9: Introduction to ML Software Tools

9.1 An Overview of Machine Learning Toolkits

Machine Learning Frameworks

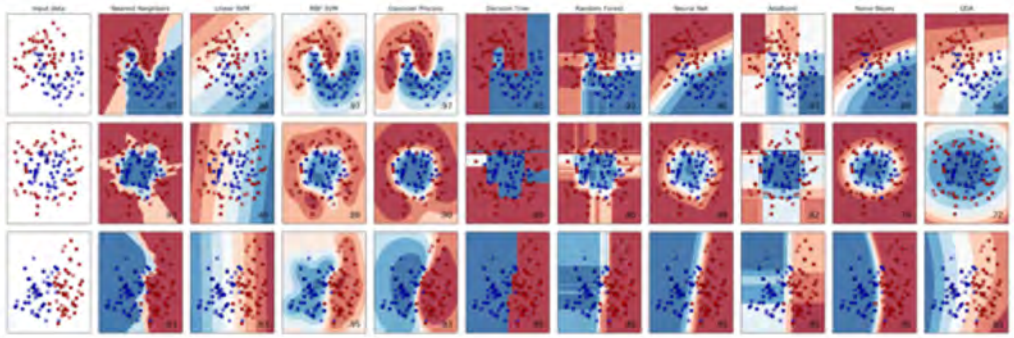
- ML frameworks offer building blocks for designing, training and validating machine learning models through a high-level programming interface
- Include libraries for the common tools we have discussed:
  - Regression
  - Neural Networks (Deep Learning)
  - Classification
  - Random Forests
  - Clustering
  - Bayesian Inference
  - PCA…

Lots of Frameworks Exist

- Many open source ML frameworks are used
  - SciKit Learn (https://scikit-learn.org/stable)
  - Google – Tensorflow (http://tensorflow.org/)
  - Facebook – PyTorch (pyrotch.org)
  - Keras (https://keras.io)
  - Microsoft – CNTK (https://github.com/Microsoft/CNTK)
  - Apache – MXNet (https://mxnet.apache.net)
  - Many more!

SciKit Learn

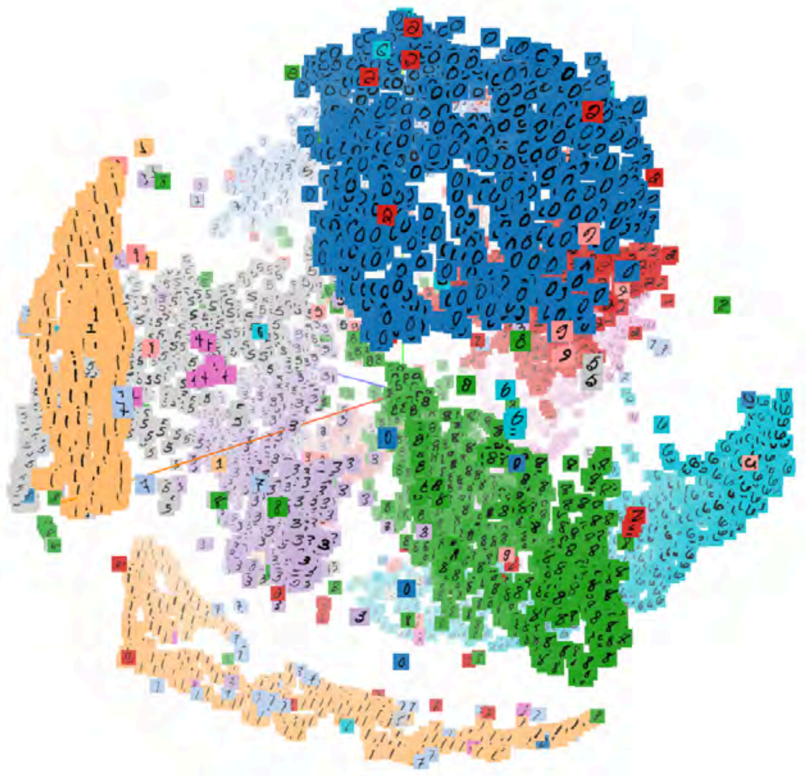
- Open Source, Python-based
- Classification, regression, clustering, dimension reduction, model morphing, feature extraction…
- Great documentation and tutorials
- Multiple online examples to work from
- CLI-based, requires some initial learning (barrier of entry)

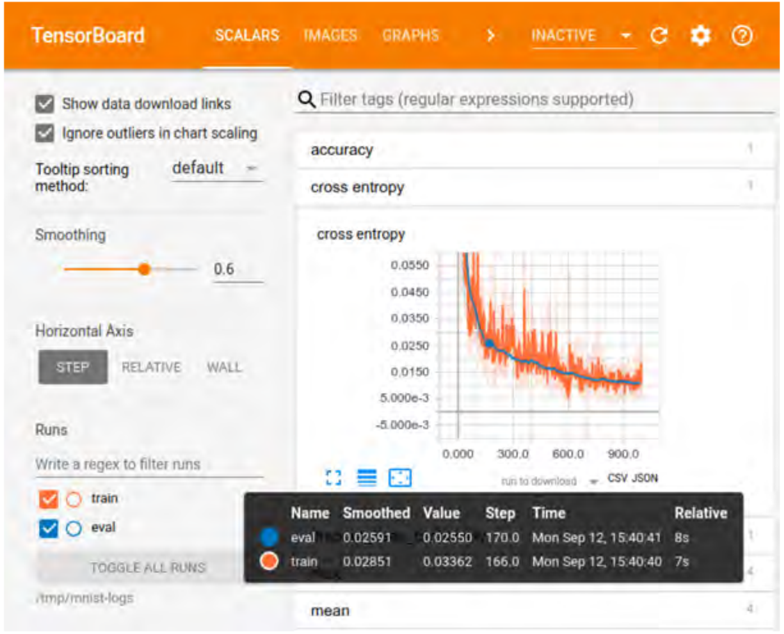


TensorFlow

- One of the most popular ML frameworks in existence
- Created by Google, written in C++
- Leveraged by Uber, AirBnB, Dropbox
- Used by Google Translate
- Excellent documentation and guidelines
- Supported by an extensive community of developers
- Primarily uses Python (knowledge of Python numpy arrays is good to have)
- Barrier to entry is a little high (not too bad though)

TensorBoard Model Visualization





PyTorch

- Developed by Facebook
- Used by Twitter, Salesforce.com, many others
- Known to have a simple architecture, giving it an easy entry point for the beginner, easy UI
- Supported by an extensive community of developers
- Lacks the visualization capabilities of TensorBoard

Keras

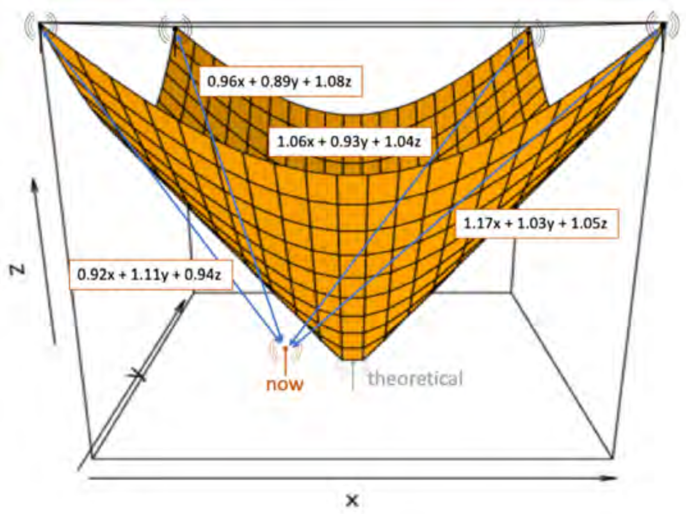
- Neural network API in Python
  - Runs on top of TensorFlow, CNTK or Theano
- Similar to Pytorch ‘in concept’, as it aims at simplifying the API and user interface
  - But Pytorch does not run on top of ‘something else’, has interactive debugging and dynamic graph definition (lacking in Keras)
  - But Keras is more mature (bigger community, plenty of tutorials)

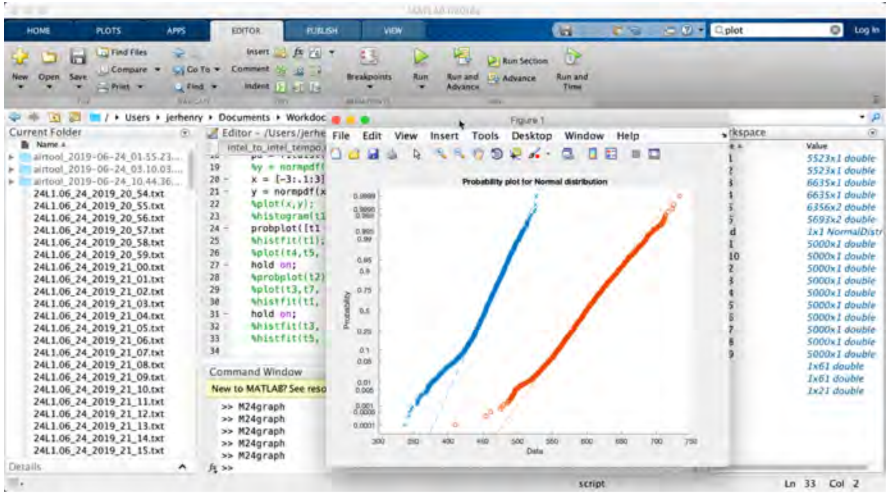
Microsoft CNTK

- CNTK = Microsoft **CogNitive ToolKit**
- More recent than the others
- Also benefited from the others' experience
- Nice function wrappers, faster execution than others
- Smaller community and smaller example set

Some Other Names

- Matlab / Octave
    - Mathematical tool
- R Studio
    - Statistical tool





Lesson 9: Introduction to ML Software Tools

9.2 A Deeper Look at TensorFlow

Project

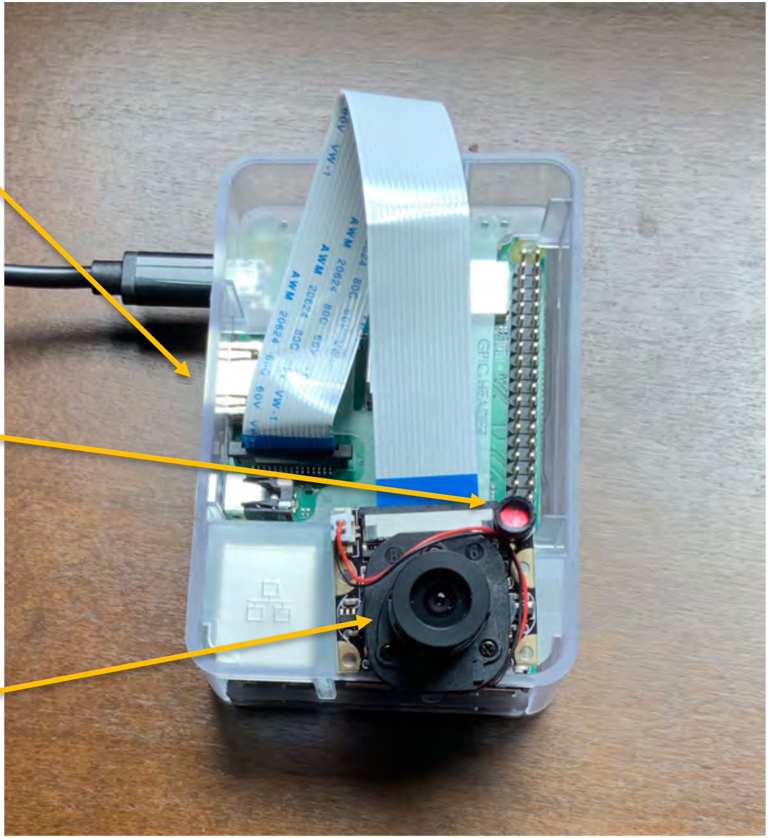
- Recognizing birds coming to a backyard feeder

Project

Raspberry Pi
(-> Linux -> Python, TensorFlow)

Presence detector

Camera



Project Preparation

1. Install motion detection-based camera trigger (e.g. https://github.com/Motion-Project/motion)
2. If running neural network locally, install Python, TensorFlow (https://www.tensorflow.org/install/source_rpi)
3. Create folders, train on something you know should work (Google example with flowers)
4. Check the result:
    1. pip3 show tensorflow,
    2. cd tensorboard
    3. python3 main.py --logdir /tmp/retrain_logs
    4. Then http://raspberrypi:6006

Running the Project

5. Get images to train from
6. Train on your images
   - python3 retrain.py --bottleneck_dir=../tensorflow/tf_files/bottlenecks --model_dir=../tensorflow/tf_files/inception --output_graph=../tensorflow/tf_files/retrained.graph.pb --output_labels=../tensorflow/tf_files/retrained.labels.txt --how_many_steps=8000 --image_dir=../tensorflow/tf_files/bird_photos/
7. Then have your script take a picture, and run the label_image.py script against that image, store the result.

Lesson 10

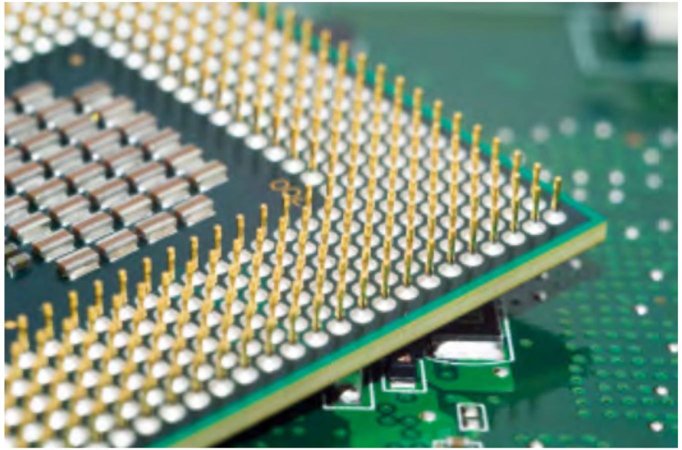
10.1 An Introduction to Graphics Processing Units (GPUs)

Machine Learning Requires Heavy Lifting!

- Regression and Classification methods require massive data sets to train the models
- Unsupervised learning involves massive amounts of seemingly "mixed up" data. Each data point needs to be processed to find the patterns
  - K-Means and other algorithms require mathematical computation for every single data point
- Neural Networks involve a significant amount of parallel processing (CPUs support at best a small number of cores)

CPUs

- CPUs are “general processors”, generally based on RISC architecture
- Have been the mainstay of computers for decades
- Extremely flexible – can be used for almost any task
- However - they are primarily focused on serial and linear processes



CPUs vs. GPUs

- CPUs are capable of almost any task – but at a price

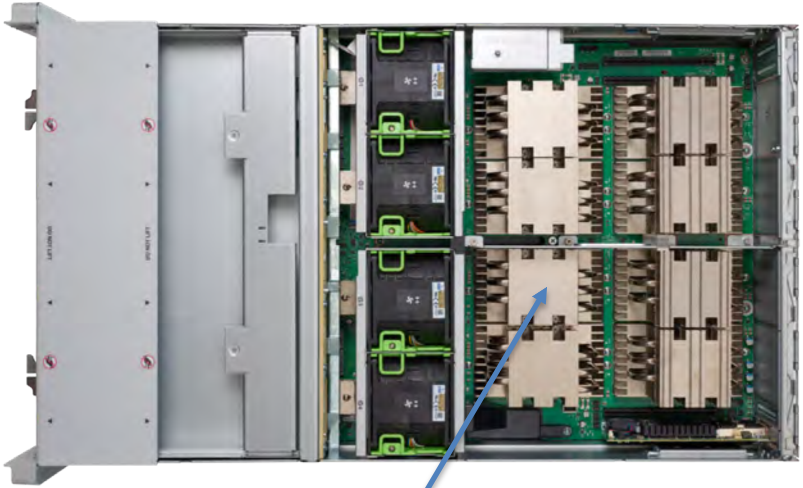
CPUs are like a swiss army knife

- GPUs are highly-specialized processors used to solve complex math problems

GPUs are like specialized surgical instruments

Graphical Processing Units

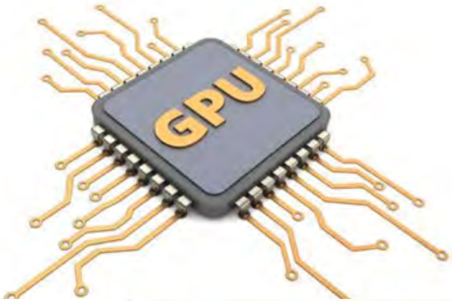
- Introduced by NVIDIA in 1999
- GPUs are well-known in the gaming industry
- Designed to process data in parallel and rapidly manipulate and alter memory to accelerate images and graphics (good for video games)
- Very good at texture-mapping
- By 2005 a new use was found for them – Machine Learning!

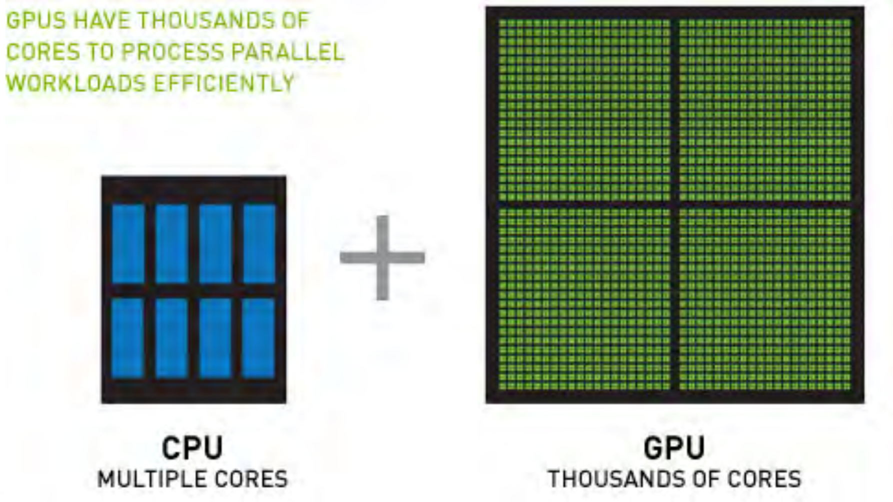


A Server with 8x Tesla V100: 640 Tensor Cores per GPU

The Power of GPUs for Deep Learning

- GPUs support parallel processing, accelerating their ability to execute algorithms that require parallel processes
- Parallel processing is what makes them ideal for AI/ML workloads
- The number of parallel computations depends on the number of cores
- Not ideal for serial-processing!





10s of threads

1000s of threads

Google Brain's YouTube Cat Video Detector

"NOW YOU CAN BUILD GOOGLE'S $1M ARTIFICIAL BRAIN ON THE CHEAP" – WIRED MAGAZINE

GOOGLE DATA CENTER	STANFORD AI LAB
1000 Servers / 16,000 cores $1,000,000 600 KWatts	3-GPU Accelerated Servers / 18,432 cores $20,000 4KWatts

Embarrassingly Parallel

Little effort is needed to break down a problem into smaller (parallel) tasks

Not logged in Talk Contributions Create account Log in

Article Talk

Read Edit View history

Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction

Help
About Wikipedia
Community portal

Embarrassingly parallel

From Wikipedia, the free encyclopedia

In parallel computing, an **embarrassingly parallel** workload or problem (also called **perfectly parallel** or **pleasingly parallel**) is one where little or no effort is needed to separate the problem into a number of parallel tasks.[1] This is often the case where there is little or no dependency or need for communication between those parallel tasks, or for results between them.[2]

Thus, these are different from distributed computing problems that need communication between tasks, especially communication of intermediate results. They are easy to perform on server farms which lack the special infrastructure used in a true supercomputer cluster. They are thus well suited to large, Internet-based distributed platforms such as BOINC, and do not suffer from parallel slowdown. The opposite of embarrassingly parallel problems are inherently serial problems, which cannot be parallelized at all.

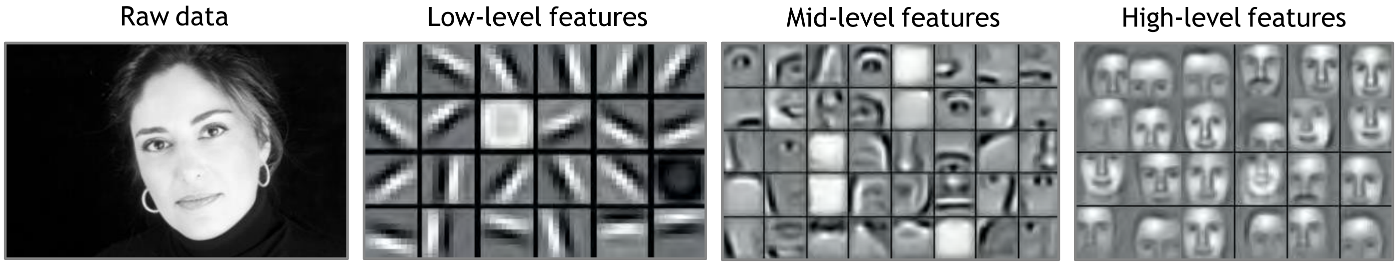
A common example of an embarrassingly parallel problem is 3D video rendering handled by a graphics processing unit, where each frame (forward method) or pixel (ray tracing method) can be handled with no interdependency. Password cracking is another embarrassingly parallel task that is easily distributed on central processing units, CPU cores, or clusters.

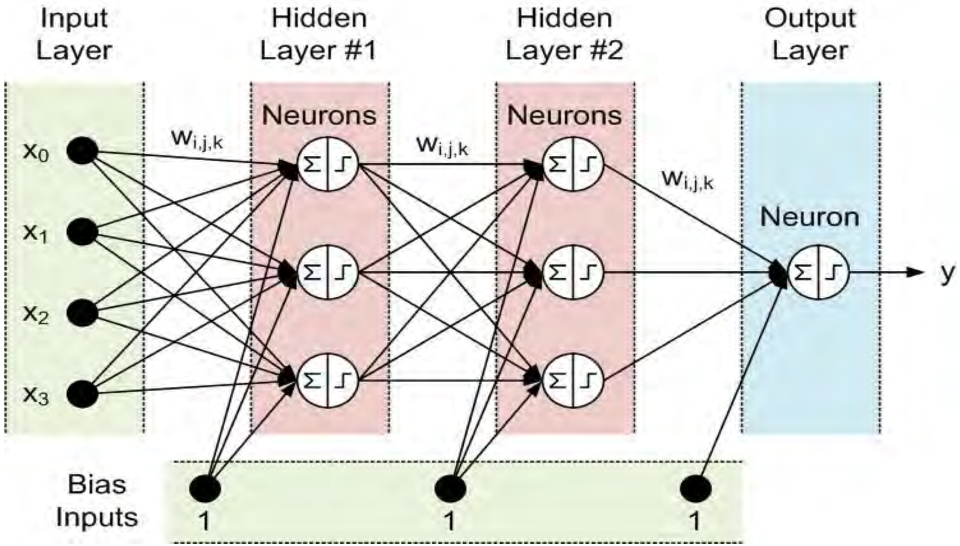
Example – GPUs and Deep Learning

What You See

What a Computer Sees

Neural Networks & GPU





	Neural Networks	GPU
Inherently Parallel	✓	✓
Matrix Operations	✓	✓
Bandwidth	✓	✓

Google Cloud Tensor Processing Unit
Google Designed AI/ML Hardware Acceleration

- Announced in 2016 by Google – a specialized ML hardware accelerator
- Specifically designed for Google's TensorFlow framework
- CISC (Complex Instruction Set Computing design) implements instructions that run complex tasks
- Includes symbolic math library which is used for ML and Neural Networks
- Capable of processing hundreds of thousands of matrix operations in a single clock cycle

Lesson 10

10.2 Programming GPUs for Machine Learning

Implementing AI – the Software and Hardware Stack

AI/ML Application
(Computer Vision, NLP)

AI/ML Frameworks
(Tensorflow, PyTorch)

Libraries for compute intensive tasks
(cuDNN)

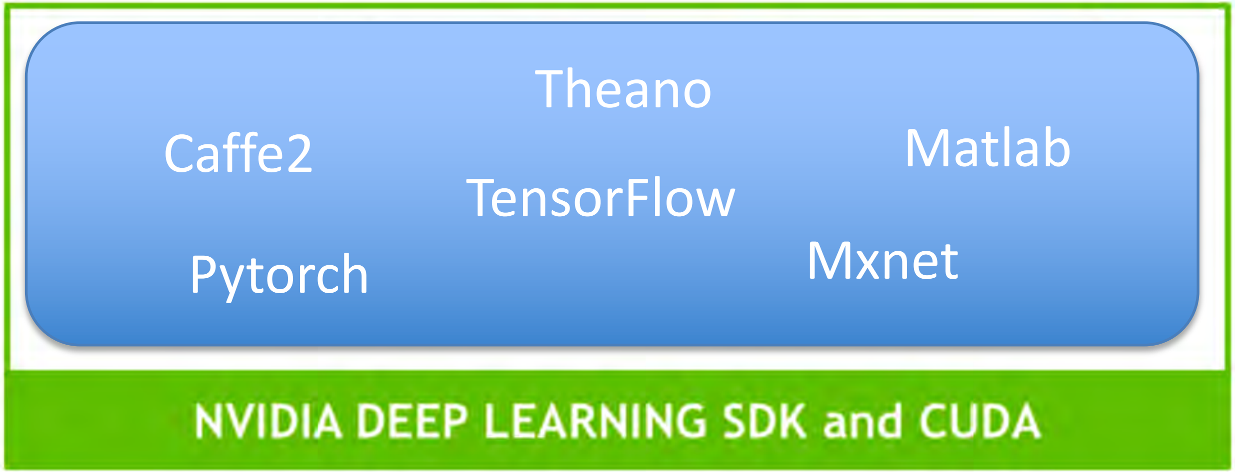
System software drivers for the GPUs
(CUDA)

Hardware accelerators for the AI job
(GPU/TPU)

Developed by Nvidia

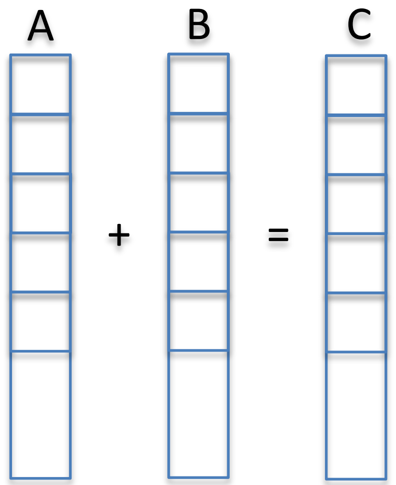
How Do You Program a GPU for Machine Learning?

- CUDA is a software API for developers on NVIDIA GPUs
- GPU is the hardware / CUDA is the software architecture for the GPU
- Includes GPU-accelerated libraries through cuDNN (CUDA Deep Neural Network)



A Quick Intro to CUDA

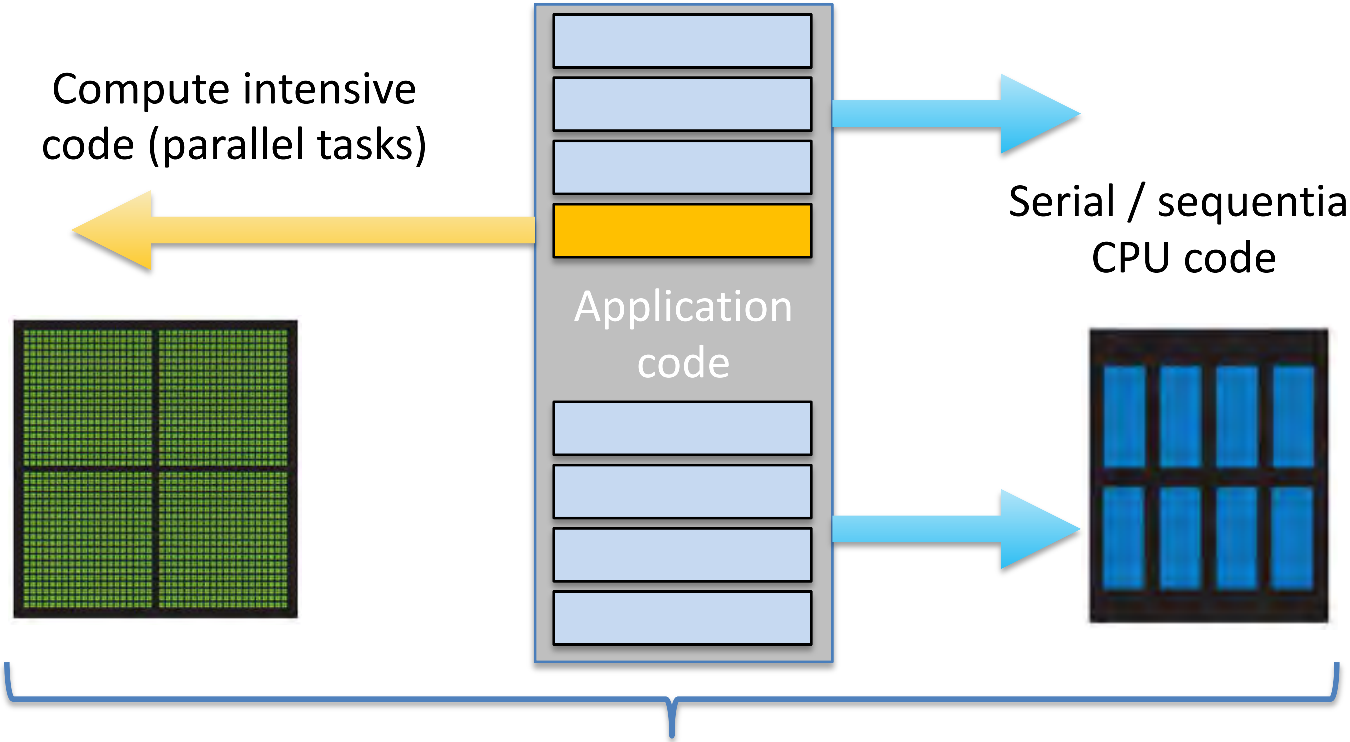
- CUDA allows programming of 100s / 1000s of cores in parallel for an AI/ML job
- Write a task for one data element, and it is replicated to multiple cores – good for computing lots of data




```
For(i = 0, n-1)
  {
    C[i] = A[i] + B[i]
  }
```

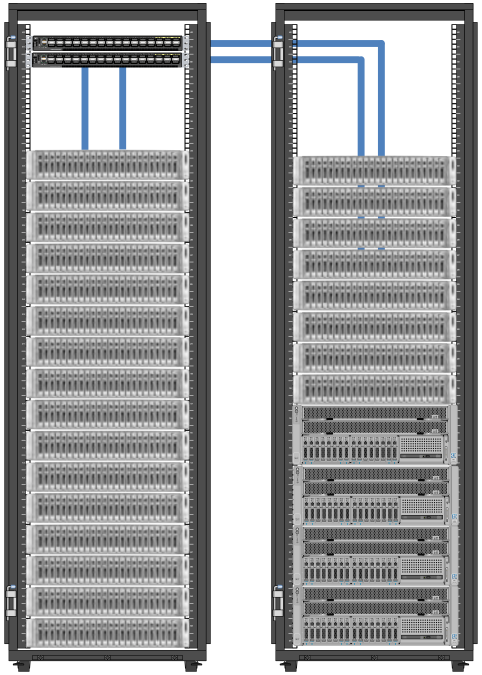
```
VecAdd(A,B,C)
  {
    int i = threadIDx
     C[i] = A[i] + B[i]
  }
//execute the kernel
vecAdd<<<n>>>(A,B,C)
```

How GPU Acceleration Works



Deploying a GPU-Enabled Hadoop Cluster
GPU-Powered Data Nodes with NVIDIA CUDA Containers and YARN

Hadoop / YARN



ML/DL with data stored on HDFS file system

GPU acceleration for ML/DL workloads

GPU Powered Data Nodes

NVIDIA CUDA containers orchestrated with YARN for Deep Learning

ML/DL frameworks

Pytorch mxnet TensorFlow

Lesson 11

11.1 New Applications in AI/ML

Investment in AI/ML Is Skyrocketing

- Machine learning patents grew at 34% compound annual growth rate between 2013 and 2017 (source: IFI Claims Patent Services)

- Machine learning market is expected to grow from $1.41B to $8.81B between 2017 and 2022 (source: Newswire)

Improved Customer Experience

Consumer experience:

- Personalized interactions and Customer Service
- Improved apps and chatbots that mimic human behavior

Interview with Eugene Goostman, the Fake Kid Who Passed the Turing Test

- Time Magazine

AI/ML in Healthcare

AI powered tools are transforming healthcare

WATERLOO NEWS

Waterloo News » News » 2017 » August »

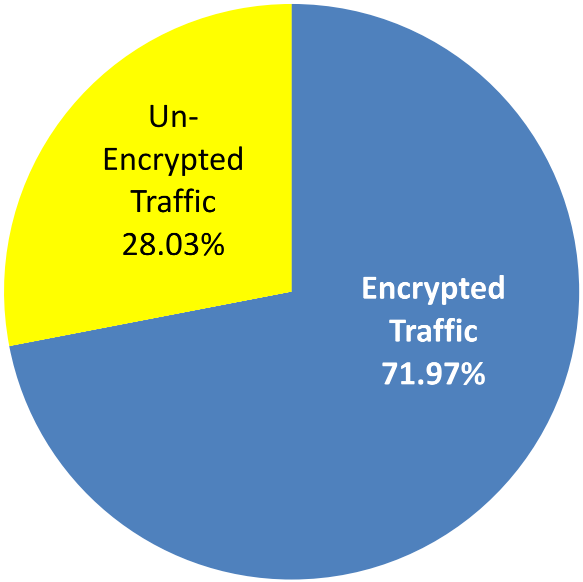
Artificial intelligence tool promises earlier detection of deadly form of skin cancer

https://uwaterloo.ca/news/news/artificial-intelligence-tool-promises-earlier-detection

AI/ML and Network Security

Cyber Security

- Faster detection of new and emerging cyber threats (using classification and anomaly detection)
- Today a majority of Internet traffic is encrypted, making detection of malware very difficult
- New AI models use Deep Learning to profile patters of behavior instead of looking inside the packet

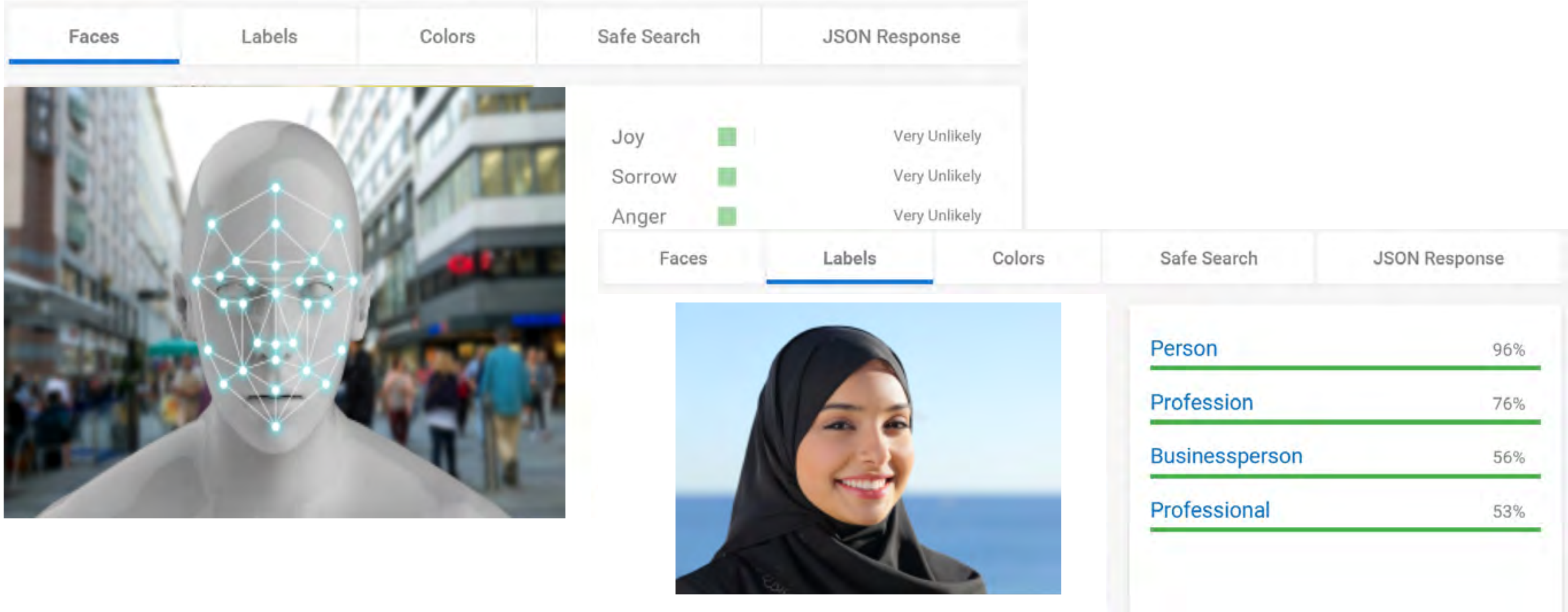


The Potential of Machine Learning

Lesson 11

11.2 Managing Bias in AI Systems

AI Development in Computer Vision



"If We Want Machines to Think, We Need to Teach Them to See"

- *Fei-Fei Li,* Stanford AI Lab Director

Bias in Machine Learning

- AI systems can easily become bias / discriminatory
- Example, a credit card company is looking at applications for new customers . . . How will they decide “creditworthiness”?
  - Decisions on creditworthiness are computed based on business metrics not on “fairness” where you come, your background, or anything else.
  - Machines have no concept of morality of fairness
- Models may sometimes be designed and trained on data from a place where the model is not going to be used (social context).

Bias in Machine Learning

- Datasets can often be the root cause of AI bias:
  1. Data may reflect existing prejudices
  2. The dataset may not reflect reality of the population

- Data Preparation needs to be considered:
  1. Data prep involves selecting the right attributes for the algorithm
     - E.g. attributes could be age, income, gender, education level, etc.
  2. Choosing which attributes to ignore and which to use in the model is something of an art

Ethical Questions are Emerging

- How do we guard against mistakes made by machines? Who is liable?
- What about self-driving cars?

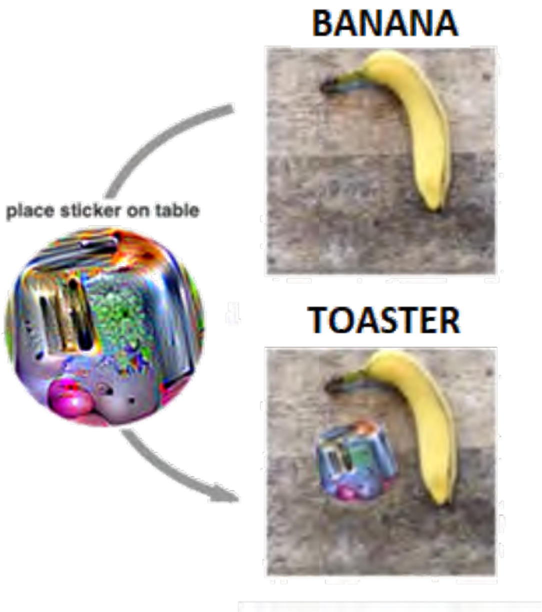
- How do we eliminate AI bias?
  - Can we allow machines to judge other humans based on a learning mechanism?

Lesson 11

11.3 An AI/ML Reality Check

Adversarial Examples in DNNs

https://youtu.be/i1sp4X57TL4



- Patches can be printed, added to any scene, photographed, and presented to image classifiers

- **Robust Physical-World Attacks** (RP2) generate robust, physically realizable adversarial perturbations
- How will an autonomous car behave ??

https://iotsecurity.engin.umich.edu/physical-adversarial-examples-for-object-detectors/

Some other examples...

A woman riding a horse on a dirt road.

An airplane is parked on the tarmac at an airport.

group of people standing on top of a beach.

An AI/ML Reality Check

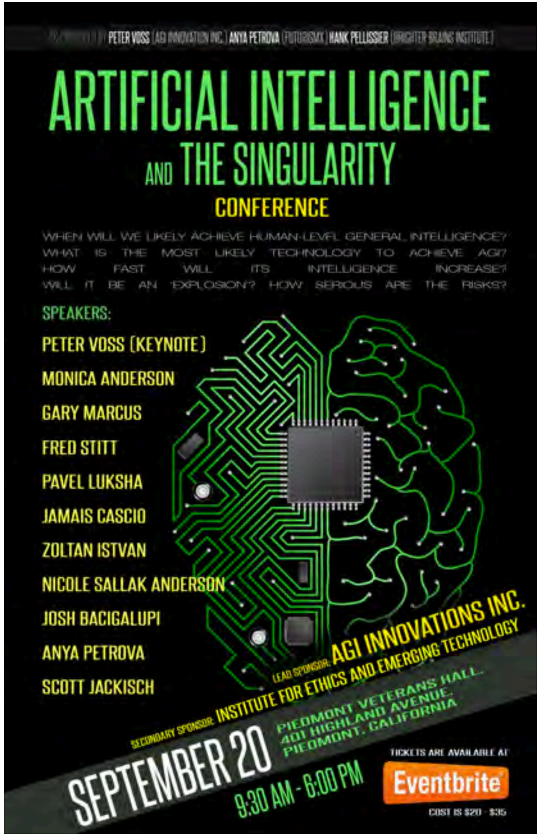
- The impressive achievements of DL amount to fitting a curve to data
- ML/DL systems operate in a statistical, or model-free mode
  - There are limits to what they can do
- ML/DL systems cannot reason about interventions and retrospection
- Data science is about interpretation of data
  - Regardless how big data sets get and how skillfully they are manipulated
- To Build Truly Intelligent Machines, teach them cause and effect

What About the “Singularity”?

- Will machines ever become self-aware?

Wikipedia Definition: *The* ***technological singularity*** *(also, simply, the* ***singularity****) is the hypothesis that the invention of artificial superintelligence (ASI) will abruptly trigger runaway technological growth, resulting in unfathomable changes to human civilization.*

- How do we control these machines if they become self-aware?
- Today - still in the realm of science fiction



Many Mysteries Remain . . .

- Deep Learning
  - Back-propagation: Why is such a simple algorithm so powerful?
  - Many adversarial examples exist (when algorithms misclassify)

- How to scale up ML Algorithms
  - How can we scale to millions of training examples, thousands of features, hundreds of classes?

- One-Shot Learning
  - How can we learn from very few training examples?