

Statistics

Parameswari Ettiappan

Data from Various Sources

- CSV file
- Flat File (tab, space, or any other separator)
- Text File (In a single file — reading data all at once)
- ZIP file
- Multiple Text Files (Data is split over multiple text files)
- Download File from Internet (File hosted on a server)
- Webpage (scraping)
- APIs (JSON)
- Text File (Reading data line by line)
- RDBMS (SQL Tables)

Common data problems

- Inconsistent column names
- Missing data
- Outliers
- Duplicate rows
- Untidy
- Need to process columns
- Column types can signal unexpected data values

Unclean data



	Continent	Country	female literacy	fertility	population
0	ASI	Chine	90.5	1.769	1.324655e+09
1	ASI	Inde	50.8	2.682	1.139965e+09
2	NAM	USA	99.0	2.077	3.040600e+08
3	ASI	Indonésie	88.8	2.132	2.273451e+08
4	LAT	Brésil	90.2	1.827	NaN

- Column name inconsistencies
- Missing data
- Country names are in French

Data Cleansing Techniques

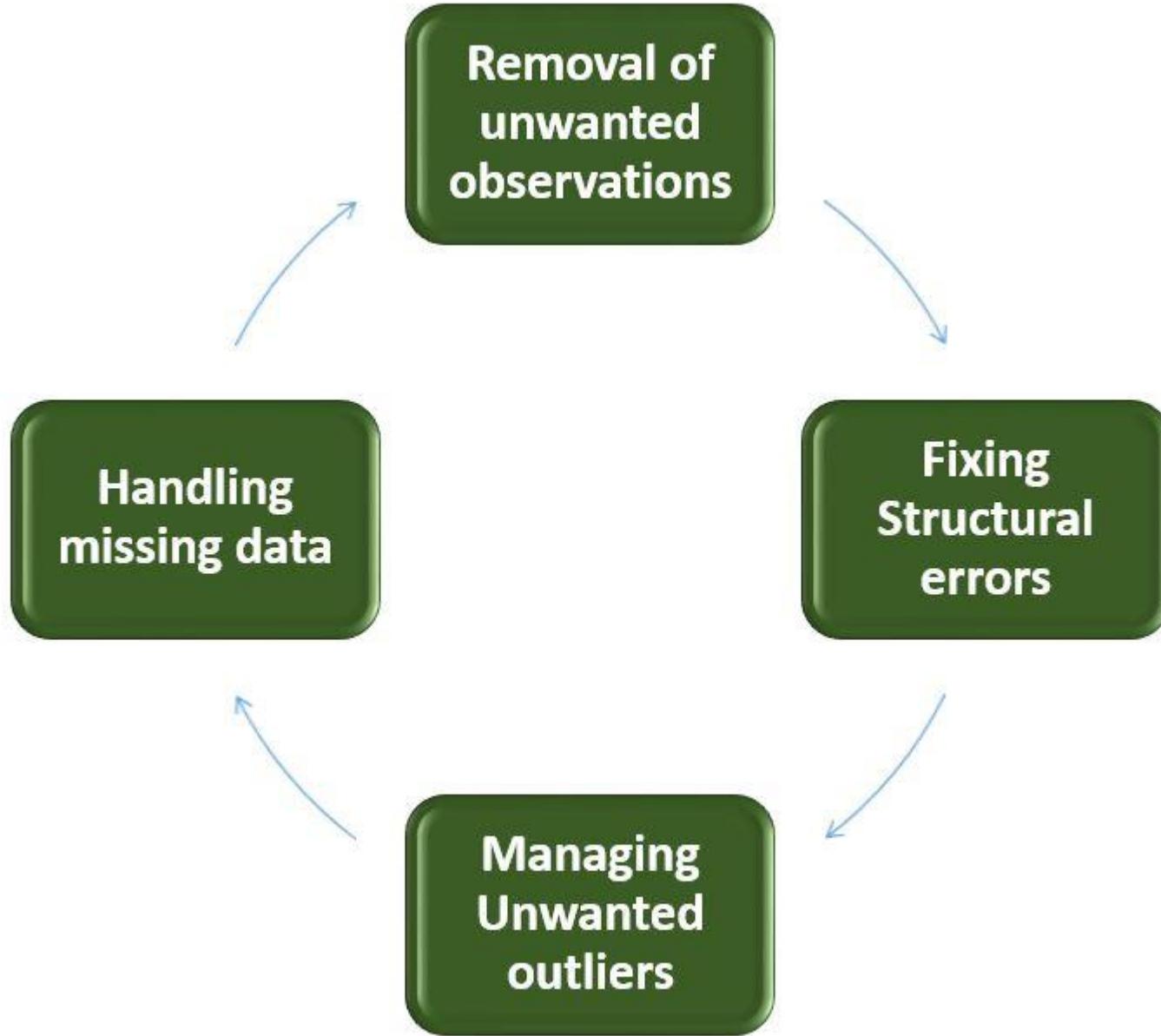
- Data forms the backbone of any data analytics you do.
- Regarding data, there are many things to go wrong – be it the construction, arrangement, formatting, spellings, duplication, extra spaces, and so on.
- To perform the data analytics properly we need various data cleaning techniques so that our data is ready for analysis. It's commonly said that,
- “Data scientists spend 80% of their time cleaning and manipulating data and only 20% of their time actually analyzing it.”

Data Cleansing Techniques

- Data cleansing or data cleaning is the process of identifying and removing (or correcting) inaccurate records from a dataset, table, or database and refers to recognising unfinished, unreliable, inaccurate or non-relevant parts of the data and then restoring, remodelling, or removing the dirty or crude data.
- Data cleaning techniques may be performed as batch processing through scripting or interactively with data cleansing tools.

Data Cleansing Techniques

- After cleaning, a dataset should be uniform with other related datasets in the operation.
- The discrepancies identified or eliminated may have been basically caused by user entry mistakes, by corruption in storage or transmission, or by various data dictionary descriptions of similar items in various stores.



How Data Cleansing is useful?

- Improves Custom Acquisition-Related Activities
- Better Decision Making
- Streamlined Business Process
- Increased Productivity and Revenue

Steps Involved in Data Cleansing

- Removal of Unwanted Observations
 - Irrelevant Observations – These observations don't fit accurately to the specific problem that the user is trying to solve. During this step, user has to review charts from the Exploratory Analysis.
 - Duplicate Observations – This type of observation arises frequently during data collection and user associated processes to it such as scrape data, combination of datasets from multiple destinations and receive data from different departments or clients.

Steps Involved in Data Cleansing

- Fixing Structural Errors
 - The next step of data cleaning is fixation of structural errors.
 - These type of errors mostly arise during data transfer, measurement and poor data-keeping.
 - Structural errors include mislabelled classes, name feature typos, use of same attribute with different names, etc.

Steps Involved in Data Cleansing

- Managing Unwanted Outliers
 - Unwanted outliers can cause serious issues with certain types of data models.
 - When a user legitimately removes an outlier, it exceptionally improves the model's performance.
 - Thing to remember here is, unless the outlier is proven unwanted or include with suspicious measurements, the user should never remove it.

Steps Involved in Data Cleansing

- Handling Missing Data
 - This one is probably the most complex step of data cleansing. As most of the algorithms don't accept missing values, the user has to manage the missing data in some way. The two most commonly recommended ways to manage missing data are:
 - To drop observations for data that have missing values.
 - To impute the required missing values based on observations.

Steps Involved in Data Cleansing

- Data missingness is always informative in itself and the user requires to inform an algorithm if a value was missing.
- Even the user builds an effective model to impute the values, it will not add any real information as it will be like reinforcing the patterns that are already provided by other features.

Steps Involved in Data Cleansing

- Missing Categorical Data – As per data science, labelling the missing data for categorical features as ‘missing’ is the best way to handle them. This step includes essentially adding a new class for the feature. This also nullifies the technical requirement for no missing values.
- Missing Numeric Data – The user has to flag and fill for missing numeric data. To perform this, the user needs to flag the observation with a missingness indicator variable. Then, replace the missing values with zero to meet the technical requirement of missing values.

What are the Tools in Data Cleansing?

- OpenRefine
- Trifacta Wrangler
- Cloudingo
- IBM Infosphere Quality Stage
- JASP
- RapidMiner
- Orange
- Talend Data Preparation

Data Cleansing Techniques in Excel

- Data Cleaning Techniques-Get Rid of Extra Spaces-TRIM
- Data Cleaning Techniques-Select and Treat All Blank Cells
- Data Cleaning Techniques-Remove Duplicates
- Data Cleaning Techniques-Highlight Errors
- Data Cleaning Techniques-Convert Numbers Stored as Text into Numbers
- Data Cleaning Techniques-Change Text to Lower/Upper/Proper Case
- Data Cleaning Techniques-Delete all Formatting

Data cleaning

- All data sources potentially include errors and missing values – data cleaning addresses these anomalies.
- Not cleaning data can lead to a range of problems, including linking errors, model misspecification, errors in parameter estimation and incorrect analysis leading users to draw false conclusions.
- The main data cleaning processes are editing, validation and imputation.

Data Removal

- This is the most frowned-upon method. For missing values, it is better to investigate the reason instead of simply eliminating the rows or columns that contain the missing values.
- This is not always avoidable, though. If an entire column is 85% missing and you cannot find another data source, you may not be able to use that column.
- Additionally, it's not optimal to remove outliers, as this is a kind of results doctoring. If you do remove datapoints, explain the reasoning for doing so (such as 85% of the data is irrecoverable) in the results and report.

Direct Correction

- For string consistency correction in smaller categorical sets, it can be trivial to run a unique values search and then write a couple of if-statements to replace errors.
- If you have something like city names, it may be difficult to go with explicit if-statements. You may want to use a fuzzy search and make corrections that way.

Direct Correction

- Numerical consistency errors, such as order of magnitude mismatches, are simple to fix by multiplication or division.
- Binary consistency issues can be corrected if you can accurately assign the non-binary input to one of the binary categories.
- In the set {on, off, broken}, you can probably safely map *broken* to *off*.
- Errors that arise from malfunctioning sensors or human input errors should also be corrected from the source, if possible.
- If you are using publicly available or large-scale, one-time-collection data sets, though, this won't be possible. In those cases, you may want to impute the values.

Scaling

- Scaling changes the ranges of data so some features do not dominate solely because they naturally produce larger values.
- For example, temperature for a city tends to have a much smaller range than the population for a city.
- Distance-based algorithms will assign much greater importance to the population variable, possibly entirely ignoring the temperature variable.
- Scaling brings variables in line with each other while retaining the proportional relationships within the variable. This is seen when you convert to percentages or baseline to 100.

Imputation

- This technique is most closely associated with filling in missing values, but it can be used for incorrect values, too, especially when a direct correction cannot be made.
- Imputation is a fancy way to say guess. However, since we are in the field of data science, this will be a data-driven guess, not just a random guess.
- You can impute values with statistical indicators (like mean, median, mode), hot-decking, stratification, and others.

Imputation

- One approach is to replace every missing value with a statistical indicator.
- In our missing building survey example above, if you just used the mean score for all missing data, you may overlook a strong negative sentiment in that building (which was why the building manager “forgot” to distribute the survey).
- Hot-decking fills in missing values by randomly selecting a value from the set of already-known values. Again, this can cause you to overlook important information believed by “missingness.”

Imputation

- Finally, stratification is beneficial if you already know some patterns in your data.
- The heights of women are, on average, shorter than the heights of men.
- You could split your data set into men and women, then use those sub-indicators for replacement or hot-deck from the subsets of men and women.
- Is it perfect? No, but it's better than using the indicators or hot-decking from the entire population.

Flagging

- This is particularly useful for missing values when you don't want to drop all of them.
- For numeric data, you can add another column to your data set and flag any missing values there.
- This will inform your algorithm of missing values, which may turn out to be influential.
- For categorical variables, simply create a “Missing” or “Unknown” class.

Editing

- Data editing can take place at different levels, and use different methods – the choice is known as the data editing strategy.
- Different data editing strategies are needed for each data type – there is no “one size fits all” solution for a S-DWH(Statistical Datawarehouse).

Macro- and micro-editing

- Macro-editing is generally subjective – eye-balling the output, in isolation and/or relative to similar outputs/previous time periods, or calculating measures of growth and applying rules of thumb to decide whether they are realistic or not.

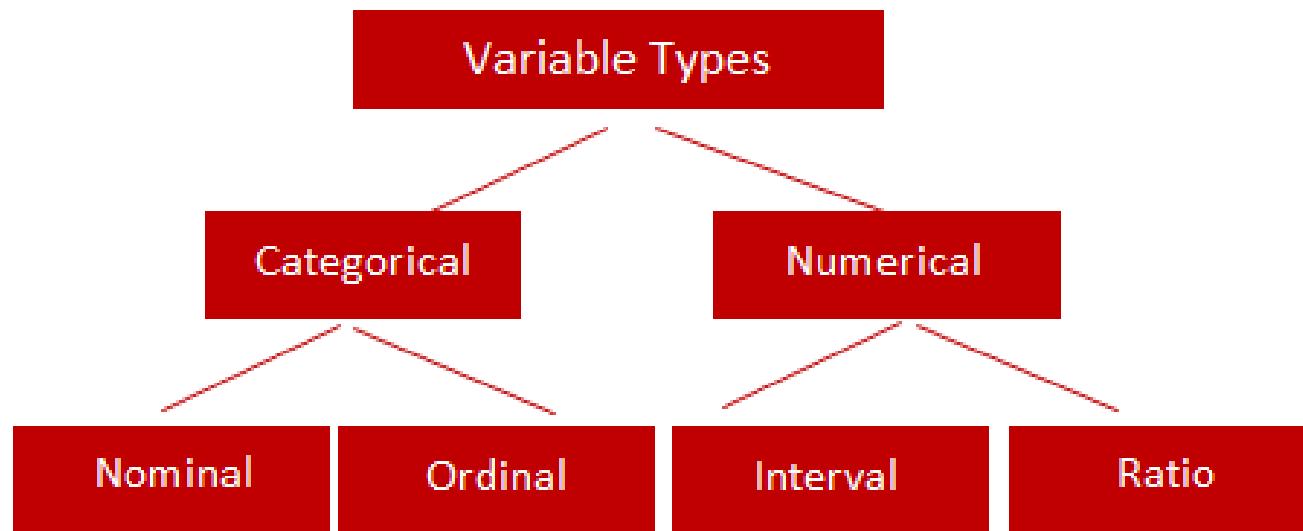
Macro- and micro-editing

- Micro-editing methods are numerous and well-established, and are appropriate for a S-DWH where editing should only take place in the sources layer.

Macro- and micro-editing

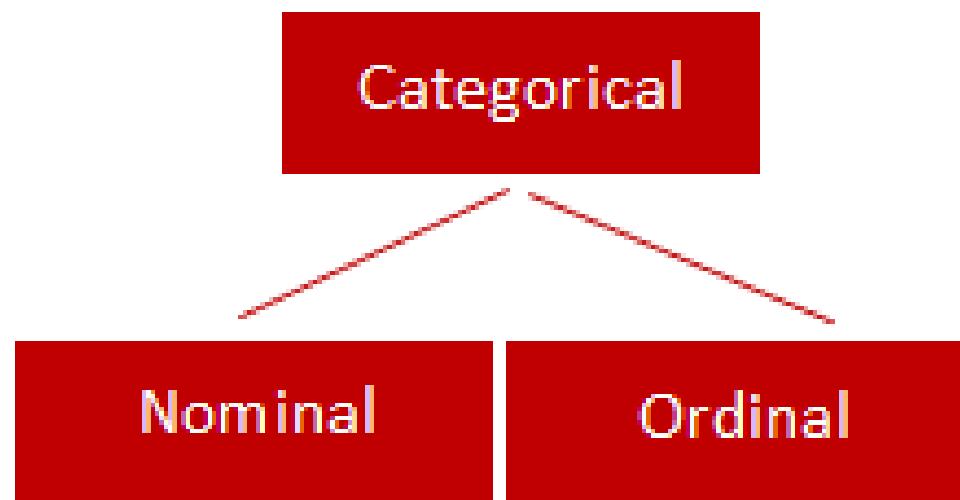
- Hard and soft edits
- Editing methods – known as rules – detect errors, but once a response fails the treatment varies dependent on the rule type.
- Hard edits (some validity, consistency, logical and statistical) do not require validation and can be treated automatically.
- Soft edits (all remaining) require external validation.

Types of Variables



Categorical

- Qualitative data are often termed **categorical data**.
- Data that can be added into **categories** according to their characteristics.
- Examples are gender, social class and blood types



Nominal Variable (Unordered list)

- A variable that has two or more categories, without any implied ordering.
- **Examples :**
 - Gender - Male, Female
 - Marital Status - Unmarried, Married, Divorcee
 - State - New Delhi, Haryana, Illinois, Michigan

Ordinal Variable (Ordered list)

- A variable that has two or more categories, with clear ordering.

Examples :

- Scale - Strongly Disagree, Disagree, Neutral, Agree, Strongly Agree
- Rating - Very low, Low, Medium, Great, Very great

Interval

- An interval variable is similar to an ordinal variable, except that the intervals between the values of the interval variable are equally spaced.
- In other words, it has order and equal intervals.

Examples :

- Temperature in Celsius - Temperature of 30°C is higher than 20°C, and temperature of 20°C is higher than 10°C. The size of these intervals is the same.
- Annual Income in Dollars - Three people who make \$5,000, \$10,000 and \$15,000. The second person makes \$5,000 more than the first person and \$5,000 less than the third person, and the size of these intervals is the same.

Ratio

- It is interval data with a natural zero point.
- When the variable equals 0.0, there is none of that variable.

Examples :

- Height
- Weight
- Temperature in Kelvin - It is a ratio variable, as 0.0 Kelvin really does mean 'no temperature.'
-

Central Location

(a) Arithmetic Mean

Of these “averages,” the most common and familiar is the arithmetic mean, defined by

$$\mu = \frac{1}{N} \sum_1^N x$$

(b) Geometric Mean

The geometric mean is defined as the nth root of the product of n observations:

$$G = \sqrt[n]{x_1 x_2 \cdots x_n},$$

Geometric Mean

- For example, if a strain of bacteria increases its population by 20% in the first hour, 30% in the next hour and 50% in the next hour, we can find out an estimate of the mean percentage growth in population using Geometric mean.
- Geometric mean of 2,3, and 6?
- First, multiply the numbers together and then take the cubed root (because there are three numbers) = $(2*3*6)^{1/3} = 3.30$
- geometric mean of $1/2$, $1/4$, $1/5$, $9/72$ and $7/4$?
First, multiply the numbers together and then take the 5th root:
$$(1/2 * 1/4 * 1/5 * 9/72 * 7/4)^{(1/5)} = 0.35.$$

Central Location

(c) Harmonic Mean

The **harmonic mean** involves inverses—i.e., one divided by each of the quantities.

The harmonic mean is the inverse of the arithmetic mean of all the inverses.

$$\frac{1}{\frac{1}{x_1} + \frac{1}{x_2} + \dots}$$

Harmonic Mean

Using the weighted harmonic mean (correct):
price–earnings ratio (P/E),

$$P/E = \frac{0.3 + 0.7}{0.3/30 + 0.7/1000} \approx 93.46$$

Harmonic Mean

- You are a stock analyst in an investment bank. Your manager asked you to determine the P/E ratio of the index that tracks the stocks of Company A and Company B. Company A reports a market capitalization of \$1 billion and earnings of \$20 million while Company B reports a market capitalization of \$20 billion and earnings of \$5 billion. The index consists of 40% of Company A and 60% Company B.
- Firstly, we need to find the P/E ratios of each company. Remember that the P/E ratio is essentially the market capitalization divided by the earnings.

Harmonic Mean

- P/E (Company A) = (\$1 billion) / (\$20 million) = 50
P/E (Company B) = (\$20 billion) / (\$5 billion) = 4
- We must use the weighted harmonic mean to calculate the P/E ratio of the index. Using the formula for the weighted harmonic mean, the P/E ratio of the index can be found in the following way:
- P/E (Index) = (0.4+0.6) / (0.4/50 + 0.6/4) = 6.33
- Note that if we calculate the P/E ratio of the index using the weighted arithmetic mean, it would be significantly overstated:
- P/E (Index) = 0.4×50 + 0.6×4 = 22.4

Median

- If all the items with which we are concerned are sorted in order of increasing magnitude (size), from the smallest to the largest, then the median is the middle item.
- Consider the five items: 12, 13, 21, 27, 31. Then 21 is the median.
- If the number of items is even, the median is given by the arithmetic mean of the two middle items. Consider the six items: 12, 13, 21, 27, 31, 33.
- The median is $(21 + 27) / 2 = 24$.

Mode

- If the frequency varies from one item to another, the mode is the value which appears most frequently.
- In the case of continuous variables the frequency depends upon how many digits are quoted, so the mode is more usefully considered as the midpoint of the class with the largest frequency.

Grouped Mean, Median and Mode

- You grew fifty baby carrots using special soil. You dig them up and measure their lengths (to the nearest mm) and group the results:

Length (mm)	Frequency
150 - 154	5
155 - 159	2
160 - 164	6
165 - 169	8
170 - 174	9
175 - 179	11
180 - 184	6
185 - 189	3

Length (mm)	Midpoint x	Frequency f	fx
150 - 154	152	5	760
155 - 159	157	2	314
160 - 164	162	6	972
165 - 169	167	8	1336
170 - 174	172	9	1548
175 - 179	177	11	1947
180 - 184	182	6	1092
185 - 189	187	3	561
	Totals:	50	8530

**Estimated
Mean = $8530 \div 50 = 170.6$ mm**

Grouped Mean, Median and Mode

- Median

The Median is the mean of the 25th and the 26th length, so is in the **170 - 174** group:

L = 169.5 (the lower class boundary of the 170 - 174 group) $\rightarrow (167+172)/2$

n = 50

B = 5 + 2 + 6 + 8 = 21

G = 9 \rightarrow frequency

w = 5 \rightarrow Group width(150-154)

Estimated Median= $169.5 + (50/2) - 219 \times 5$

$$= 169.5 + 2.22\dots$$

$$= \mathbf{171.7 \text{ mm}} \text{ (to 1 decimal)}$$

Grouped Mean, Median and Mode

- Mode

The Modal group is the one with the highest frequency, which is **175 - 179**:

$L = 174.5$ (the lower class boundary of the 175 - 179 group)

$$f_{m-1} = 9$$

$$f_m = 11$$

$$f_{m+1} = 6$$

$$w = 5$$

$$\text{Estimated Mode} = 174.5 + \frac{11 - 9}{(11 - 9) + (11 - 6)} \times 5$$

$$= 174.5 + 1.42\dots$$

$$= \mathbf{175.9 \text{ mm}} \text{ (to 1 decimal)}$$

Variability or Spread of the Data

- **Mean Deviation from the Mean**

The mean deviation from the mean, defined as –

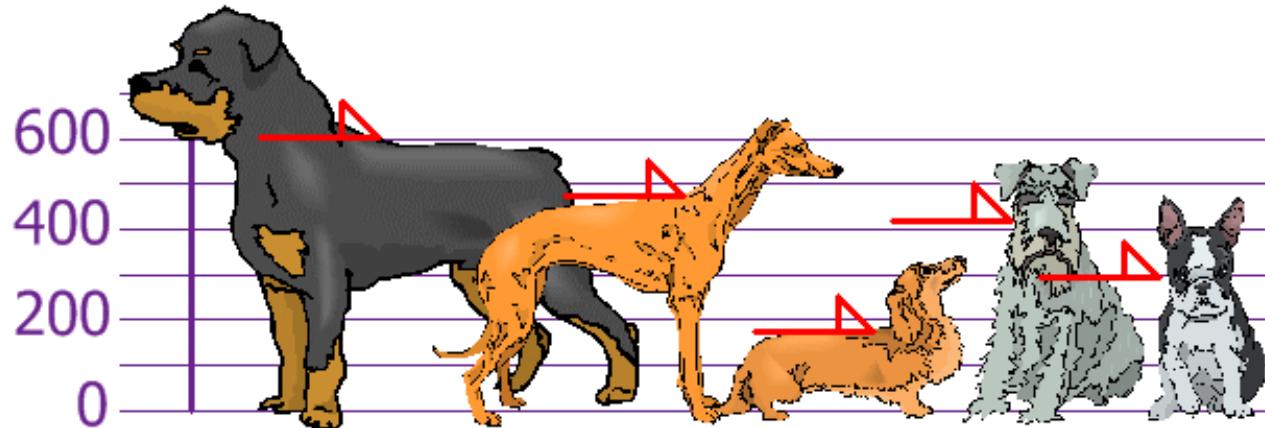
$$\sum_{i=1}^N (x_i - \bar{x}) / N$$

- **Mean Absolute Deviation from the Mean**

$$\sum_{i=1}^N |x_i - \bar{x}| / N$$

Mean Absolute Deviation

- Example: You and your friends have just measured the heights of your dogs (in millimeters):



The heights (at the shoulders) are: 600mm, 470mm, 170mm, 430mm and 300mm.

Mean Absolute Deviation

- Step 1: Find the **mean**:
- $\mu = 600 + 470 + 170 + 430 + 3005 = 19705 = 394$
- Step 2: Find the **Absolute Deviations**:

x	x - μ
600	206
470	76
170	224
430	36
300	94
	$\Sigma x - \mu = 636$

Step 3. Find the **Mean Deviation**:

$$\text{Mean Deviation} = \Sigma|x - \mu| / N = 636/5 = 127.2$$

So, on average, the dogs' heights are **127.2 mm from the mean**.

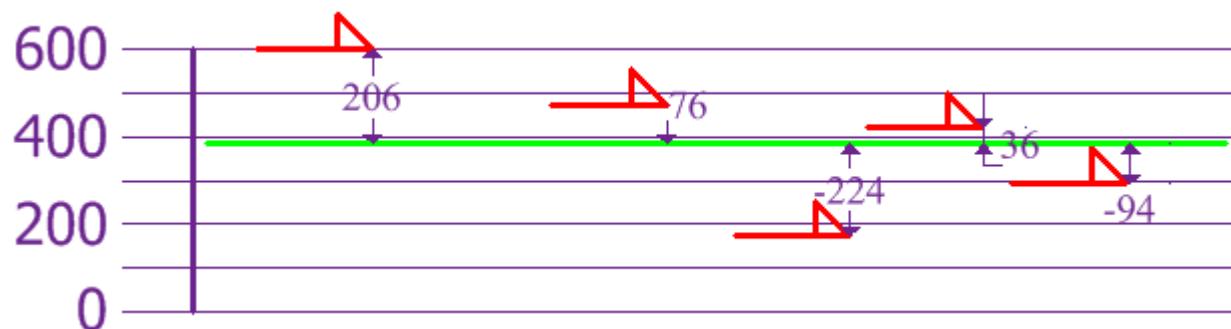
Variability or Spread of the Data

- **Variance**
- Since squares of both positive and negative real numbers are always positive, the variance is always positive.

$$\sigma^2 = \frac{\sum_{i=1}^N (x_i - \mu)^2}{N}$$

Variability or Spread of the Data

- **Variance**
- To calculate the Variance, take each difference, square it, and then average the result:



Variance
 σ^2

$$\begin{aligned} &= (206^2 + 76^2 + (-224)^2 + \\ &\quad 36^2 + (-94)^2) / 5 \\ &= (42436 + 5776 + \\ &\quad 50176 + 1296 + \\ &\quad 8836) / 5 \\ &= 108520 / 5 \\ &= 21704 \end{aligned}$$

So the Variance is **21,704**

Variability or Spread of the Data

- Standard Deviation

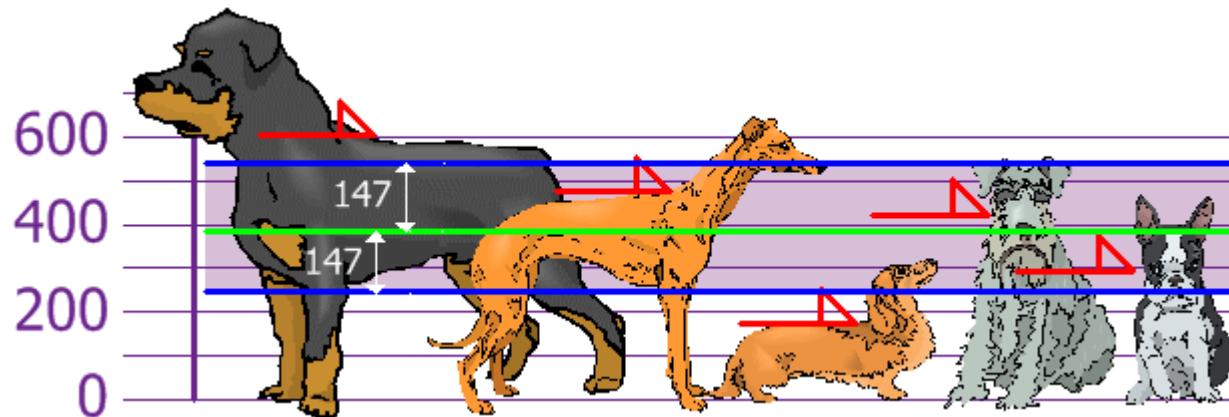
$$\sigma = \sqrt{\frac{\sum_{i=1}^N (x_i - \mu)^2}{N}}$$

Variability or Spread of the Data

- Standard Deviation
- And the Standard Deviation is just the square root of Variance, so:

Standard Deviation

$$\begin{aligned}\sigma &= \sqrt{21704} \\ &= 147.32... \\ &= 147 \text{ (to the nearest mm)}\end{aligned}$$



Multi Dimensional Array

- Create a multidimensional array as follows:
- In: `m = array([arange(2), arange(2)])`
- In: `m`
- Out:
- `array([[0, 1],
[0, 1]])`
- Show the array shape as follows:

- In: `m.shape`
- Out: `(2, 2)`

NumPy numerical types

Type	Description
bool	Boolean (True or False) stored as a bit
inti	Platform integer (normally either int32 or int64)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (- 2^{31} to $2^{31}-1$)
int64	Integer (- 2^{63} to $2^{63}-1$)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to $2^{32}-1$)
uint64	Unsigned integer (0 to $2^{64}-1$)
float16	Half precision float: sign bit, 5 bits exponent, and 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, and 23 bits mantissa

NumPy numerical types

float64 or float	Double precision float: sign bit, 11 bits exponent, and 52 bits mantissa
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128 or complex	Complex number, represented by two 64-bit floats (real and imaginary components)

NumPy numerical types

Type	Character code
integer	i
Unsigned integer	u
Single precision float	f
Double precision float	d
bool	b
complex	D
string	S
unicode	U
Void	V

Manipulating array shapes

- Flattening is transforming a multidimensional array into a one-dimensional array.

Quartiles, Deciles, Percentiles, and Quantiles

- Quartiles, deciles, and percentiles divide a frequency distribution into a number of parts containing equal frequencies.
- **Quartiles** divide the range of values into four parts, each containing one quarter of the values.
- **Percentiles** divide into a hundred parts, each containing one hundredth of the total frequency.
- **Quantile** divides a frequency distribution into parts containing stated proportions of a distribution.

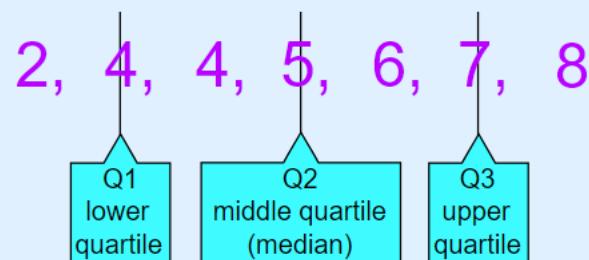
Quartiles

- Quartiles are the values that divide a list of numbers into quarters:
- Put the list of numbers **in order**
- Then cut the list into **four equal parts**
- The Quartiles are at the "cuts"

Example: 5, 7, 4, 4, 6, 2, 8

Put them in order: 2, 4, 4, 5, 6, 7, 8

Cut the list into quarters:



And the result is:

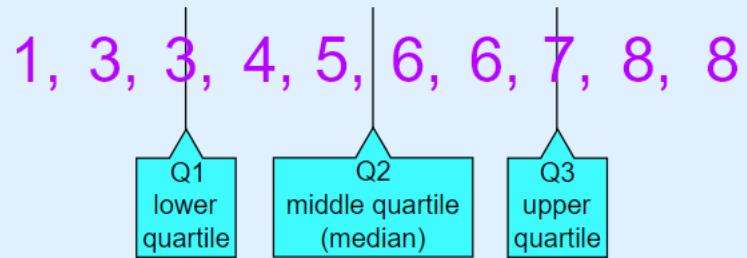
- Quartile 1 (Q1) = 4
- Quartile 2 (Q2), which is also the Median, = 5
- Quartile 3 (Q3) = 7

Quartiles

Example: 1, 3, 3, 4, 5, 6, 6, 7, 8, 8

The numbers are already in order

Cut the list into quarters:



In this case Quartile 2 is half way between 5 and 6:

$$Q2 = (5+6)/2 = 5.5$$

And the result is:

- Quartile 1 (Q1) = 3
- Quartile 2 (Q2) = 5.5
- Quartile 3 (Q3) = 7

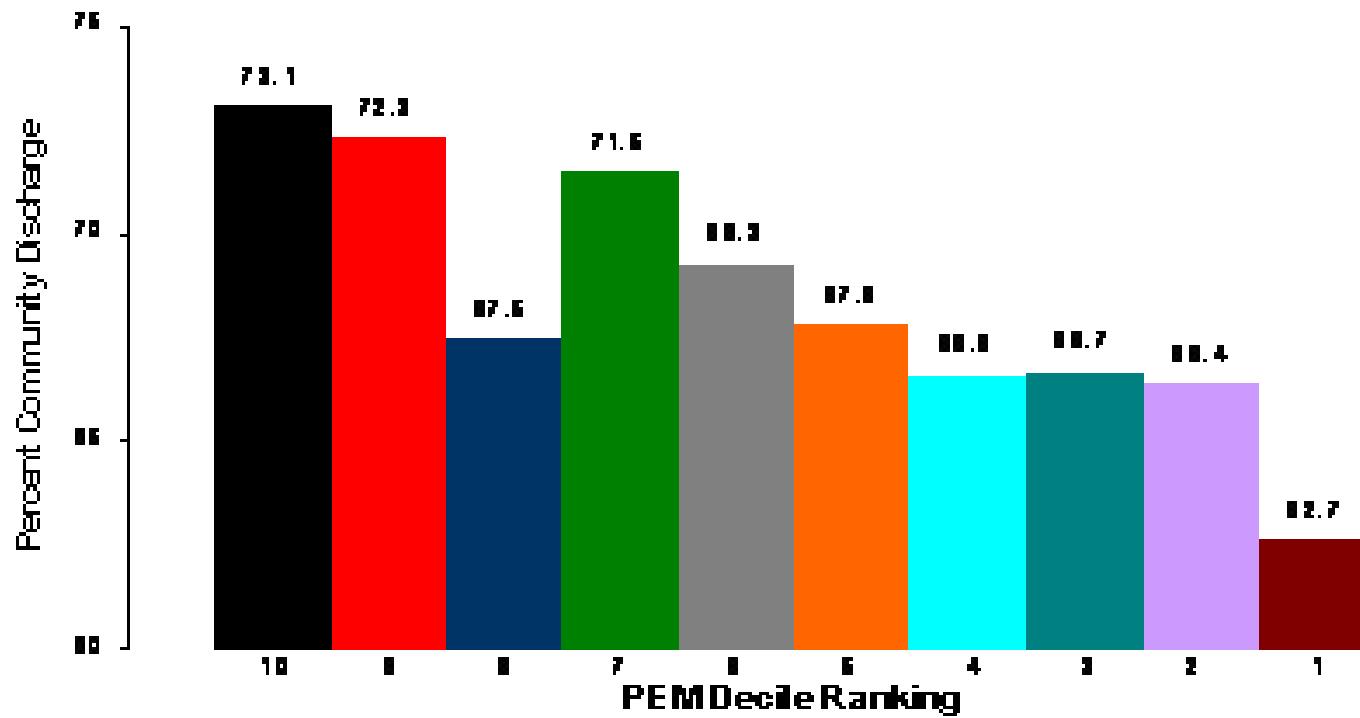
Decile

- What is a Decile used for in Real Life?
- Deciles and decile ranks are used more often in real life than in the classroom. For example, [Australia](#) uses decile ranks to report drought data. Deciles 1-2 represent the lowest 20% (“much below normal”). That means droughts that are “much below normal” don’t occur more than 20% of the time.
- Deciles are also commonly used for college admissions and high school rankings. For example, [this chart](#) from Roanoke College shows the high school decile rankings for the student body.

Decile

Decile Rank	Percentile
1	10th
2	20th
3	30th
4	40th
5	50th
6	60th
7	70th
8	80th
9	90th

Decile



Normal Distribution

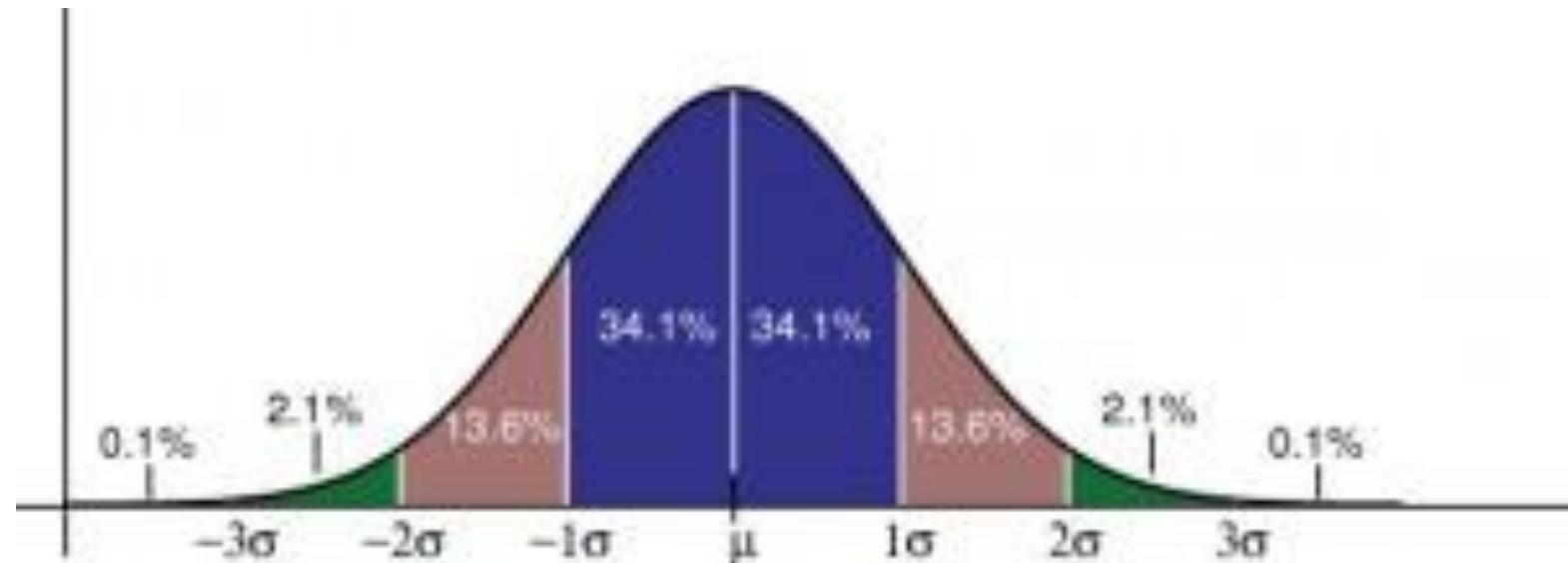
- A normal distribution, sometimes called the bell curve, is a distribution that occurs naturally in many situations.
- For example, the bell curve is seen in tests like the SAT and GRE.
- The bulk of students will score the average (C), while smaller numbers of students will score a B or D.
- An even smaller percentage of students score an F or an A.
- This creates a distribution that resembles a bell (hence the nickname).
- The bell curve is symmetrical. Half of the data will fall to the left of the mean; half will fall to the right.

Normal Distribution

- Many groups follow this type of pattern. That's why it's widely used in business, statistics and in government bodies like the [FDA](#):
- Heights of people.
- Measurement errors.
- Blood pressure.
- Points on a test.
- IQ scores.
- Salaries.

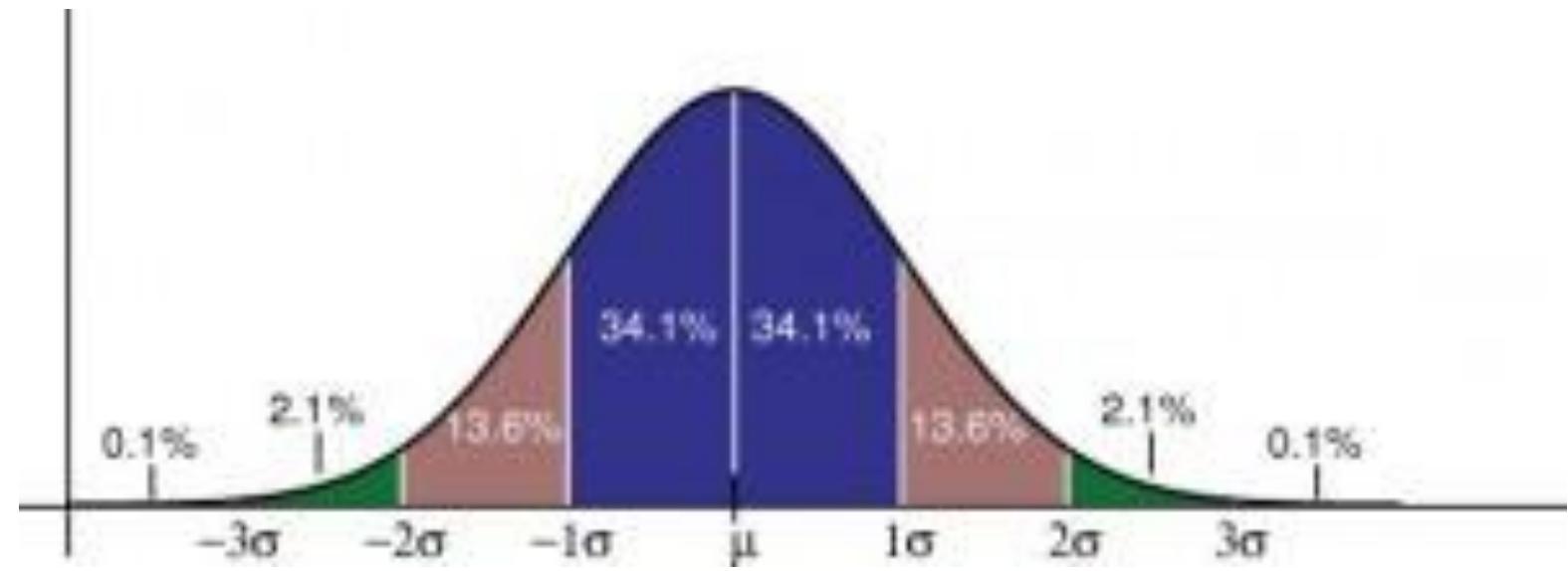
Normal Distribution

- The empirical rule tells you what percentage of your data falls within a certain number of standard deviations from the mean:
 - 68% of the data falls within one standard deviation of the mean.
 - 95% of the data falls within two standard deviations of the mean.
 - 99.7% of the data falls within three standard deviations of the mean.

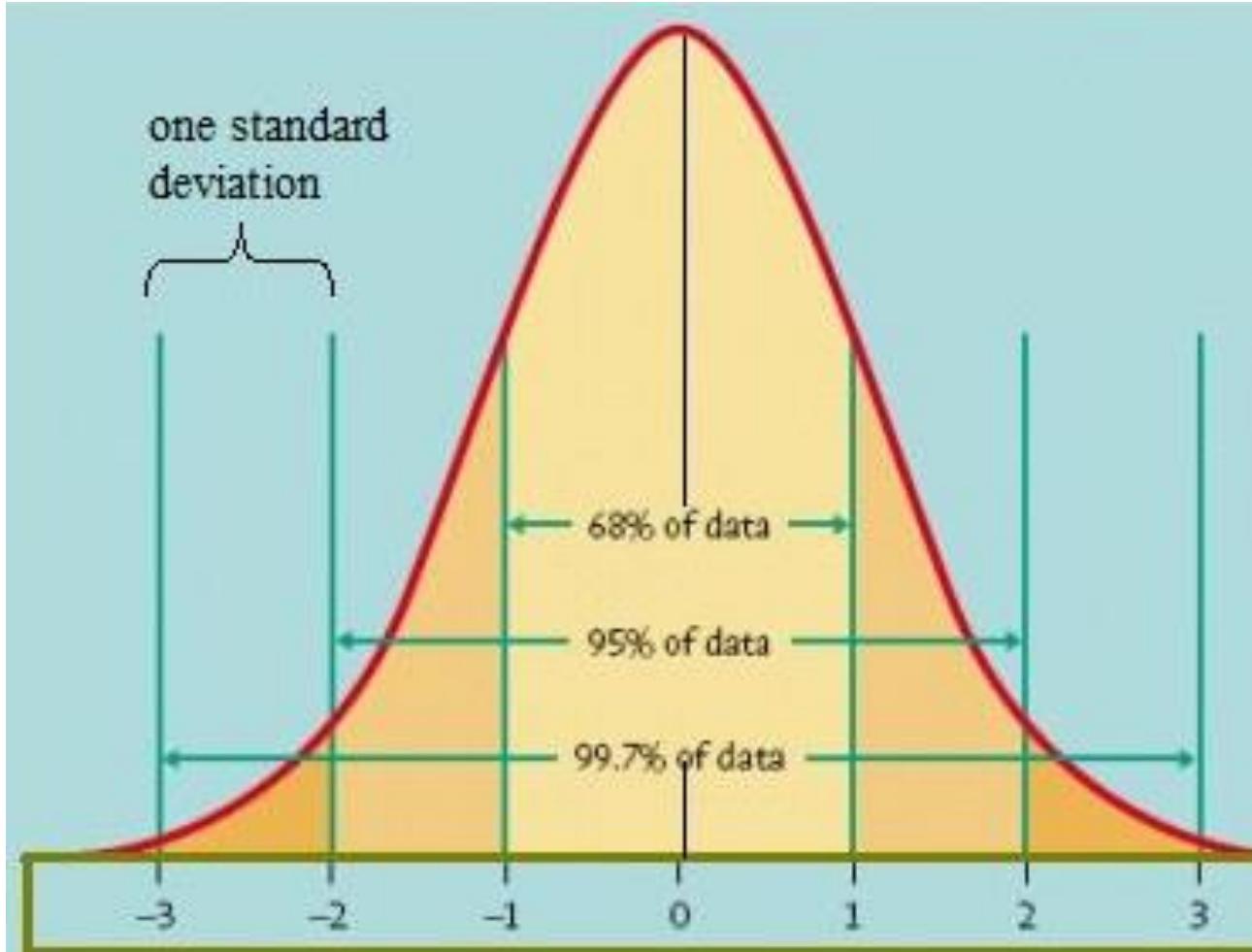


Normal Distribution

- The standard deviation controls the spread of the distribution.
- A smaller standard deviation indicates that the data is tightly clustered around the mean; the normal distribution will be taller.
- A larger standard deviation indicates that the data is spread out around the mean; the normal distribution will be flatter and wider.



Standard Normal Model



Normal Distribution

- Properties of a normal distribution
 - The mean, mode and median are all equal.
 - The curve is symmetric at the center (i.e. around the mean, μ).
 - Exactly half of the values are to the left of center and exactly half the values are to the right.
 - The total area under the curve is 1.

Word problems with normal distribution: “Between”: Steps

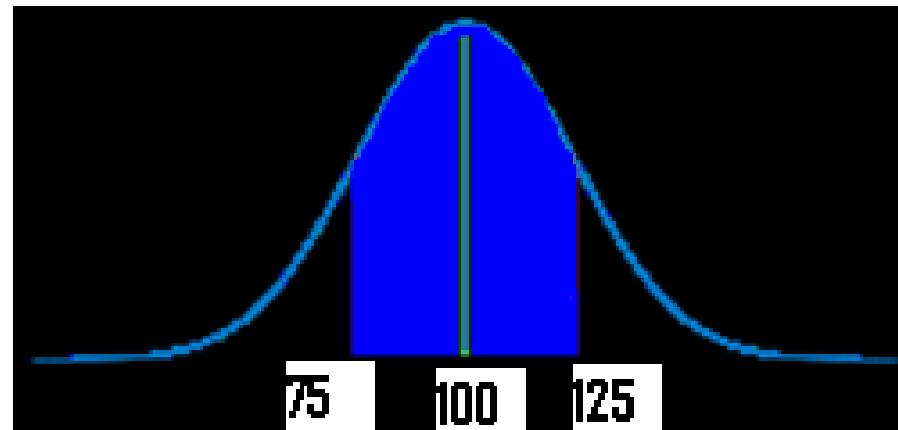
- **Step 1:** *Identify the parts of the word problem.* The word problem will identify:
 - The mean average or μ .
 - Standard deviation (σ).
 - Number selected (i.e. “choose one at random” or “select ten at random”).
 - X: the numbers associated with “between” (i.e. “between \$5,000 and \$10,000” would have X as 5,000 and as \$10,000).

Word problems with normal distribution: “Between”: Steps

- **Step 1:** *Identify the parts of the word problem.* The word problem will identify:
 - The mean average or μ .
 - Standard deviation (σ).
 - Number selected (i.e. “choose one at random” or “select ten at random”).
 - X: the numbers associated with “between” (i.e. “between \$5,000 and \$10,000” would have X as 5,000 and as \$10,000).

Word problems with normal distribution: “Between”: Steps

- **Step 2:** *Draw a graph.* Put the mean you identified in Step 1 in the center.
- Put the number associated with “between” on the graph (take a guess at where the numbers would fall—it doesn’t have to be exact).
- For example, if your mean was \$100, and you were asked for “hourly wages between \$75 and \$125”) your graph will look something like this:

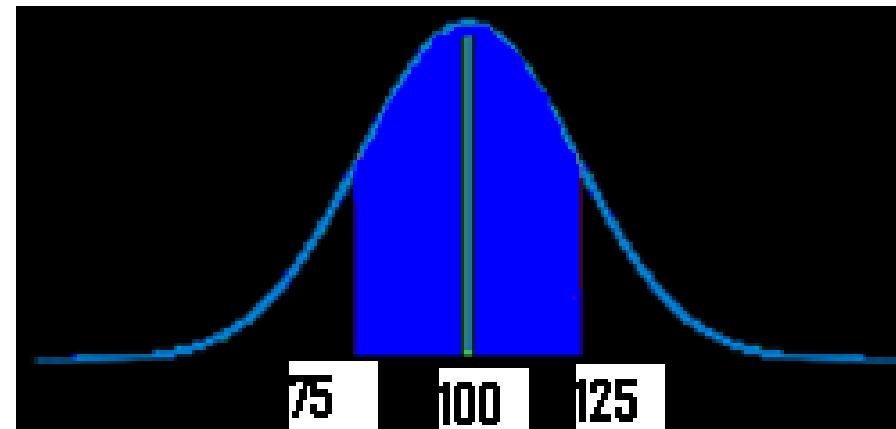


Word problems with normal distribution: “Between”: Steps

- Step 3:Figure out the z-scores.
- Plug the first X value (in my graph above, it's 75) into the z value formula and solve. The μ (the mean), is 100 from the sample graph. You can get these figures (including σ , the standard deviation) from your answers in step 1 :

- z score formula
$$z = \frac{X - \mu}{\sigma} =$$

- *Note: if the formula confuses you, all this formula is asking you to do is:
- subtract the mean from X
- divide by the standard deviation.



Word problems with normal distribution: “Between”: Steps

- **Step 4:** Repeat step 3 for the second X.
- **Step 5:** Take the numbers from step 3 and 4 and use them to find the area in the z-table.
- **Step 6a:**
 - Convert the answer from step 5 into a percentage.
 - For example, 0.1293 is 12.93%.
 - That's it—skip step 6b!
- **Step 6b**
 - Multiply the sample size (found in step 1) by the z-value you found in step 4. For example, $0.300 * 100 = 30$.
 - That's it!

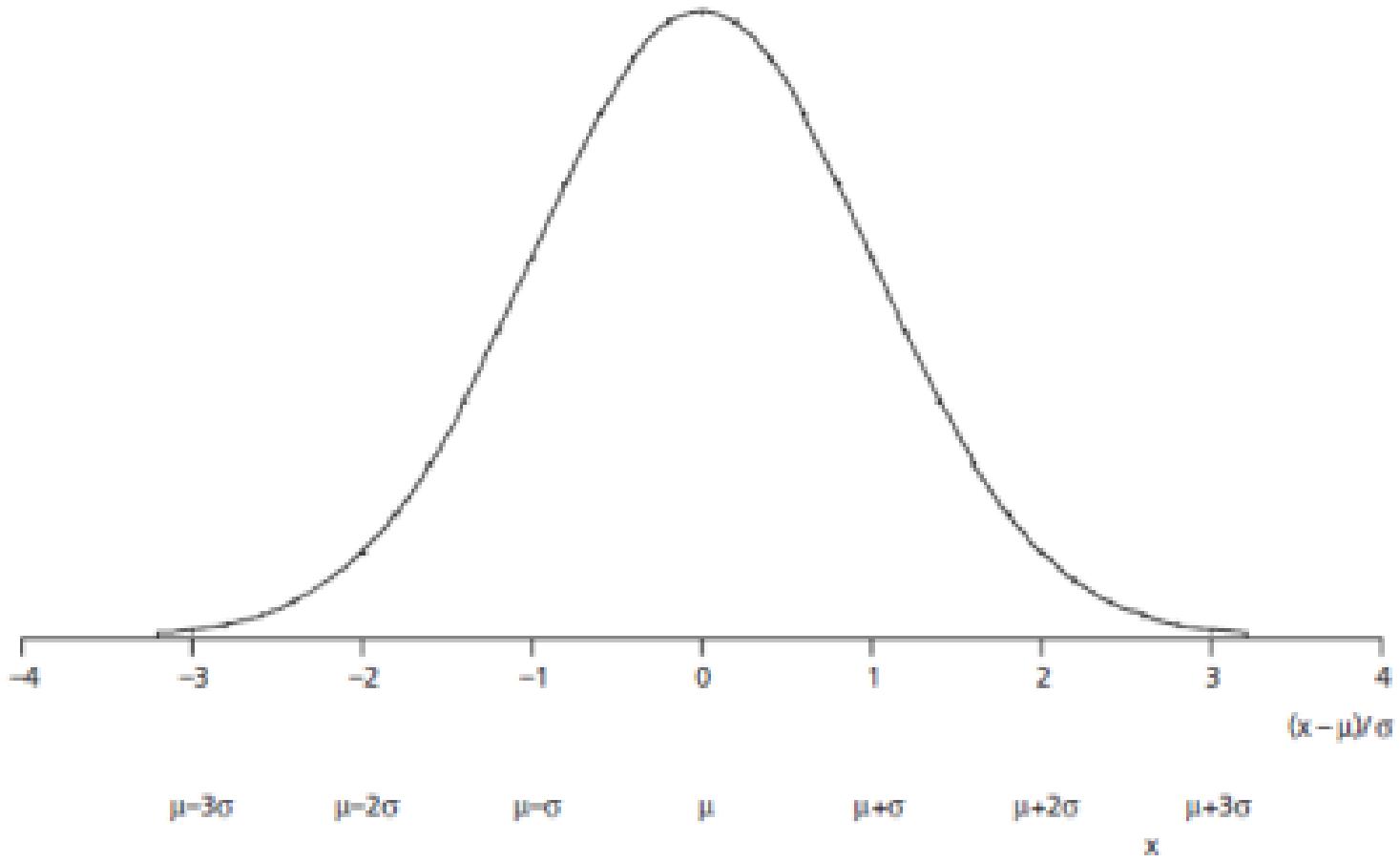
Normal Distribution

- They are approximately symmetrical, and the mode is close to the centre of the distribution.
- The mean, median, and mode are close together.
- The shape of the distribution can be approximated by a bell: nearly flat on top, then decreasing more quickly, then decreasing more slowly toward the tails of the distribution.
- This implies that values close to the mean are relatively frequent, and values farther from the mean tend to occur less frequently.

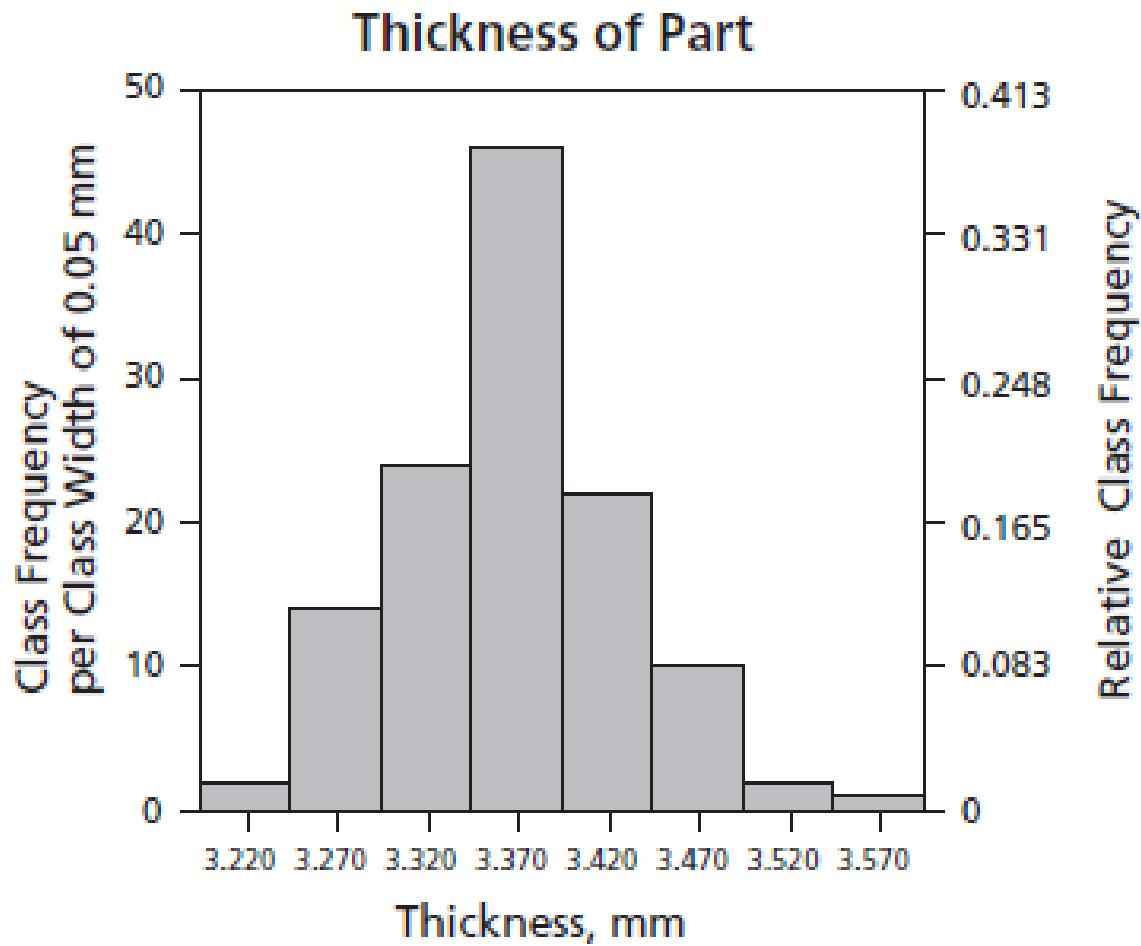
Probability Density Function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Normal Distribution



Normal Distribution



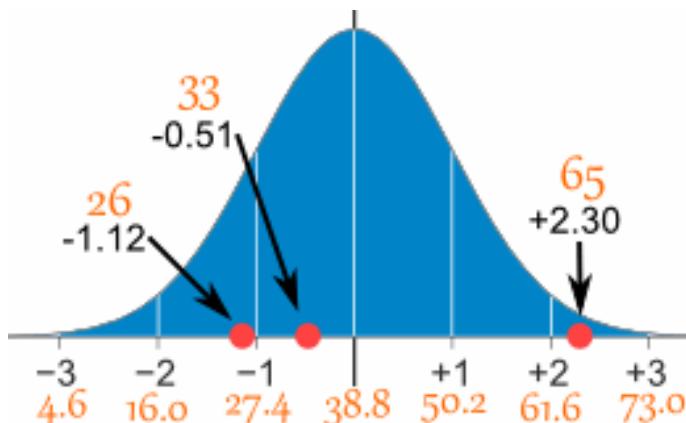
Normal Distribution

- A survey of daily travel time had these results (in minutes):
- 26, 33, 65, 28, 34, 55, 25, 44, 50, 36, 26, 37, 43, 62, 35, 38, 45, 32, 28, 34
- The **Mean is 38.8 minutes**, and the **Standard Deviation is 11.4 minutes**
- To convert **26**:
- first subtract the mean: $26 - 38.8 = -12.8$,
- then divide by the Standard Deviation: $-12.8/11.4 = -1.12$
- So **26 is -1.12 Standard Deviations from the Mean**

Normal Distribution

Here are the first three conversions

Original Value	Calculation	Standard Score (z-score)
26	$(26-38.8) / 11.4 =$	-1.12
33	$(33-38.8) / 11.4 =$	-0.51
65	$(65-38.8) / 11.4 =$	+2.30
...



Feature Scaling

- StandardScaler
- MinMaxScaler
- RobustScaler
- Normalizer

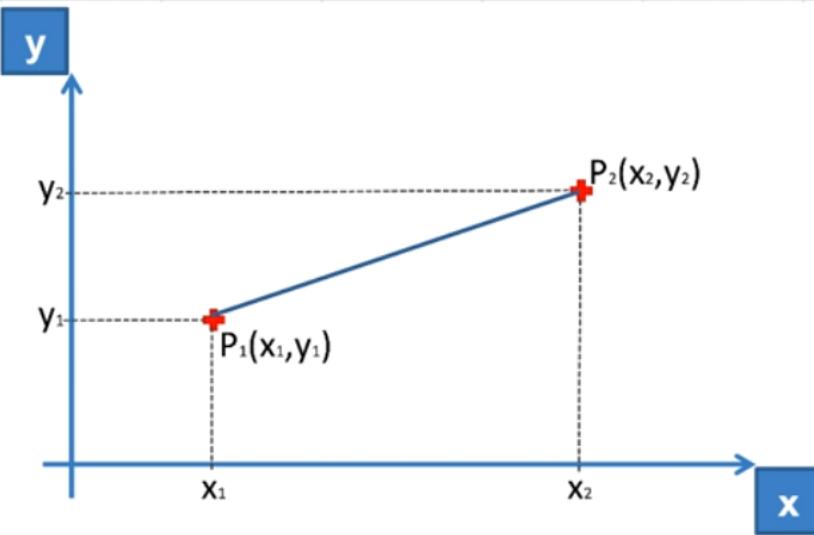
Standard Scaler

The `StandardScaler` assumes your data is normally distributed within each feature and will scale them such that the distribution is now centred around 0, with a standard deviation of 1.

The mean and standard deviation are calculated for the feature and then the feature is scaled based on:

$$\frac{x_i - \text{mean}(x)}{\text{stdev}(x)}$$

1	Country	Age	Salary	Purchased
2	France	44	72000	No
3	Spain	27	48000	Yes
4	Germany	30	54000	No
5	Spain	38	61000	No
6	Germany	40	63777.77778	Yes
7	France	35	58000	Yes
8	Spain	38.77777778	52000	No
9	France	48	79000	Yes
10	Germany	50	83000	No
11	France	37	67000	Yes
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Feature Scaling

Standardisation	Normalisation
$x_{\text{stand}} = \frac{x - \text{mean}(x)}{\text{standard deviation } (x)}$	$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$

Feature Scaling

- StandardScaler
- MinMaxScaler
- RobustScaler
- Normalizer

Min-Max Scaler

The `MinMaxScaler` is probably the most famous scaling algorithm, and follows the following formula for each feature:

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

It essentially shrinks the range such that the range is now between 0 and 1 (or -1 to 1 if there are negative values).

Feature Scaling

Robust Scaler

The `RobustScaler` uses a similar method to the Min-Max scaler but it instead uses the interquartile range, rather than the min-max, so that it is robust to outliers. Therefore it follows the formula:

$$\frac{x_i - Q_1(x)}{Q_3(x) - Q_1(x)}$$

For each feature.

Feature Scaling

Normalizer

The normalizer scales each value by dividing each value by its magnitude in n -dimensional space for n number of features.

Say your features were x, y and z Cartesian co-ordinates your scaled value for x would be:

$$\frac{x_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}$$

Each point is now within 1 unit of the origin on this Cartesian co-ordinate system.

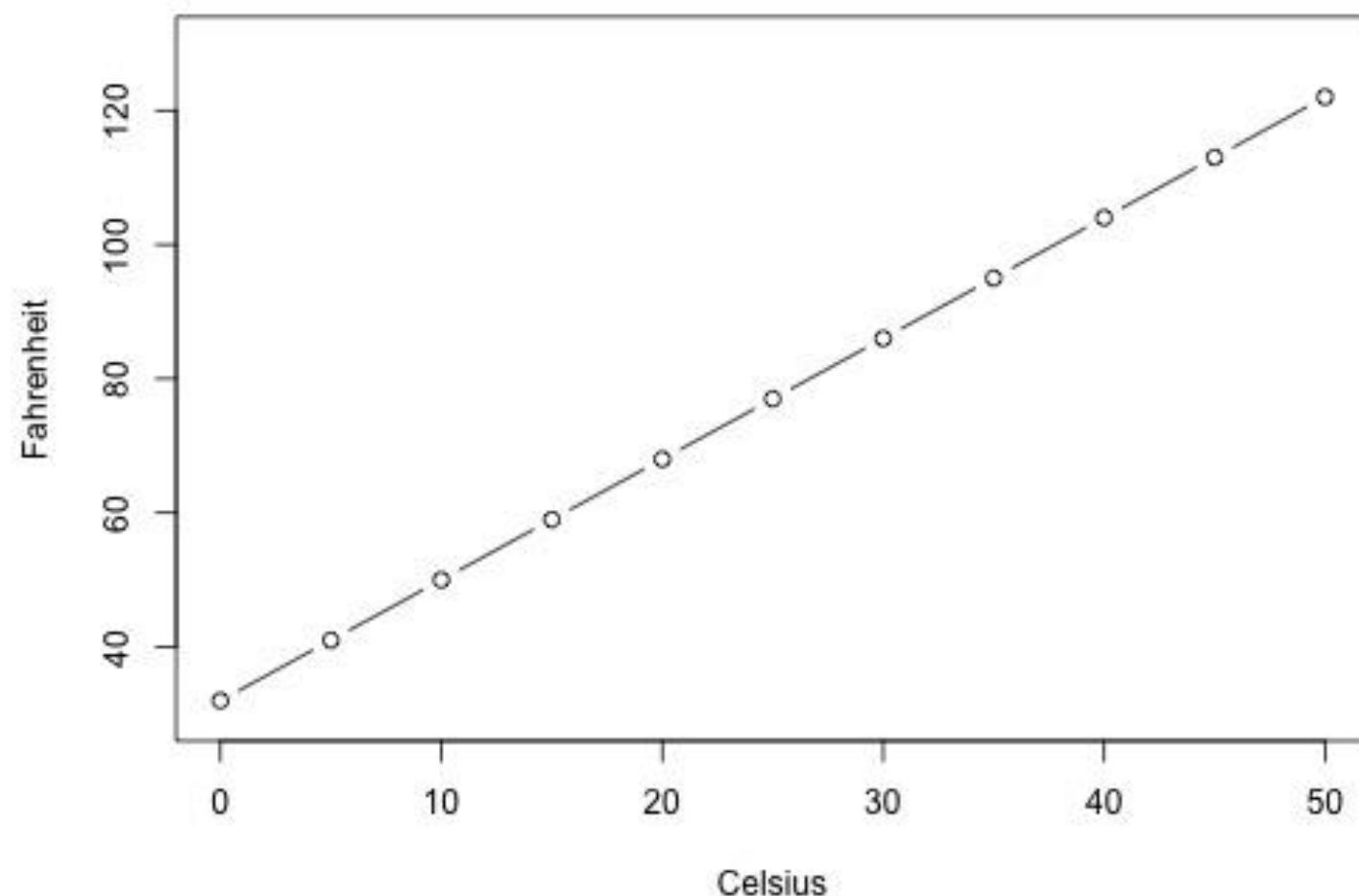
Probability

- Many events can't be predicted with total certainty. The best we can say is how likely they are to happen, using the idea of probability.

Simple Linear Regression

- **Simple linear regression** is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables:
- One variable, denoted x , is regarded as the **predictor, explanatory**, or **independent** variable.
- The other variable, denoted y , is regarded as the **response, outcome**, or **dependent** variable.

Simple Linear Regression



Types of Relationships

- Deterministic
 - Temperature conversion
 - Circumference = $\pi \times$ diameter
 - Hooke's Law: $Y = \alpha + \beta X$, where Y = amount of stretch in a spring, and X = applied weight.
 - Ohm's Law: $I = V/r$, where V = voltage applied, r = resistance, and I = current.
 - Boyle's Law: For a constant temperature, $P = \alpha/V$, where P = pressure, α = constant for each gas, and V = volume of gas.

Types of Relationships

- Statistical
- Height and weight — as height increases, you'd expect weight to increase, but not perfectly.
- Alcohol consumed and blood alcohol content — as alcohol consumption increases, you'd expect one's blood alcohol content to increase, but not perfectly.
- Vital lung capacity and pack-years of smoking — as amount of smoking increases (as quantified by the number of pack-years of smoking), you'd expect lung function (as quantified by vital lung capacity) to decrease, but not perfectly.
- Driving speed and gas mileage — as driving speed increases, you'd expect gas mileage to decrease, but not perfectly.

Real Time Example

- We have a dataset which contains information about relationship between ‘number of hours studied’ and ‘marks obtained’.
- Many students have been observed and their hours of study and grade are recorded.
- This will be our training data. Goal is to design a model that can predict marks if given the number of hours studied.
- Using the training data, a regression line is obtained which will give minimum error.
- This linear equation is then used for any new data.
- That is, if we give number of hours studied by a student as an input, our model should predict their mark with minimum error.

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Constant Coefficient

Dependent variable (DV) Independent variable (IV)

The diagram illustrates the components of a simple linear regression equation. The equation is $y = b_0 + b_1 * x_1$. A green arrow points from the label "Constant" to the term b_0 . Another green arrow points from the label "Coefficient" to the term b_1 . A third green arrow points from the label "Dependent variable (DV)" to the term y . A fourth green arrow points from the label "Independent variable (IV)" to the term x_1 .

Simple Linear Regression

- The values b_0 and b_1 must be chosen so that they minimize the error.
- If sum of squared error is taken as a metric to evaluate the model, then goal to obtain a line that best reduces the error.

$$\text{Error} = \sum_{i=1}^n (\text{actual_output} - \text{predicted_output}) ^\star 2$$

If we don't square the error, then positive and negative point will cancel out each other.

Simple Linear Regression

- For model with one predictor,

$$b_0 = \bar{y} - b_1 \bar{x}$$

$$b_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Simple Linear Regression

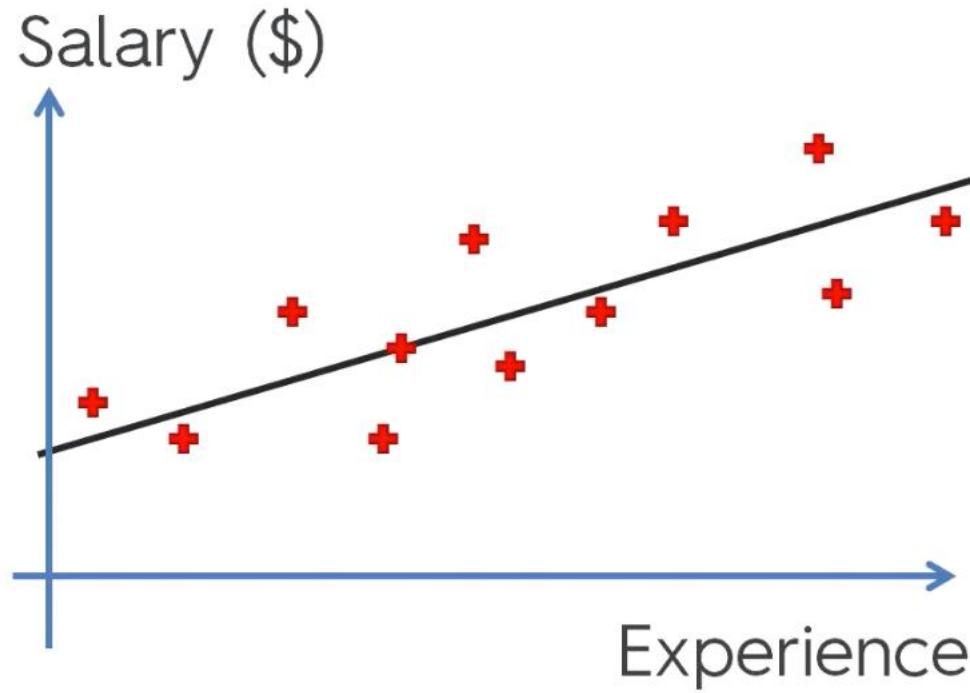
- *Exploring 'b1'*
- If $b1 > 0$, then x(predictor) and y(target) have a positive relationship. That is increase in x will increase y.
- If $b1 < 0$, then x(predictor) and y(target) have a negative relationship. That is increase in x will decrease y.

Simple Linear Regression

- ***Exploring ‘b0’***
- If the model does not include $x=0$, then the prediction will become meaningless with only b_0 . For example, we have a dataset that relates height(x) and weight(y). Taking $x=0$ (that is height as 0), will make equation have only b_0 value which is completely meaningless as in real-time height and weight can never be zero. This resulted due to considering the model values beyond its scope.
- If the model includes value 0, then ‘ b_0 ’ will be the average of all predicted values when $x=0$. But, setting zero for all the predictor variables is often impossible.
- The value of b_0 guarantee that residual have mean zero. If there is no ‘ b_0 ’ term, then regression will be forced to pass over the origin. Both the regression co-efficient and prediction will be biased.

Regressions

Simple Linear Regression:



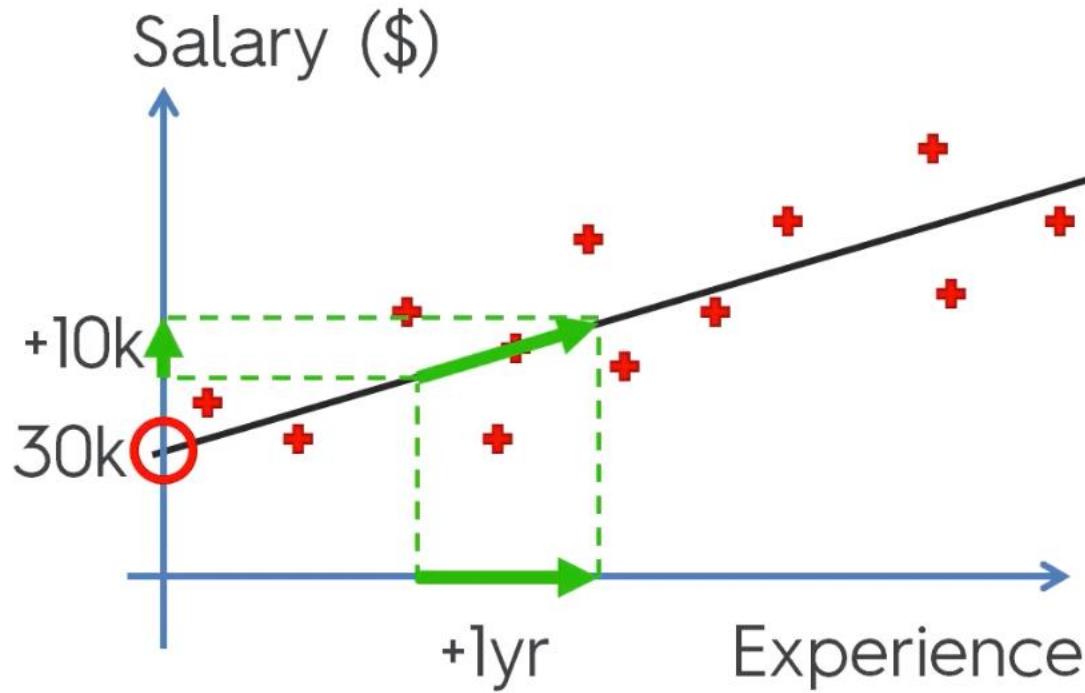
$$y = b_0 + b_1 * x$$



$$\text{Salary} = b_0 + b_1 * \text{Experience}$$

Regressions

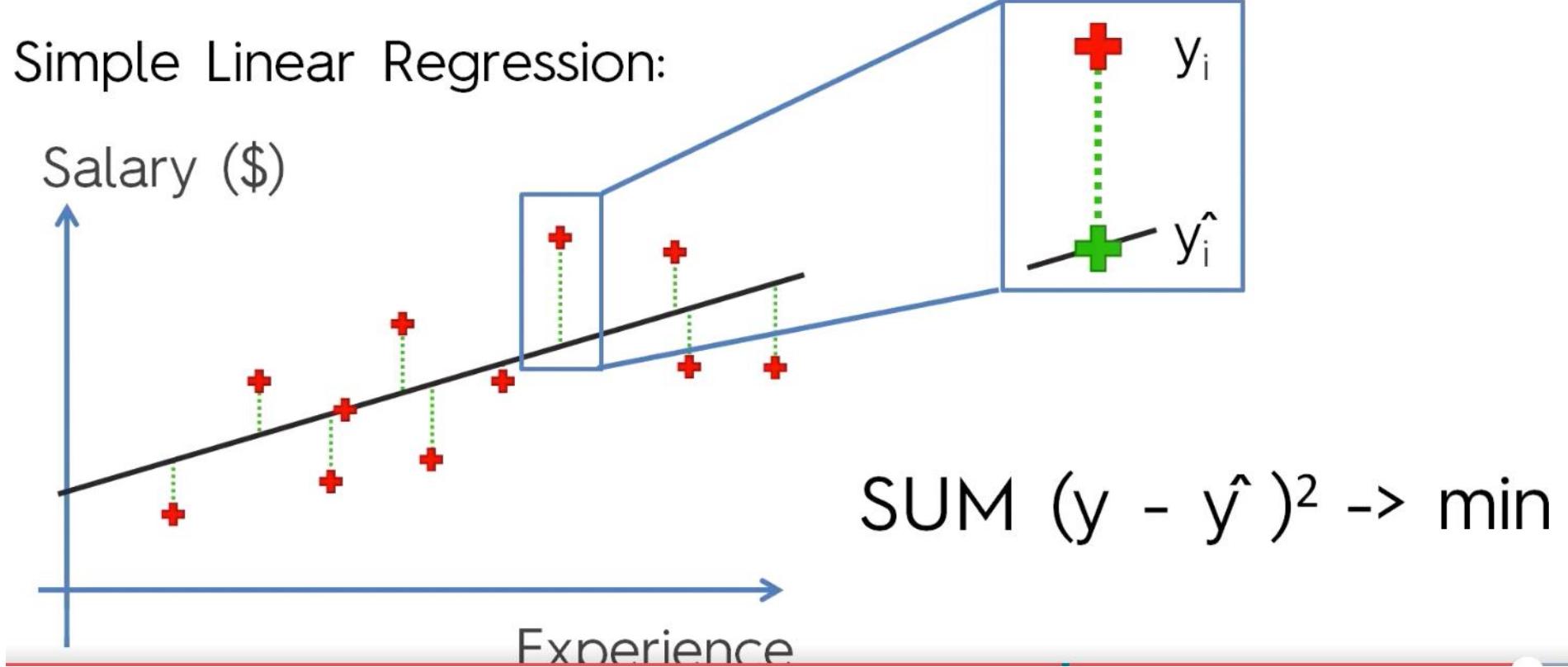
Simple Linear Regression:



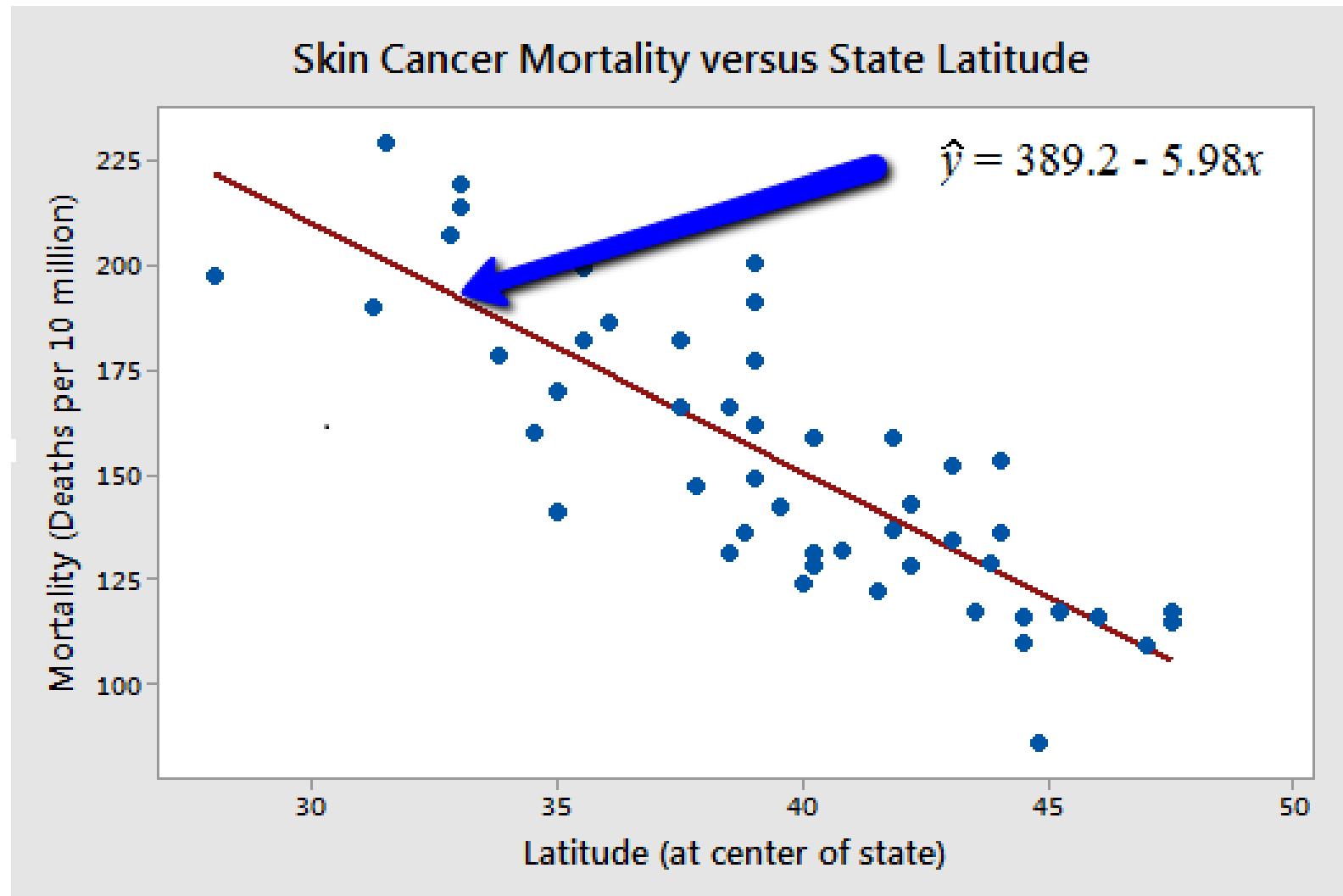
$$y = b_0 + b_1 * x$$

↓
Salary = b₀ + b₁ * Experience

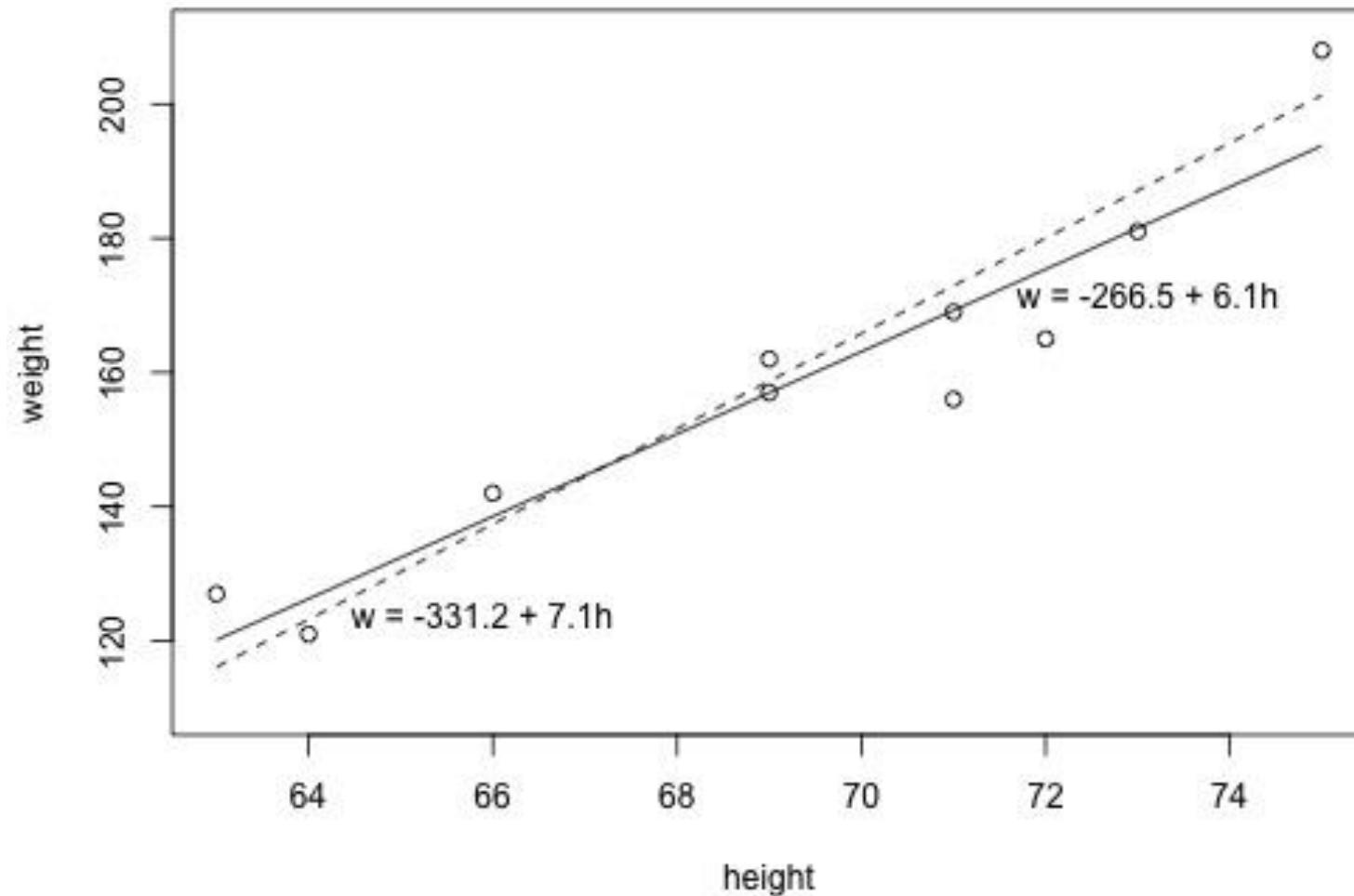
Ordinary Least Square Method



Types of Relationships



What is the "Best Fitting Line"?



What is the "Best Fitting Line"?

- y_i denotes the observed response for experimental unit i
- x_i denotes the predictor value for experimental unit i
- \hat{y}_i is the predicted response (or fitted value) for experimental unit i

Then, the equation for the best fitting line is:

i	x_i	y_i	\hat{y}_i
1	63	127	120.1
2	64	121	126.3
3	66	142	138.5
4	69	157	157.0
5	69	162	157.0
6	71	156	169.2
7	71	169	169.2
8	72	165	175.4
9	73	181	181.5
10	75	208	193.8

$$\hat{y}_i = b_0 + b_1 x_i$$

it has some "**prediction error**" (or "**residual error**").

In fact, the size of its prediction error is $127 - 120.1$ or 6.9 pounds.

As you can see, the size of the prediction error depends on the data point.

If we didn't know the weight of student 5, the equation of the line would predict his or her weight to be $-266.53 + 6.1376(69)$ or 157 pounds.

The size of the prediction error here is $162 - 157$, or 5 pounds.

What is the "Best Fitting Line"?

In general, when we use $\hat{y}_i = b_0 + b_1 x_i$ to predict the actual response y_i , we make a prediction error (or residual error) of size:

$$e_i = y_i - \hat{y}_i$$

A line that fits the data "**best**" will be one for which the ***n* prediction errors** — one for each observed data point — **are as small as possible in some overall sense**. One way to achieve this goal is to invoke the "**least squares criterion**," which says to "minimize the sum of the squared prediction errors." That is:

- The equation of the best fitting line is: $\hat{y}_i = b_0 + b_1 x_i$
- We just need to find the values b_0 and b_1 that make the sum of the squared prediction errors the smallest it can be.
- That is, we need to find the values b_0 and b_1 that minimize:

$$Q = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Here's how you might think about this quantity Q :

- The quantity $e_i = y_i - \hat{y}_i$ is the prediction error for data point i .
- The quantity $e_i^2 = (y_i - \hat{y}_i)^2$ is the squared prediction error for data point i .
- And, the symbol $\sum_{i=1}^n$ tells us to add up the squared prediction errors for all n data points.

What is the "Best Fitting Line"?

$w = -331.2 + 7.1 h$ (the dashed line)					
<i>i</i>	<i>x_i</i>	<i>y_i</i>	\hat{y}_i	$(y_i - \hat{y}_i)$	$(y_i - \hat{y}_i)^2$
1	63	127	116.1	10.9	118.81
2	64	121	123.2	-2.2	4.84
3	66	142	137.4	4.6	21.16
4	69	157	158.7	-1.7	2.89
5	69	162	158.7	3.3	10.89
6	71	156	172.9	-16.9	285.61
7	71	169	172.9	-3.9	15.21
8	72	165	180.0	-15.0	225.00
9	73	181	187.1	-6.1	37.21
10	75	208	201.3	6.7	44.89
					<u>766.5</u>

$w = -266.53 + 6.1376 h$ (the solid line)					
<i>i</i>	<i>x_i</i>	<i>y_i</i>	\hat{y}_i	$(y_i - \hat{y}_i)$	$(y_i - \hat{y}_i)^2$
1	63	127	120.139	6.8612	47.076
2	64	121	126.276	-5.2764	27.840
3	66	142	138.552	3.4484	11.891
4	69	157	156.964	0.0356	0.001
5	69	162	156.964	5.0356	25.357
6	71	156	169.240	-13.2396	175.287
7	71	169	169.240	-0.2396	0.057
8	72	165	175.377	-10.3772	107.686
9	73	181	181.515	-0.5148	0.265
10	75	208	193.790	14.2100	201.924
					<u>597.4</u>

What is the "Best Fitting Line"?

- Based on the least squares criterion, which equation best summarizes the data?
- The sum of the squared prediction errors is 766.5 for the dashed line, while it is only 597.4 for the solid line.
- Therefore, of the two lines, the solid line, $w = -266.53 + 6.1376h$, best summarizes the data.

What is the "Best Fitting Line"?

If we used the above approach for finding the equation of the line that minimizes the sum of the squared prediction errors, we'd have our work cut out for us. We'd have to implement the above procedure for an infinite number of possible lines — clearly, an impossible task! Fortunately, somebody has done some dirty work for us by figuring out formulas for the **intercept** b_0 and the **slope** b_1 for the equation of the line that minimizes the sum of the squared prediction errors.

The formulas are determined using methods of calculus. We minimize the equation for the sum of the squared prediction errors:

$$Q = \sum_{i=1}^n (y_i - (b_0 + b_1 x_i))^2$$

(that is, take the derivative with respect to b_0 and b_1 , set to 0, and solve for b_0 and b_1) and get the "**least squares estimates**" for b_0 and b_1 :

$$b_0 = \bar{y} - b_1 \bar{x}$$

and:

$$b_1 = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Because the formulas for b_0 and b_1 are derived using the least squares criterion, the resulting equation — $\hat{y}_i = b_0 + b_1 x_i$ — is often referred to as the "**least squares regression line**," or simply the "**least squares line**." It is also sometimes called the "**estimated regression equation**." Incidentally, note that in deriving the above formulas, we made no assumptions about the data other than that they follow some sort of linear trend.

We can see from these formulas that the least squares line passes through the point (\bar{x}, \bar{y}) , since when $x = \bar{x}$, then $y = b_0 + b_1 \bar{x} = \bar{y} - b_1 \bar{x} + b_1 \bar{x} = \bar{y}$.

In practice, you won't really need to worry about the formulas for b_0 and b_1 . Instead, you are going to let statistical software, such as Minitab, find least squares lines for you. But, we can still learn something from the formulas — for b_1 in particular.

Slope and Intercept calculation

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

How to Find a Linear Regression Equation: Steps

Step 1: Make a chart of your data, filling in the columns in the same way as you would fill in the chart if you were finding the Pearson's Correlation Coefficient.

SUBJECT	AGE X	GLUCOSE LEVEL Y	XY	X ²	Y ²
1	43	99	4257	1849	9801
2	21	65	1365	441	4225
3	25	79	1975	625	6241
4	42	75	3150	1764	5625
5	57	87	4959	3249	7569
6	59	81	4779	3481	6561
Σ	247	486	20485	11409	40022

From the above table, $\Sigma x = 247$, $\Sigma y = 486$, $\Sigma xy = 20485$, $\Sigma x^2 = 11409$, $\Sigma y^2 = 40022$. n is the sample size (6, in our case).

How to Find a Linear Regression Equation: Steps

- **Step 2:** Use the following equations to find a and b.

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

$$\begin{aligned} a &= 65.1416 \\ b &= .385225 \end{aligned}$$

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

How to Find a Linear Regression Equation: Steps

- **Step 2:** Use the following equations to find a and b.

Find a:

$$((486 \times 11,409) - ((247 \times 20,485)) / 6 (11,409) - 247^2)$$

$$484979 / 7445$$

$$= 65.14$$

Find b:

$$(6(20,485) - (247 \times 486)) / (6 (11409) - 247^2)$$

$$(122,910 - 120,042) / 68,454 - 247^2$$

$$2,868 / 7,445$$

$$= .385225$$

How to Find a Linear Regression Equation: Steps

- **Step 3:** *Insert the values into the equation.*

$$y' = a + bx$$

$$y' = 65.14 + .385225x$$

- *That's how to find a linear regression equation by hand!*

Interpolation vs Regression

- In the mathematical field of numerical analysis, **interpolation** is a method of constructing new data points within the range of a discrete set of known data points.
- Interpolation is the process of deriving a simple function from a set of discrete data points so that the function passes through all the given data points (i.e. reproduces the data points exactly) and can be used to estimate data points in-between the given ones.
- Newton method

Interpolation vs Regression

- Regression is different from interpolation in that it allows us to approximate overdetermined system, which has more equations than unknowns.
- This is useful when the exact solution is too expensive or unnecessary due to errors in the data, such as measurement errors or random noise.

Interpolation vs Regression

- In interpolation you are given some data points, and you are supposed to find a curve which fits the input/output relationship perfectly. In case of interpolation, you don't have to worry about variance of the fitted curve. As long as your curve is giving perfect output(for a given dataset), you are done.
- In regression, your curve should not be over-fitting. It should give you a regularized model so that you can predict future values. Variance of your function should be as low as possible. Your model should be generalized so that you can accurately predict future values.

Interpolation vs Regression

- Interpolation is like an algorithm without a "brain": it tries to achieve perfect match to the given data
- Regression is the same algorithm with the power to **generalize**. It won't fit perfectly to your data but at least it will try to learn some insights from it.
-

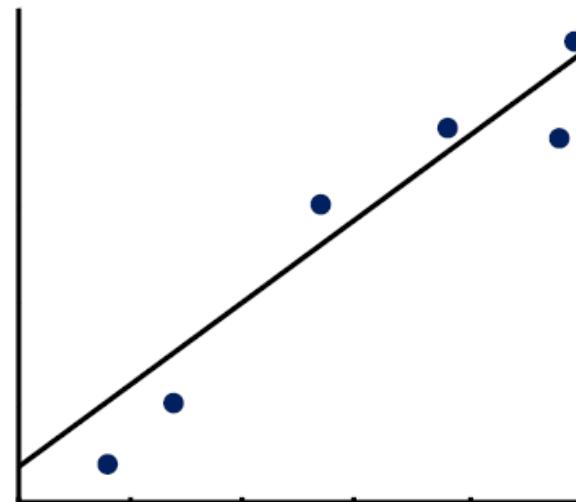
Example: Regression

- Given the following data:

x	0.8	1.4	2.7	3.8	4.8	4.9
y	0.69	1.00	2.02	2.39	2.34	2.83

Regression:

Obtain a straight line that
best fits the data



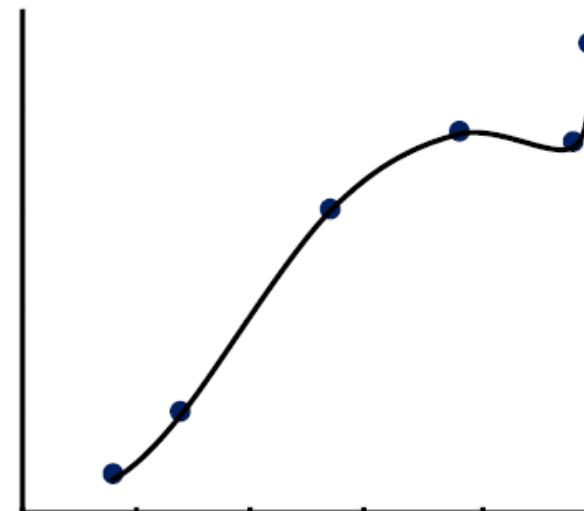
Example: Interpolation

- Given the following data:

x	0.8	1.4	2.7	3.8	4.8	4.9
y	0.69	1.00	2.02	2.39	2.34	2.83

Interpolation:

“Join the dots” and
find a curve passing
through the data.



Regression vs. Interpolation

- Given the following data:

x	0.8	1.4	2.7	3.8	4.8	4.9
y	0.69	1.00	2.02	2.39	2.34	2.83

- In regression, we are interested in fitting a chosen function to data

$$y = 0.45 + 0.47x$$

- In interpolation, given finite amount of data, we are interested in obtaining new data-points within this range.

At $x = 2.0, y = 1.87$

- Linear Regression: Fit a straight line to the given data
- Newton's Interpolation: For values at intermediate points

Example: Temperature variation in a day

time	00	01	02	03	04	05	06	07	08	09	10	11	12
T	25.6	25.4	25.1	24.9	24.9	25.2	25.9	26.3	27.1	29.3	30.8	31.2	32.1
time		13	14	15	16	17	18	19	20	21	22	23	24
T		31.0	30.3	31.4	30.6	31.8	29.6	28.4	28.1	28.2	27.4	26.8	26.1

We are interested in finding temperature at various times during the day, in addition to the ones where data is available.

We *interpolate* or “fill in” the missing data

Regressions

Multiple Linear Regression

Simple
Linear
Regression

$$y = b_0 + b_1 * x_1$$

Multiple
Linear
Regression

$$y = \text{Constant} + \text{Coefficients}_1 * \text{Independent variable}_1 + \text{Coefficients}_2 * \text{Independent variable}_2 + \dots + \text{Coefficients}_n * \text{Independent variable}_n$$

Dependent variable (DV) Independent variables (IVs)

Constant Coefficients

```
graph TD; DV[Dependent variable (DV)] --> Eq[y = b0 + b1*x1 + b2*x2 + ... + bn*xn]; IVs[Independent variables (IVs)] --> Eq; Constant[Constant] --> b0[b0]; Coefficients[Coefficients] --> terms["b1*x1 + b2*x2 + ... + bn*xn"]
```

Dummy Variables

Profit	R&D Spend	Admin	Marketing	State
192,261.83	165,349.20	136,897.80	471,784.10	New York
191,792.06	162,597.70	151,377.59	443,898.53	California
191,050.39	153,441.51	101,145.55	407,934.54	California
182,901.99	144,372.41	118,671.85	383,199.62	New York
166,187.94	142,107.34	91,391.77	366,168.42	California

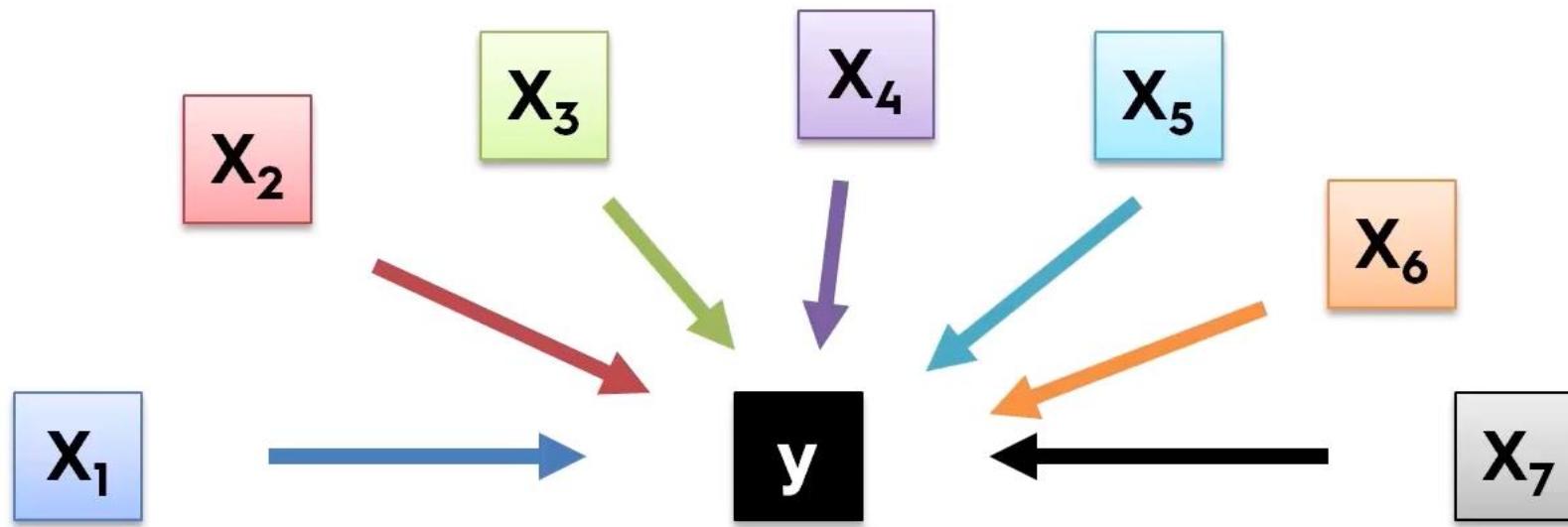
Dummy Variables

New York	California
1	0
0	1
0	1
1	0
0	1

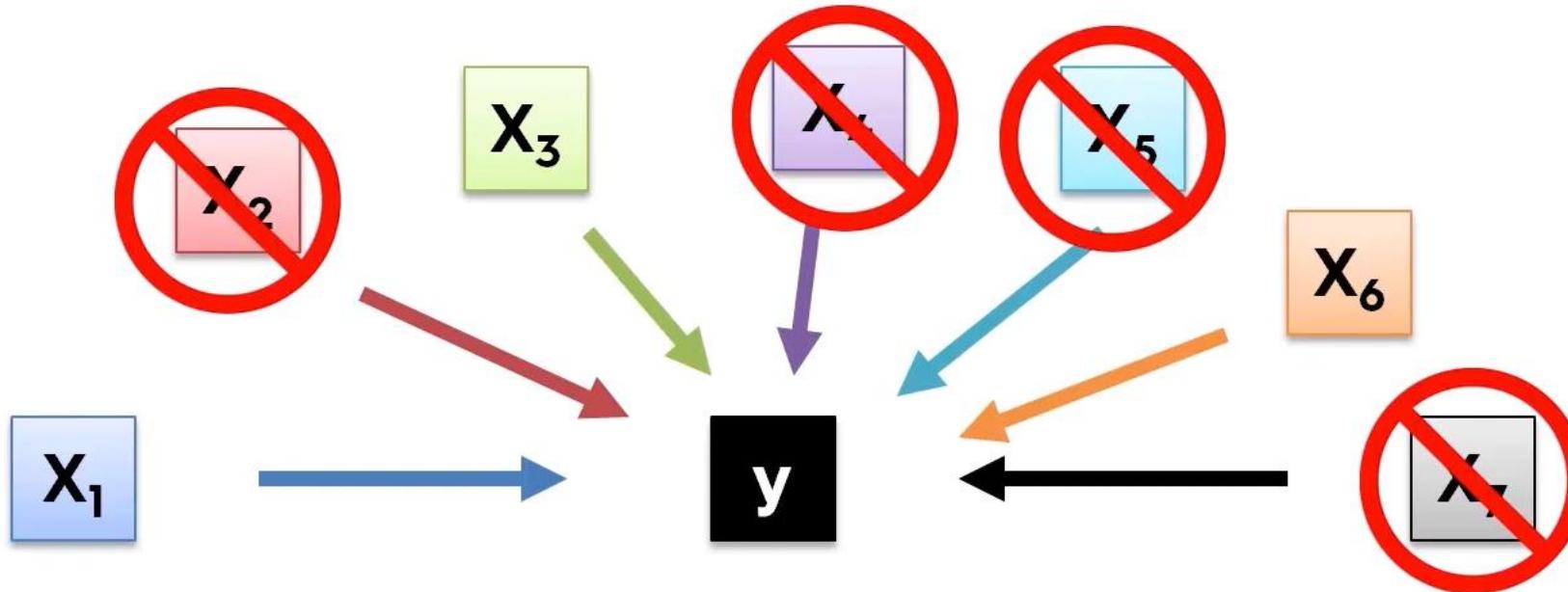
$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_3 * x_3 + b_4 * D_1$$



Building Model

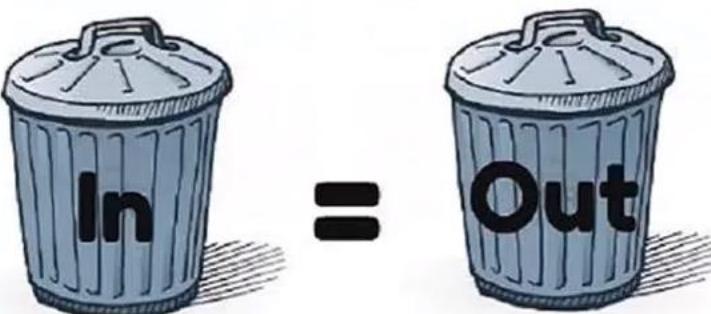


Building A Model



Why?

1)



2)

A photograph of a handwritten mathematical proof or derivation on yellowed paper. The handwriting is in red ink, with some parts circled in red. The text is dense and includes various mathematical symbols and equations.

Building A Model

5 methods of building models:

1. All-in
 2. Backward Elimination
 3. Forward Selection
 4. Bidirectional Elimination
 5. Score Comparison
- 
- Stepwise
Regression

Building A Model

Press Esc to exit full screen

Backward Elimination

STEP 1: Select a significance level to stay in the model (e.g. SL = 0.05)



STEP 2: Fit the full model with all possible predictors



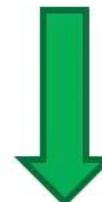
STEP 3: Consider the predictor with the highest P-value. If $P > SL$, go to STEP 4, otherwise go to FIN



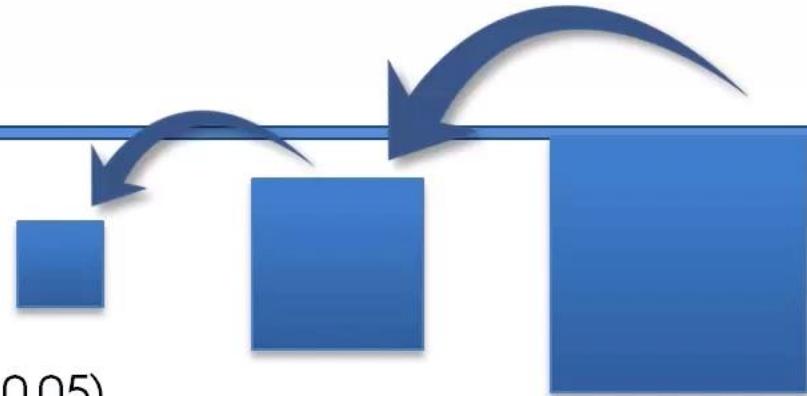
STEP 4: Remove the predictor



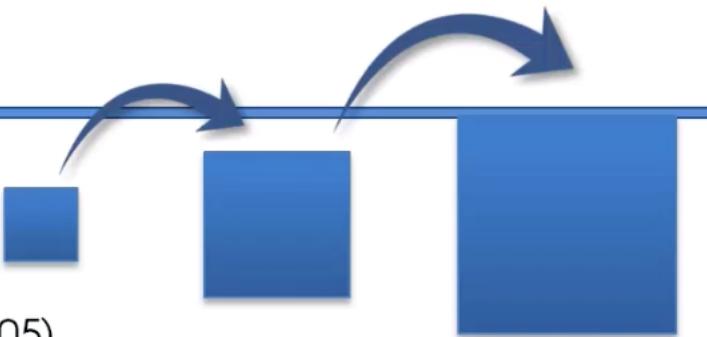
STEP 5: Fit model without this variable*



FIN: Your Model Is Ready



Building A Model



Forward Selection

STEP 1: Select a significance level to enter the model (e.g. SL = 0.05)



STEP 2: Fit all simple regression models $y \sim x_n$. Select the one with the lowest P-value



STEP 3: Keep this variable and fit all possible models with one extra predictor added to the one(s) you already have



STEP 4: Consider the predictor with the lowest P-value. If $P < SL$, go to STEP 3, otherwise go to FIN

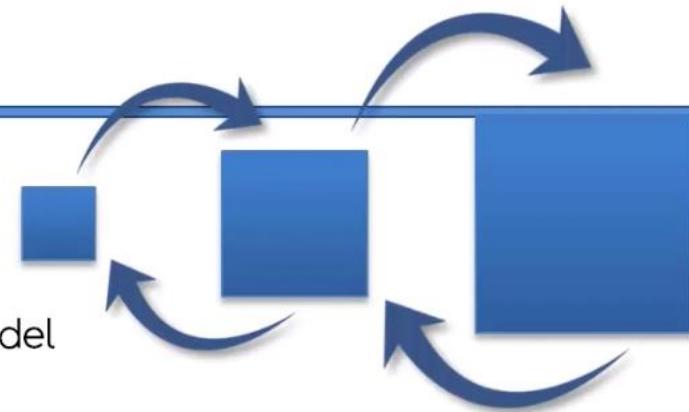


FIN: Keep the previous model

Building A Model

Bidirectional Elimination

STEP 1: Select a significance level to enter and to stay in the model
e.g.: SLENTER = 0.05, SLSTAY = 0.05



- ↓
- STEP 2: Perform the next step of Forward Selection (new variables must have: $P < \text{SLENTER}$ to enter)
- ↓
- STEP 3: Perform ALL steps of Backward Elimination (old variables must have $P < \text{SLSTAY}$ to stay)
- ↓
- STEP 4: No new variables can enter and no old variables can exit



FIN: Your Model Is Ready

Building A Model

All Possible Models

STEP 1: Select a criterion of goodness of fit (e.g. Akaike criterion)



STEP 2: Construct All Possible Regression Models: $2^N - 1$ total combinations



STEP 3: Select the one with the best criterion



FIN: Your Model Is Ready



Example:
**10 columns means
1,023 models**

Building A Model

5 methods of building models:

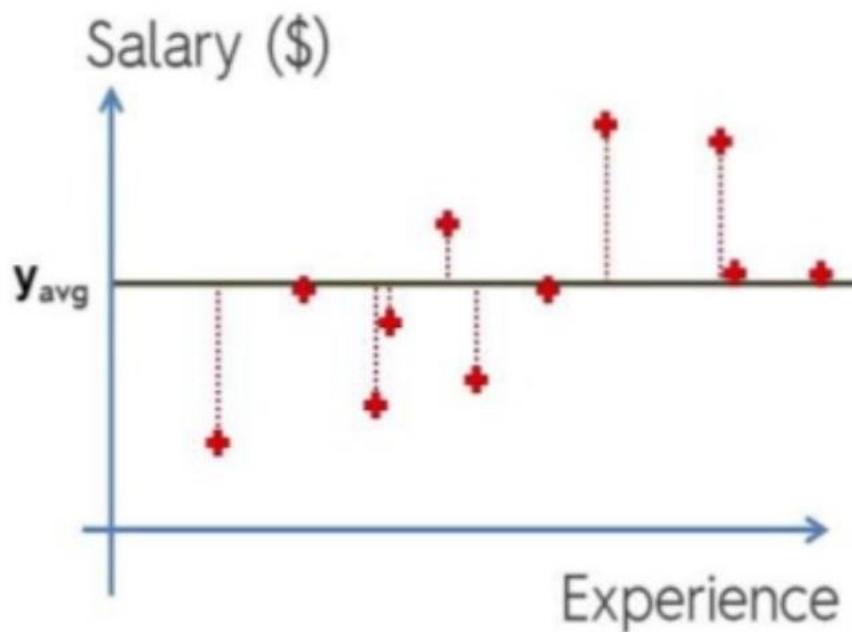
1. All-in
2. Backward Elimination
3. Forward Selection
4. Bidirectional Elimination
5. Score Comparison

```
import statsmodels.formula.api as sm
def backwardElimination(x, sl):
    numVars = len(x[0])
    for i in range(0, numVars):
        regressor_OLS = sm.OLS(y, x).fit()
        maxVar = max(regressor_OLS.pvalues).astype(float)
        if maxVar > sl:
            for j in range(0, numVars - i):
                if (regressor_OLS.pvalues[j].astype(float) == maxVar):
                    x = np.delete(x, j, 1)
            regressor_OLS.summary()
    return x
```

```
SL = 0.05
X_opt = X[:, [0, 1, 2, 3, 4, 5]]
X_Modeled = backwardElimination(X_opt, SL)
```

R SQUARED INTUITION

Simple Linear Regression:



$$SS_{res} = \text{SUM } (y_i - \hat{y}_i)^2$$

$$SS_{tot} = \text{SUM } (y_i - y_{avg})^2$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

- Goal : find a line with SSres as low as possible
- Value of R^2 **generally** lies between 0 and 1
- The more closer the R^2 is to 1 the better our line is.
- Why we square ? To deal with positive values and to deal with outliers , however we can use power of 4 or 6 or etc. but square is convention and that is widely accepted hence we will use that for now.
- R^2 can take negative values (if our data fits worstly to our linear regression)

PROBLEMS WITH R²

As number of independent values increase the value of R² will either increase but will never decrease.

Hence we will not know how good of an influence does this newly added independent variable has on our dependent variable.

Reason for this is that any independent variable has tendency of slightly correlate with the dependent variable. This might help reducing the SSres value hence the value of R² increases.

To overcome this we use adjusted R²

Adjusted R²

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

R² - Goodness of fit
(greater is better)

$$y = b_0 + b_1 * x_1$$

Problem:

$$y = b_0 + b_1 * x_1 + b_2 * x_2$$

$$+ b_3 * x_3$$

SS_{res} → Min

R² will never decrease

ADJUSTED R²

Adjusted R² deals with additional independent variables.

This r squared value tends to penalize the value of the rsquare if our choice of independent variable wasn't good (i.e independent variable had no effect on dependent variable)

Also the bias of R SQUARE to not to decrease is handled pretty well in this adjusted R SQUARED method.

DIFFERENCE

The adjusted R-squared is a modified version of R-squared that has been adjusted for the number of predictors in the model. The adjusted R-squared increases only if the new term improves the model more than would be expected by chance. It decreases when a predictor improves the model by less than expected by chance. The adjusted R-squared can be negative, but it's usually not. It is always lower than the R-squared.

R-squared or R² explains the degree to which your input variables explain the variation of your output / predicted variable. So, if R-square is 0.8, it means 80% of the variation in the output variable is explained by the input variables. So, in simple terms, higher the R squared, the more variation is explained by your input variables and hence better is your model.

However, the problem with R-squared is that it will either stay the same or increase with addition of more variables, even if they do not have any relationship with the output variables. This is where “Adjusted R square” comes to help. Adjusted R-square penalizes you for adding variables which do not improve your existing model.

Hence, if you are building Linear regression on multiple variable, it is always suggested that you use Adjusted R-squared to judge goodness of model. In case you only have one input variable, R-square and Adjusted R squared would be exactly same.

Typically, the more non-significant variables you add into the model, the gap in R-squared and Adjusted R-squared increases.

Ordinary Least Square Method

```
model1=sm.OLS(y_train,x_train)
```

```
In [221]: result=model1.fit()
```

```
In [222]: print(result.summary())
```

OLS Regression Results

```
=====
```

Dep. Variable:	0	R-squared:	0.959
Model:	OLS	Adj. R-squared:	0.958
Method:	Least Squares	F-statistic:	711.8
Date:	Sun, 16 Apr 2017	Prob (F-statistic):	8.37e-263
Time:	21:23:08	Log-Likelihood:	-1210.8
No. Observations:	404	AIC:	2448.
Df Residuals:	391	BIC:	2500.
Df Model:	13		
Covariance Type:	nonrobust		

```
=====
```

	coef	std err	t	P> t	[95.0% Conf. Int.]
CRIM	-0.1077	0.039	-2.779	0.006	-0.184 -0.031
ZN	0.0484	0.016	2.952	0.003	0.016 0.081
INDUS	-0.0232	0.073	-0.317	0.751	-0.167 0.121
CHAS	2.9930	1.062	2.819	0.005	0.906 5.080
NOX	-2.1626	3.662	-0.591	0.555	-9.362 5.036
RM	5.9590	0.339	17.584	0.000	5.293 6.625
AGE	-0.0169	0.015	-1.094	0.274	-0.047 0.013
DIS	-1.0273	0.220	-4.661	0.000	-1.461 -0.594
RAD	0.1669	0.075	2.240	0.026	0.020 0.313
TAX	-0.0105	0.004	-2.368	0.018	-0.019 -0.002
PTRATIO	-0.3753	0.124	-3.018	0.003	-0.620 -0.131
B	0.0143	0.003	4.733	0.000	0.008 0.020
LSTAT	-0.3463	0.057	-6.129	0.000	-0.457 -0.235

```
=====
```

Omnibus: 151.837 Durbin-Watson: 1.804

We can drop few variables and select only those that have p values < 0.5 and then we can check improvement in the model.

A general approach to compare two different models is AIC(Akaike Information Criteria) and the model with minimum AIC is the best one.

```
model2=sm.OLS(y_train,x_train[['CRIM','ZN','CHAS','RM','DIS','RAD','TAX','PTRATIO','B','LSTAT']])
result2=model2.fit()
print(result2.summary())
```

```
OLS Regression Results
=====
Dep. Variable:                      0    R-squared:                 0.959
Model:                            OLS    Adj. R-squared:            0.958
Method:                           Least Squares    F-statistic:             926.5
Date:                Sun, 16 Apr 2017    Prob (F-statistic):      1.08e-266
Time:                    20:40:51    Log-Likelihood:          -1212.1
No. Observations:                  404    AIC:                     2444.
Df Residuals:                      394    BIC:                     2484.
Df Model:                           10
Covariance Type:                nonrobust

=====
```

	coef	std err	t	P> t	[95.0% Conf. Int.]
CRIM	-0.1040	0.039	-2.695	0.007	-0.180 -0.028
ZN	0.0521	0.016	3.229	0.001	0.020 0.084
CHAS	2.7772	1.046	2.655	0.008	0.721 4.834
RM	5.7093	0.271	21.103	0.000	5.177 6.241
DIS	-0.8541	0.188	-4.542	0.000	-1.224 -0.484
RAD	0.1845	0.071	2.607	0.009	0.045 0.324
TAX	-0.0125	0.004	-3.412	0.001	-0.020 -0.005
PTRATIO	-0.3939	0.123	-3.197	0.002	-0.636 -0.152
B	0.0138	0.003	4.640	0.000	0.008 0.020
LSTAT	-0.3920	0.048	-8.168	0.000	-0.486 -0.298

OLS Summary Interpretation

- Omnibus/Prob(Omnibus) – a test of the skewness and kurtosis of the residual (characteristic #2). We hope to see a value close to zero which would indicate normalcy. The Prob (Omnibus) performs a statistical test indicating the probability that the residuals are normally distributed. We hope to see something close to 1 here. In this case Omnibus is relatively low and the Prob (Omnibus) is relatively high so the data is somewhat normal, but not altogether ideal. A linear regression approach would probably be better than random guessing but likely not as good as a nonlinear approach.
- Skew – a measure of data symmetry. We want to see something close to zero, indicating the residual distribution is normal. Note that this value also drives the Omnibus. This result has a small, and therefore good, skew.
- Kurtosis – a measure of "peakiness", or curvature of the data. Higher peaks lead to greater Kurtosis. Greater Kurtosis can be interpreted as a tighter clustering of residuals around zero, implying a better model with few outliers.

OLS Summary Interpretation

- Durbin-Watson – tests for homoscedasticity (characteristic #3). We hope to have a value between 1 and 2. In this case, the data is close, but within limits.
- Jarque-Bera (JB)/Prob(JB) – like the Omnibus test in that it tests both skew and kurtosis. We hope to see in this test a confirmation of the Omnibus test. In this case we do.
- Condition Number – This test measures the sensitivity of a function's output as compared to its input (characteristic #4). When we have multicollinearity, we can expect much higher fluctuations to small changes in the data, hence, we hope to see a relatively small number, something below 30. In this case we are well below 30, which we would expect given our model only has two variables and one is a constant.

OLS Summary Interpretation

- the AIC or BIC for a model is usually written in the form $[-2\log L + kp]$, where L is the likelihood function, p is the number of parameters in the model, and k is 2 for AIC and $\log(n)$ for BIC.
- AIC is an estimate of a constant plus the relative distance between the unknown true likelihood function of the data and the fitted likelihood function of the model, so that a lower AIC means a model is considered to be closer to the truth.
- BIC is an estimate of a function of the posterior probability of a model being true, under a certain Bayesian setup, so that a lower BIC means that a model is considered to be more likely to be the true model.

Regressions

Simple
Linear
Regression

$$y = b_0 + b_1 x_1$$

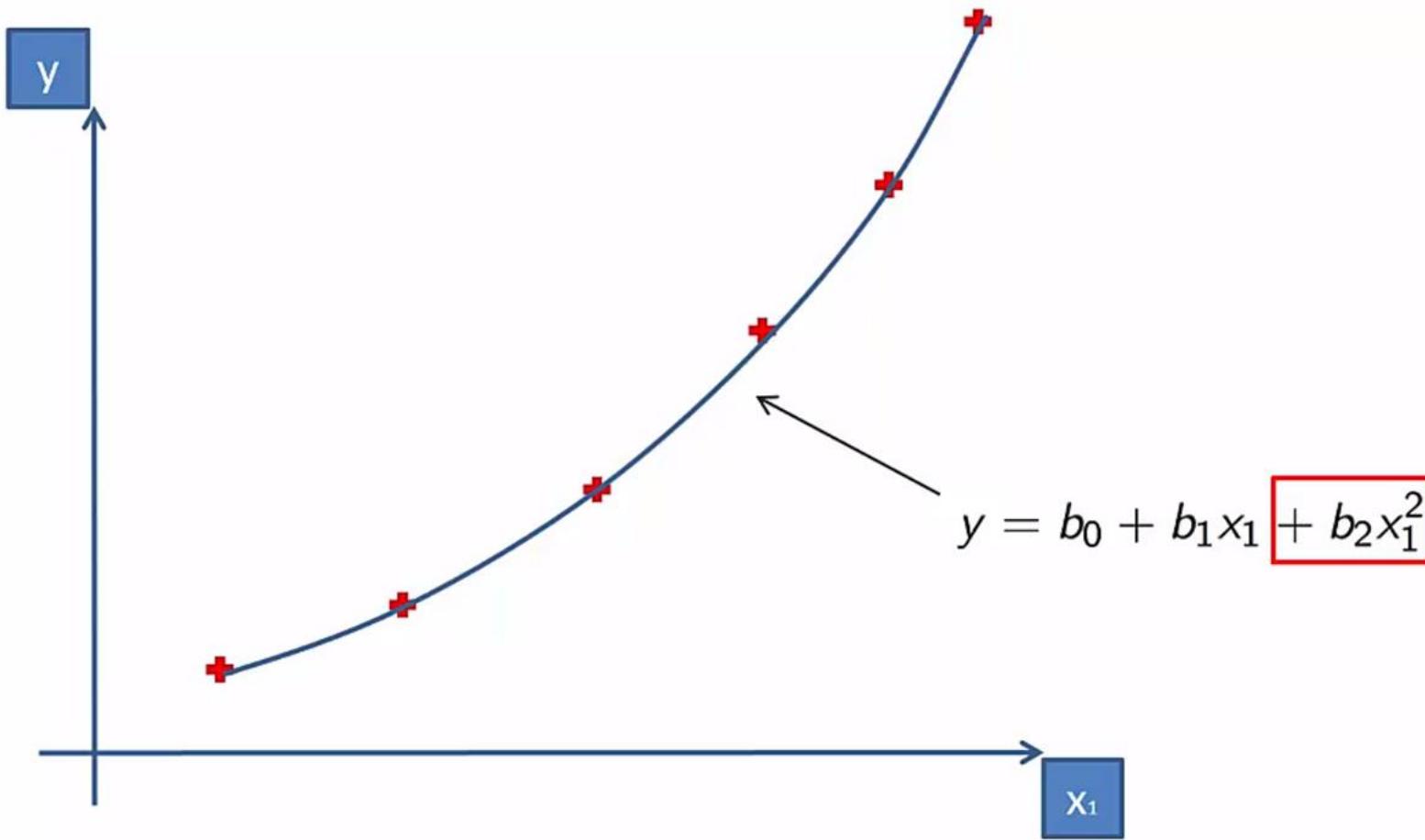
Multiple
Linear
Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_n x_n$$

Polynomial
Linear
Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \dots + b_n x_1^n$$

Polynomial Regression



Support Vector Regression

- SVM is a *supervised* learning model
- Problem:
- I have a business and I receive a lot of emails from customers every day. Some of these emails are complaints and should be answered very quickly. I would like a way to identify them quickly so that I answer these email in priority.

Support Vector Regression

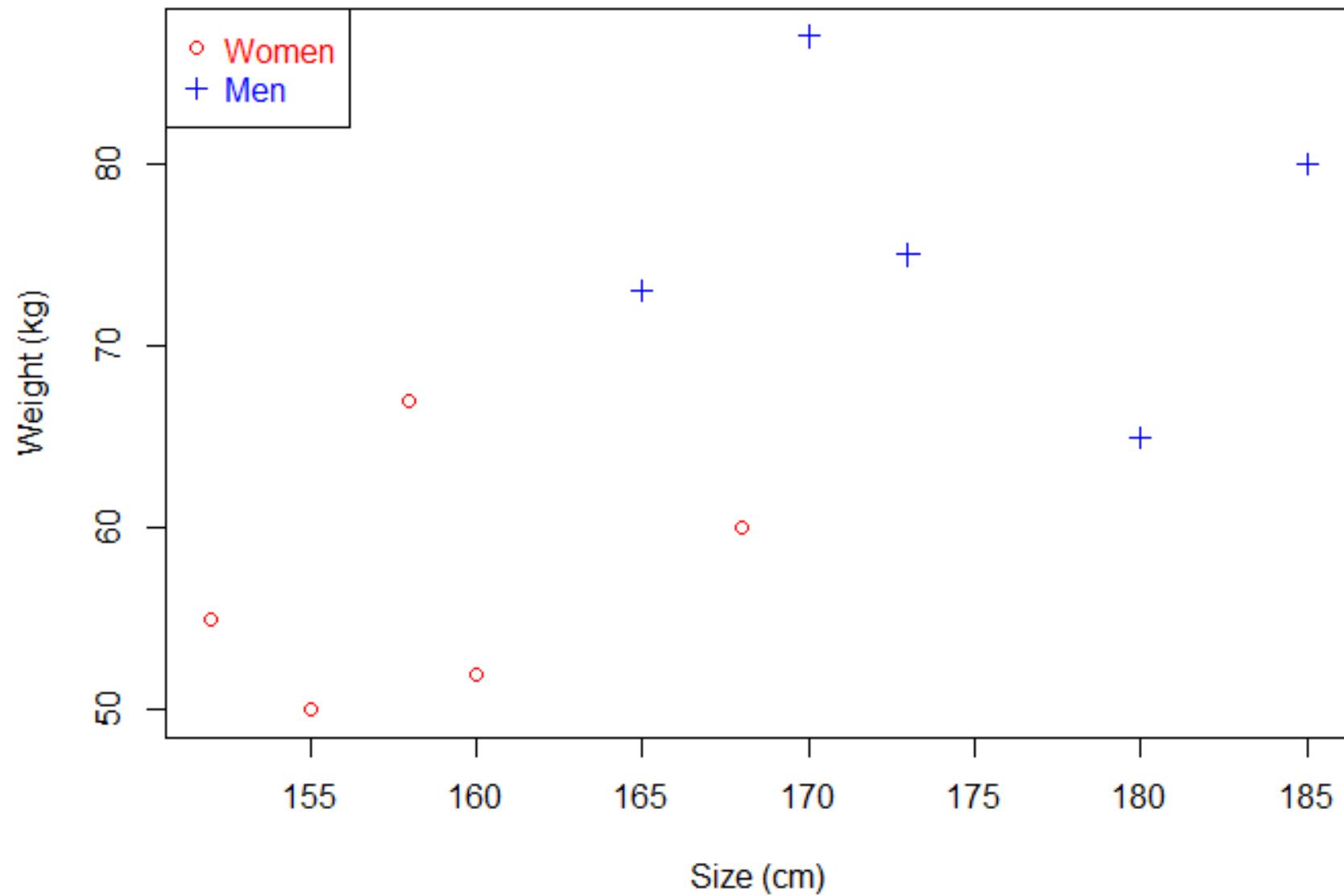
- Approach: I can use a supervised machine learning algorithm.
- Step 1: I need a lot of emails, the more the better.
Step 2: I will read the title of each email and classify it by saying "it is a complaint" or "it is not a complaint". It put a **label** on each email.
Step 3: I will **train** a model on this dataset
Step 4: I will assess the quality of the prediction (using cross validation)
Step 5: I will use this model to **predict** if an email is a complaint or not.
- In this case, if I have trained the model with a lot of emails then it will perform well. SVM is just one among many models you can use to learn from this data and make predictions.

Support Vector Regression

- **SVMs - Support Vector Machines**
- Wikipedia tells us that SVMs can be used to do two things: classification or regression.
- **SVM** is used for classification
- **SVR** (Support Vector Regression) is used for regression

What is the goal of the Support Vector Machine (SVM)?

- The goal of a support vector machine is to find the optimal separating hyperplane which maximizes the margin of the training data.
- The first thing we can see from this definition, is that a SVM needs training data. Which means it is a supervised learning algorithm.
- It is also important to know that SVM is a classification algorithm. Which means we will use it to predict if something belongs to a particular class.



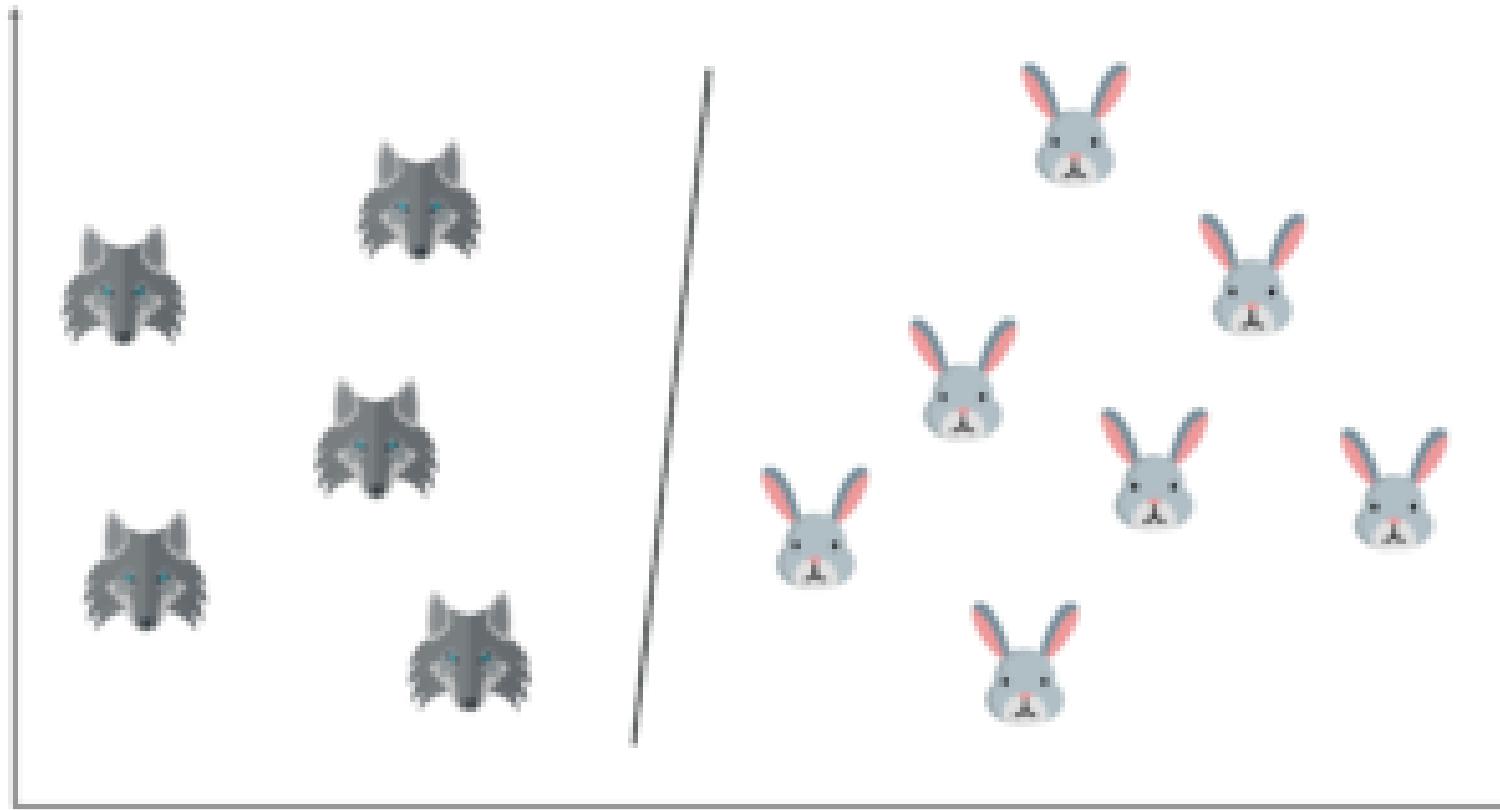
SVM

- We have plotted the size and weight of several people, and there is also a way to distinguish between men and women.
- With such data, using a SVM will allow us to answer the following question:
- Given a particular data point (weight and size), is the person a man or a woman ?
- For instance: if someone measures 175 cm and weights 80 kg, is it a man or a woman?

For a second, pretend you own a farm and you have a problem—you need to set up a fence to protect your rabbits from a pack of wolves. But where do you build your fence?

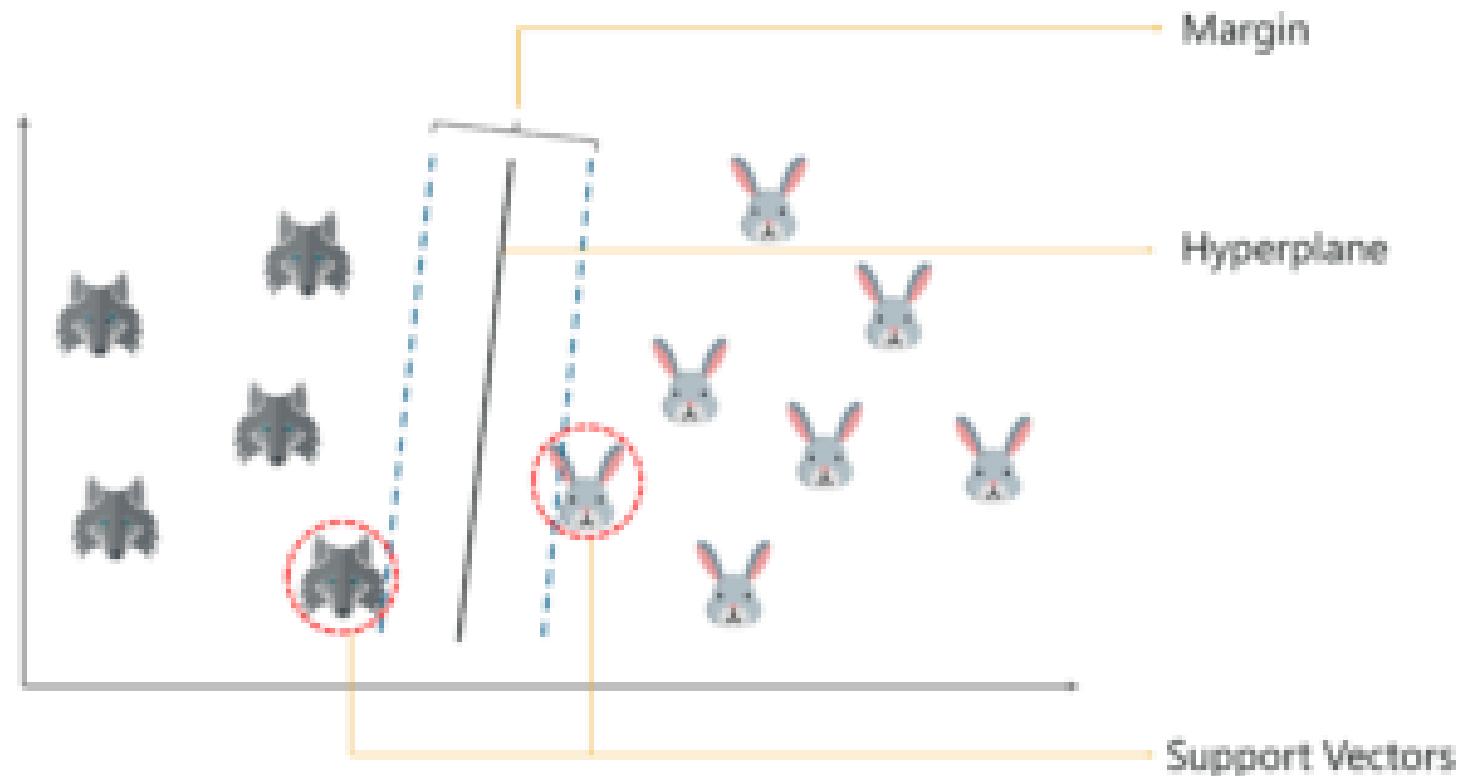


Hyperplane between any two classes in order to separate them or classify them

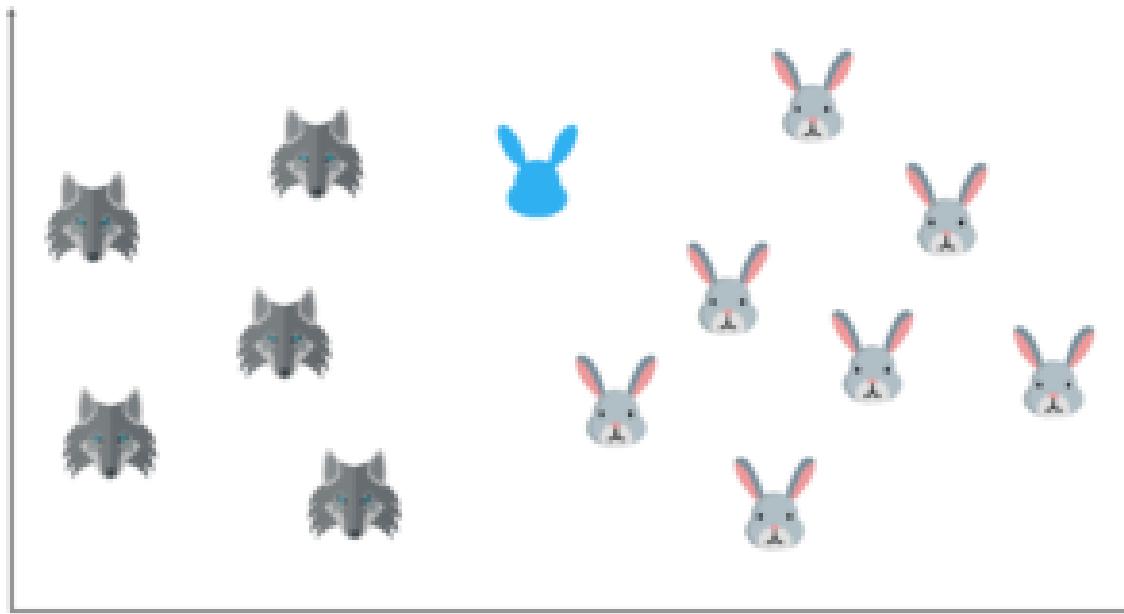


What is a Support Vector in SVM?

So, you start off by drawing a random hyperplane and then you check the distance between the hyperplane and the closest data points from each class. These closest data points to the hyperplane are known as support vectors. And that's where the name comes from, support vector machine.

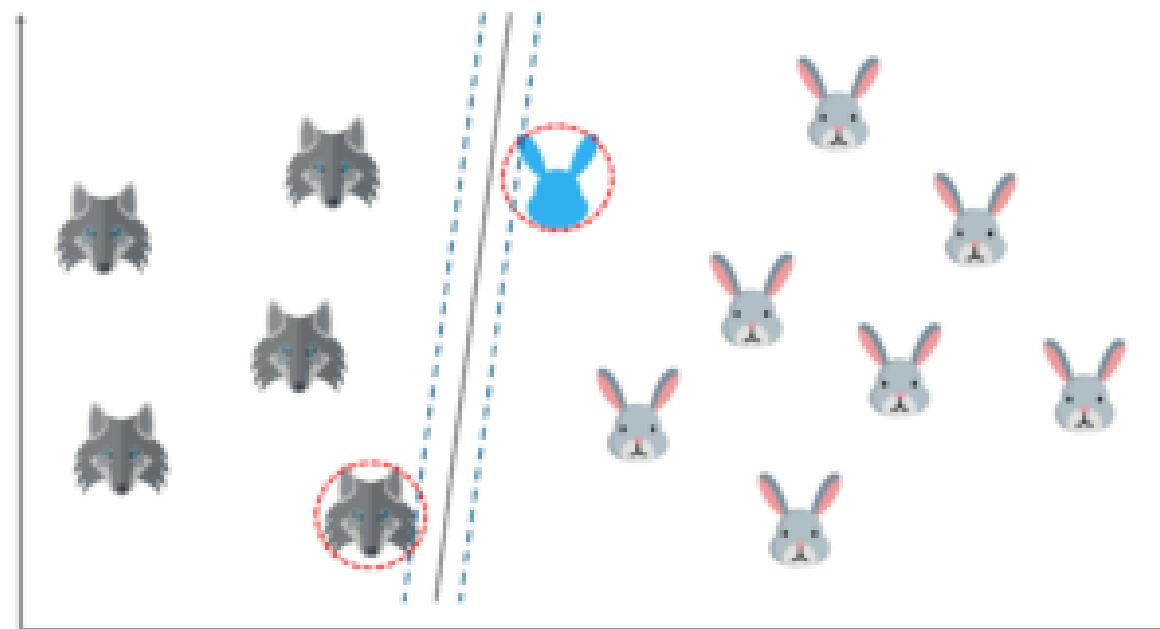


- The hyperplane is drawn based on these support vectors and an optimum hyperplane will have a maximum distance from each of the support vectors.
- And this distance between the hyperplane and the support vectors is known as the margin.
- To sum it up, SVM is used to classify data by using a hyperplane, such that the distance between the hyperplane and the support vectors is maximum.

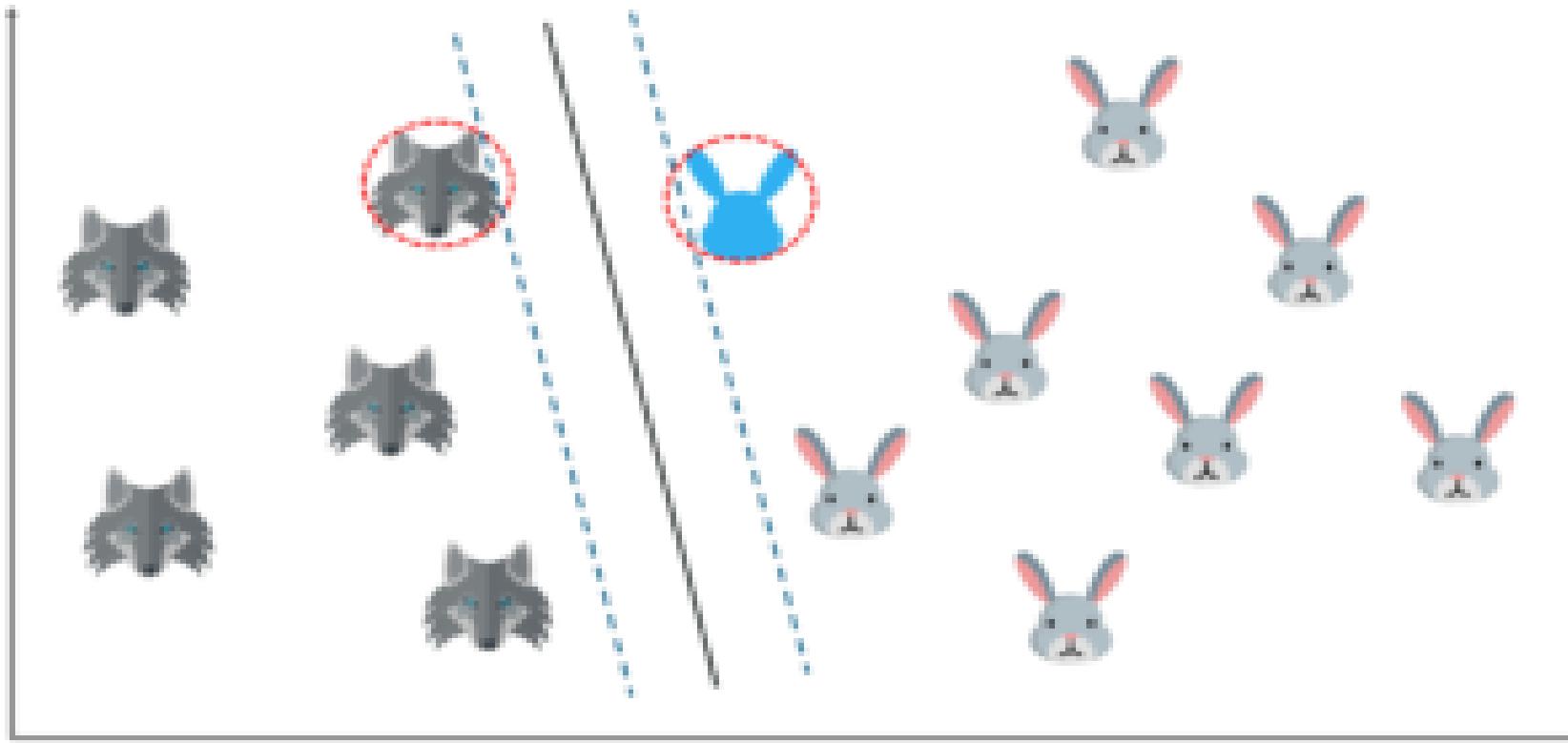


Let's say that I input a new data point and now I want to draw a hyperplane such that it best separates these two classes.

So, I start off by drawing a hyperplane and then I check the distance between the hyperplane and the support vectors. Here I'm basically trying to check if the margin is maximum for this hyperplane.

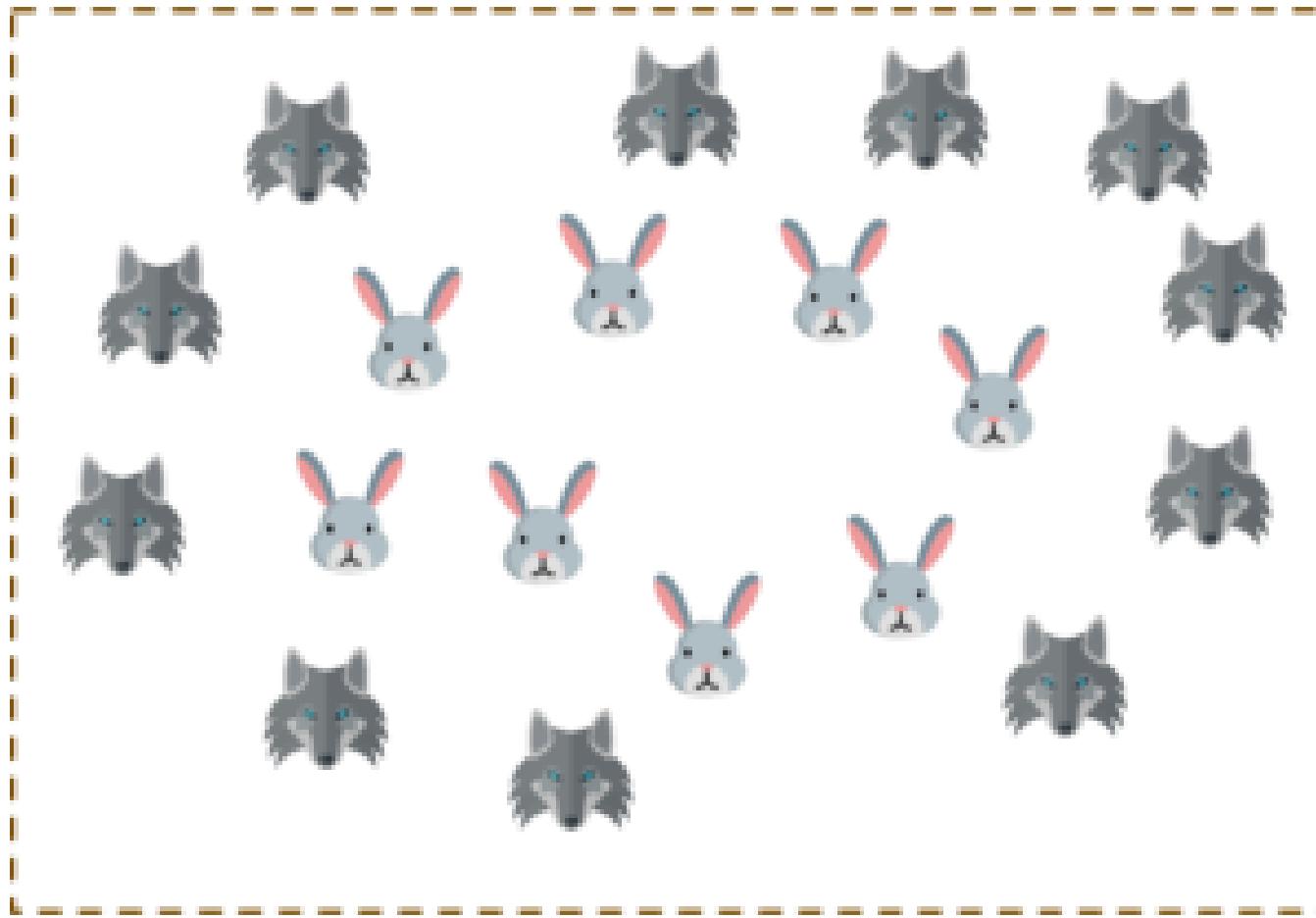


But what if I draw the hyperplane like this? The margin for this hyperplane is clearly more than the previous one. So, this is my optimal hyperplane.

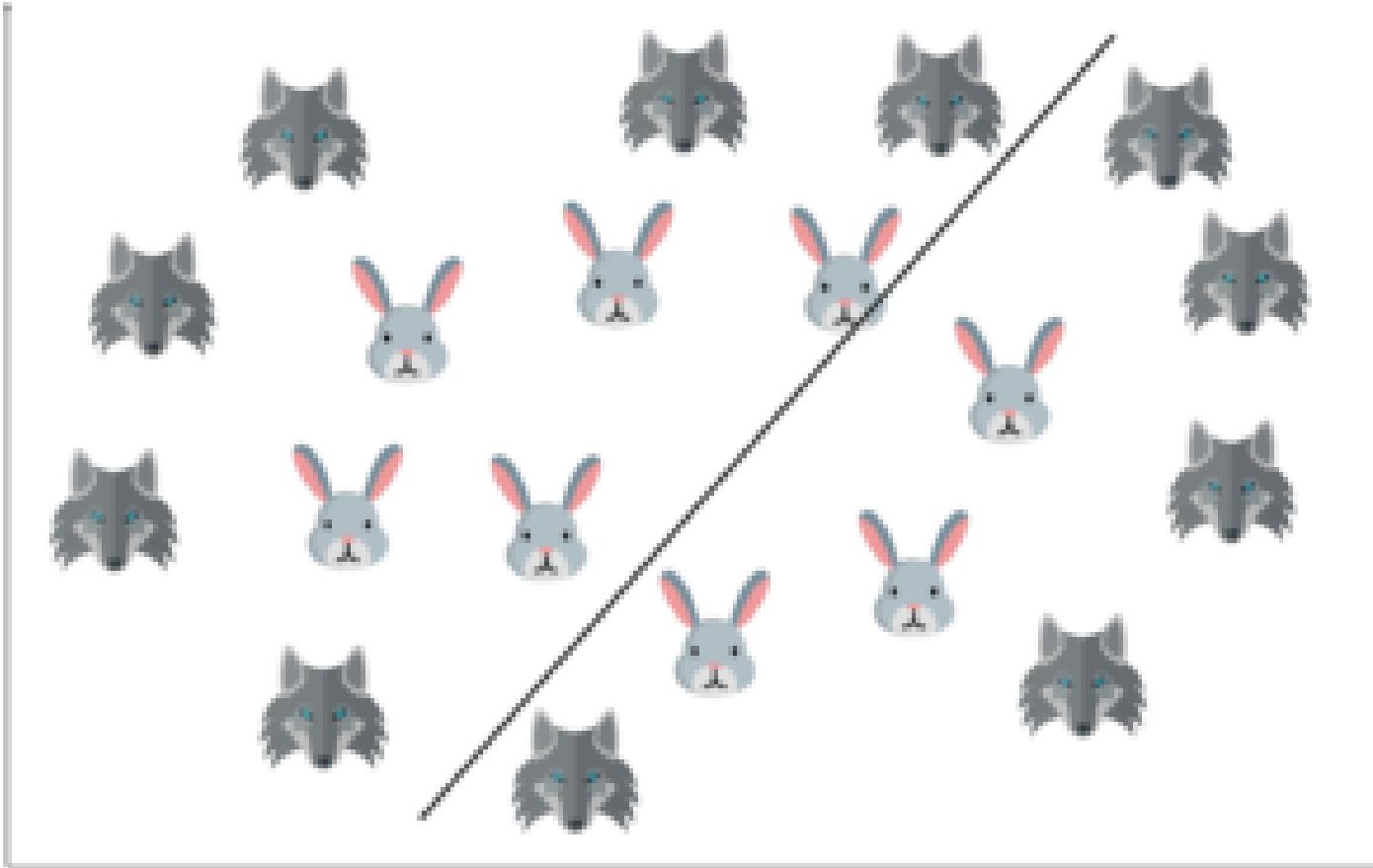


So far it was quite easy, our data was linearly separable, which means that you could draw a straight line to separate the two classes!

But what will you do if the data set is like this?

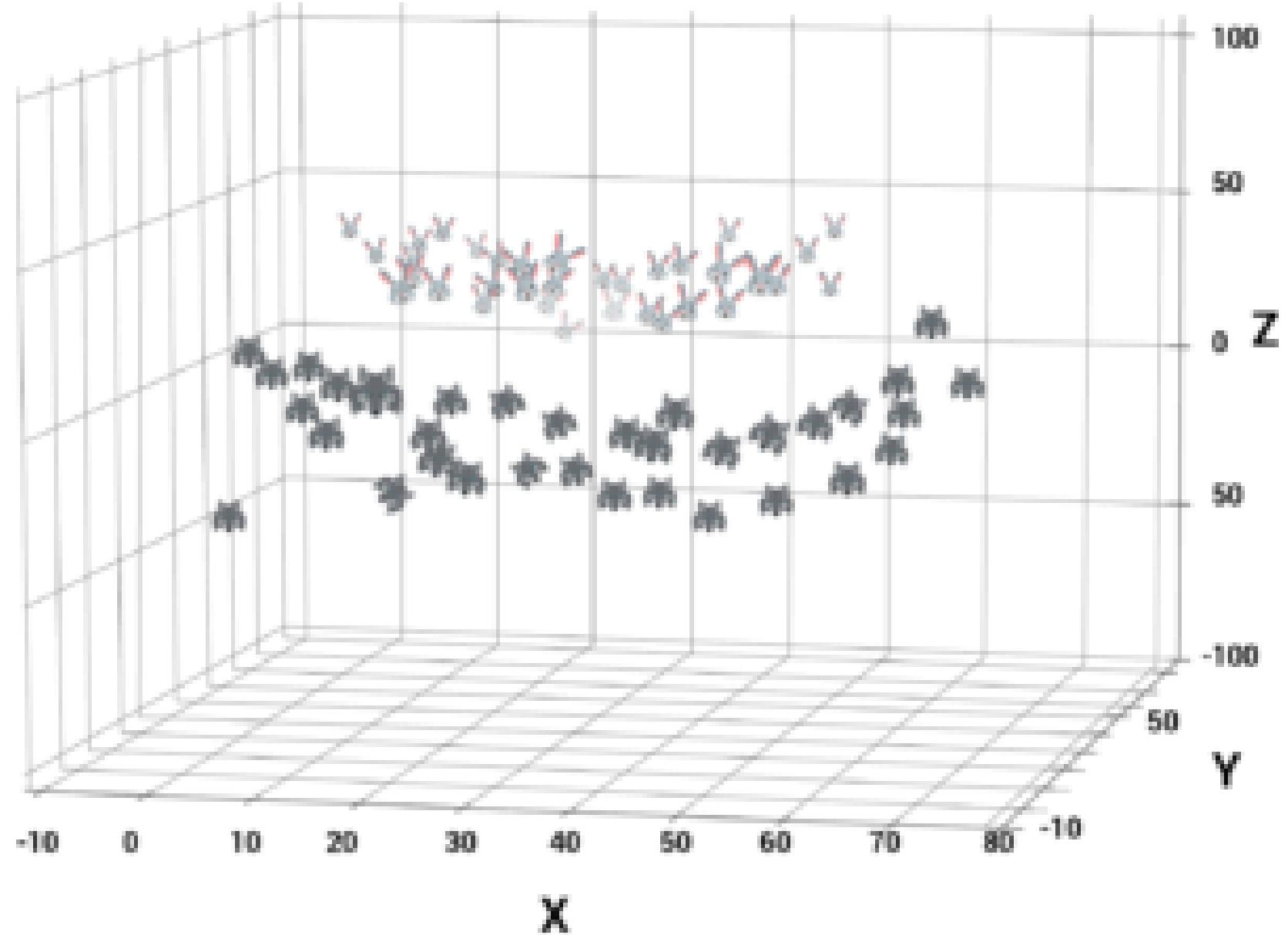


You possibly can't draw a hyperplane like this! It doesn't separate the two classes.



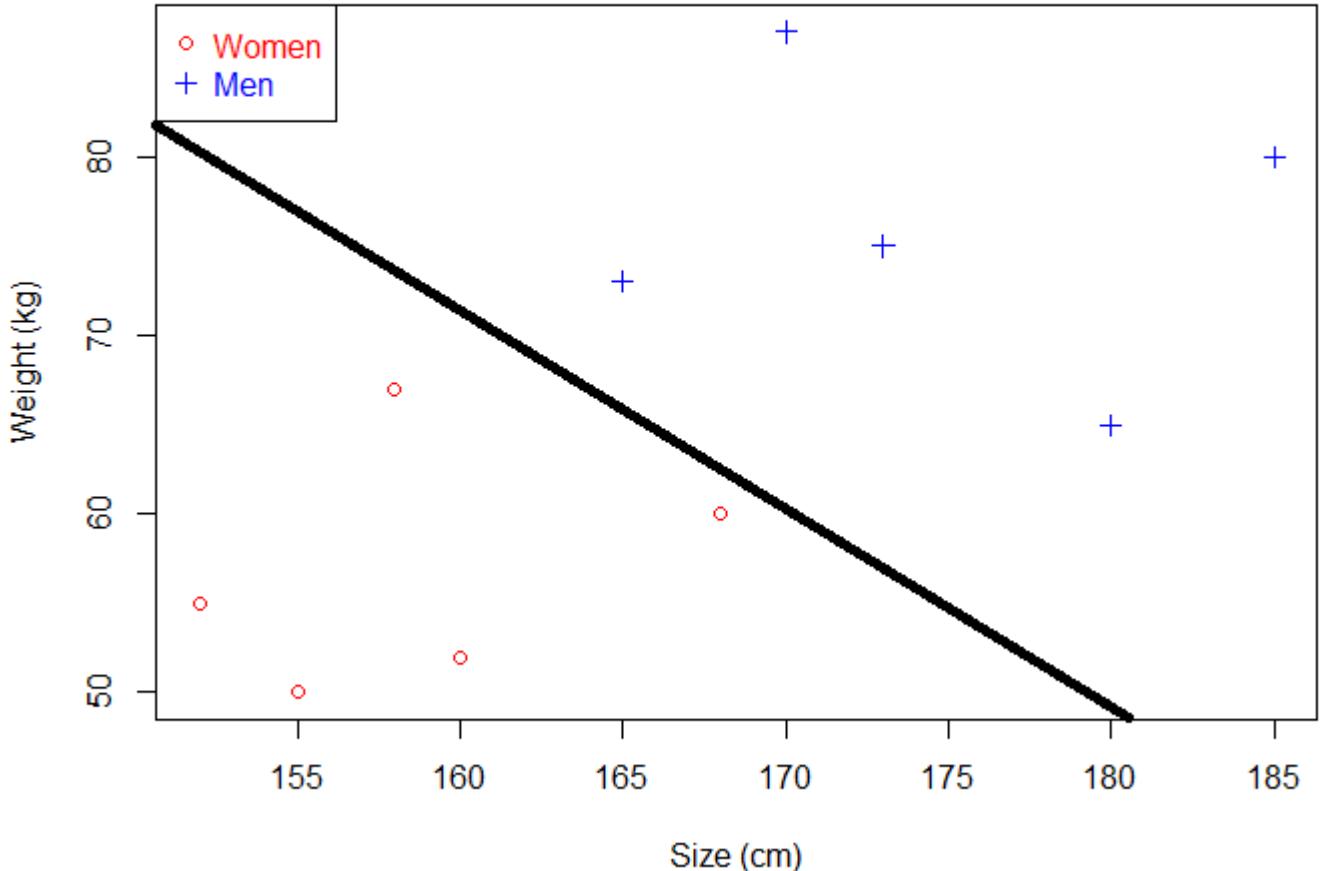
Non-Linear Support Vector Machine (SVM)

- Kernel functions offer the user the option of transforming nonlinear spaces into linear ones.
- Until this point, we were plotting our data on 2-dimensional space. So, we had only 2 variables, x and y.
- A simple trick would be transforming the two variables x and y into a new feature space involving a new variable z. Basically, we're visualizing the data on a 3-dimensional space.
- When you transform the 2D space into a 3D space you can clearly see a dividing margin between the 2 classes of data. And now you can go ahead and separate the two classes by drawing the best hyperplane between them.



What is a separating hyperplane?

- For instance, we could trace a line and then all the data points representing men will be above the line, and all the data points representing women will be below the line.
- Such a line is called a **separating hyperplane** and is depicted below:

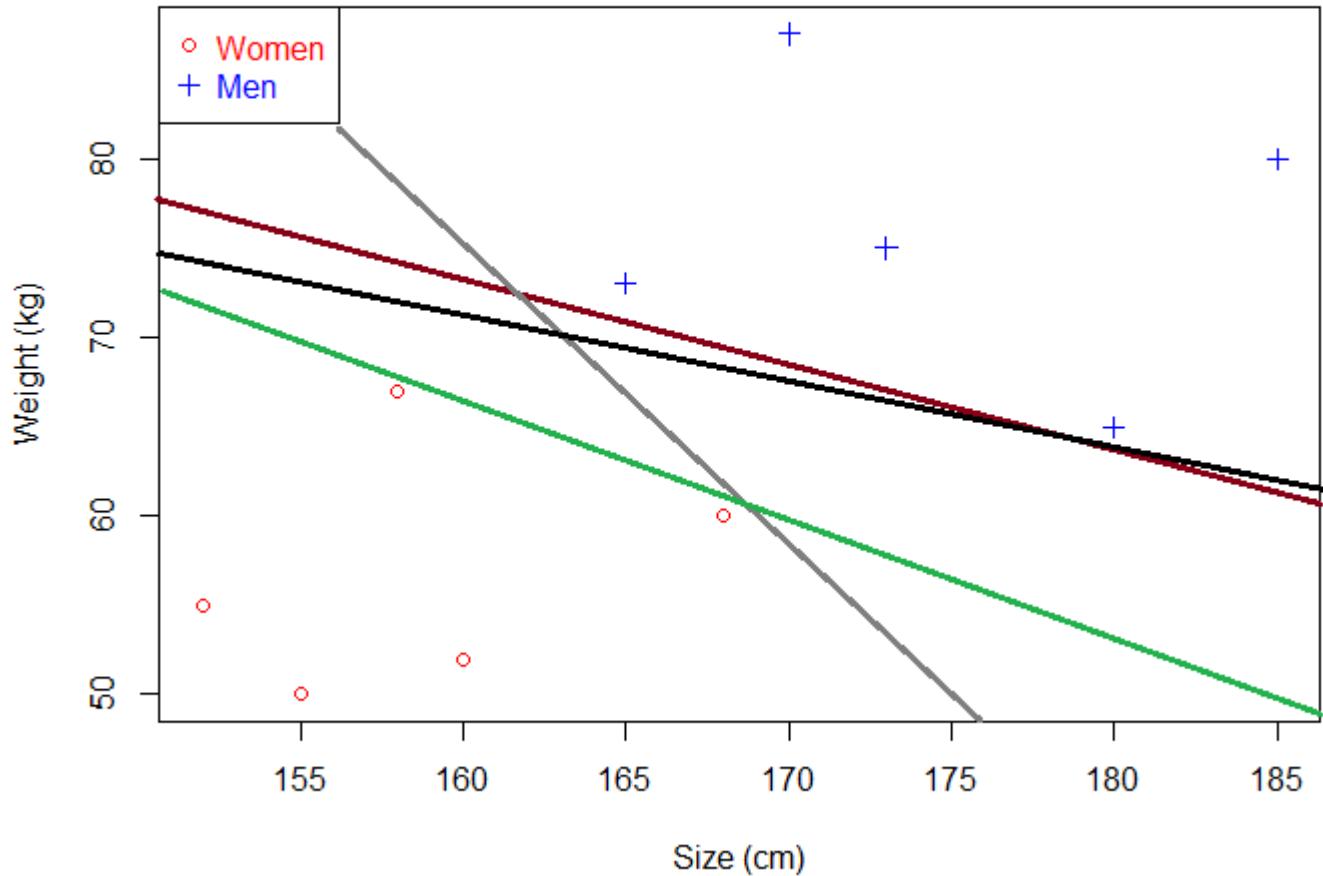


If it is just a line, why do we call it an hyperplane ?

- An hyperplane is a generalization of a plane.
- in one dimension, an hyperplane is called a point
- in two dimensions, it is a line
- in three dimensions, it is a plane
- in more dimensions you can call it an hyperplane

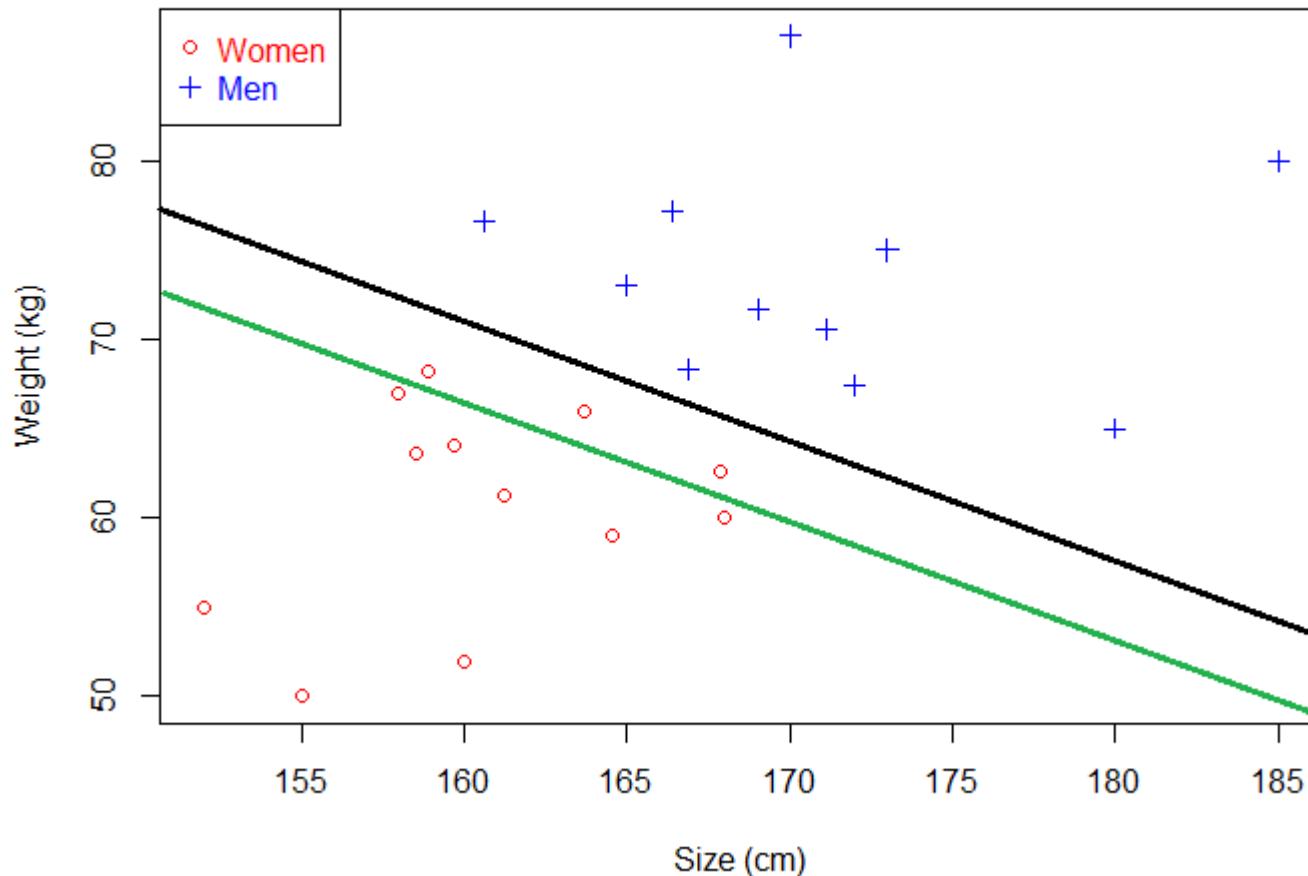
What is the *optimal* separating hyperplane?

- The fact that you can find a **separating hyperplane**, does not mean it is the best one !
- In the example below there is several separating hyperplanes. Each of them is valid as it successfully separates our data set with men on one side and women on the other side.



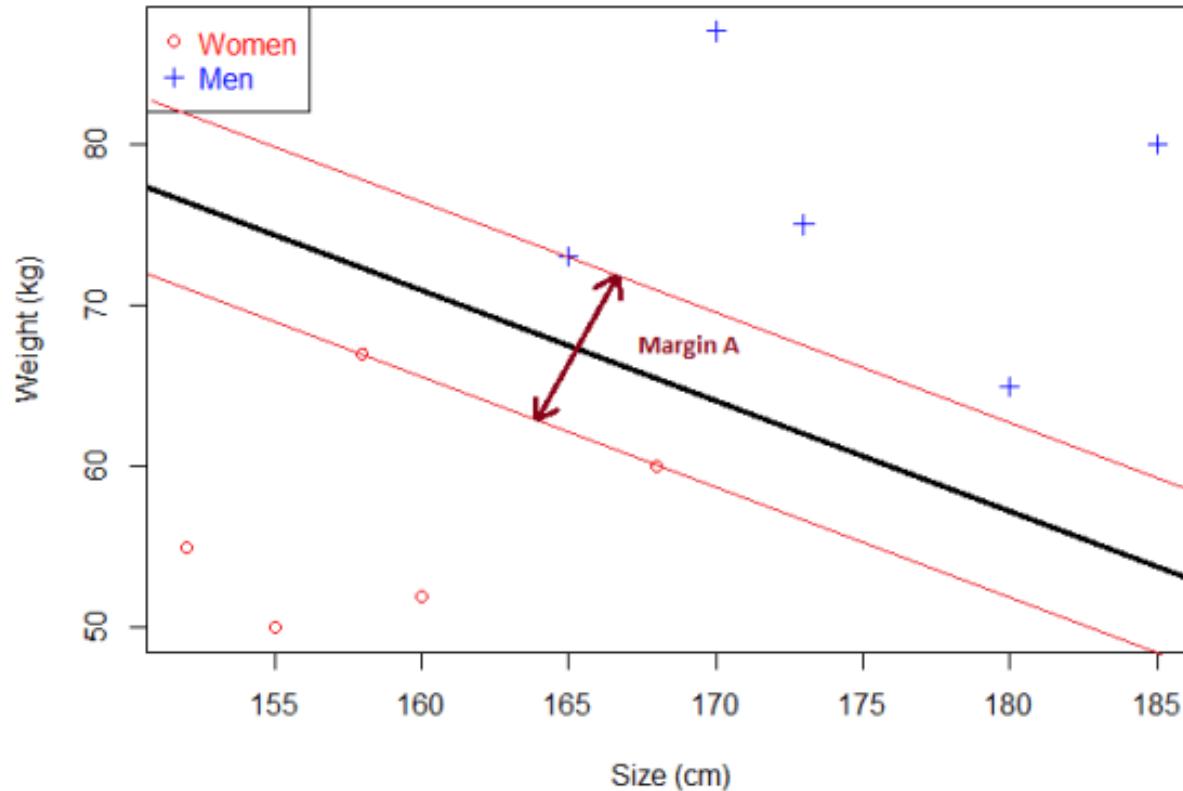
Optimal Hyperplane

The black hyperplane classifies more accurately than the green one



What is the margin and how does it help choosing the optimal hyperplane?

- Given a particular hyperplane, we can compute the distance between the hyperplane and the closest data point.
- Once we have this value, if we double it we will get what is called the **margin**.
- Basically the margin is a no man's land. There will never be any data point inside the margin.

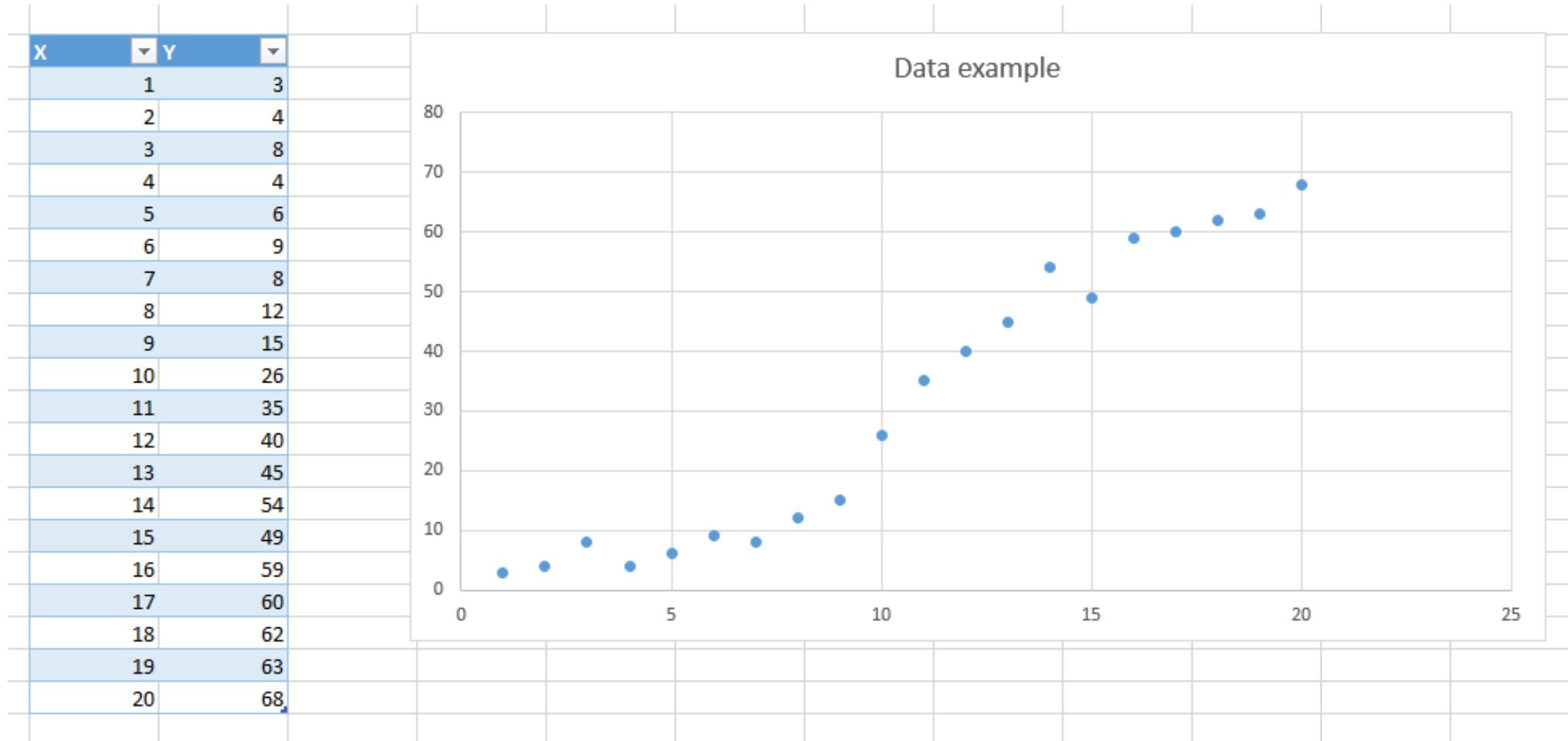


Optimal Hyperplane

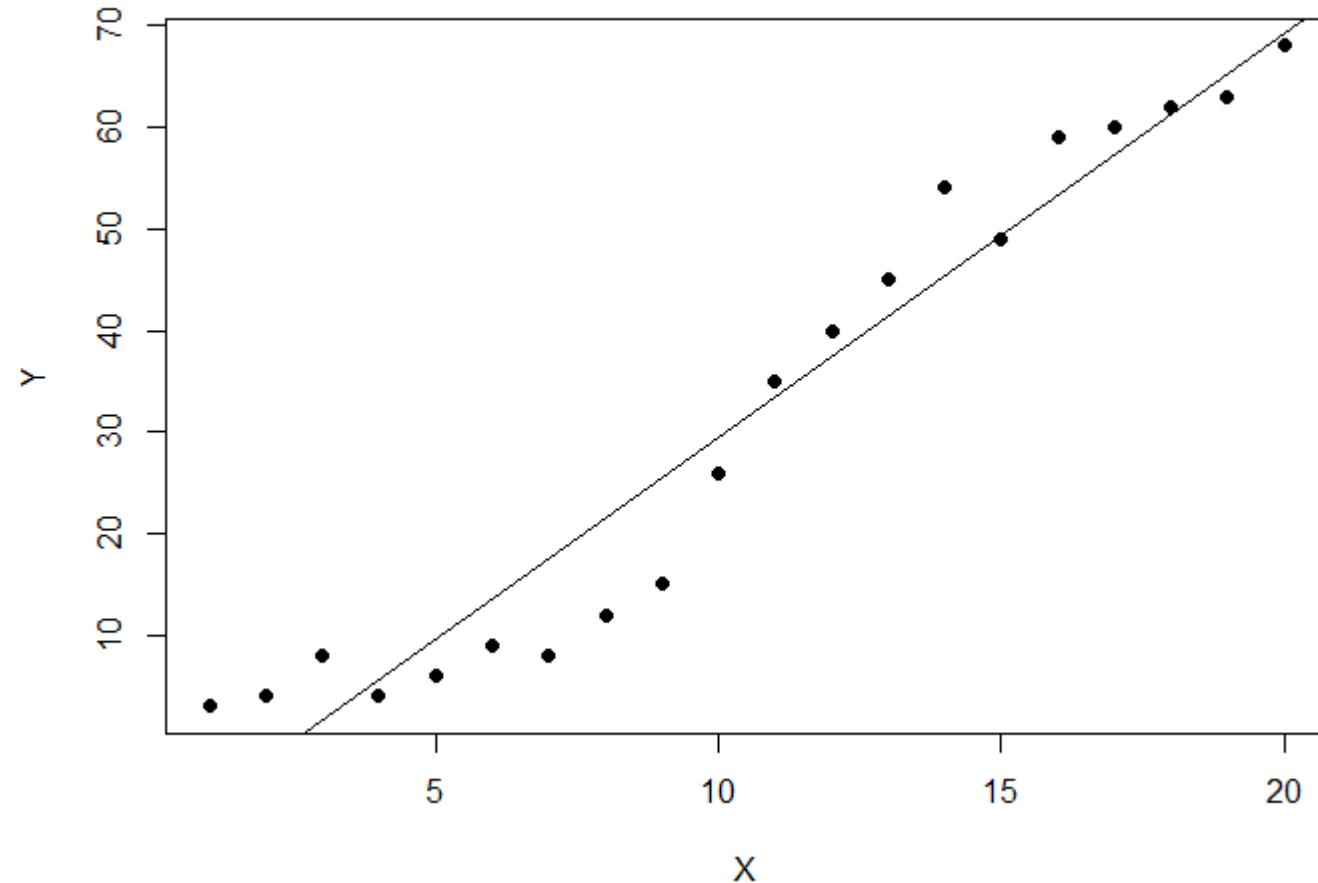
We can make the following observations:

- If an hyperplane is very close to a data point, its margin will be small.
- The further an hyperplane is from a data point, the larger its margin will be.
- This means that **the optimal hyperplane will be the one with the biggest margin.**

Data Points



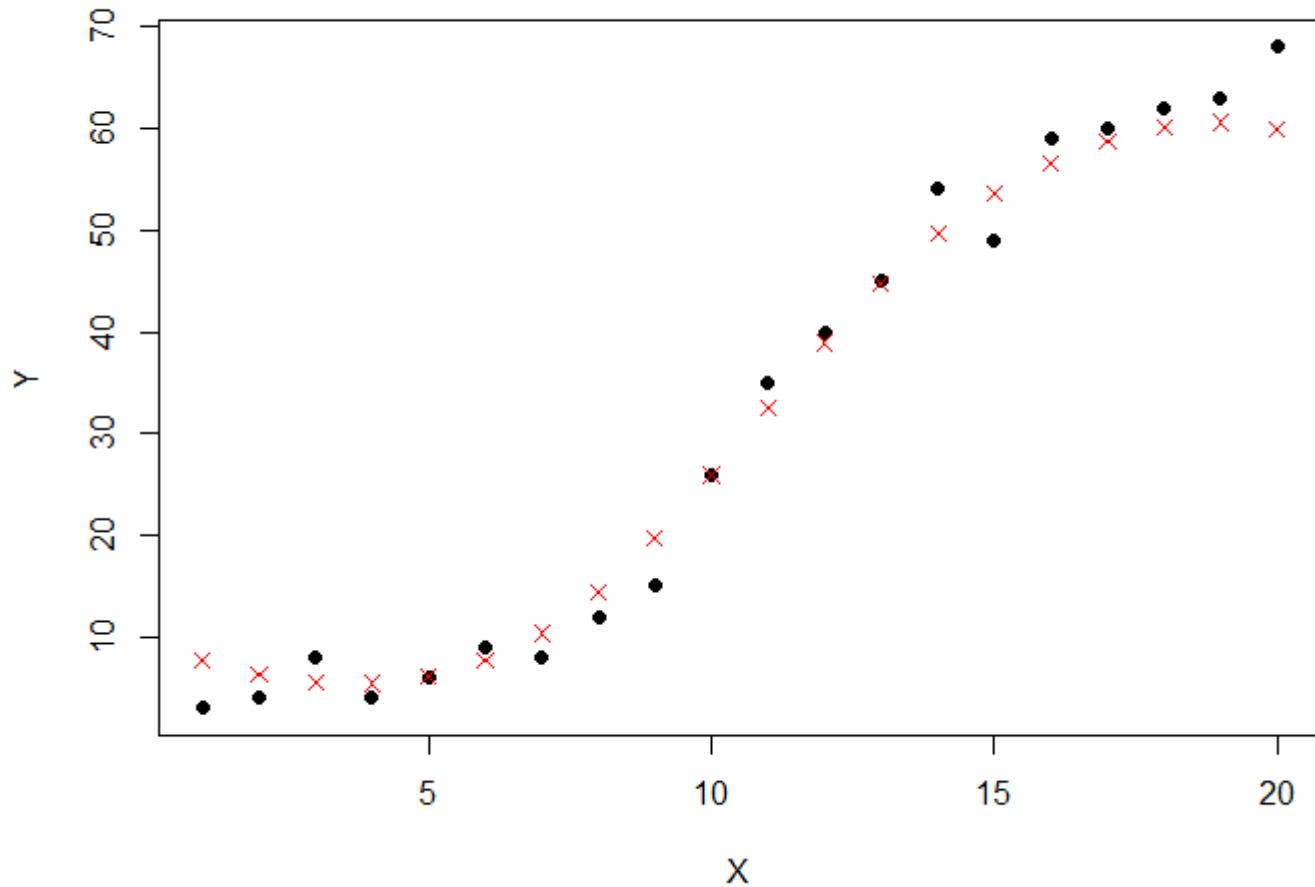
Linear Fit



Linear Fit

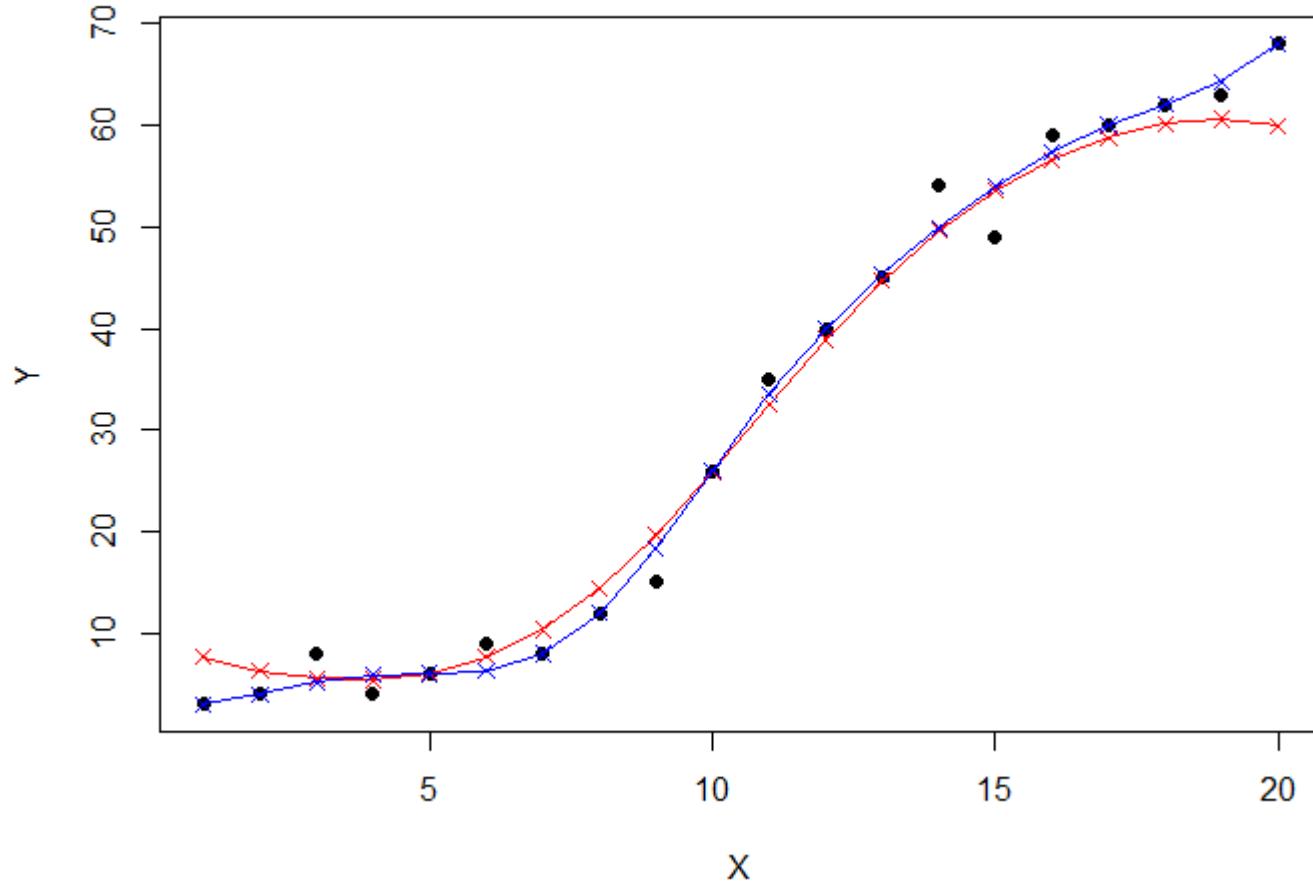
- rmse <- function(error)
- {
- sqrt(mean(error^2))
- }
-
- error <- model\$residuals # same as data\$Y - predictedY
- predictionRMSE <- rmse(error) # 5.703778
- How to Improve?

SVM Regression



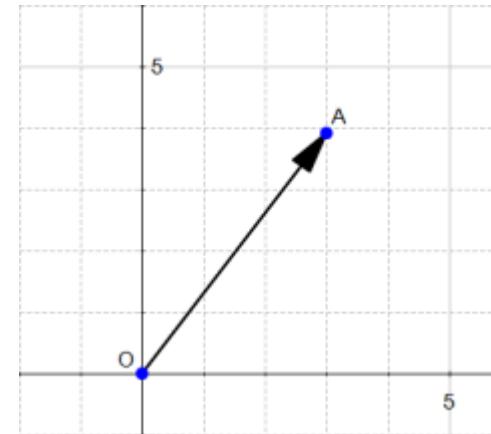
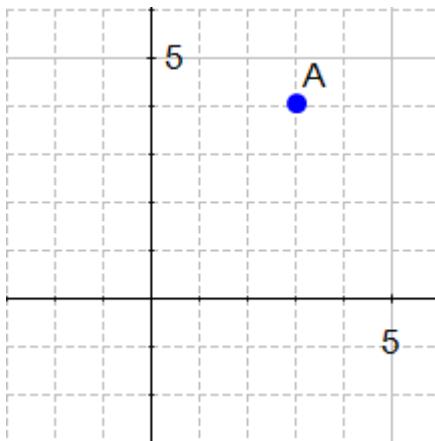
```
# !\\ this time svrModel$residuals is not the  
# same as data$Y - predictedY  
# so we compute the error like this  
error <- data$Y - predictedY  
svrPredictionRMSE <- rmse(error) #  
3.157061
```

Improved SVM with eps factor adjustment



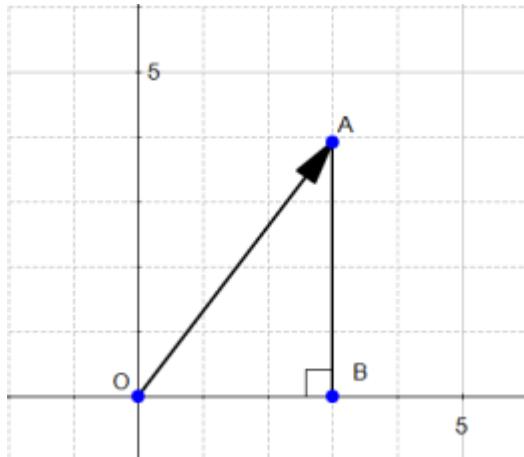
What is a vector?

- *A vector is an object that has both a magnitude and a direction.*
- If we define a point $A(3,4)$ in R^2 we can plot it like this.
- Definition: Any point $x=(x_1,x_2), x \neq 0$, in R^2 specifies a vector in the plane, namely the vector starting at the origin and ending at x .
- If we say that the point at the origin is the point $O(0,0)$ then the vector above is the vector $OA\rightarrow$. We could also give it an arbitrary name such as u .



The magnitude

- The magnitude or length of a vector x is written $\|x\|$ and is called its norm.
- For our vector $OA \rightarrow$, $\|OA\|$ is the length of the segment OA



$$OA^2 = OB^2 + AB^2$$

$$OA^2 = 3^2 + 4^2$$

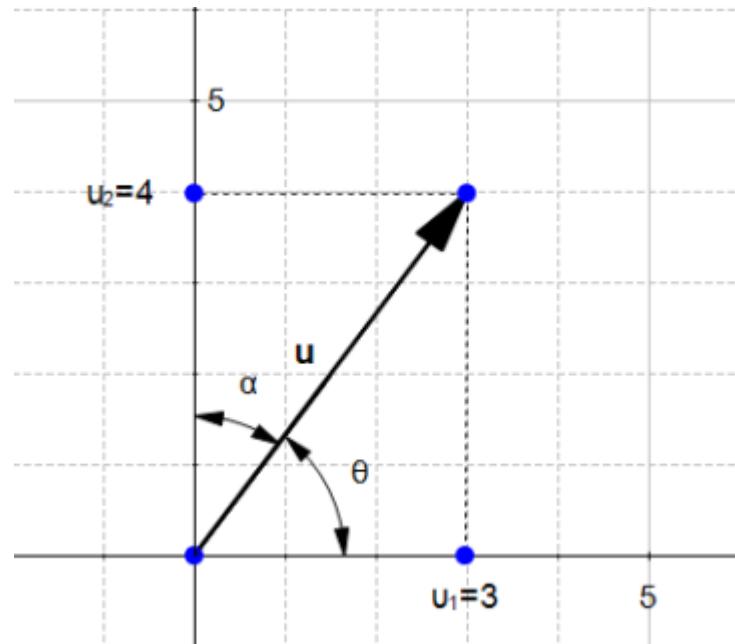
$$OA^2 = 25$$

$$OA = \sqrt{25}$$

$$\|OA\| = OA = 5$$

The direction

- The direction is the second component of a vector.
- Definition : The direction of a vector $u(u_1, u_2)$ is the vector $w(u_1\|u\|, u_2\|u\|)$

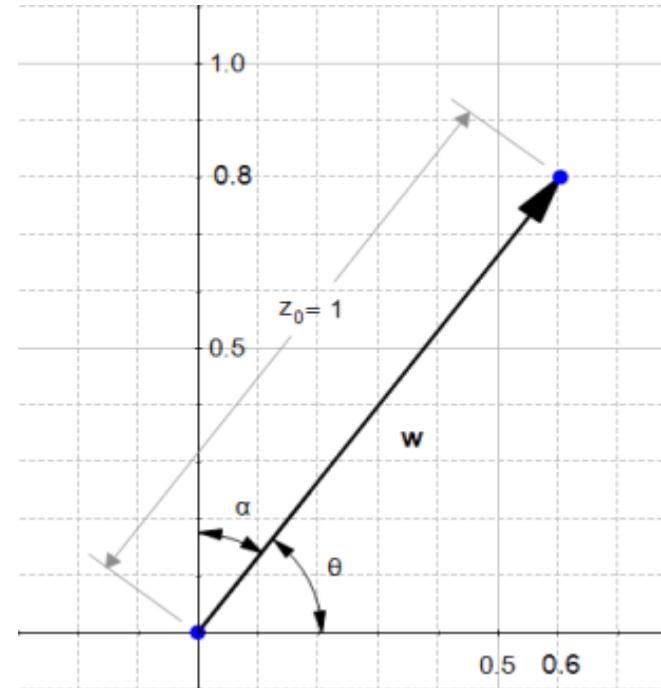


Computing the direction vector

- The direction of $u(3,4)$ is the vector $w(0.6,0.8)$

$$\cos(\theta) = \frac{u_1}{\|u\|} = \frac{3}{5} = 0.6$$

$$\cos(\alpha) = \frac{u_2}{\|u\|} = \frac{4}{5} = 0.8$$



The sum of two vectors

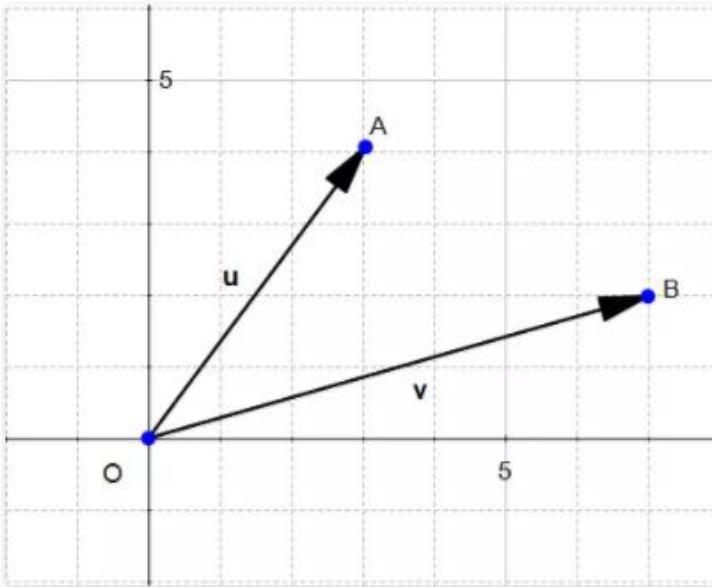
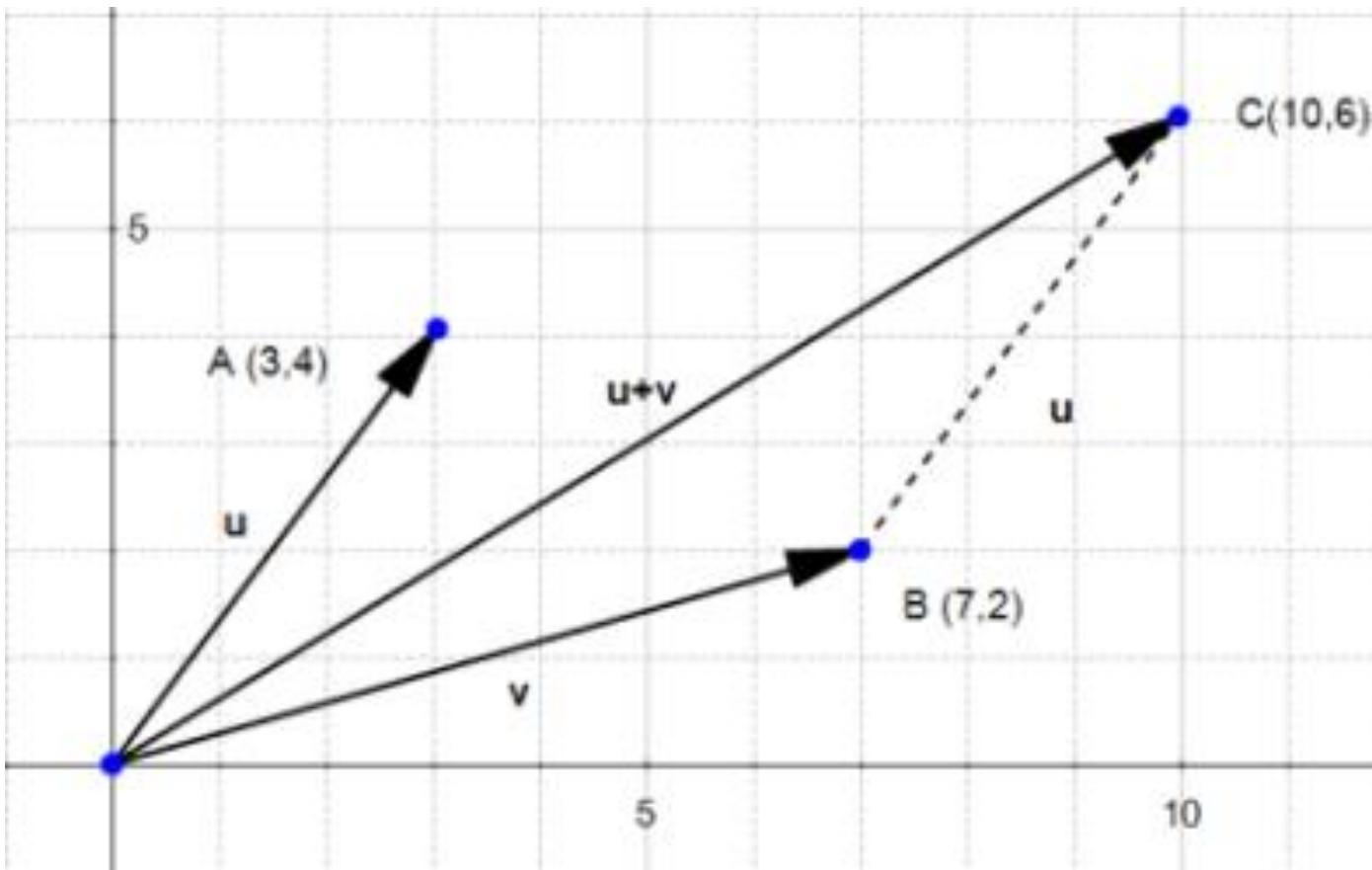


Figure 6: two vectors u and v

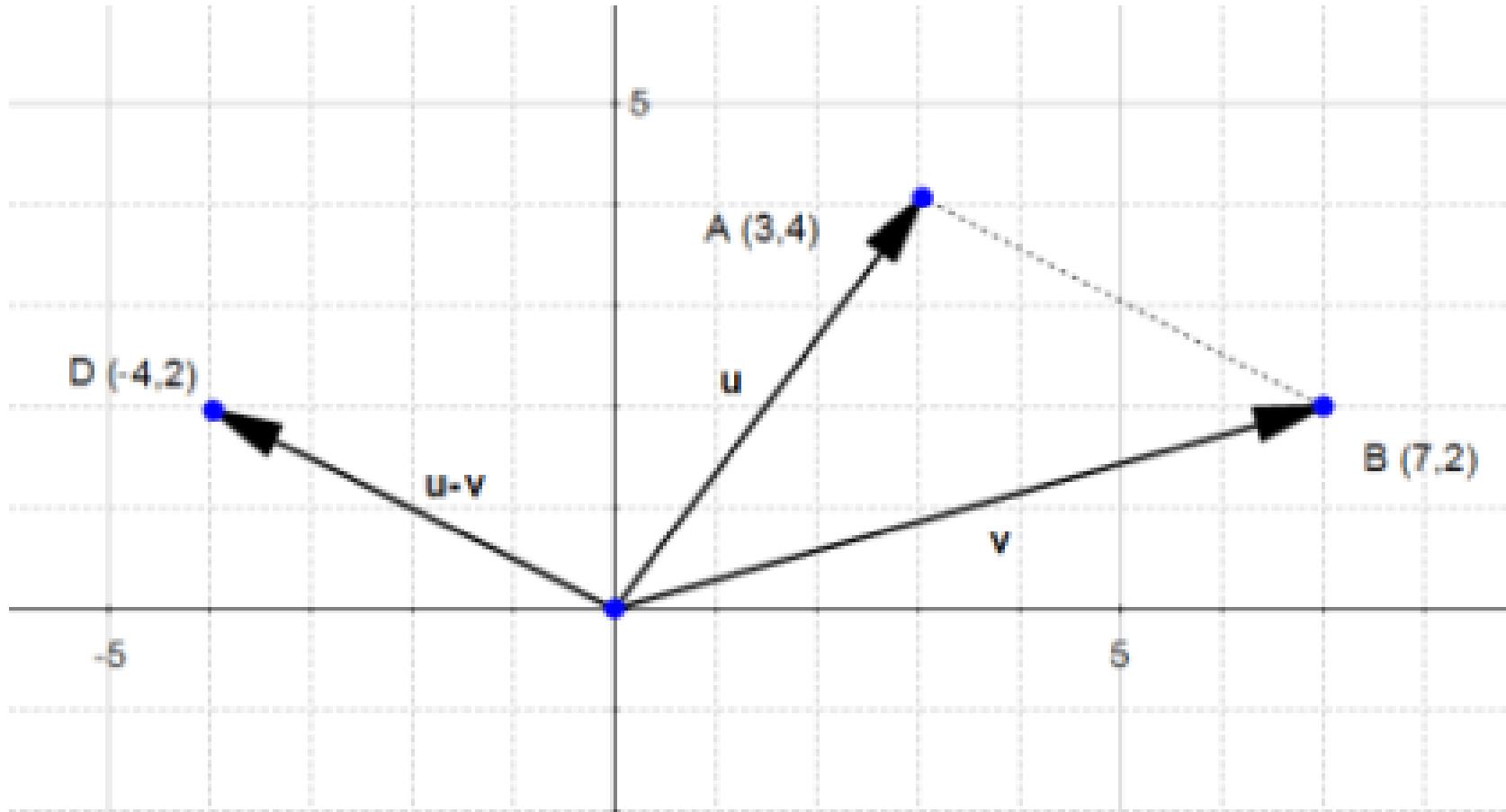
Given two vectors $\mathbf{u}(u_1, u_2)$ and $\mathbf{v}(v_1, v_2)$ then :

$$\mathbf{u} + \mathbf{v} = (u_1 + v_1, u_2 + v_2)$$

Third Vector

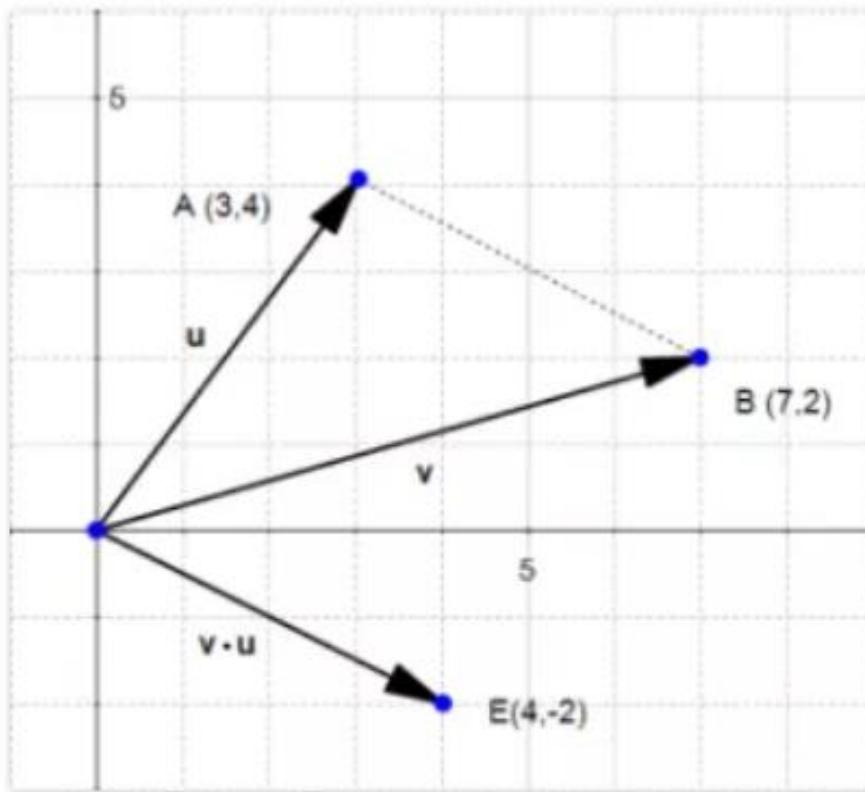


The difference between two vectors



The difference between two vectors

$$\mathbf{v} - \mathbf{u} = (v_1 - u_1, v_2 - u_2)$$



The dot product

- *Definition: Geometrically, it is the product of the Euclidian magnitudes of the two vectors and the cosine of the angle between them*

Which means if we have two vectors \mathbf{x} and \mathbf{y} and there is an angle θ (theta) between them, their dot product is :

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\theta)$$

Why ?

To understand let's look at the problem geometrically.

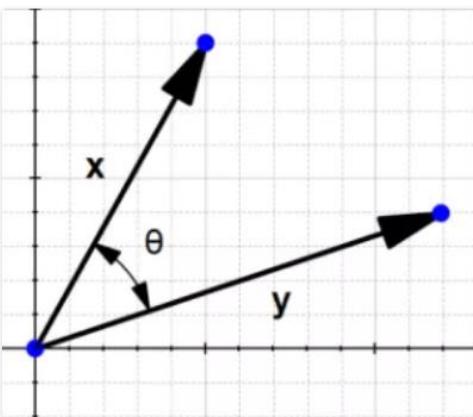
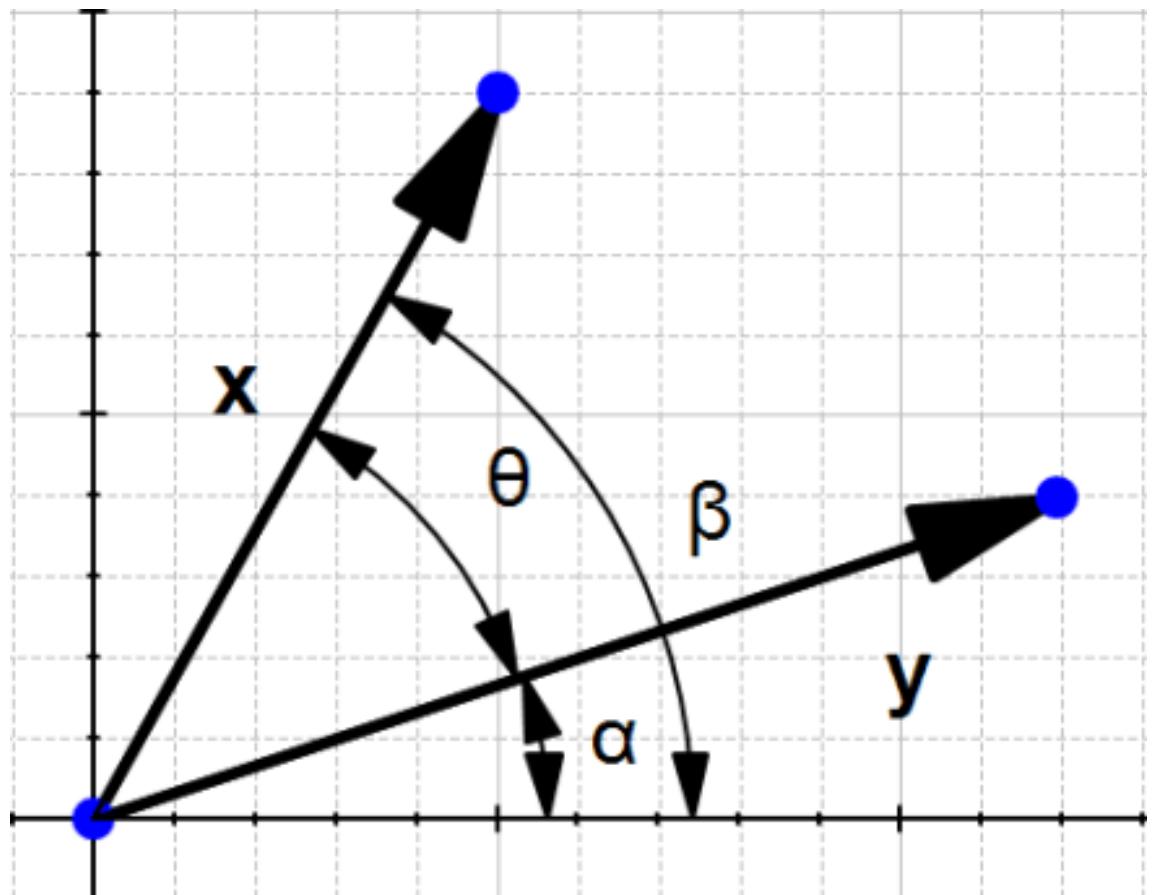


Figure 12



- We can see that
- $\theta = \beta - \alpha$
- So computing $\cos(\theta)$ is like computing $\cos(\beta - \alpha)$
- There is a special formula called the difference identity for cosine which says that:
- $\cos(\beta - \alpha) = \cos(\beta)\cos(\alpha) + \sin(\beta)\sin(\alpha)$

Let's use this formula!

$$\cos(\beta) = \frac{\text{adjacent}}{\text{hypotenuse}} = \frac{x_1}{\|x\|}$$

$$\sin(\beta) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{x_2}{\|x\|}$$

$$\cos(\alpha) = \frac{\text{adjacent}}{\text{hypotenuse}} = \frac{y_1}{\|y\|}$$

$$\sin(\alpha) = \frac{\text{opposite}}{\text{hypotenuse}} = \frac{y_2}{\|y\|}$$

So if we replace each term

$$\cos(\theta) = \cos(\beta - \alpha) = \cos(\beta)\cos(\alpha) + \sin(\beta)\sin(\alpha)$$

$$\cos(\theta) = \frac{x_1}{\|x\|} \frac{y_1}{\|y\|} + \frac{x_2}{\|x\|} \frac{y_2}{\|y\|}$$

$$\cos(\theta) = \frac{x_1 y_1 + x_2 y_2}{\|x\| \|y\|}$$

If we multiply both sides by $\|x\| \|y\|$ we get:

$$\|x\| \|y\| \cos(\theta) = x_1 y_1 + x_2 y_2$$

Which is the same as :

$$\|x\| \|y\| \cos(\theta) = \mathbf{x} \cdot \mathbf{y}$$

We just found the geometric definition of the dot product !

Eventually from the two last equations we can see that :

$$\mathbf{x} \cdot \mathbf{y} = x_1 y_1 + x_2 y_2 = \sum_{i=1}^2 (x_i y_i)$$

This is the algebraic definition of the dot product !

The SVM hyperplane

- Understanding the equation of the hyperplane
- You probably learnt that an equation of a line is : $y=ax+b$. However when reading about hyperplane, you will often find that the equation of an hyperplane is defined by :

$$\mathbf{w}^T \mathbf{x} = 0$$

The SVM hyperplane

- How does these two forms relate ?
- In the hyperplane equation you can see that the name of the variables are in bold. Which means that they are vectors !
- Moreover, $w^T x$ is how we compute the inner product of two vectors, and if you recall, the inner product is just another name for the dot product !

The SVM hyperplane

Note that

$$y = ax + b$$

is the same thing as

$$y - ax - b = 0$$

Given two vectors $\mathbf{w} \begin{pmatrix} -b \\ -a \\ 1 \end{pmatrix}$ and $\mathbf{x} \begin{pmatrix} 1 \\ x \\ y \end{pmatrix}$

$$\mathbf{w}^T \mathbf{x} = -b \times (1) + (-a) \times x + 1 \times y$$

$$\mathbf{w}^T \mathbf{x} = y - ax - b$$

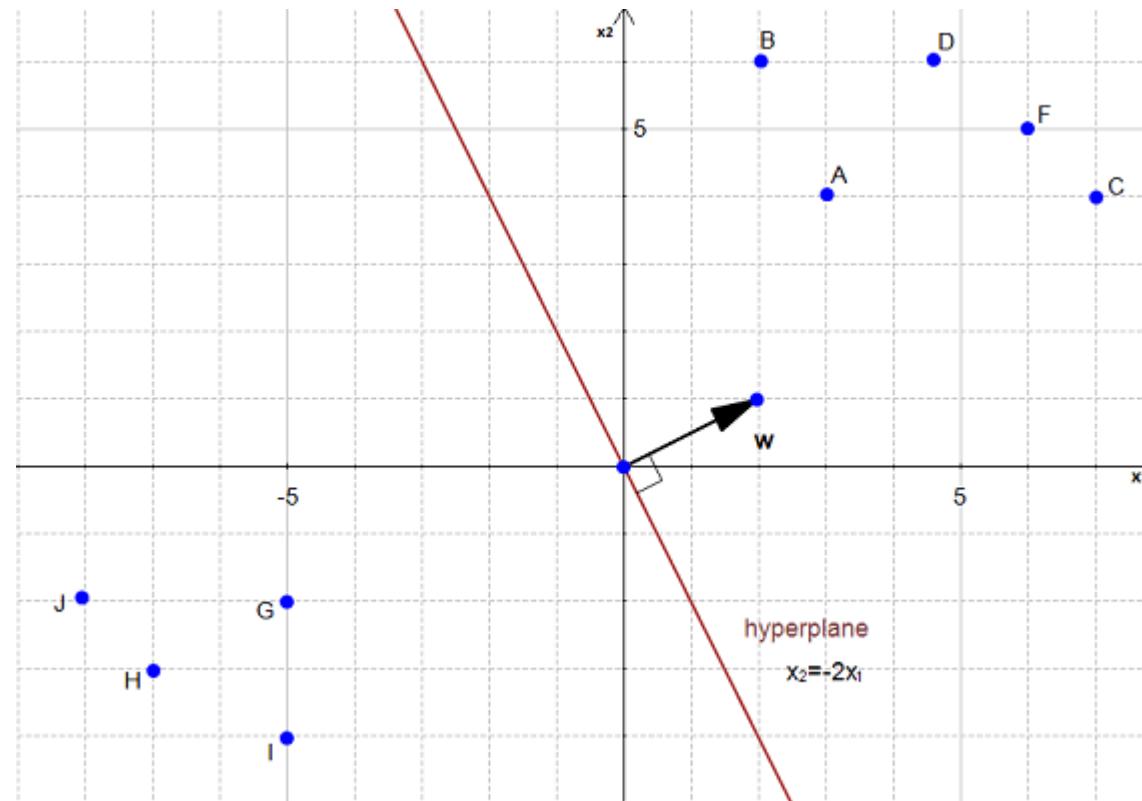
The two equations are just different ways of expressing the same thing.

It is interesting to note that w_0 is $-b$, which means that this value determines the intersection of the line with the vertical axis.

Why do we use the hyperplane equation $\mathbf{w}^T \mathbf{x}$ instead of $y = ax + b$?

The SVM hyperplane

- For two reasons:
- it is easier to work in more than two dimensions with this notation,
- the vector w will always be normal to the hyperplane.



SVM hyperplane

- Our goal is to find the distance between the point $A(3,4)$ and the hyperplane.
- We start with two vectors, $w=(2,1)$ which is normal to the hyperplane, and $a=(3,4)$ which is the vector between the origin and A .

$$\|w\| = \sqrt{2^2 + 1^2} = \sqrt{5}$$

Let the vector \mathbf{u} be the direction of \mathbf{w}

$$\mathbf{u} = \left(\frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right)$$

\mathbf{p} is the orthogonal projection of \mathbf{a} onto \mathbf{w} so :

$$\mathbf{p} = (\mathbf{u} \cdot \mathbf{a})\mathbf{u}$$

$$\mathbf{p} = \left(3 \times \frac{2}{\sqrt{5}} + 4 \times \frac{1}{\sqrt{5}} \right) \mathbf{u}$$

$$\mathbf{p} = \left(\frac{6}{\sqrt{5}} + \frac{4}{\sqrt{5}} \right) \mathbf{u}$$

$$\mathbf{p} = \frac{10}{\sqrt{5}} \mathbf{u}$$

$$\mathbf{p} = \left(\frac{10}{\sqrt{5}} \times \frac{2}{\sqrt{5}}, \frac{10}{\sqrt{5}} \times \frac{1}{\sqrt{5}} \right)$$

$$\mathbf{p} = \left(\frac{20}{5}, \frac{10}{5} \right)$$

$$\mathbf{p} = (4, 2)$$

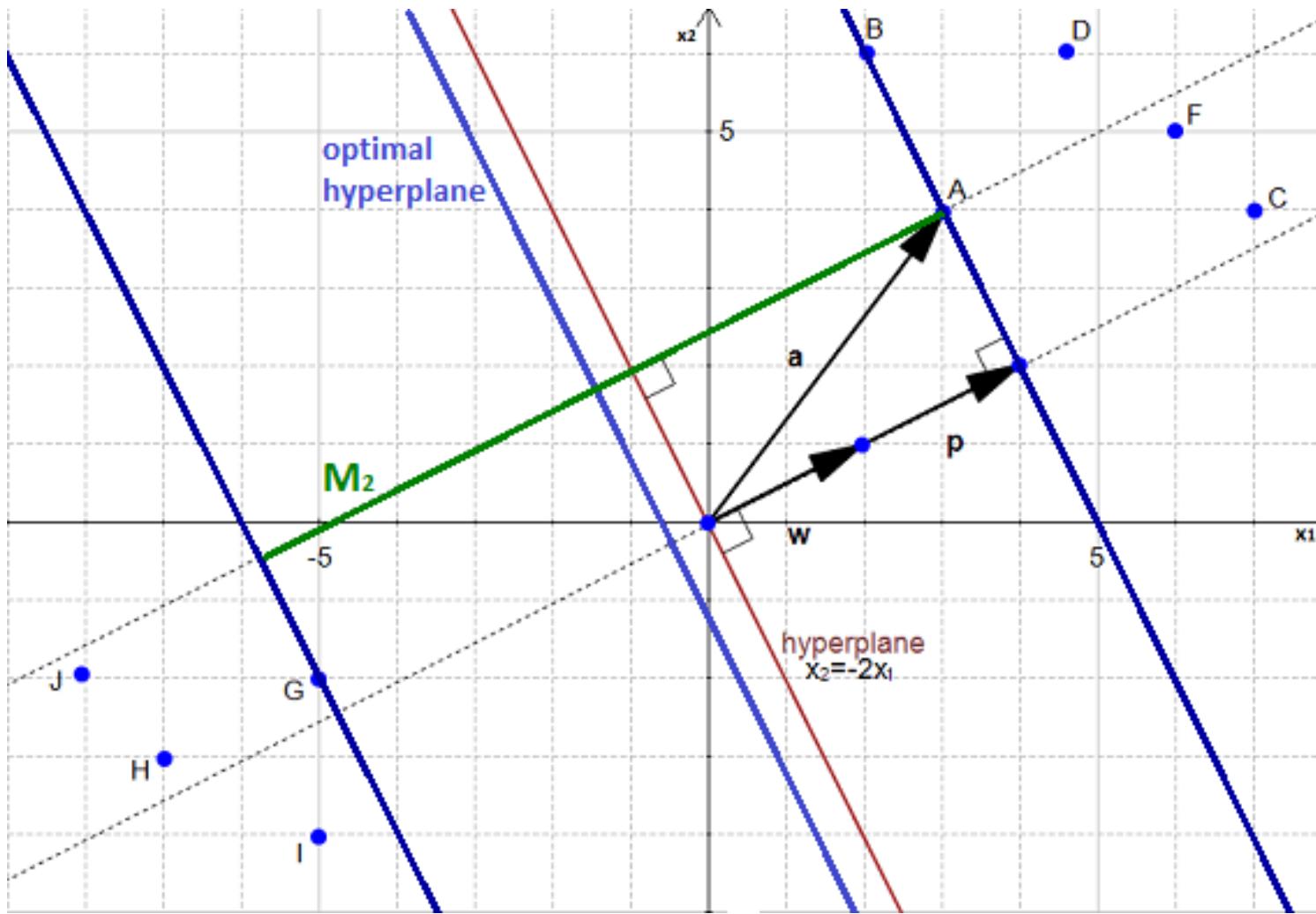
$$\|p\| = \sqrt{4^2 + 2^2} = 2\sqrt{5}$$

SVM Hyperplane

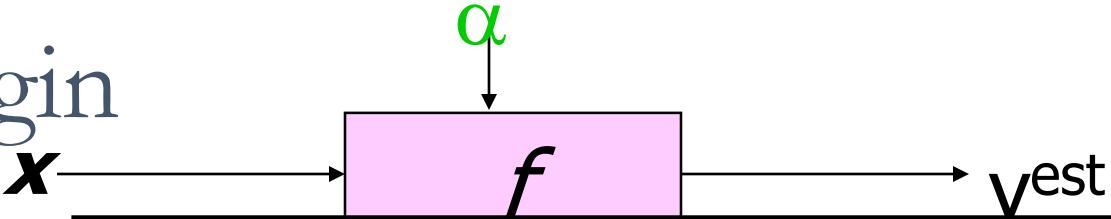
- Compute the margin of the hyperplane
- Now that we have the distance $\|p\|$ between A and the hyperplane, the margin is defined by :

$$\text{margin} = 2\|p\| = 4\sqrt{5}$$

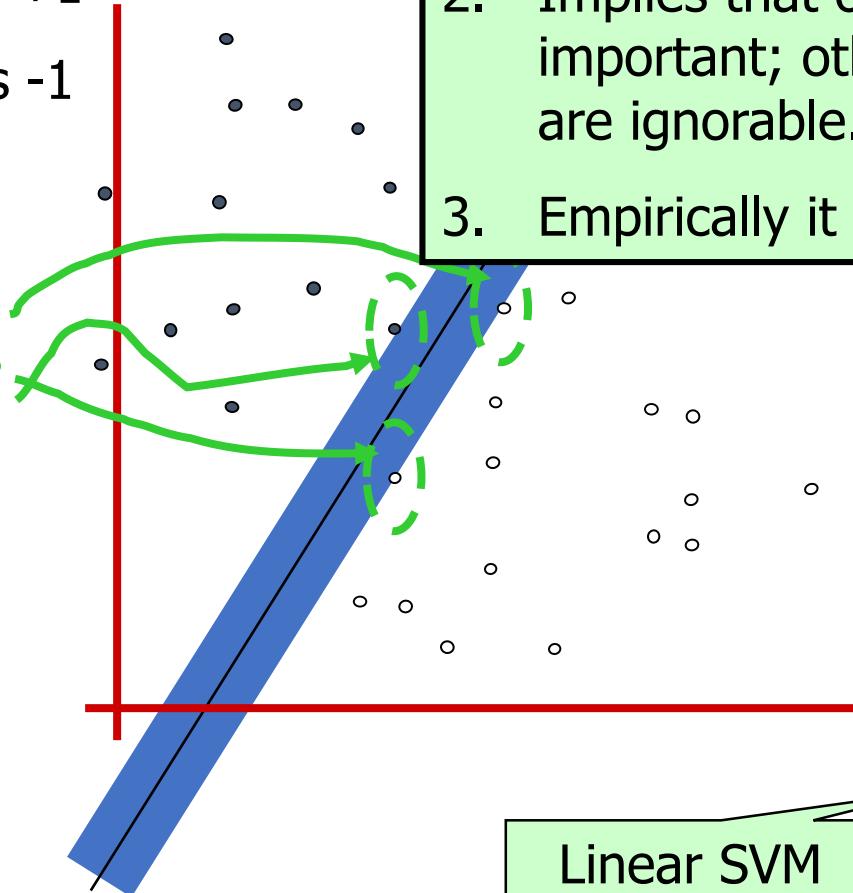
Optimal hyperplane



Maximum Margin



- denotes +1
 - denotes -1
- Support Vectors** are those datapoints that the margin pushes up against



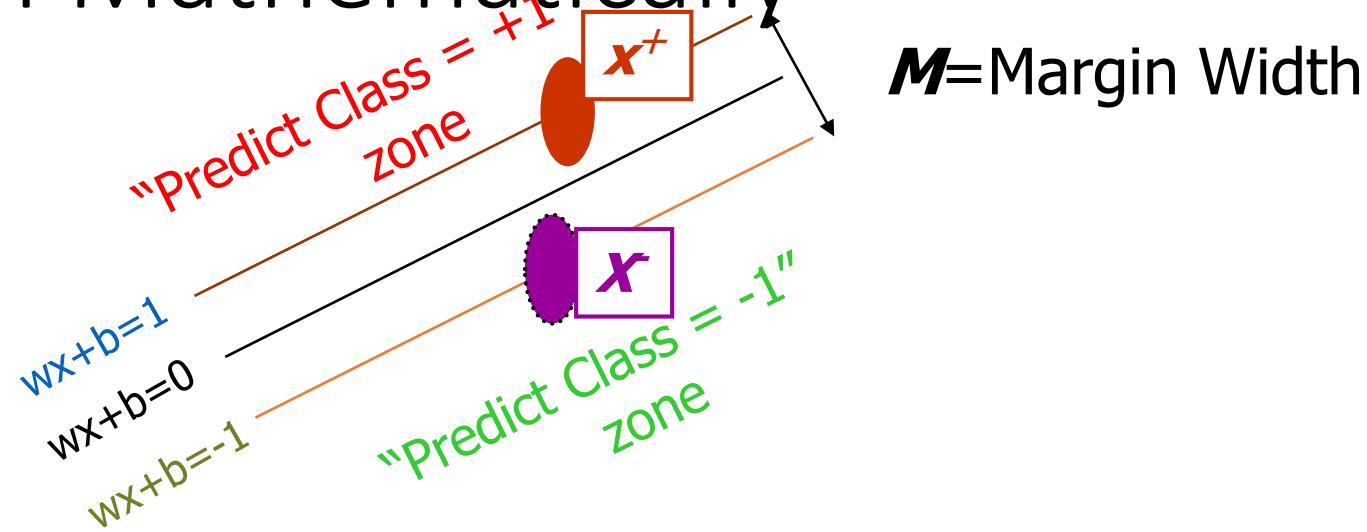
Linear SVM

linear classifier
with the, um,
maximum margin.

This is the
simplest kind of
SVM (Called an
LSVM)

1. Maximizing the margin is good according to intuition and PAC theory
2. Implies that only support vectors are important; other training examples are ignorable.
3. Empirically it works very very well.

Linear SVM Mathematically



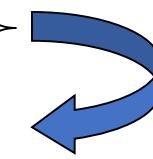
What we know:

- $w \cdot x^+ + b = +1$
- $w \cdot x^- + b = -1$
- $w \cdot (x^+ - x^-) = 2$

$$M = \frac{(x^+ - x^-) \cdot w}{|w|} = \frac{2}{|w|}$$

Linear SVM Mathematically

- Goal: 1) Correctly classify all training data

$$\begin{aligned} w\mathbf{x}_i + b &\geq 1 & \text{if } y_i = +1 \\ w\mathbf{x}_i + b &\leq -1 & \text{if } y_i = -1 \\ y_i(w\mathbf{x}_i + b) &\geq 1 \end{aligned} \quad \left. \begin{array}{l} \\ \\ \text{for all } i \end{array} \right\}$$


- 2) Maximize the Margin
same as minimize

$$M = \frac{1}{2} \frac{w^t w}{|w|}$$

- We can formulate a Quadratic Optimization Problem and solve for w and b

- Minimize $\Phi(w) = \frac{1}{2} w^t w$

- subject to $y_i(w\mathbf{x}_i + b) \geq 1 \quad \forall i$

Solving the Optimization Problem

Find \mathbf{w} and b such that

$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w}$ is minimized;

and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

- Need to optimize a *quadratic* function subject to *linear* constraints.
- Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.
- The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:

Find $\alpha_1 \dots \alpha_N$ such that

$\mathbf{Q}(\boldsymbol{\alpha}) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and

$$(1) \quad \sum \alpha_i y_i = 0$$

$$(2) \quad \alpha_i \geq 0 \text{ for all } \alpha_i$$

The Optimization Problem Solution

- The solution has the form:

$$\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i \quad b = y_k - \mathbf{w}^T \mathbf{x}_k \text{ for any } \mathbf{x}_k \text{ such that } \alpha_k \neq 0$$

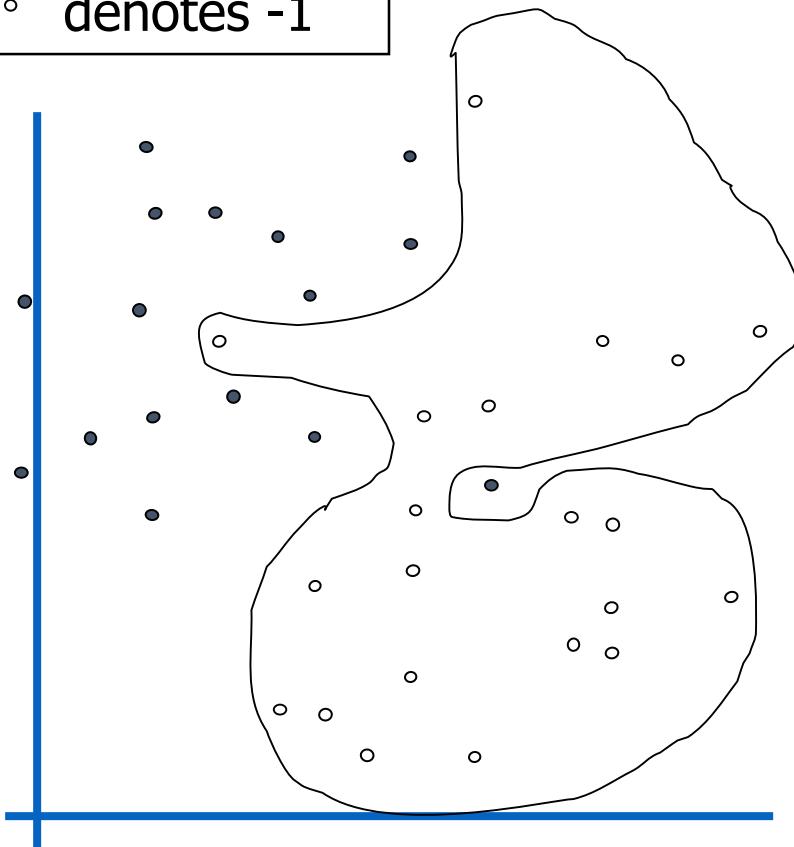
- Each non-zero α_i indicates that corresponding \mathbf{x}_i is a support vector.
- Then the classifying function will have the form:

$$f(\mathbf{x}) = \sum \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$$

- Notice that it relies on an *inner product* between the test point \mathbf{x} and the support vectors \mathbf{x}_i – we will return to this later.
- Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^T \mathbf{x}_j$ between all pairs of training points.

Dataset with noise

- denotes +1
- denotes -1

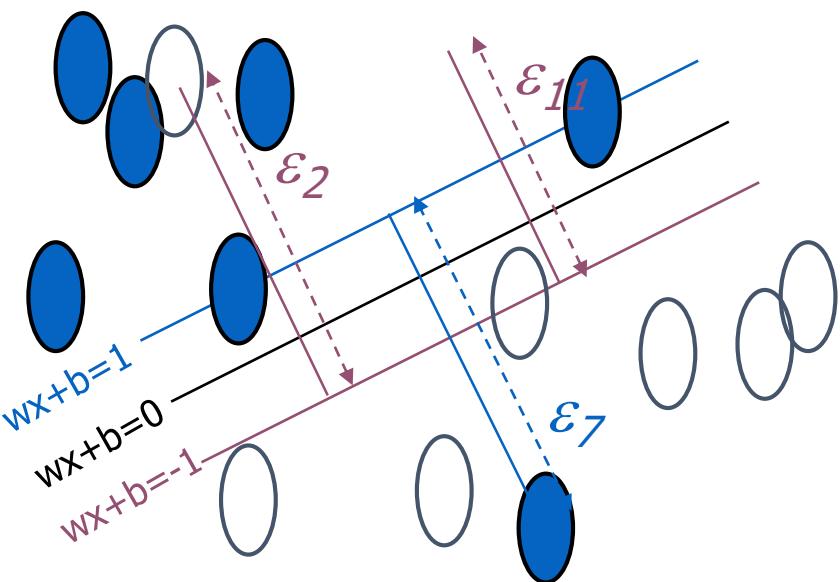


- **Hard Margin:** So far we require all data points be classified correctly
 - No training error
- **What if the training set is noisy?**
 - **Solution 1:** use very powerful kernels

OVERFITTING!

Soft Margin Classification

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples.



What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

Hard Margin v.s. Soft Margin

- **The old formulation:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

- **The new formulation incorporating slack variables:**

Find \mathbf{w} and b such that

$$\Phi(\mathbf{w}) = \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_i \text{ is minimized and for all } \{(\mathbf{x}_i, y_i)\}$$

$$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{and} \quad \xi_i \geq 0 \text{ for all } i$$

- **Parameter C can be viewed as a way to control overfitting.**

Linear SVMs: Overview

- The classifier is a *separating hyperplane*.
- Most “important” training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i .
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j$ is maximized and

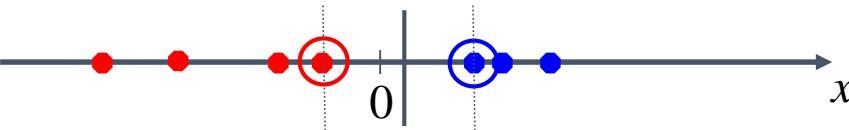
$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

$$f(x) = \sum \alpha_i y_i x_i^T x + b$$

Non-linear SVMs

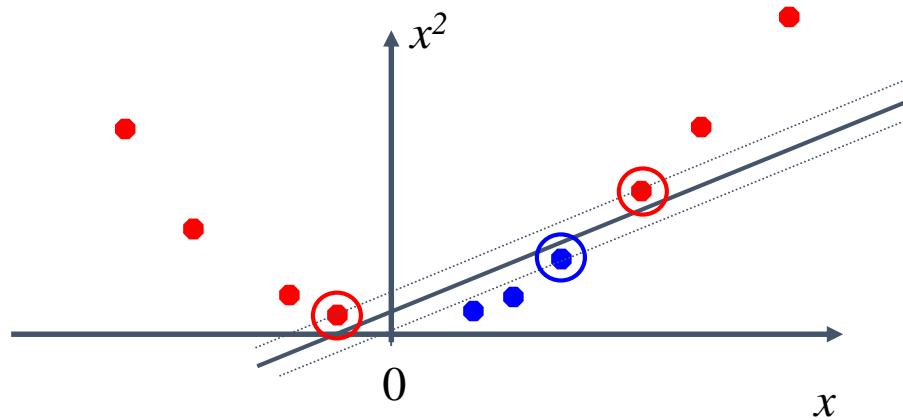
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?

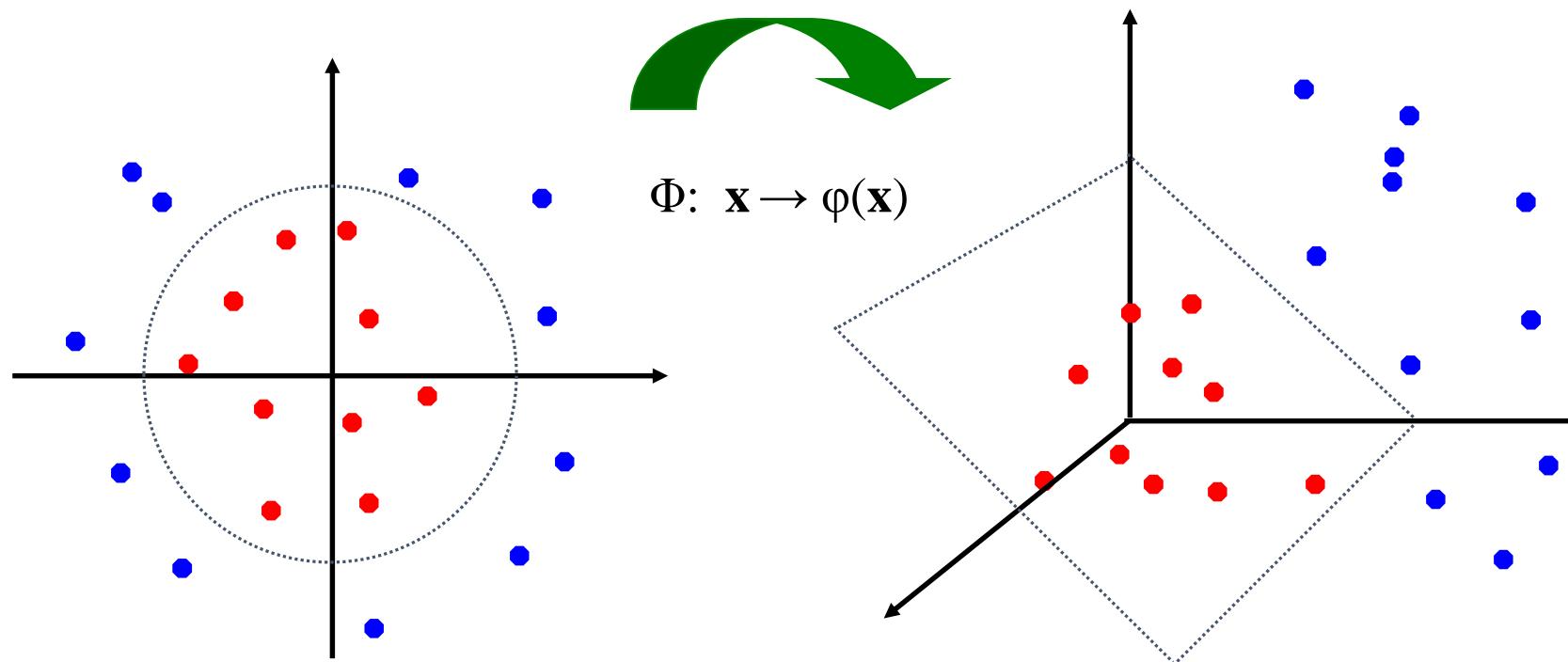


- How about... mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



The “Kernel Trick”

- The linear classifier relies on dot product between vectors $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation $\Phi: \mathbf{x} \rightarrow \phi(\mathbf{x})$, the dot product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.
- Example:

2-dimensional vectors $\mathbf{x} = [x_1 \ x_2]$; let $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$:

$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2, \\ &= 1 + x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{j1} x_{i2} x_{j2} + x_{i2}^2 x_{j2}^2 + 2 x_{i1} x_{j1} + 2 x_{i2} x_{j2} \\ &= [1 \ x_{i1}^2 \ \sqrt{2} x_{i1} x_{i2} \ x_{i2}^2 \ \sqrt{2} x_{i1} \ \sqrt{2} x_{i2}]^T [1 \ x_{j1}^2 \ \sqrt{2} x_{j1} x_{j2} \ x_{j2}^2 \ \sqrt{2} x_{j1} \ \sqrt{2} x_{j2}] \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j), \quad \text{where } \phi(\mathbf{x}) = [1 \ x_1^2 \ \sqrt{2} x_1 x_2 \ x_2^2 \ \sqrt{2} x_1 \ \sqrt{2} x_2] \end{aligned}$$

What Functions are Kernels?

- For some functions $K(\mathbf{x}_i, \mathbf{x}_j)$ checking that $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem:
Every semi-positive definite symmetric function is a kernel
- Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

$K =$

$K(\mathbf{x}_1, \mathbf{x}_1)$	$K(\mathbf{x}_1, \mathbf{x}_2)$	$K(\mathbf{x}_1, \mathbf{x}_3)$...	$K(\mathbf{x}_1, \mathbf{x}_N)$
$K(\mathbf{x}_2, \mathbf{x}_1)$	$K(\mathbf{x}_2, \mathbf{x}_2)$	$K(\mathbf{x}_2, \mathbf{x}_3)$		$K(\mathbf{x}_2, \mathbf{x}_N)$
...
$K(\mathbf{x}_N, \mathbf{x}_1)$	$K(\mathbf{x}_N, \mathbf{x}_2)$	$K(\mathbf{x}_N, \mathbf{x}_3)$...	$K(\mathbf{x}_N, \mathbf{x}_N)$

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
- Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):
$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$
- Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^\top \mathbf{x}_j + \beta_1)$

Non-linear SVMs Mathematically

- Dual problem formulation:

Find $\alpha_1 \dots \alpha_N$ such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j K(x_i, x_j)$ is maximized and

(1) $\sum \alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

- The solution is:

$$f(\mathbf{x}) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b$$

- Optimization techniques for finding α_i 's remain the same!

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**
 - only support vectors are used to specify the separating hyperplane
- **Ability to handle large feature spaces**
 - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property:** a simple convex optimization problem which is guaranteed to converge to a single global solution
- **Feature Selection**

SVM Applications

- SVM has been used successfully in many real-world problems
 - text (and hypertext) categorization
 - image classification
 - bioinformatics (Protein classification,
Cancer classification)
 - hand-written character recognition

Application 1: Cancer Classification

- High Dimensional

- $p > 1000$; $n < 100$

- Imbalanced

- less positive samples

$$K[x, x] = k(x, x) + \lambda \frac{n^+}{N}$$

- Many irrelevant features

- Noisy

SVM is sensitive to noisy (mis-labeled) data ☹

Patients	Genes			
	g-1	g-2	g-p
P-1				
p-2				
.....				
p-n				

FEATURE SELECTION

In the linear case,
 w_i^2 gives the ranking of dim i

Weakness of SVM

- It is sensitive to noise

- A relatively small number of mislabeled examples can dramatically decrease the performance

- It only considers two classes

- how to do multi-class classification with SVM?

- Answer:

- 1) with output arity m, learn m SVM's

- SVM 1 learns "Output==1" vs "Output != 1"
 - SVM 2 learns "Output==2" vs "Output != 2"
 - :
 - SVM m learns "Output==m" vs "Output != m"

- 2) To predict the output for a new input, just predict with each SVM and find out which one puts the prediction the furthest into the positive region.

Application 2: Text Categorization

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content.
 - email filtering, web searching, sorting documents by topic, etc..
- A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

Representation of Text

IR's vector space model (aka bag-of-words representation)

- A doc is represented by a vector indexed by a pre-fixed set or dictionary of terms
- Values of an entry can be binary or weights

$$\phi_i(x) = \frac{\text{tf}_i \log (\text{idf}_i)}{\kappa},$$

- Normalization, stop words, word stems
- Doc $x \Rightarrow \varphi(x)$

Text Categorization using SVM

- The distance between two documents is $\phi(x) \cdot \phi(z)$
- $K(x,z) = \langle \phi(x) \cdot \phi(z) \rangle$ is a valid kernel, SVM can be used with $K(x,z)$ for discrimination.
- Why SVM?
 - High dimensional input space
 - Few irrelevant features (dense concept)
 - Sparse document vectors (sparse instances)
 - Text categorization problems are linearly separable

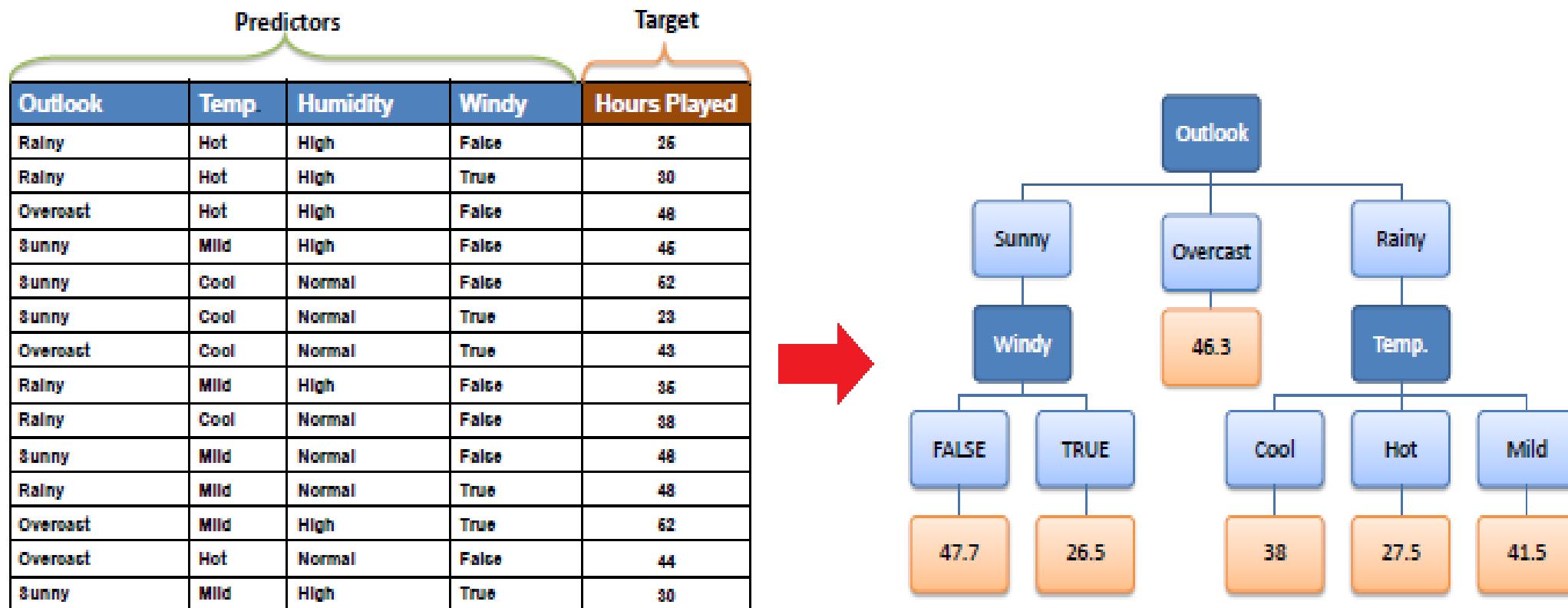
Some Issues

- **Choice of kernel**
 - Gaussian or polynomial kernel is default
 - if ineffective, more elaborate kernels are needed
 - domain experts can give assistance in formulating appropriate similarity measures
- **Choice of kernel parameters**
 - e.g. σ in Gaussian kernel
 - σ is the distance between closest points with different classifications
 - In the absence of reliable criteria, applications rely on the use of a validation set or cross-validation to set such parameters.
- **Optimization criterion** – Hard margin v.s. Soft margin
 - a lengthy series of experiments in which various parameters are tested

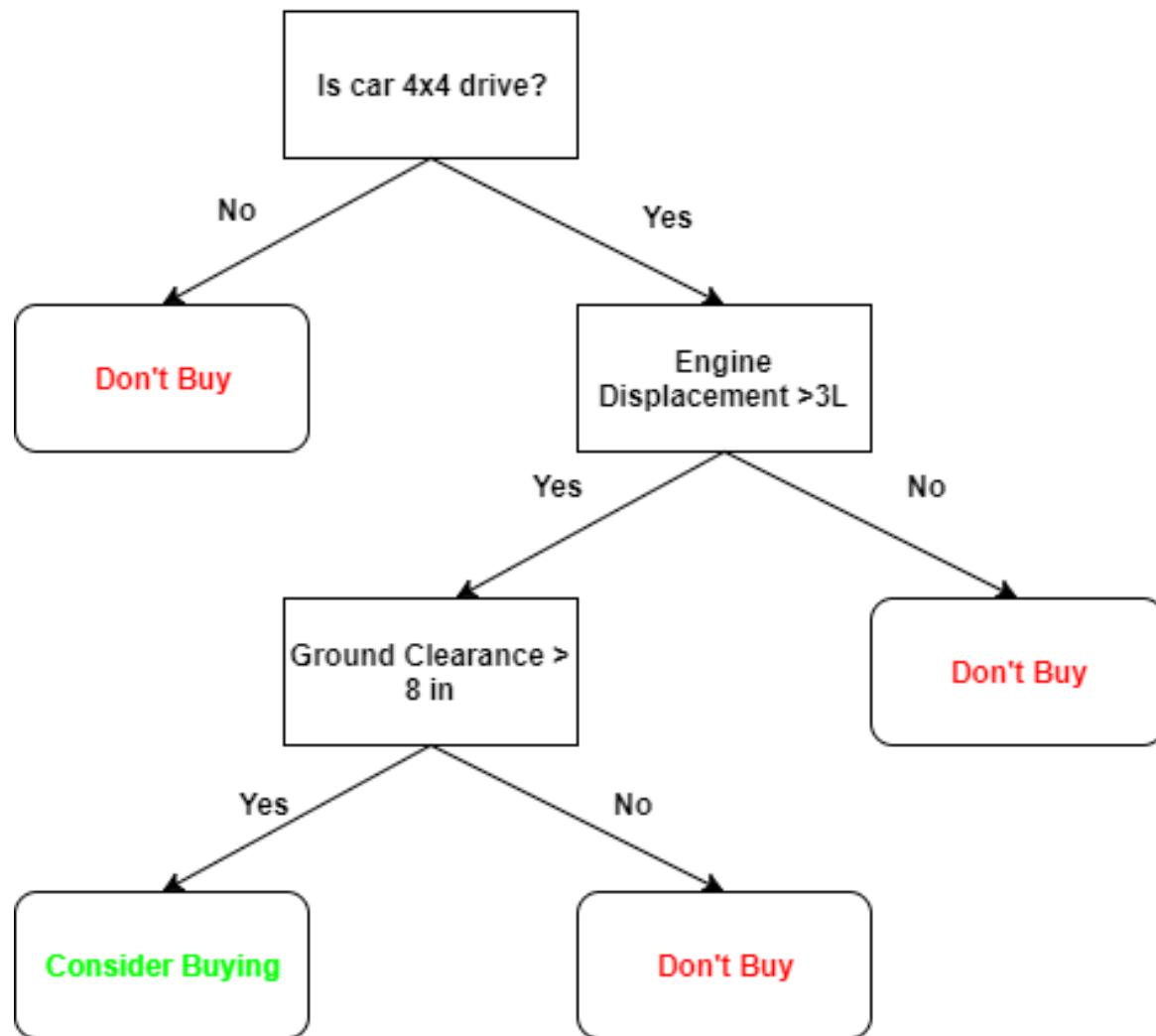
Decision Tree Regression

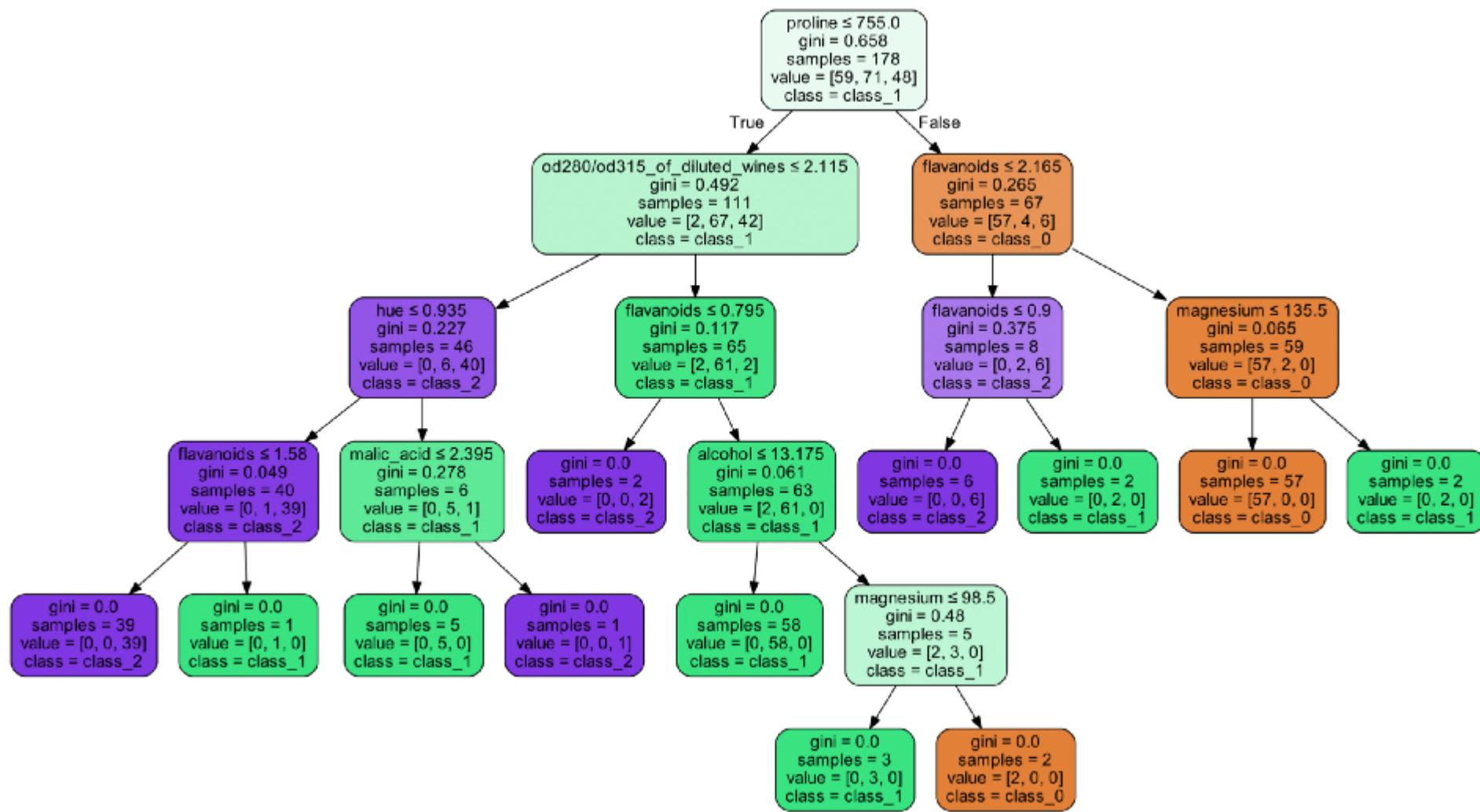
- Decision tree builds regression or classification models in the form of a tree structure.
- It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.
- The final result is a tree with **decision nodes** and **leaf nodes**.
- A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy), each representing values for the attribute tested.
- Leaf node (e.g., Hours Played) represents a decision on the numerical target.
- The topmost decision node in a tree which corresponds to the best predictor called **root node**.
- Decision trees can handle both categorical and numerical data.

Decision Tree Regression



Decision Tree for Classification and Prediction





Decision Tree Regression

- The core algorithm for building decision trees called **ID3** by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking.
- The ID3 algorithm can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*.

Decision Tree Regression

Entropy

Entropy $H(S)$ is a measure of the amount of uncertainty in the (data) set S (i.e. entropy characterizes the (data) set S).

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

Where,

- S – The current (data) set for which entropy is being calculated (changes every iteration of the ID3 algorithm)
- C – Set of classes in S $C=\{ \text{yes}, \text{no} \}$
- $p(c)$ – The proportion of the number of elements in class c to the number of elements in set S

When $H(S) = 0$, the set S is perfectly classified (i.e. all elements in S are of the same class).

In ID3, entropy is calculated for each remaining attribute. The attribute with the **smallest** entropy is used to split the set S on this iteration. The higher the entropy, the higher the potential to improve the classification here.

Decision Tree Regression

Information gain

Information gain $IG(A)$ is the measure of the difference in entropy from before to after the set S is split on an attribute A . In other words, how much uncertainty in S was reduced after splitting set S on attribute A .

$$IG(A, S) = H(S) - \sum_{t \in T} p(t)H(t)$$

Where,

- $H(S)$ – Entropy of set S
- T – The subsets created from splitting set S by attribute A such that $S = \bigcup_{t \in T} t$
- $p(t)$ – The proportion of the number of elements in t to the number of elements in set S
- $H(t)$ – Entropy of subset t

In ID3, information gain can be calculated (instead of entropy) for each remaining attribute. The attribute with the **largest** information gain is used to split the set S on this iteration.

Decision Tree Regression

1. compute the entropy for data-set
2. for every attribute/feature:
 1. calculate entropy for all categorical values
 2. take average information entropy for the current attribute
 3. calculate gain for the current attribute
3. pick the highest gain attribute.
4. Repeat until we get the tree we desired.

Decision Tree Regression

Compute the entropy for the weather data set:

$$H(S) = \sum_{c \in C} -p(c) \log_2 p(c)$$

$$C = \{\text{yes}, \text{no}\}$$

Out of 14 instances, 9 are classified as yes,
and 5 as no

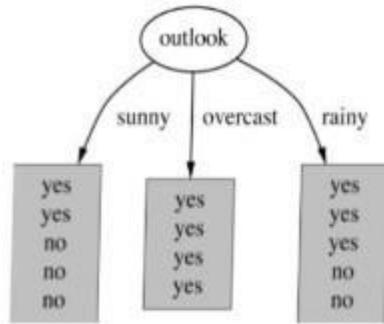
$$p_{\text{yes}} = -(9/14) * \log_2(9/14) = 0.41$$

$$p_{\text{no}} = -(5/14) * \log_2(5/14) = 0.53$$

$$H(S) = p_{\text{yes}} + p_{\text{no}} = 0.94$$

Decision Tree Regression

For every feature calculate the entropy and information gain



$$E(\text{Outlook}=\text{sunny}) = -\frac{2}{5} \log\left(\frac{2}{5}\right) - \frac{3}{5} \log\left(\frac{3}{5}\right) = 0.971$$

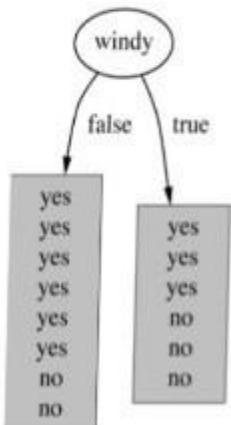
$$E(\text{Outlook}=\text{overcast}) = -1 \log(1) - 0 \log(0) = 0$$

$$E(\text{Outlook}=\text{rainy}) = -\frac{3}{5} \log\left(\frac{3}{5}\right) - \frac{2}{5} \log\left(\frac{2}{5}\right) = 0.971$$

Average Entropy information for Outlook

$$I(\text{Outlook}) = \frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 = 0.693$$

$$\text{Gain}(\text{Outlook}) = E(S) - I(\text{outlook}) = 0.94 - 0.693 = 0.247$$



$$E(\text{Windy}=\text{false}) = -\frac{6}{8} \log\left(\frac{6}{8}\right) - \frac{2}{8} \log\left(\frac{2}{8}\right) = 0.811$$

$$E(\text{Windy}=\text{true}) = -\frac{3}{6} \log\left(\frac{3}{6}\right) - \frac{3}{6} \log\left(\frac{3}{6}\right) = 1$$

Average entropy information for Windy

$$I(\text{Windy}) = \frac{8}{14} * 0.811 + \frac{6}{14} * 1 = 0.892$$

$$\text{Gain}(\text{Windy}) = E(S) - I(\text{Windy}) = 0.94 - 0.892 = 0.048$$

$$H(S, \text{Outlook})$$

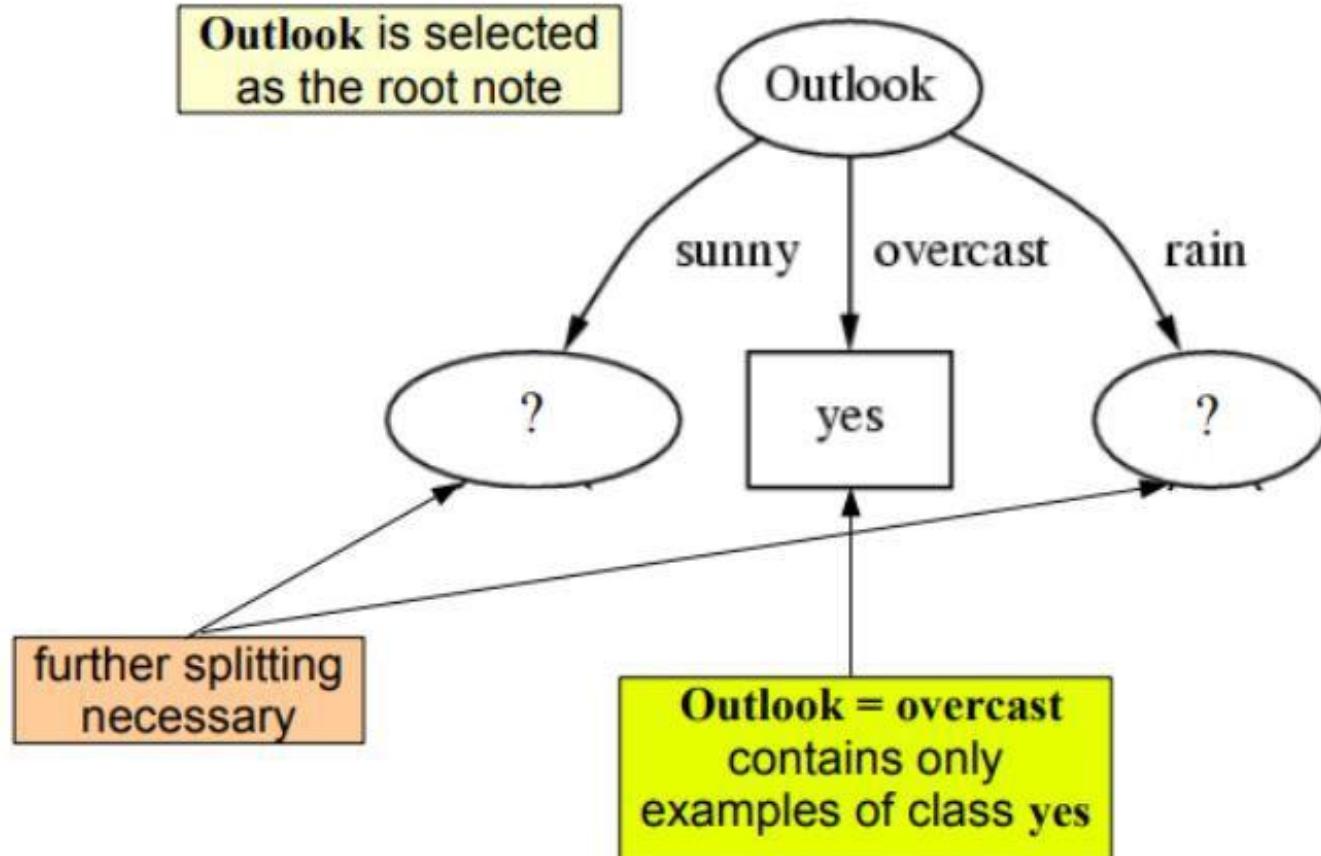
$$\sum_{t \in T} p(t) H(t)$$

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

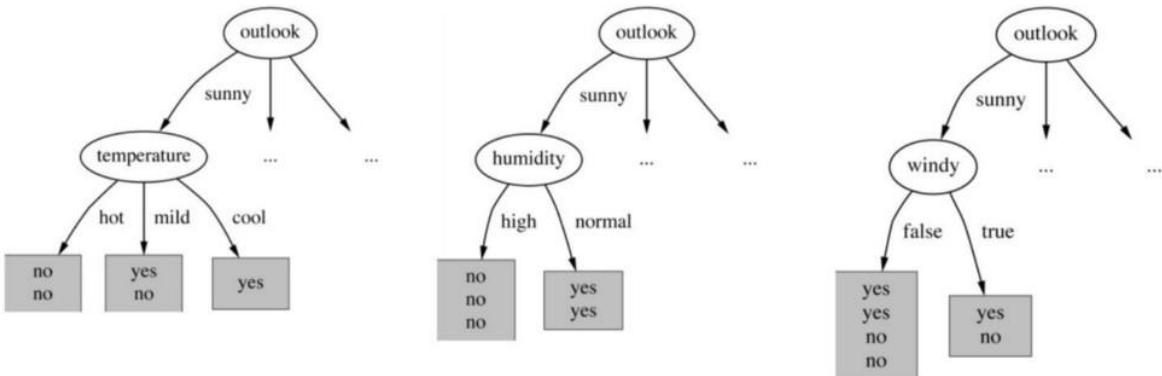
Pick the highest gain attribute.

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048

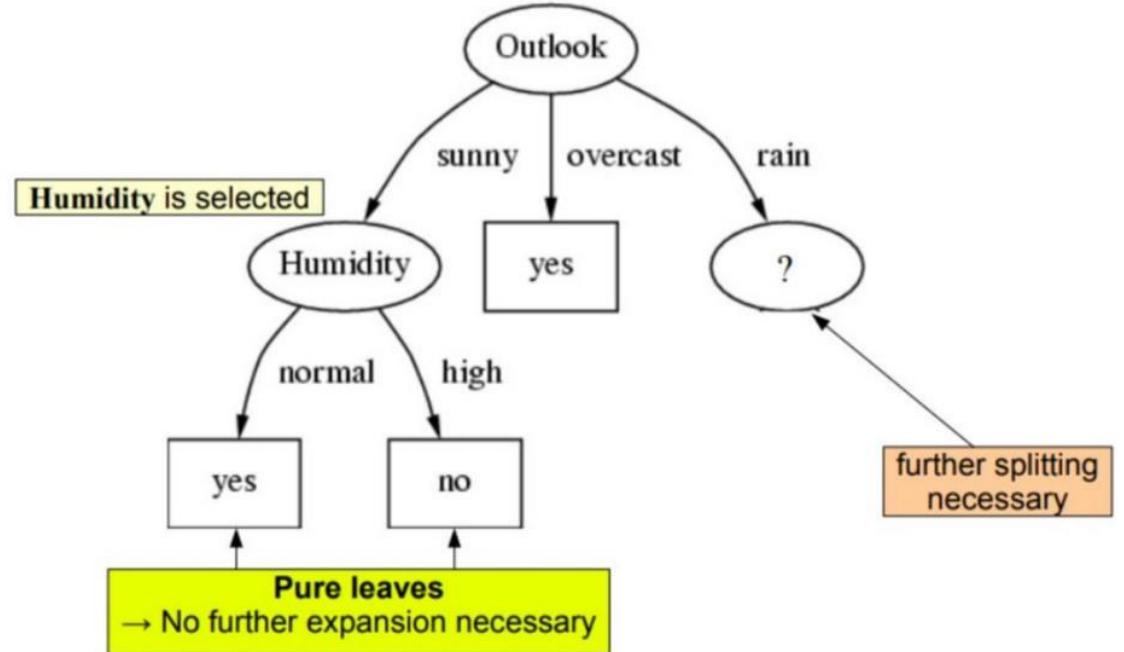
So our root node is Outlook.



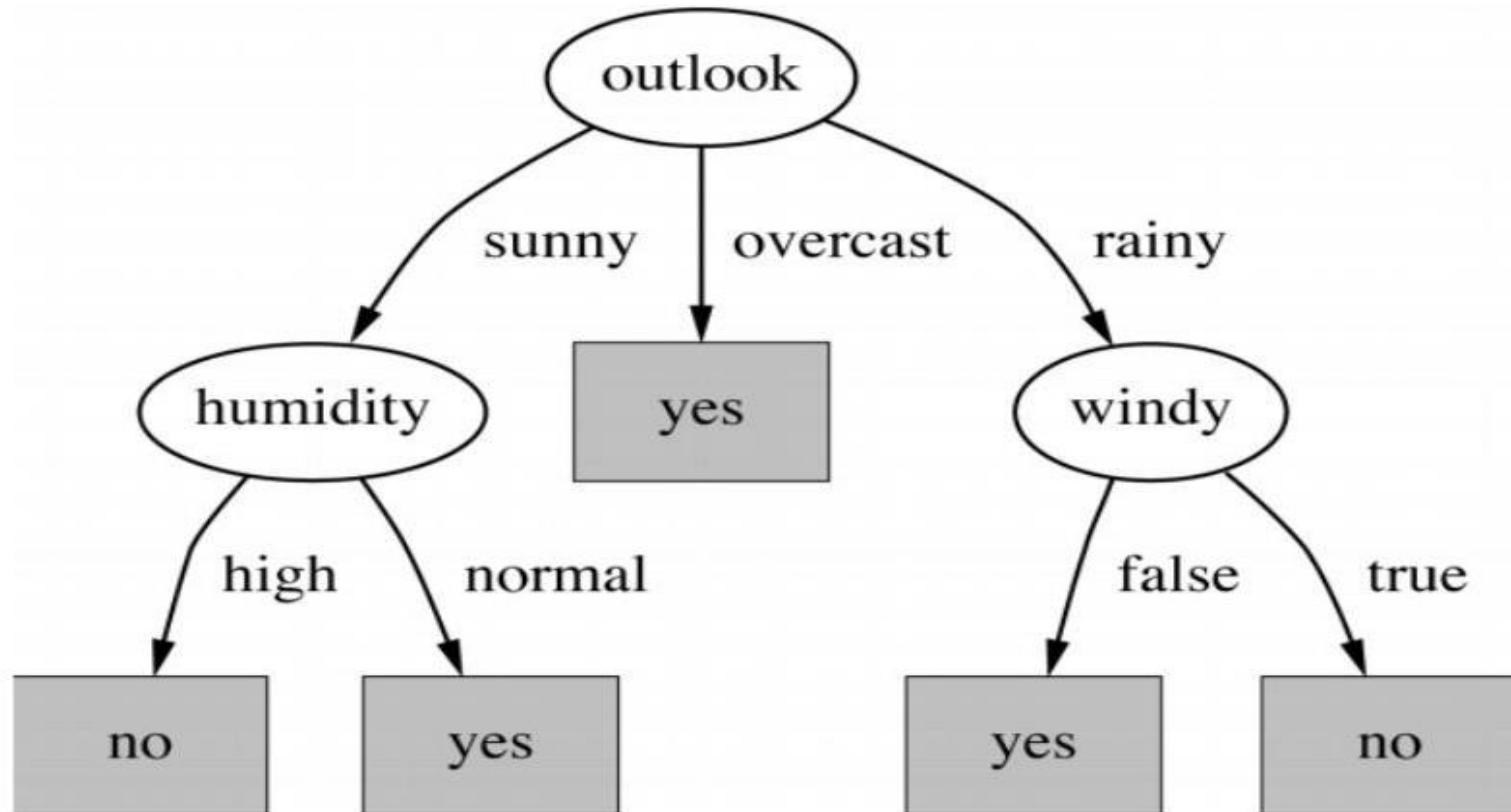
Repeat the same thing for sub-trees till we get the tree.



$$\begin{aligned}
 \text{Gain}(Temperature) &= 0.571 \text{ bits} \\
 \text{Gain}(Humidity) &= 0.971 \text{ bits} \\
 \text{Gain}(Windy) &= 0.020 \text{ bits}
 \end{aligned}
 \left. \right\} \text{Humidity is selected}$$



Final Decision Tree



Performance

- The Receiver Operating Characteristic (ROC) is a plot that can be used to determine the performance and robustness of a binary or multi-class classifier.
- The x-axis is the false positive rate (FPR) and the y-axis is the true positive rate (TPR).
- The ROC plot gives information about the true positive/negative rate and false positive/negative rate and something called the C-statistic or area under ROC curve (AUROC) for each class predicted by the classifier (there is one ROC for each class predicted by the classifier).

Performance

- The AUROC is defined as the probability that a randomly selected positive sample will have a higher prediction value than a randomly selected negative sample.
- *Interpretation of the AUC is easy: the higher the AUC, the better, with 0.50 indicating random performance and 1.00 denoting perfect performance.”*

GINI INDEX

GINI Index

$$Gini = \sum_{i \neq j} p(i)p(j)$$

i and j are levels of the target variable

GINI Index

- Example, banks and financial institutions grant credit facility after evaluating credit risk involved. Credit risk involved in credit decisions is evaluated using Credit Scorecard [Credit Score: What is it and how is it developed?]. Also, there are a few additional decisions involved in credit underwriting [Credit Underwriting: Minimize credit risk losses using Data Science and Analytics].
- Decision Tree: Non Technical Explanation
- The last 2 years of customer performance on meeting credit obligations is available with us. We want to understand the variable(s) explains high risk of customers who defaulted on a credit facility given to them.
- The sample has 24 customers. And for making it simple, only customer age and gender are considered. Age is a continuous variable and Gender is nominal variable.
- Input sample has 12 customers who have defaulted on the credit facility. So, default rate is 50%.
- We have an example in which input node, parent node, has equal number of Target variable values- “Yes” and “No”. Overall number of observations are 24.

GINI Index

Gender variable is considered to split the node. Gini Split value is calculated as below.

Gender	Target Value		Total
	No	Yes	
Female	6	2	8
Male	6	10	16
Total	12	12	24

Gini index for this node will be

$$= 1 - (1/2)^2 - (1/2)^2 \rightarrow 12/24 = 1/2$$

$$= 1 - 0.25 - 0.25$$

$$= 0.5$$

Now we want to split the code based on Gender Variable. After the split we will have following summary.

Now, let's calculate GINI index of the split using Gender variable.

$$\text{GINI}(s,t) = \text{GINI}(t) - P_L \text{GINI}(t_L) - P_R \text{GINI}(t_R)$$

$$\text{GINI}(t_L) = 1 - (6/8)^2 - (2/8)^2$$

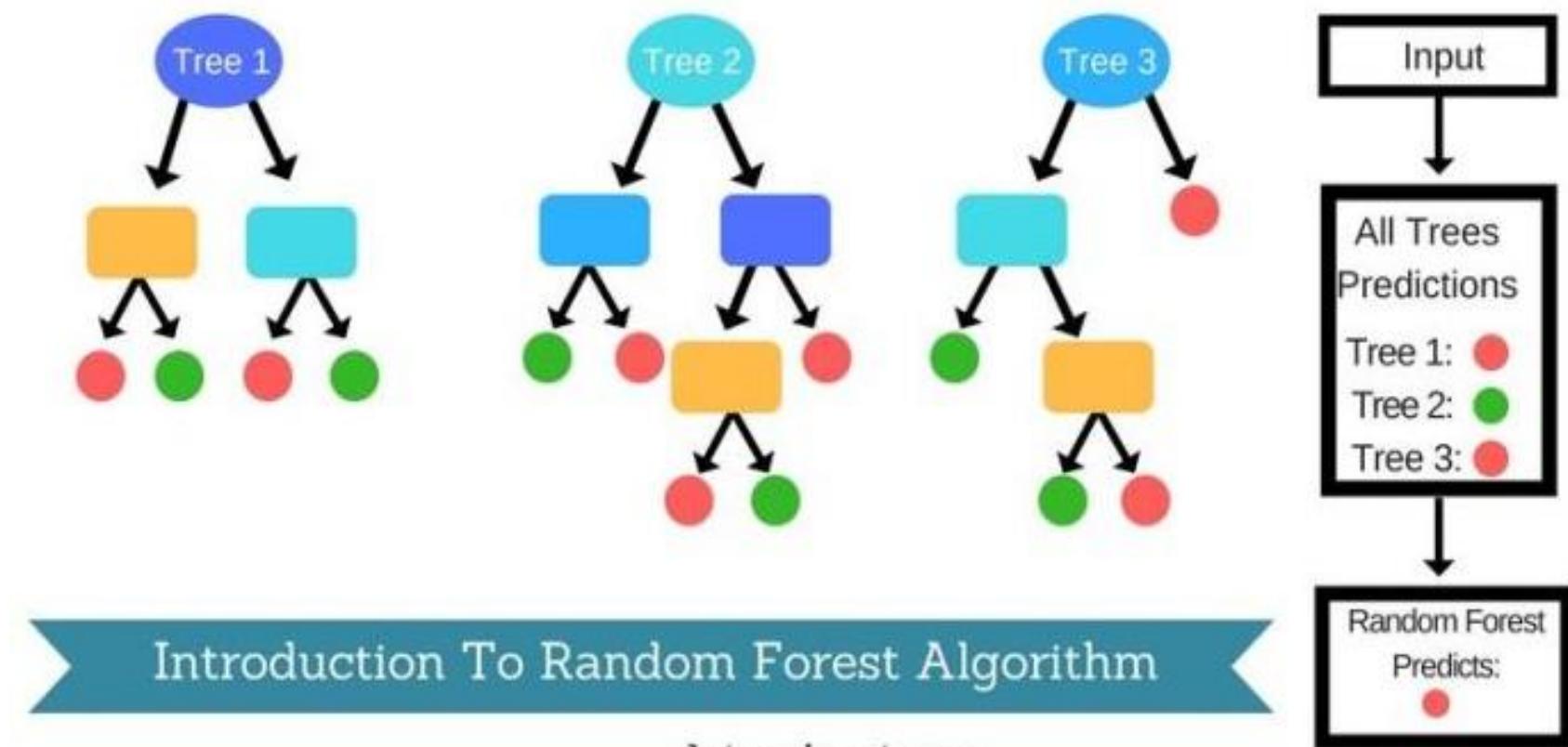
$$= 1 - 0.5625 - 0.0625$$

$$= 0.375$$

GINI Index

- GINI (t_R) = $1 - (6/16)^2 - (10/16)^2$
- = $1 - 0.140625 - 0.390625$
- = 0.469
- GINI (s,t) = $0.5 - (8/24)*0.375 - (16/24)*0.469$
- = $0.5 - 0.125 - 0.313$
- = 0.0625
- **The attribute value that provides the smallest SPLIT Gini (T) is chosen to split the node.

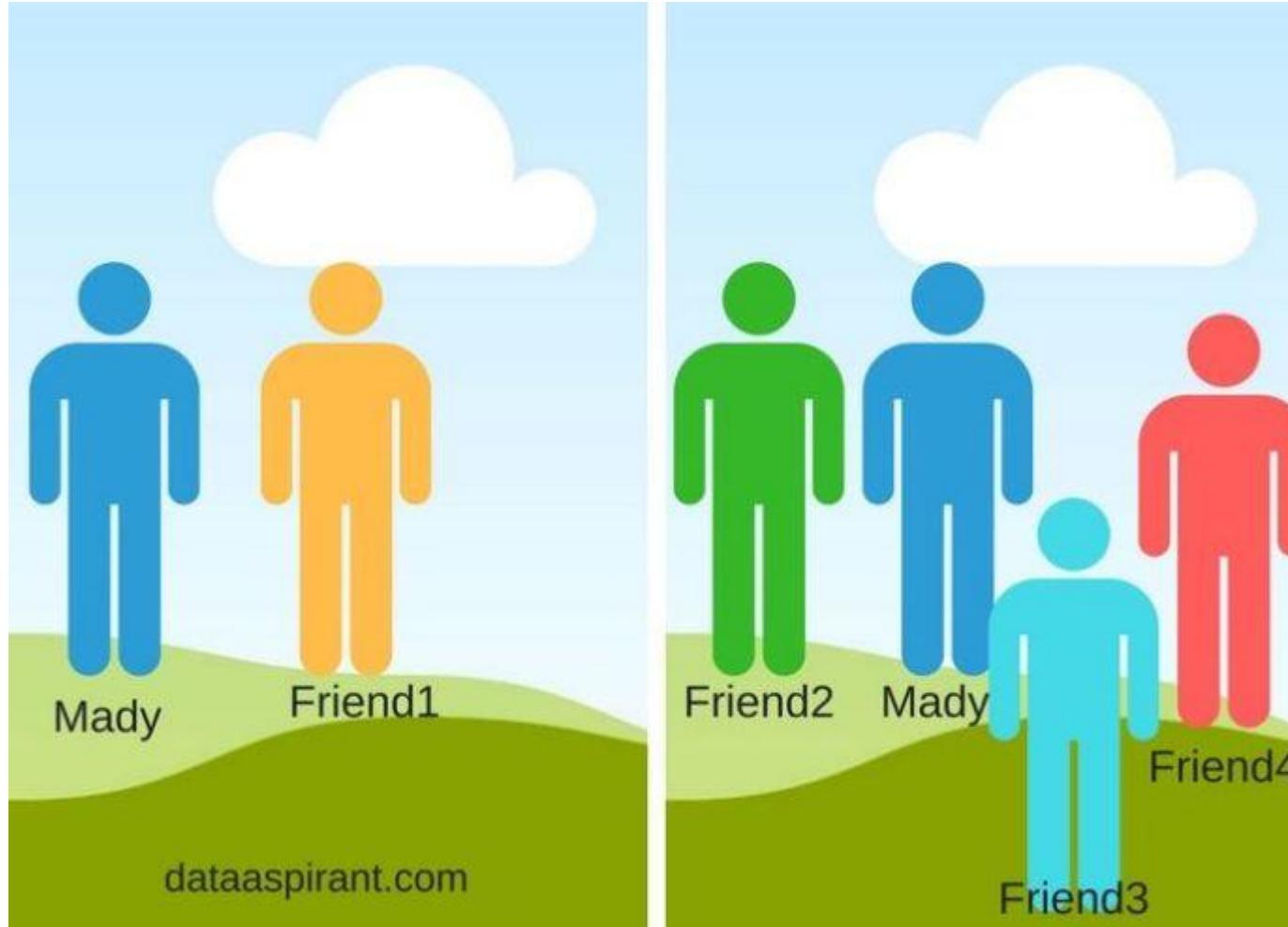
Random Forest Algorithm



Random Forest Algorithm

- Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees.
- In general, the **more trees in the forest** the more robust the forest looks like. In the same way in the random forest classifier, the **higher the number** of trees in the forest gives **the high accuracy** results.

Random forest algorithm real life example



Random forest algorithm real life example

- Suppose **Mady** somehow got 2 weeks leave from his office. He wants to spend his 2 weeks by traveling to the different place. He also wants to go to the place he may like.
- So he decided to ask his **best friend** about the places he may like. Then his friend started asking about his past trips. It's just like his best friend will ask, You have been visited the X place did you like it?
- Based on the answers which are given by Mady, his best start recommending the place Mady may like. Here his best formed the decision tree with the answer given by Mady.
- As his best friend may recommend his best place to Mady as a friend. The model will be **biased** with the closeness of their friendship. So he decided to ask few more friends to recommend the best place he may like.
- Now his friends asked some **random questions** and each one recommended one place to Mady. Now Mady considered the place which is **high votes** from his friends as the final place to visit.
- In the above Mady trip planning, two main interesting algorithms decision tree algorithm and random forest algorithm used. I hope you find it already. Anyhow, I would like to highlight it again.

Decision Tree

- To recommend the best place to Mady, his best friend asked some questions. Based on the answers given by mady, he recommended a place. This is **decision tree algorithm** approach.
- Mady friend used the answers given by mady to create rules. Later he used the created rules to recommend the best place which mady will like. These rules could be, mady like a place with lots of tree or waterfalls ..etc
- In the above approach mady best friend is the decision tree. The vote (recommended place) is the leaf of the decision tree (Target class). The target is finalized by a single person, In a technical way of saying, using an only single decision tree.

Random Forest Algorithm:

- In the other case when mady asked his friends to recommend the best place to visit. Each friend asked him different questions and come up their recommend a place to visit. Later mady consider all the recommendations and calculated the votes. Votes basically is to pick the popular place from the recommend places from all his friends.
- Mady will consider each recommended place and if the same place recommended by some other place he will increase the count. At the end the high count place where mady will go.
- In this case, the recommended place (**Target Prediction**) is considered by many friends. Each friend is the tree and the combined all friends will form the forest. This forest is the random forest. As each friend asked random questions to recommend the best place visit.

How Random Forest Algorithm Works

f11	f12	f13	f14	f15	t1
f21	f22	f23	f24	f25	t2
f31	f32	f33	f34	f35	t3
:	:	:	:	:	:
:	:	:	:	:	:
fm1	fm2	fm3	fm4	fm5	tm

Dataset

f11	f12	f13	f14	f15	t1
f81	f82	f83	f84	f85	t8
f71	f72	f73	f74	f75	t7
:	:	:	:	:	:
:	:	:	:	:	:
fj1	fj2	fj3	fj4	fj5	tj

Random Dataset
for Tree-01

f21	f22	f23	f24	f25	t2
f51	f52	f53	f54	f55	t5
f31	f32	f33	f34	f35	t3
:	:	:	:	:	:
:	:	:	:	:	:
fm1	fm2	fm3	fm4	fm5	tm

Random Dataset
for Tree-02

f31	f32	f33	f34	f35	t3
f61	f62	f63	f64	f65	t6
f91	f92	f73	f94	f95	t9
:	:	:	:	:	:
:	:	:	:	:	:
fk1	fk2	fk3	fk4	fk5	tk

Random Dataset
for Tree-03

Random Forest pseudocode:

- Randomly select “k” features from total “m” features.
- Where $k \ll m$
- Among the “k” features, calculate the node “d” using the best split point.
- Split the node into daughter nodes using the best split.
- Repeat 1 to 3 steps until “l” number of nodes has been reached.
- Build forest by repeating steps 1 to 4 for “n” number times to create “n” number of trees.

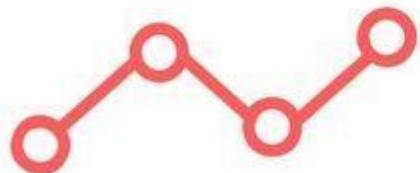
Random Forest pseudocode:

- To perform prediction using the trained random forest algorithm uses the below pseudocode.
- Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
- Calculate the votes for each predicted target.
- Consider the high voted predicted target as the final prediction from the random forest algorithm.

Random forest algorithm applications



Medicine



Stock Market



Banking:

- In the banking sector, random forest algorithm widely used in two main application. These are for finding the loyal customer and finding the fraud customers.
- The loyal customer means not the customer who pays well, but also the customer whom can take the huge amount as loan and pays the loan interest properly to the bank. As the growth of the bank purely depends on the loyal customers. The bank customers data highly analyzed to find the pattern for the loyal customer based the customer details.
- In the same way, there is need to identify the customer who are not profitable for the bank, like taking the loan and not paying the loan interest properly or find the outlier customers. If the bank can identify theses kind of customer before giving the loan the customer. Bank will get a chance to not approve the loan to these kinds of customers. In this case, also random forest algorithm is used to identify the customers who are not profitable for the bank.

Random Forest Applications

2. Medicine

- In medicine field, random forest algorithm is used identify the correct combination of the components to validate the medicine. Random forest algorithm also helpful for identifying the disease by analyzing the patient's medical records.

3. Stock Market

- In the stock market, random forest algorithm used to identify the stock behavior as well as the expected loss or profit by purchasing the particular stock.

E-Commerce

- In e-commerce, the random forest used only in the small segment of the recommendation engine for identifying the likely hood of customer liking the recommend products base on the similar kinds of customers.
- Running random forest algorithm on very large dataset requires **high-end GPU** systems. If you are not having any GPU system. You can always run the machine learning models in cloud hosted desktop. You can use cloud desktop online platform to run high-end machine learning models from sitting any corner of the world.

Logistic Regression

- Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy).
- The prediction is based on the use of one or several predictors (numerical and categorical).
- A linear regression is not appropriate for predicting the value of a binary variable for two reasons:
- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

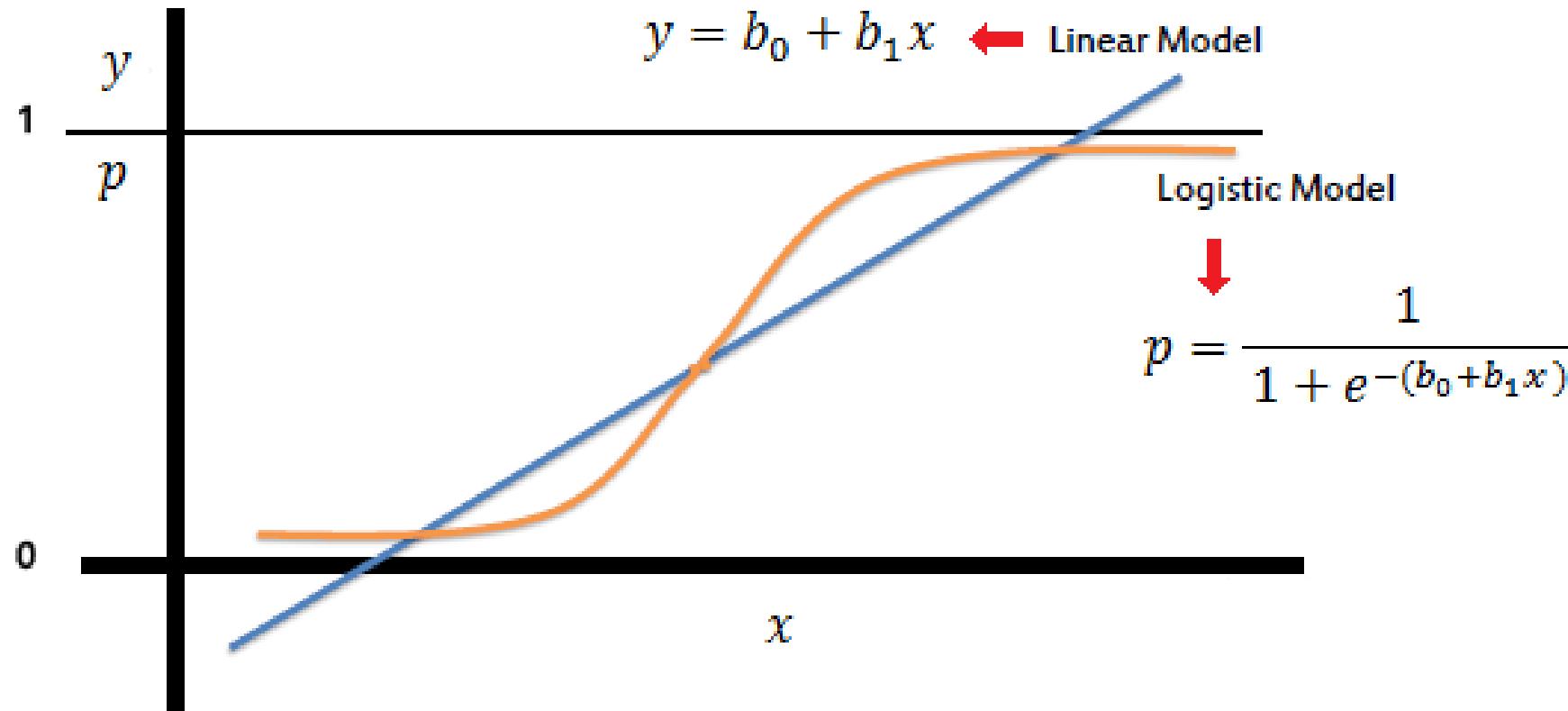
Logistic Regression

- On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1.
- Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the “odds” of the target variable, rather than the probability.
- Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

Example

- Example 1: Suppose that we are interested in the factors that influence whether a political candidate wins an election.
- The outcome (response) variable is binary (0/1); win or lose.
- The predictor variables of interest are the amount of money spent on the campaign, the amount of time spent campaigning negatively and whether or not the candidate is an incumbent(is he right person to hold the role/position).
- Example 2: A researcher is interested in how variables, such as GRE (Graduate Record Exam scores), GPA (grade point average) and prestige of the undergraduate institution, effect admission into graduate school. The response variable, admit/don't admit, is a binary variable.

Logistic Regression



Logistic Regression

- In the logistic regression the constant (b_0) moves the curve left and right and the slope (b_1) defines the steepness of the curve. By simple transformation, the logistic regression equation can be written in terms of an odds ratio.

$$\frac{p}{1-p} = \exp(b_0 + b_1 x)$$

Logistic Regression

- Finally, taking the natural log of both sides, we can write the equation in terms of log-odds (logit) which is a linear function of the predictors. The coefficient (b_1) is the amount the logit (log-odds) changes with a one unit change in x .

$$\ln\left(\frac{p}{1-p}\right) = b_0 + b_1 x$$

Logistic Regression

- As mentioned before, logistic regression can handle any number of numerical and/or categorical variables.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2 + \dots + b_p x_p)}}$$

$$\beta^1 = \beta^0 + [X^T W X]^{-1} \cdot X^T (y - \mu)$$

β is a vector of the logistic regression coefficients.

W is a square matrix of order N with elements $n_i \pi_i (1 - \pi_i)$ on the diagonal and zeros everywhere else.

μ is a vector of length N with elements $\mu_i = n_i \pi_i$.

Logistic Regression Assumptions

- Binary logistic regression requires the dependent variable to be binary.
- For a binary regression, the factor level 1 of the dependent variable should represent the desired outcome.
- Only the meaningful variables should be included.
- The independent variables should be independent of each other. That is, the model should have little or no multicollinearity.
- The independent variables are linearly related to the log odds.
- Logistic regression requires quite large sample sizes.

Data

- The dataset comes from the UCI Machine Learning repository, and it is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict whether the client will subscribe (1/0) to a term deposit (variable y).

Input Variables

- age (numeric)
- job : type of job (categorical: “admin”, “blue-collar”, “entrepreneur”, “housemaid”, “management”, “retired”, “self-employed”, “services”, “student”, “technician”, “unemployed”, “unknown”)
- marital : marital status (categorical: “divorced”, “married”, “single”, “unknown”)
- education (categorical: “basic.4y”, “basic.6y”, “basic.9y”, “high.school”, “illiterate”, “professional.course”, “university.degree”, “unknown”)
- default: has credit in default? (categorical: “no”, “yes”, “unknown”)
- housing: has housing loan? (categorical: “no”, “yes”, “unknown”)
- loan: has personal loan? (categorical: “no”, “yes”, “unknown”)
- contact: contact communication type (categorical: “cellular”, “telephone”)
- month: last contact month of year (categorical: “jan”, “feb”, “mar”, ..., “nov”, “dec”)
- day_of_week: last contact day of the week (categorical: “mon”, “tue”, “wed”, “thu”, “fri”)

Input Variables

- duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). The duration is not known before a call is performed, also, after the end of the call, y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model
- campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
- previous: number of contacts performed before this campaign and for this client (numeric)
- poutcome: outcome of the previous marketing campaign (categorical: “failure”, “nonexistent”, “success”)
- emp.var.rate: employment variation rate — (numeric)
- cons.price.idx: consumer price index — (numeric)
- cons.conf.idx: consumer confidence index — (numeric)
- euribor3m: euribor 3 month rate — (numeric)
- nr.employed: number of employees — (numeric)

Predict Variable

- y — has the client subscribed a term deposit? (binary: “1”, means “Yes”, “0” means “No”)
- The education column of the dataset has many categories and we need to reduce the categories for a better modelling. The education column has the following categories:

Observations

- The average age of customers who bought the term deposit is higher than that of the customers who didn't.
- The pdays (days since the customer was last contacted) is understandably lower for the customers who bought it. The lower the pdays, the better the memory of the last call and hence the better chances of a sale.
- Surprisingly, campaigns (number of contacts or calls made during the current campaign) are lower for customers who bought the term deposit.

Supervised Learning

An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- **A decision is needed:** whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU, those patients who may survive less than a month are given higher priority.
- **Problem:** to predict **high-risk patients** and discriminate them from **low-risk patients**.

Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
 - age
 - Marital status
 - annual salary
 - outstanding debts
 - credit rating
 - etc.
- **Problem:** to decide whether an application should be approved, or to classify applications into two categories, **approved** and **not approved**.

Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

The data and the goal

- **Data:** A set of data records (also called examples, instances or cases) described by
 - k attributes: $A_1, A_2, \dots A_k$.
 - a class: Each example is labelled with a pre-defined class.
- **Goal:** To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
 - Yes (approved) and
 - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

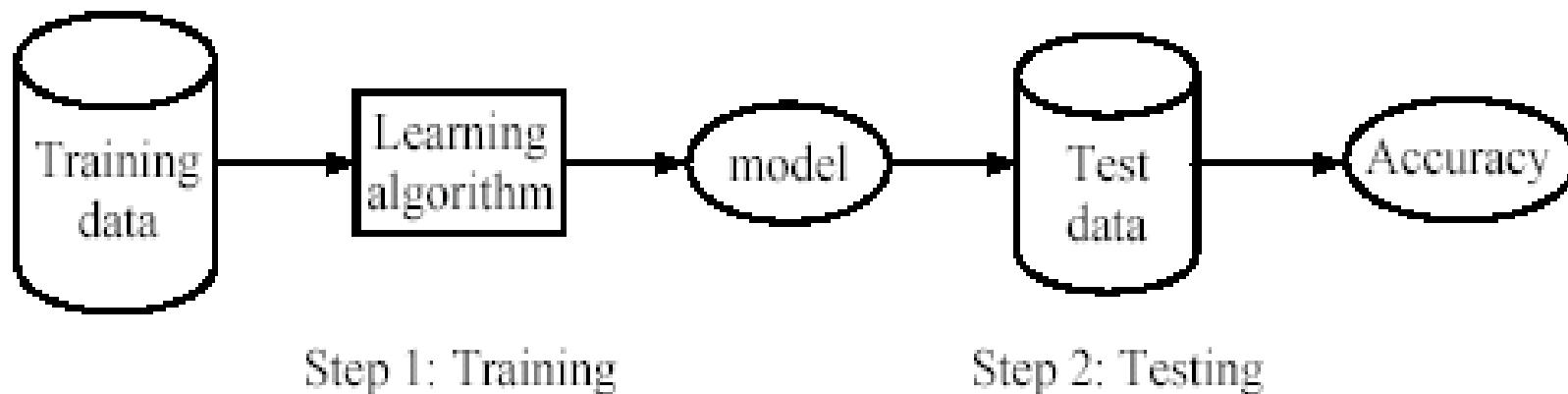
Supervised vs. unsupervised Learning

- **Supervised learning:** classification is seen as supervised learning from examples.
 - **Supervision:** The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (**supervision**).
 - Test data are classified into these classes too.
- **Unsupervised learning (clustering)**
 - **Class labels of the data are unknown**
 - Given a set of data, the task is to establish the existence of classes or clusters in the data

Supervised learning process: two steps

- **Learning (training):** Learn a model using the training data
 - **Testing:** Test the model using **unseen** test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



What do we mean by learning?

- Given

- a data set D ,
- a task T , and
- a performance measure M ,

a computer system is said to **learn** from D to perform the task T if after learning the system's performance on T improves as measured by M .

- In other words, the learned model helps the system to perform T better as compared to no learning.

An example

- **Data**: Loan application data
- **Task**: Predict whether a loan should be approved or not.
- **Performance measure**: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

$$\text{Accuracy} = 9/15 = 60\%.$$

- We can do better than 60% with learning.

Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

Introduction

- Decision tree learning is one of the most widely used techniques for classification.
 - Its classification accuracy is competitive with other methods, and
 - it is very efficient.
- The classification model is a tree, called **decision tree**.
- **C4.5** by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

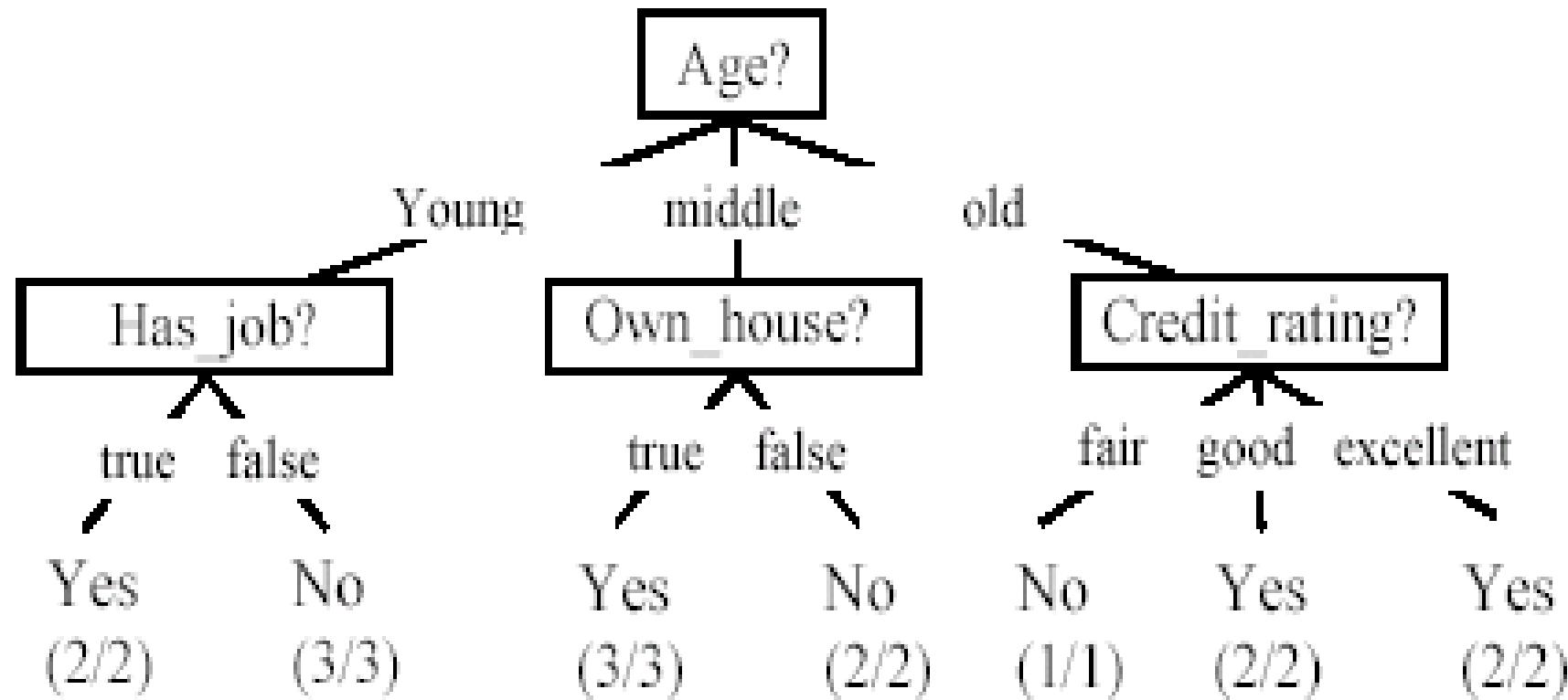
The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

A decision tree from the loan data

- Decision nodes and leaf nodes (classes)



Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	? No

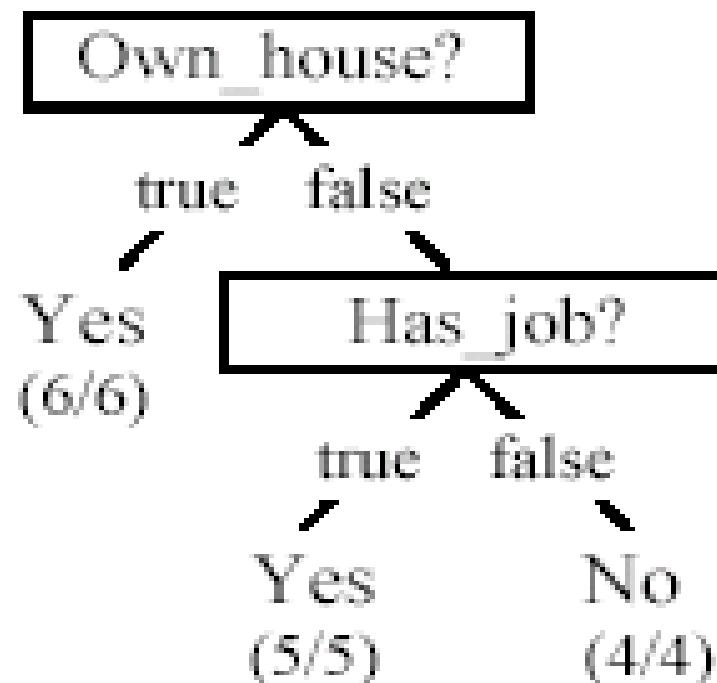

```
graph TD; A[Age?] --> B[Young]; A --> C[middle]; A --> D[old]; B --> E[Has_job?]; C --> F[Own_house?]; D --> G[Credit_rating?]; E --> H[true]; E --> I[false]; H --> J[Yes]; H --> K[No]; I --> L[Yes]; I --> M[No]; L --> N["(2/2)"]; M --> O["(3/3)"]; F --> P[true]; F --> Q[false]; P --> R[Yes]; P --> S[No]; R --> T["(3/3)"]; S --> U["(2/2)"]; G --> V[fair]; G --> W[good]; G --> X[excellent]; V --> Y[No]; V --> Z[Yes]; Y --> AA["(1/1)"]; Z --> BB["(2/2)"]; X --> CC["(2/2)"];
```

The decision tree diagram illustrates the classification process for a loan application based on four attributes: Age, Has_job, Own_house, and Credit_rating. The tree starts at the root node 'Age?' and branches into three categories: 'Young', 'middle', and 'old'. The 'Young' branch leads to the 'Has_job?' node, which further branches into 'true' and 'false'. The 'true' branch leads to a leaf node 'Yes' with a count of '(2/2)', while the 'false' branch leads to a leaf node 'No' with a count of '(3/3)'. The 'middle' branch leads to the 'Own_house?' node, which branches into 'true' and 'false'. The 'true' branch leads to a leaf node 'Yes' with a count of '(3/3)', while the 'false' branch leads to a leaf node 'No' with a count of '(2/2)'. The 'old' branch leads to the 'Credit_rating?' node, which branches into 'fair', 'good', and 'excellent'. The 'fair' branch leads to a leaf node 'No' with a count of '(1/1)', the 'good' branch leads to a leaf node 'Yes' with a count of '(2/2)', and the 'excellent' branch leads to a leaf node 'Yes' with a count of '(2/2)'.

Is the decision tree unique?

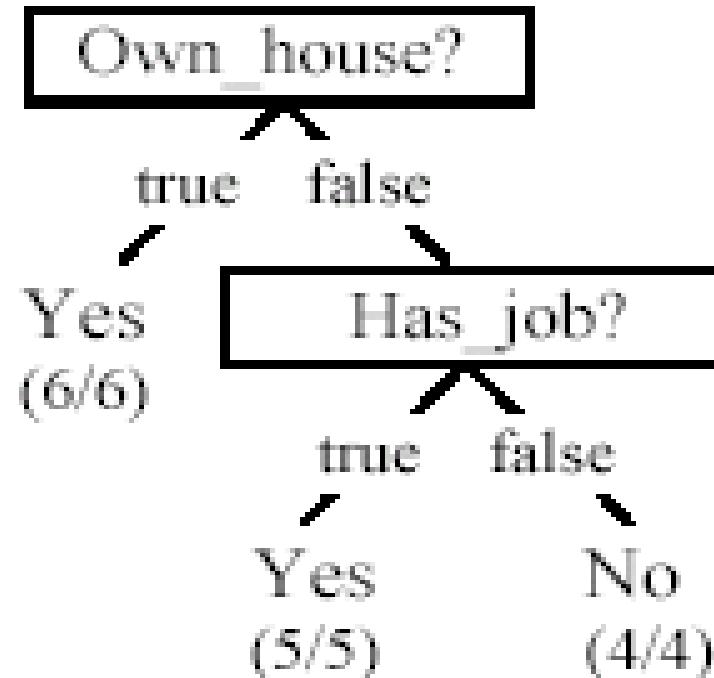
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
 - Easy to understand and perform better.

- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.



$\text{Own_house} = \text{true} \rightarrow \text{Class} = \text{Yes}$ [sup=6/15, conf=6/6]
 $\text{Own_house} = \text{false}, \text{Has_job} = \text{true} \rightarrow \text{Class} = \text{Yes}$ [sup=5/15, conf=5/5]
 $\text{Own_house} = \text{false}, \text{Has_job} = \text{false} \rightarrow \text{Class} = \text{No}$ [sup=4/15, conf=4/4]

Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
 - Assume attributes are categorical now (continuous attributes can be handled too)
 - Tree is constructed in a **top-down recursive manner**
 - At start, all the training examples are at the root
 - Examples are partitioned recursively based on selected attributes
 - Attributes are selected on the basis of an impurity function (e.g., **information gain**)
- Conditions for stopping partitioning
 - All examples for a given node belong to the same class
 - There are no remaining attributes for further partitioning – majority class is the leaf
 - There are no examples left

Decision tree learning algorithm

```
. Algorithm decisionTree( $D, A, T$ )
1   if  $D$  contains only training examples of the same class  $c_j \in C$  then
2       make  $T$  a leaf node labeled with class  $c_j$ ;
3   elseif  $A = \emptyset$  then
4       make  $T$  a leaf node labeled with  $c_j$ , which is the most frequent class in  $D$ 
5   else //  $D$  contains examples belonging to a mixture of classes. We select a single
6       // attribute to partition  $D$  into subsets so that each subset is purer
7        $p_0 = \text{impurityEval-1}(D)$ ;
8       for each attribute  $A_i \in \{A_1, A_2, \dots, A_k\}$  do
9            $p_i = \text{impurityEval-2}(A_i, D)$ 
10      end
11      Select  $A_g \in \{A_1, A_2, \dots, A_k\}$  that gives the biggest impurity reduction,
12          computed using  $p_0 - p_i$ 
13      if  $p_0 - p_g < \text{threshold}$  then //  $A_g$  does not significantly reduce impurity  $p_0$ 
14          make  $T$  a leaf node labeled with  $c_j$ , the most frequent class in  $D$ .
15      else //  $A_g$  is able to reduce impurity  $p_0$ 
16          Make  $T$  a decision node on  $A_g$ ;
17          Let the possible values of  $A_g$  be  $v_1, v_2, \dots, v_m$ . Partition  $D$  into  $m$ 
18          disjoint subsets  $D_1, D_2, \dots, D_m$  based on the  $m$  values of  $A_g$ .
19          for each  $D_j$  in  $\{D_1, D_2, \dots, D_m\}$  do
20              if  $D_j \neq \emptyset$  then
21                  create a branch (edge) node  $T_j$  for  $v_j$  as a child node of  $T$ ;
22                  decisionTree( $D_j, A - \{A_g\}, T_j$ ) //  $A_g$  is removed
23              end
24          end
25      end
26  end
```

Choose an attribute to partition data

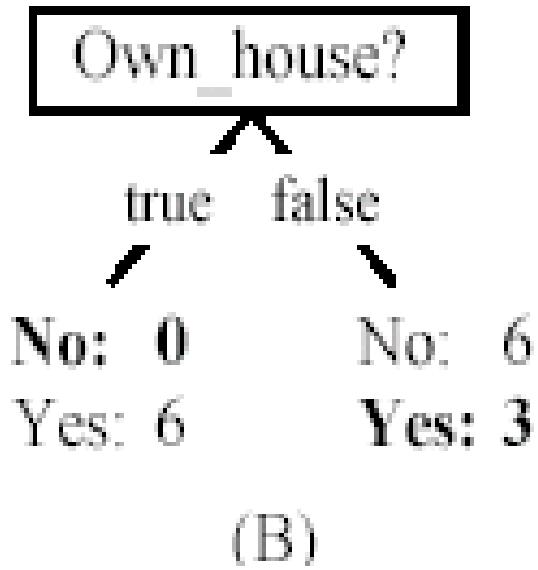
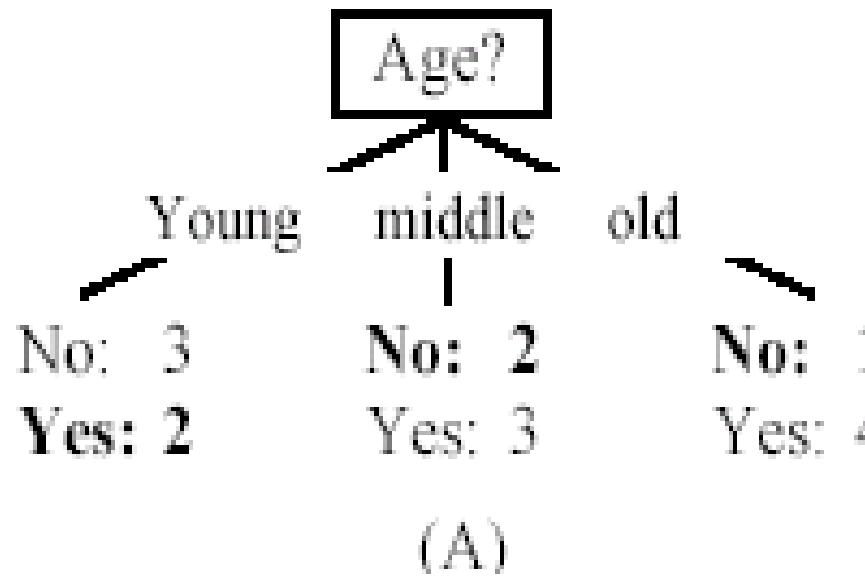
- The **key** to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
 - A subset of data is **pure** if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum **Information Gain** or **Gain Ratio** based on information theory.

The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Two possible roots, which is better?



- Fig. (B) seems to be better.

Information theory

- **Information theory** provides a mathematical basis for measuring the information content.
- To understand the notion of information, think about it as providing the answer to a question, for example, whether a coin will come up heads.
 - If one already has a good guess about the answer, then the actual answer is less informative.
 - If one already knows that the coin is rigged so that it will come with heads with probability 0.99, then a message (advanced information) about the actual outcome of a flip is worth less than it would be for a honest coin (50-50).

Information theory (cont ...)

- For a fair (honest) coin, you have no information, and you are willing to pay more (say in terms of \$) for advanced information - less you know, the more valuable the information.
- **Information theory** uses this same intuition, but instead of measuring the value for information in dollars, it measures information contents in **bits**.
- One bit of information is enough to answer a yes/no question about which one has no idea, such as the flip of a fair coin

Information theory: Entropy measure

- The entropy formula,

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

- $\Pr(c_j)$ is the probability of class c_j in data set D
- We use entropy as a **measure of impurity or disorder** of data set D . (Or, a measure of information in a tree)

Entropy measure: let us get a feeling

1. The data set D has 50% positive examples ($\Pr(\text{positive}) = 0.5$) and 50% negative examples ($\Pr(\text{negative}) = 0.5$).

$$\text{entropy}(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(\text{positive}) = 0.2$) and 80% negative examples ($\Pr(\text{negative}) = 0.8$).

$$\text{entropy}(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(\text{positive}) = 1$) and no negative examples, ($\Pr(\text{negative}) = 0$).

$$\text{entropy}(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

Information gain

- Given a set of examples D , we first compute its entropy:

$$\text{entropy}(D) = - \sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

- If we make ~~.....~~, ~~.....~~, ~~.....~~, ~~.....~~ current tree, this will partition D into v subsets $D_1, D_2 \dots, D_v$. The expected entropy if A_i is used as the current root:

$$\text{entropy}_{A_i}(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times \text{entropy}(D_j)$$

Information gain (cont ...)

- Information gained by selecting attribute A_i to branch or to partition the data is

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

- We choose the attribute with the highest gain to branch/split the current tree.

Tested all the calculations

An example

$$\text{entropy}(D) = \frac{6}{15} \times \log_2 \frac{6}{15} + \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

Entropy age -> refer manual cal -> $-2/5 \log(2/5) - 3/5 \log(3/5)$

$$\begin{aligned}\text{entropy}_{\text{Own_house}}(D) &= \frac{6}{15} \times \text{entropy}(D_1) + \frac{9}{15} \times \text{entropy}(D_2) \\ &= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918 \\ &= 0.551\end{aligned}$$

$$\begin{aligned}\text{entropy}_{\text{Age}}(D) &= \frac{5}{15} \times \text{entropy}(D_1) + \frac{5}{15} \times \text{entropy}(D_2) + \frac{5}{15} \times \text{entropy}(D_3) \\ &= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722 \\ &= 0.888\end{aligned}$$

- Own_house is the best choice for the root.

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

	Age	Yes	No	entropy(Di)
5	young	2	3	0.971
5	middle	3	2	0.971
5	old	4	1	0.722

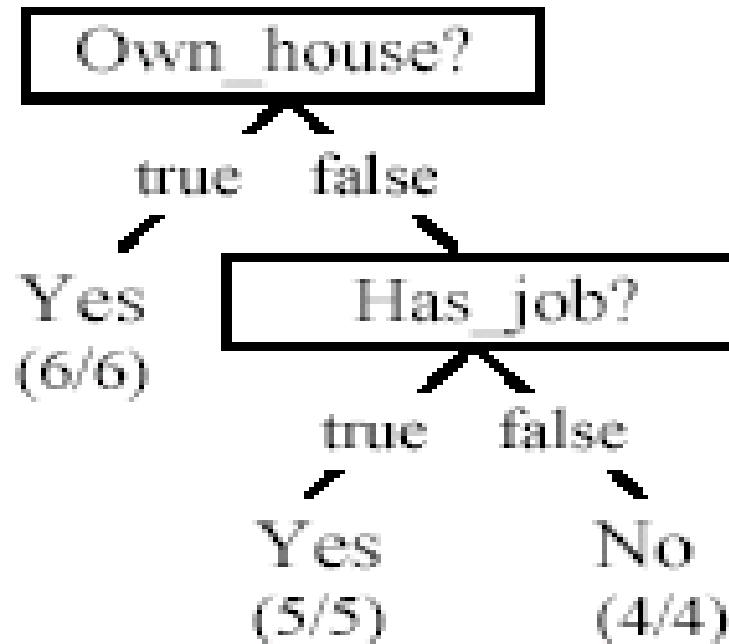
$$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$$

~~$$gain(D, \text{Own_house}) = 0.971 - 0.551 = 0.420$$~~

$$gain(D, \text{Has_Job}) = 0.971 - 0.647 = 0.324$$

$$gain(D, \text{Credit_Rating}) = 0.971 - 0.608 = 0.363$$

We build the final tree

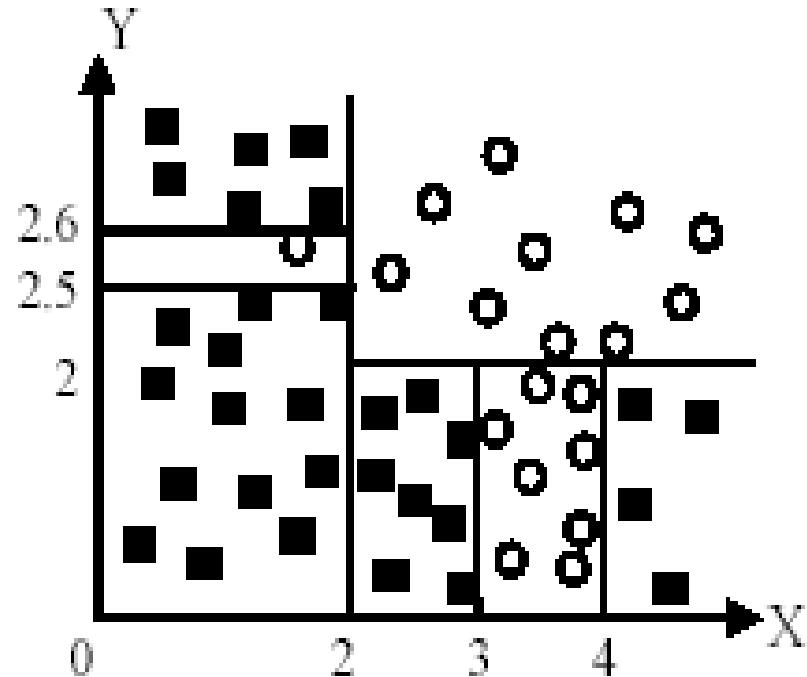


- We can use information gain ratio to evaluate the impurity as well (see the handout)

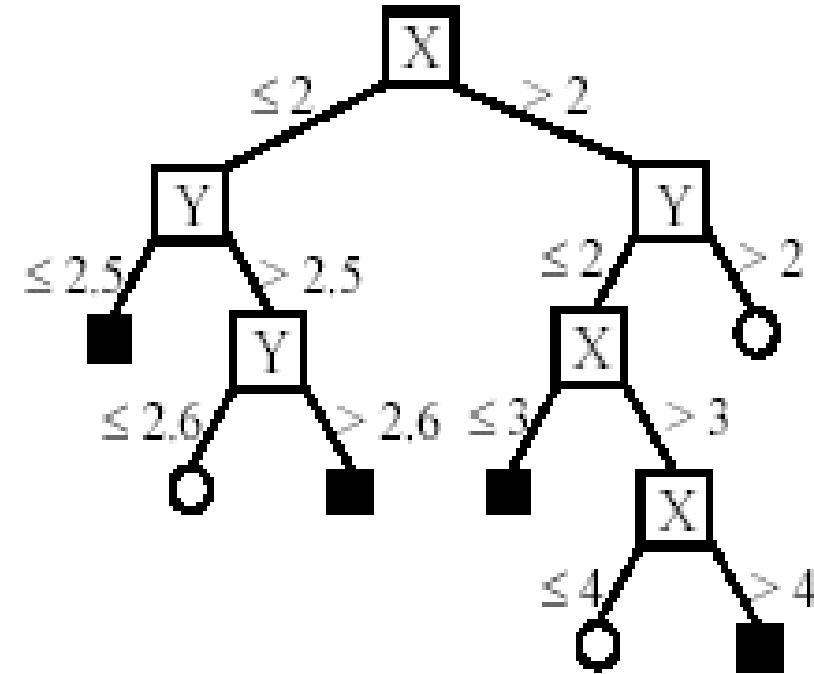
Handling continuous attributes

- Handle continuous attribute by splitting into two intervals (can be more) at each node.
- How to find the best threshold to divide?
 - Use information gain or gain ratio again
 - Sort all the values of an continuous attribute in increasing order $\{v_1, v_2, \dots, v_r\}$,
 - One possible threshold between two adjacent values v_i and v_{i+1} . Try all possible thresholds and find the one that maximizes the gain (or gain ratio).

An example in a continuous space



(A) A partition of the data space

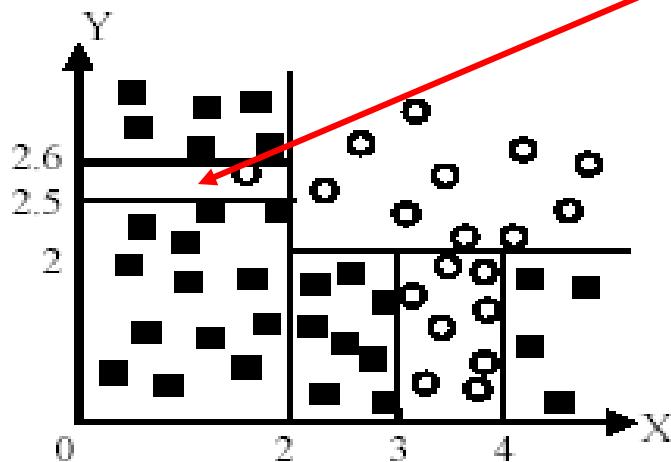


(B). The decision tree

Avoid overfitting in classification

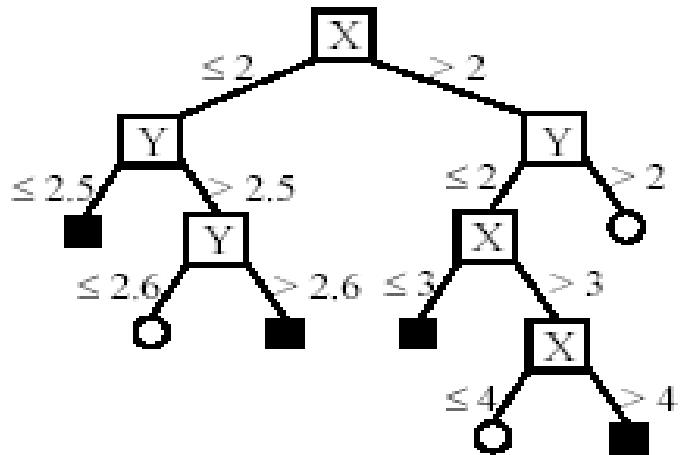
- **Overfitting:** A tree may overfit the training data
 - Good accuracy on training data but poor on test data
 - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
 - **Pre-pruning:** Halt tree construction early
 - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
 - **Post-pruning:** Remove branches or sub-trees from a “fully grown” tree.
 - This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.
 - A validation set may be used for pruning as well.

An example

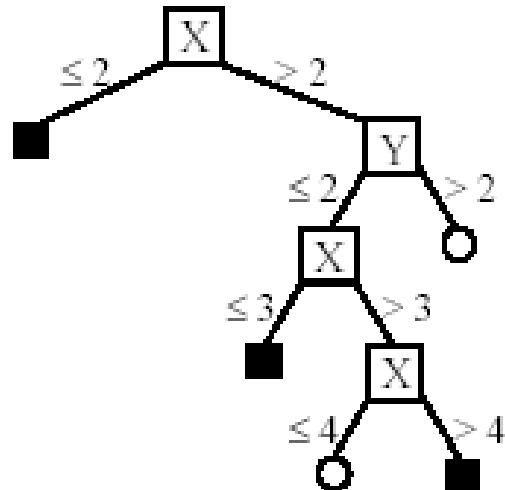
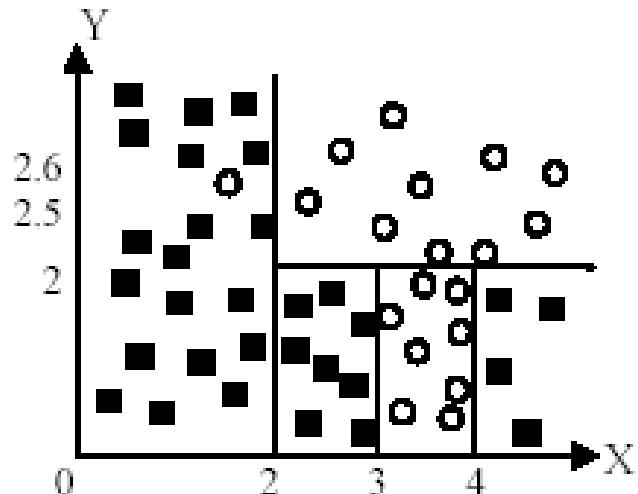


(A) A partition of the data space

Likely to overfit the data



(B). The decision tree



Other issues in decision tree learning

- From tree to rules, and rule pruning
- Handling of miss values
- Handing skewed distributions
- Handling attributes and classes with different costs.
- Attribute construction
- Etc.

Supervised learning vs. unsupervised learning

- **Supervised learning:** discover patterns in the data that relate data attributes with a target (class) attribute.
 - These patterns are then utilized to predict the values of the target attribute in future data instances.
- **Unsupervised learning:** The data have no target attribute.
 - We want to explore the data to find some intrinsic structures in them.

Natural Language Processing

- Natural Language Processing (NLP) refers to AI method of communicating with an intelligent systems using a natural language such as English.
- Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.
- The field of NLP involves making computers to perform useful tasks with the natural languages humans use. The input and output of an NLP system can be –
 - Speech
 - Written Text

Components of NLP

- Natural Language Understanding (NLU)
- Understanding involves the following tasks –
 - Mapping the given input in natural language into useful representations.
 - Analyzing different aspects of the language.
- Natural Language Generation (NLG)
 - It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.

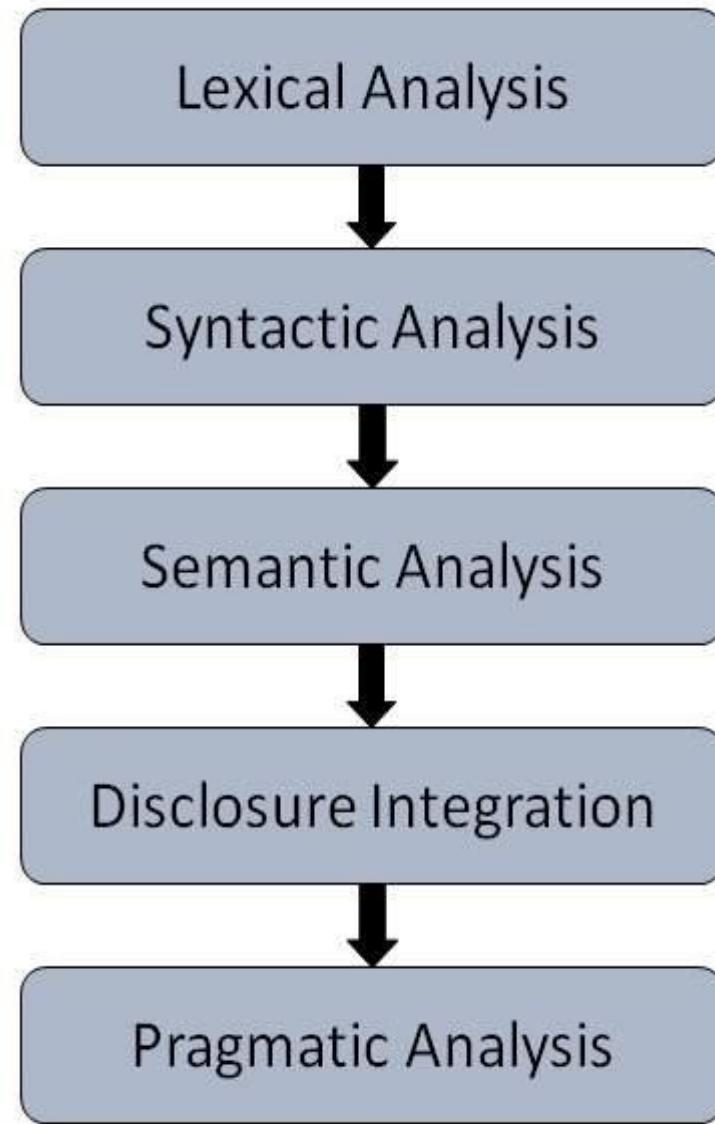
Components of NLP

- It involves –
- **Text planning** – It includes retrieving the relevant content from knowledge base.
- **Sentence planning** – It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
- **Text Realization** – It is mapping sentence plan into sentence structure.

Difficulties in NLU

- **Lexical ambiguity** – It is at very primitive level such as word-level.
- For example, treating the word “board” as noun or verb?
- **Syntax Level ambiguity** – A sentence can be parsed in different ways.
- For example, “He lifted the beetle with red cap.” – Did he use cap to lift the beetle or he lifted a beetle that had red cap?
- **Referential ambiguity** – Referring to something using pronouns. For example, Rima went to Gauri. She said, “I am tired.” – Exactly who is tired?

Steps in NLP



Where Can Developers Use NLP Algorithms For?

- **Summarize blocks of text** (Summarizer tool)
- <https://algorithmia.com/>
- Create a **chat bot**
- **Parsey McParseface** --- Parse sentences with ease.

- **Automatically generate keyword tags**
- **Identify the type of entity extracted,**
- Use Sentiment Analysis to **identify the sentiment of a string of text**,
- **Reduce words to their root**, or stem, using PorterStemmer, or **break up text into tokens** using Tokenizer.

Open Source NLP Libraries

- Apache OpenNLP: a machine learning toolkit that provides tokenizers, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, coreference resolution, and more.
- Natural Language Toolkit (NLTK): a Python library that provides modules for processing text, classifying, tokenizing, stemming, tagging, parsing, and more.
- Standford NLP: a suite of NLP tools that provide part-of-speech tagging, the named entity recognizer, coreference resolution system, sentiment analysis, and more.
- MALLET: a Java package that provides Latent Dirichlet Allocation, document classification, clustering, topic modeling, information extraction, and more.

NLTK

- Install NLTK 3.0, downloadable for free from <http://nltk.org/>
- The book module contains the data
- Any time we want to find out about these texts, we just have to enter their names
- A concordance view shows us every occurrence of a given word, together with some context.
- A concordance permits us to see words in context.
- It is one thing to automatically detect that a particular word occurs in a text, and to display some words that appear in the same context. However, we can also determine the *location* of a word in the text: how many words from the beginning it appears. This positional information can be displayed using a **dispersion plot**.
- **Counting Vocabulary**

NLTK

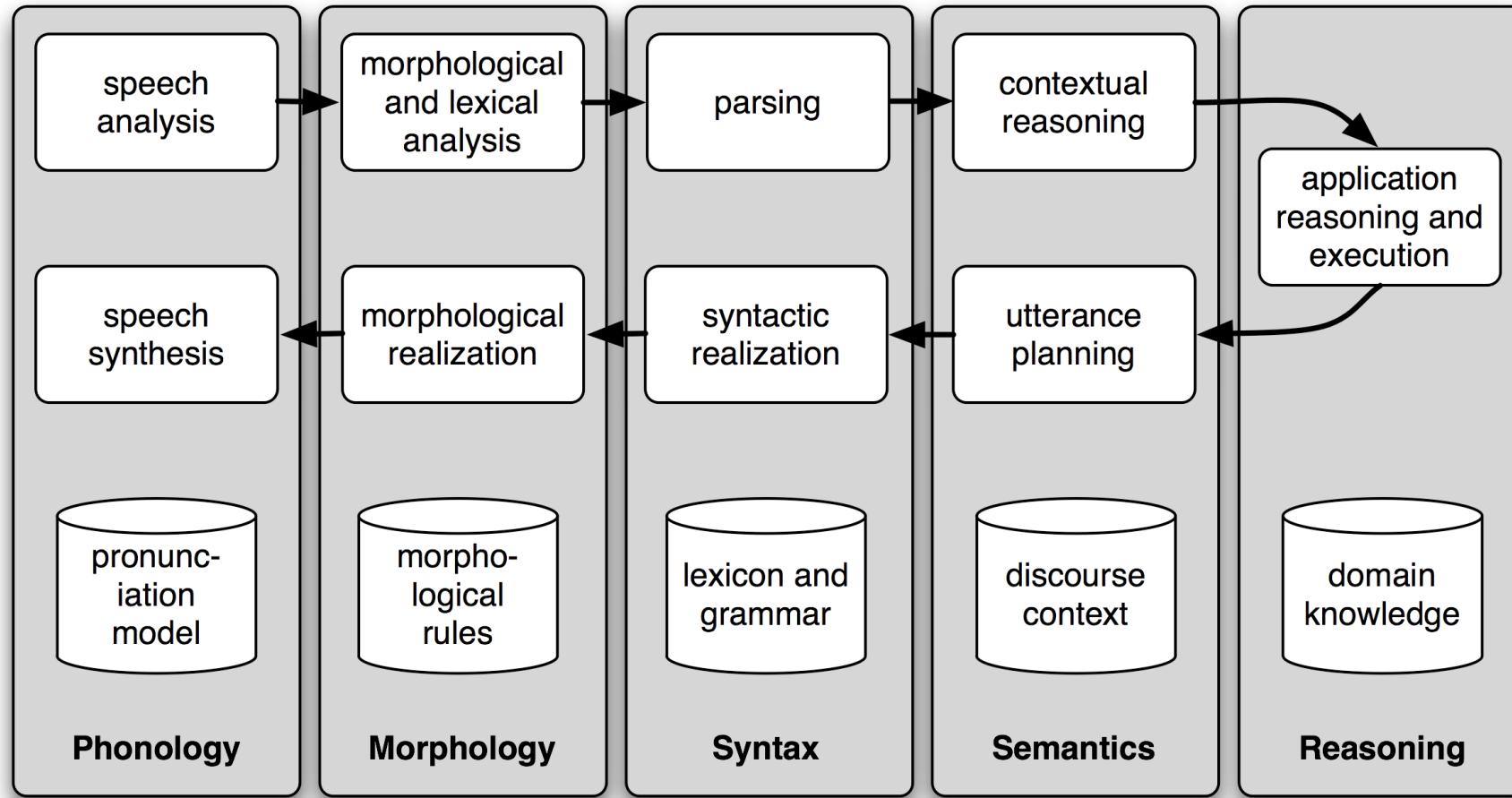
- **Frequency Distributions**
- **Fine-grained Selection of Words(Long Words)**
- **Collocations and Bigrams**
- A **collocation** is a sequence of words that occur together unusually often. Thus *red wine* is a collocation, whereas *the wine* is not. , *maroon wine* sounds definitely odd.

Automatic Natural Language Understanding

- **Word Sense Disambiguation**
 - serve: help with food or drink; hold an office; put ball into play
 - dish: plate; course of a meal; communications device
- **Pronoun Resolution**
 - A deeper kind of language understanding is to work out "who did what to whom"
- **Generating Language Output**
 - a. Text: ... The thieves stole the paintings. They were subsequently sold. ...
 - b. Human: Who or what was sold? (**Question to machine**)
 - c. Machine: The paintings. (**Machine Answers**)

Automatic Natural Language Understanding

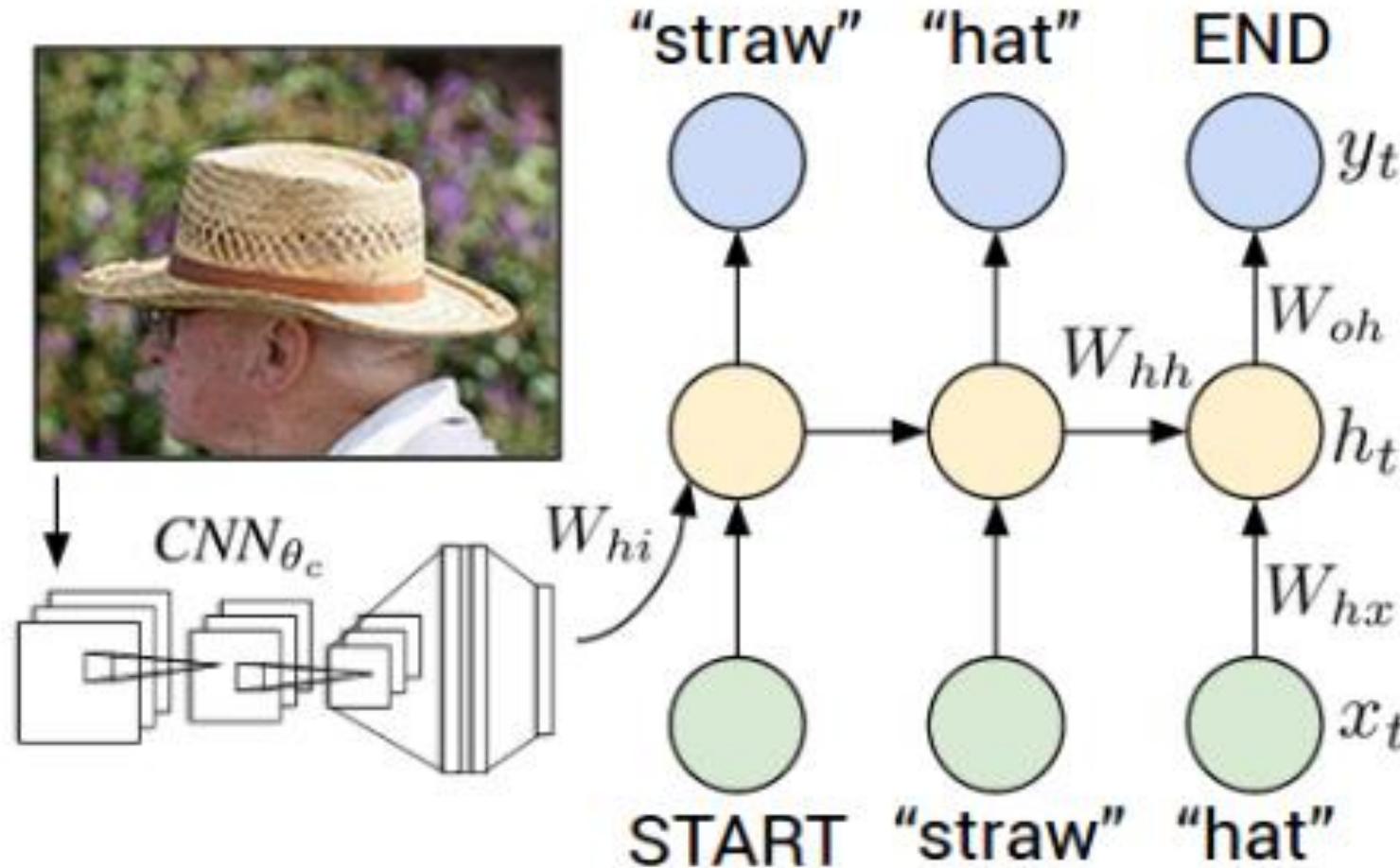
- Spoken Dialog Systems



Deep Learning

- conda install theano
- Tensor flow
- Keras

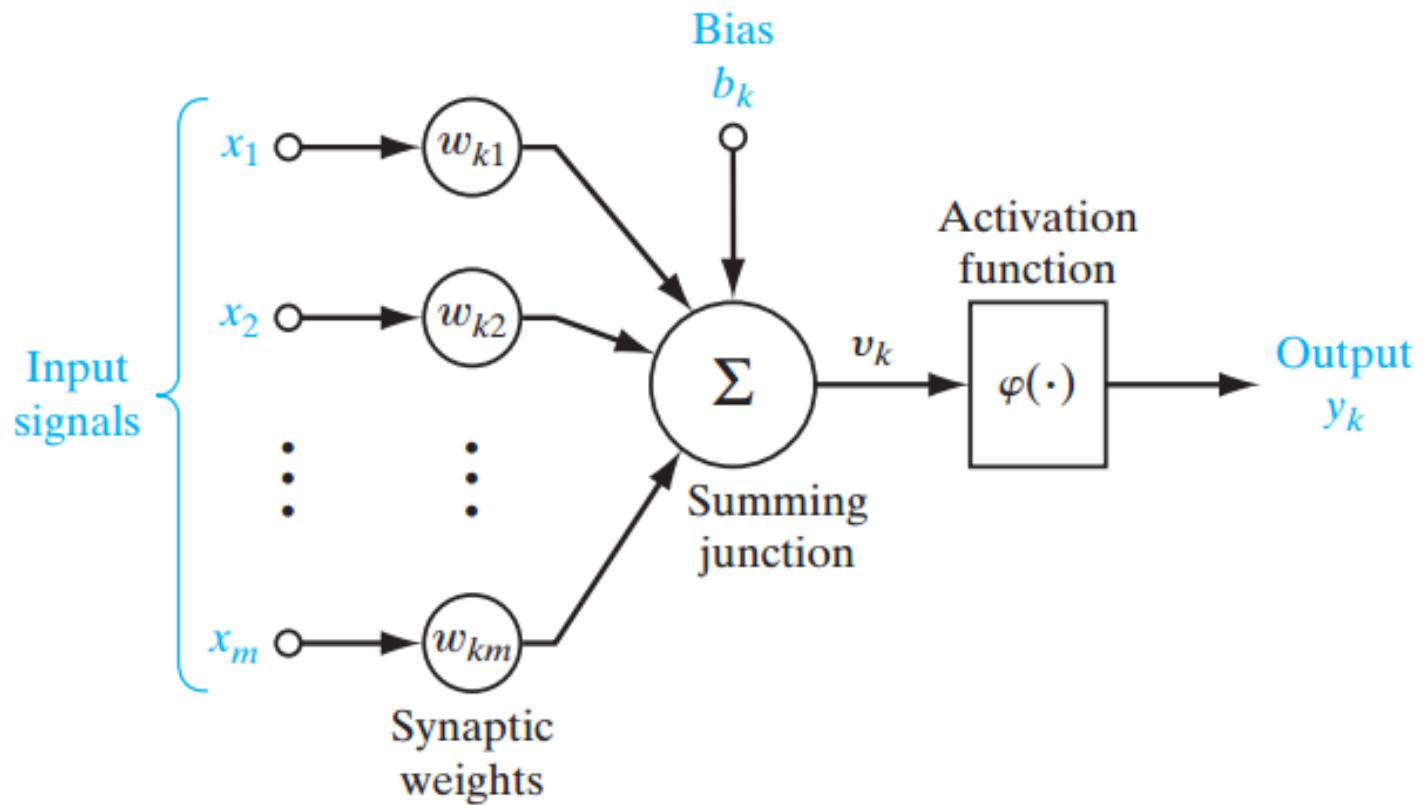
Image Captioning in Deep Learning



Artificial Neural Network

- Neural Networks are a machine learning framework that attempts to mimic the learning pattern of natural biological neural networks.
- Biological neural networks have interconnected neurons with dendrites that receive inputs, then based on these inputs they produce an output signal through an axon to another neuron.
- We will try to mimic this process through the use of Artificial Neural Networks (ANN), which we will just refer to as neural networks from now on.
- The process of creating a neural network begins with the most basic form, a single perceptron.

Single Layer Network

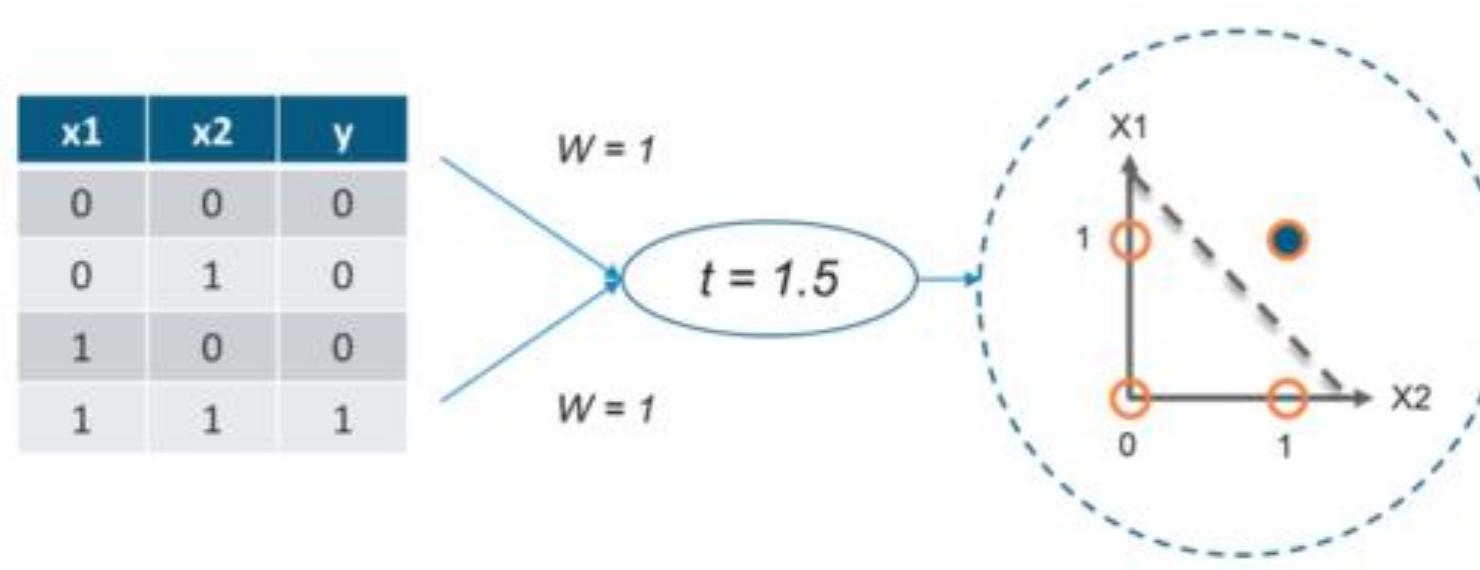


Artificial Neural Network

- Perceptron as AND Gate
- A perceptron can be used as a separator or a decision line that divides the input set of AND Gate, into two classes:
- **Class 1:** Inputs having output as 0 that lies below the decision line.
- **Class 2:** Inputs having output as 1 that lies above the decision line or separator.

Artificial Neural Network

- The below diagram shows the above idea of classifying the inputs of AND Gate using a perceptron:



Artificial Neural Network

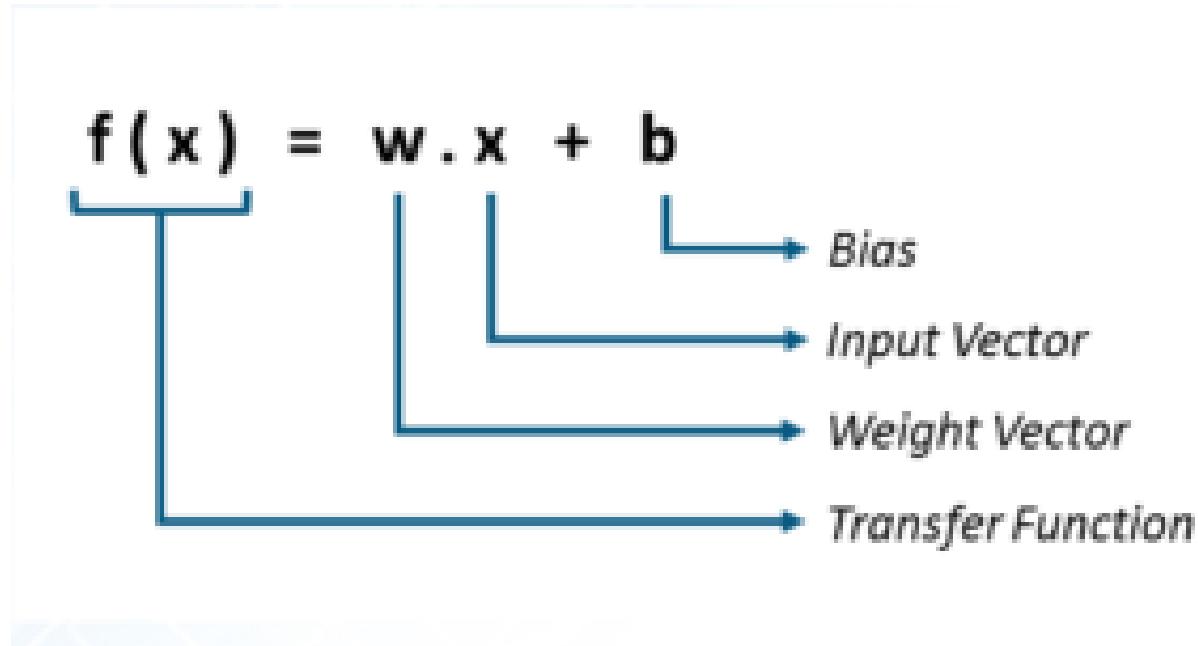
- Mathematically, one can represent a perceptron as a function of weights, inputs and bias (vertical offset):

$$f(x) = w \cdot x + b$$

The diagram illustrates the mathematical representation of a perceptron. The equation $f(x) = w \cdot x + b$ is shown in a white box. Below the equation, four arrows point from left to right, each labeled with a component: 'Transfer Function' points to the entire equation, 'Weight Vector' points to the term $w \cdot x$, 'Input Vector' points to the term x , and 'Bias' points to the term b .

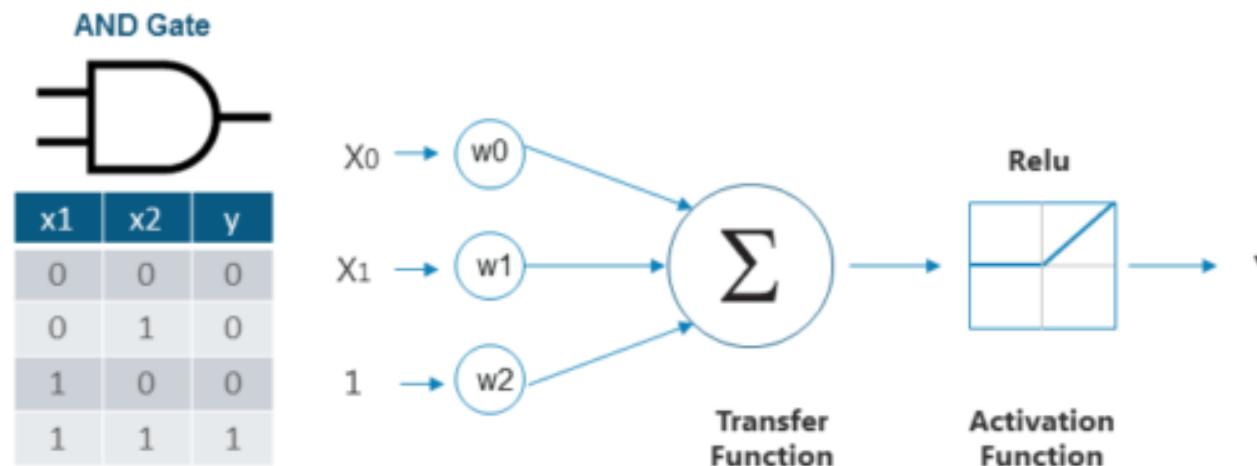
Artificial Neural Network

- Each of the input received by the perceptron has been weighted based on the amount of its contribution for obtaining the final output.
- Bias allows us to shift the decision line so that it can best separate the inputs into two classes.

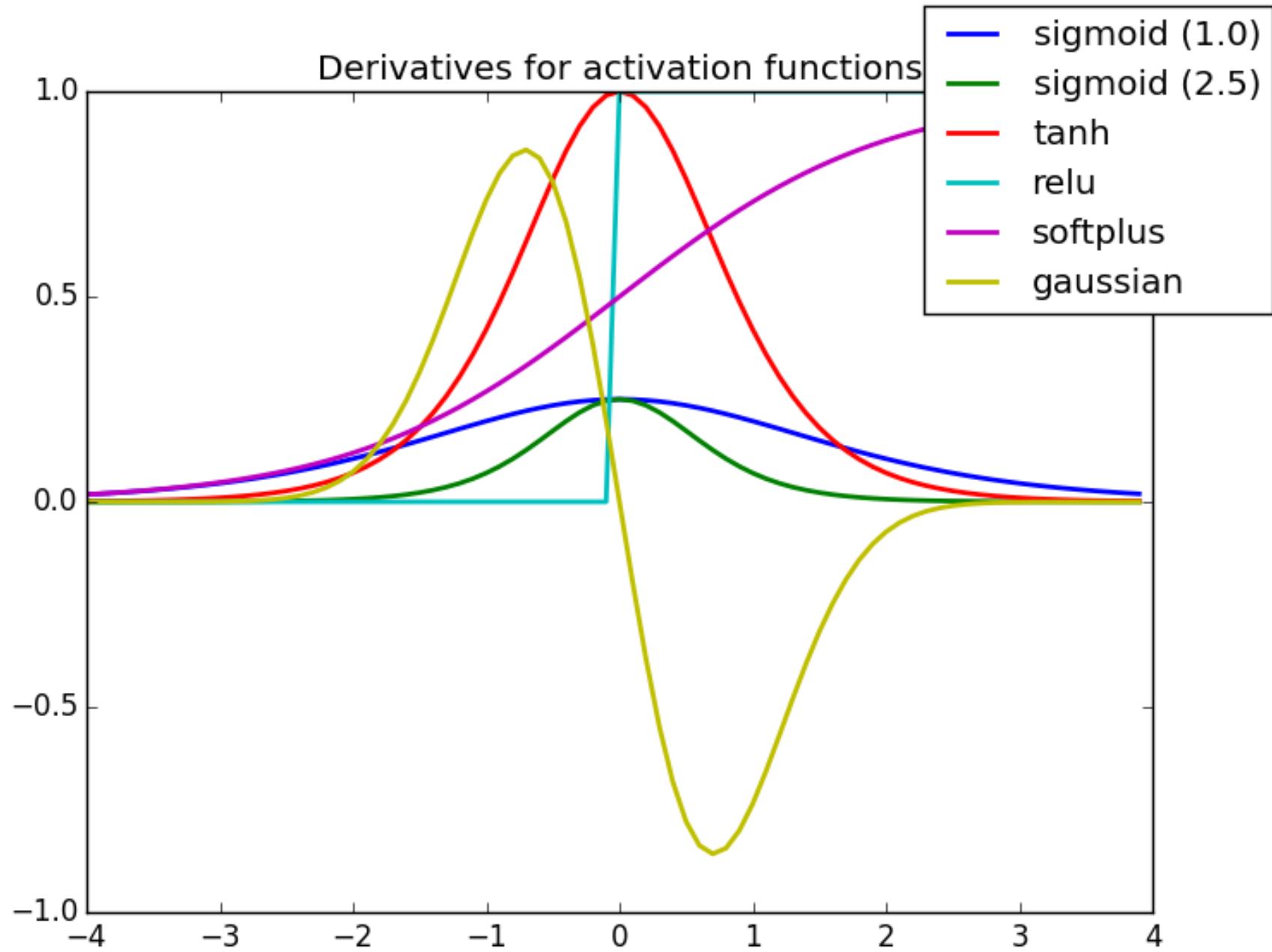


Activation Function

- The input received by a perceptron is first multiplied by the respective weights and then, all these weighted inputs are summed together.
- This summed value is then fed to activation for obtaining the final result as shown in the image below followed by the the code:



Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) \stackrel{?}{=} \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$



Activation Function

The sequence of exemplars presented may go like this:

input x	network fires or not (y)	told correct answer (O)	what to do with w's	what to do with t
(0,1,0,0)	0	1	increase	reduce
(1,0,0,0)	0	0	no change	no change
(0,1,1,1)	1	0	reduce	increase
(1,0,1,0)	1	0	reduce	increase
(1,1,1,1)	0	1	increase	reduce
(0,1,0,0)	1	1	no change	no change
....

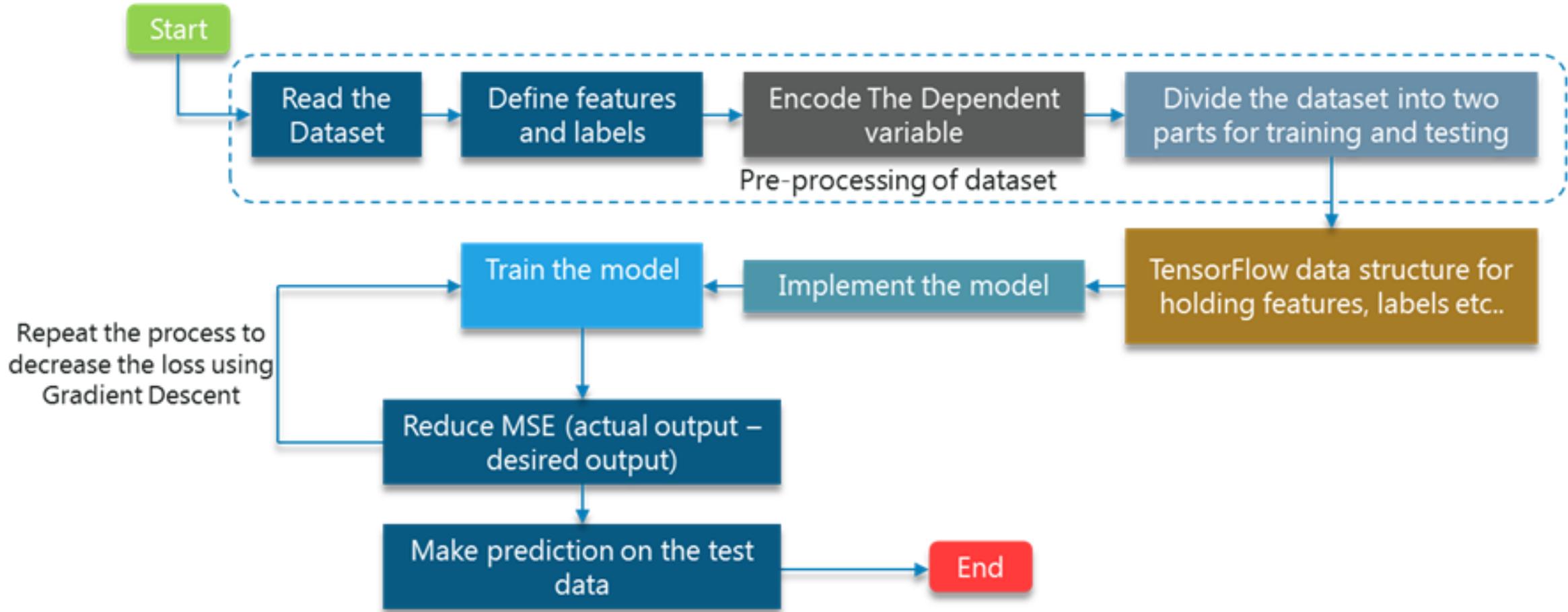
Limitations of Single-Layer Perceptron:

- Well, there are two major problems:
- Single-Layer Perceptrons cannot classify non-linearly separable data points.
- Complex problems, that involve a lot of parameters cannot be solved by Single-Layer Perceptrons.

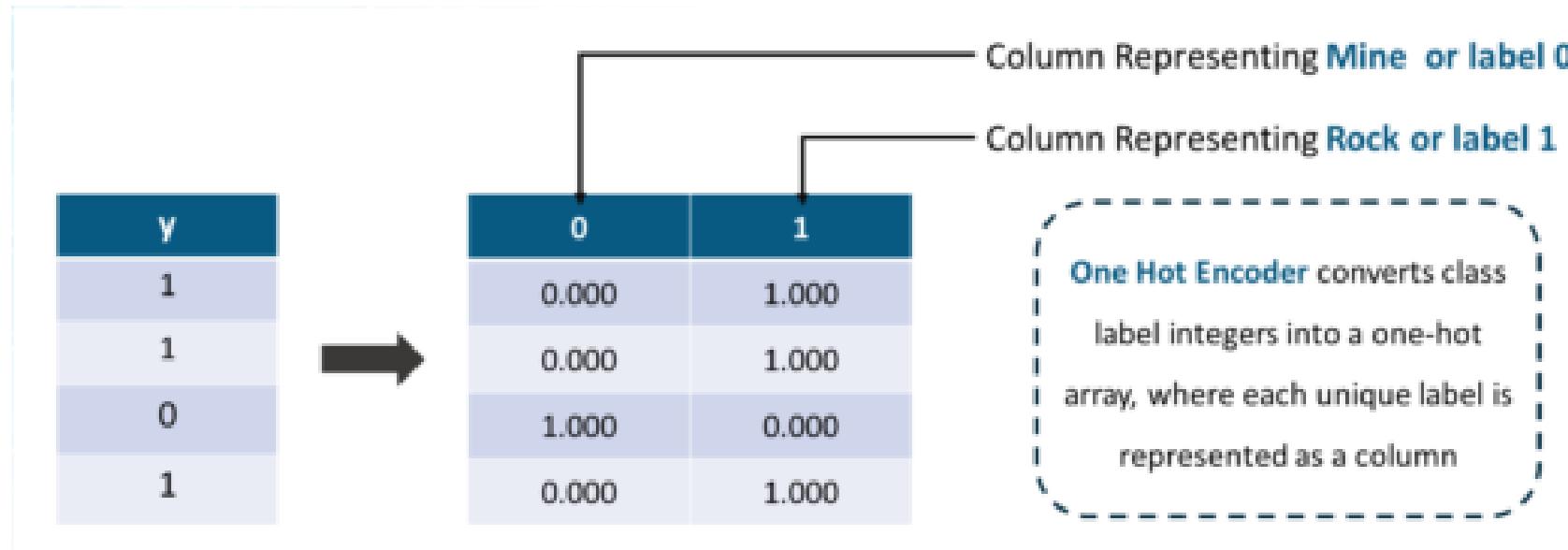
SONAR Data Classification Using Single Layer Perceptrons



SONAR Data Classification Using Single Layer Perceptrons



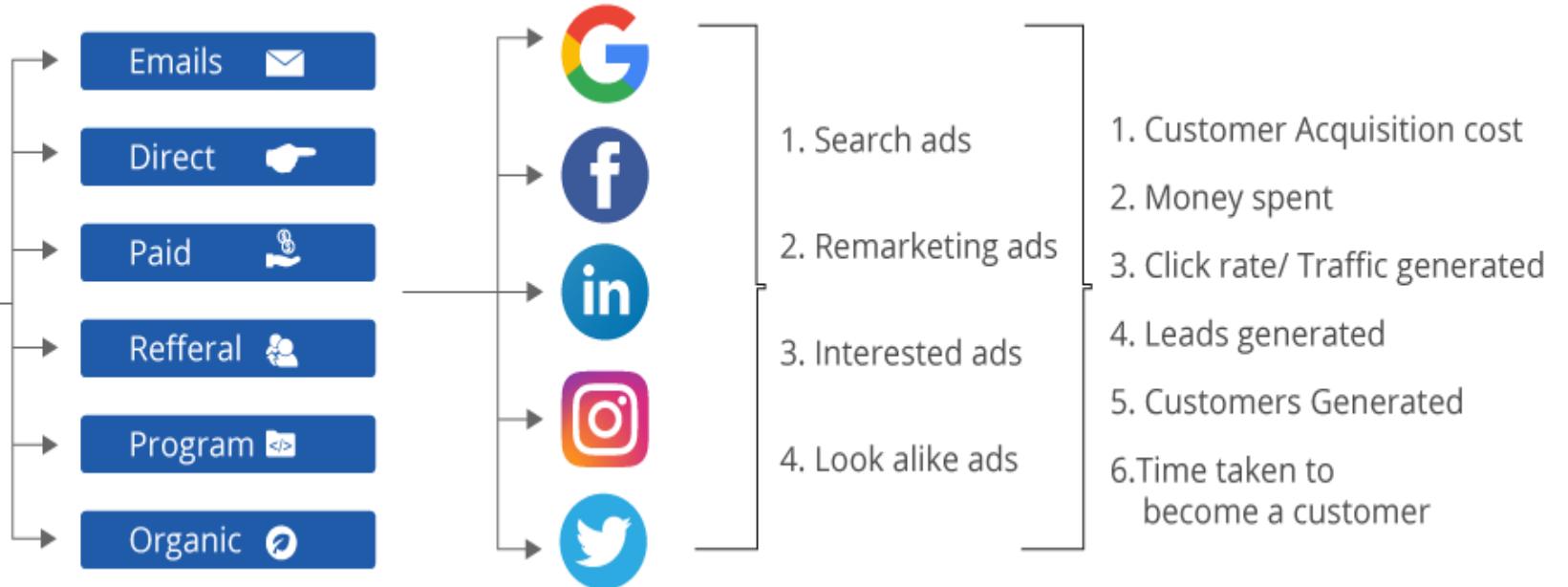
SONAR Data Classification Using Single Layer Perceptrons



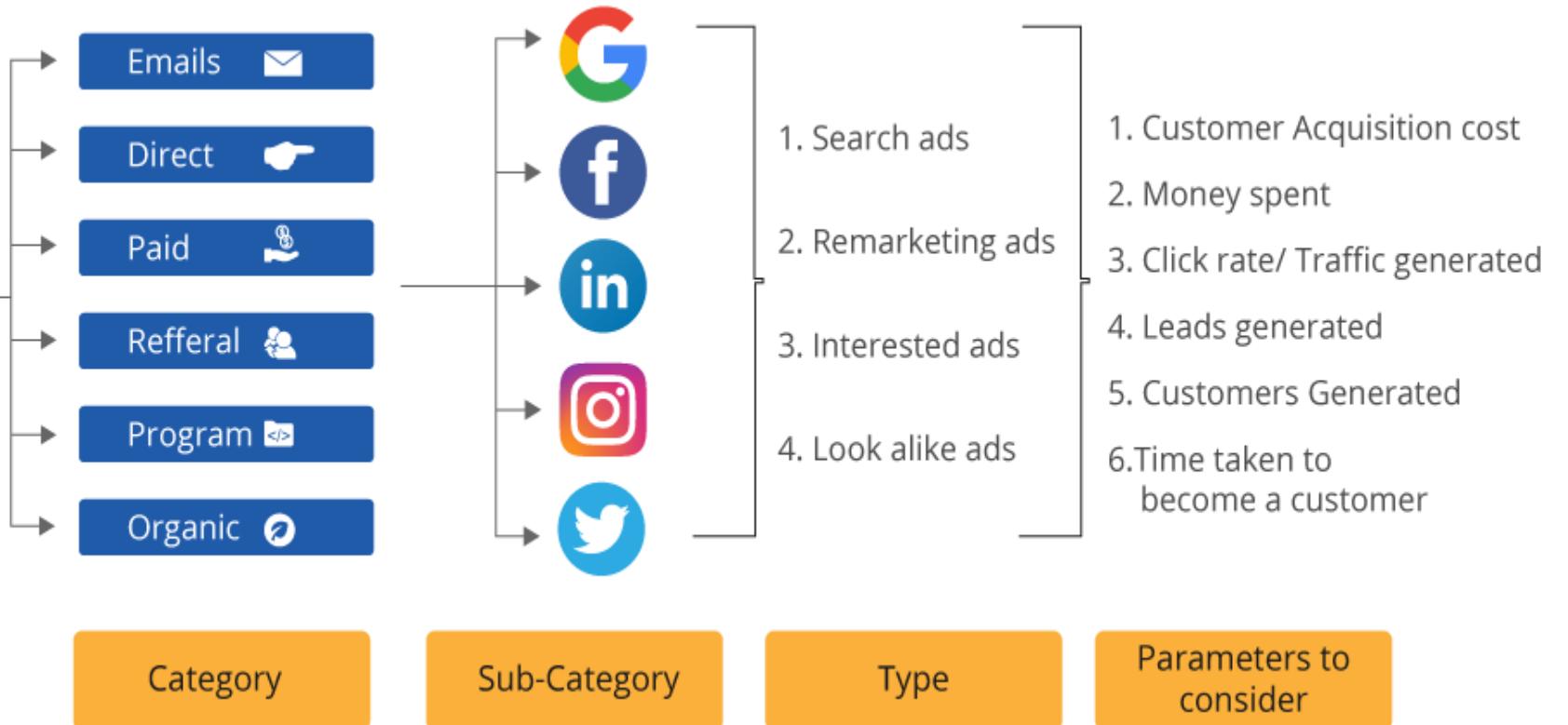
Complex problems, that involve a lot of parameters cannot be solved by Single-Layer Perceptrons:

- The marketing team can market your product through various ways, such as:
- Google Ads
- Personal emails
- Sale advertisement on relevant sites
- Reference program
- Blogs and so on . . .
- Considering all the factors and options available, marketing team has to decide a strategy to do optimal and efficient marketing, but this task is too complex for a human to analyse, because number of parameters are quite high.
- This problem will have to be solved using Deep Learning.

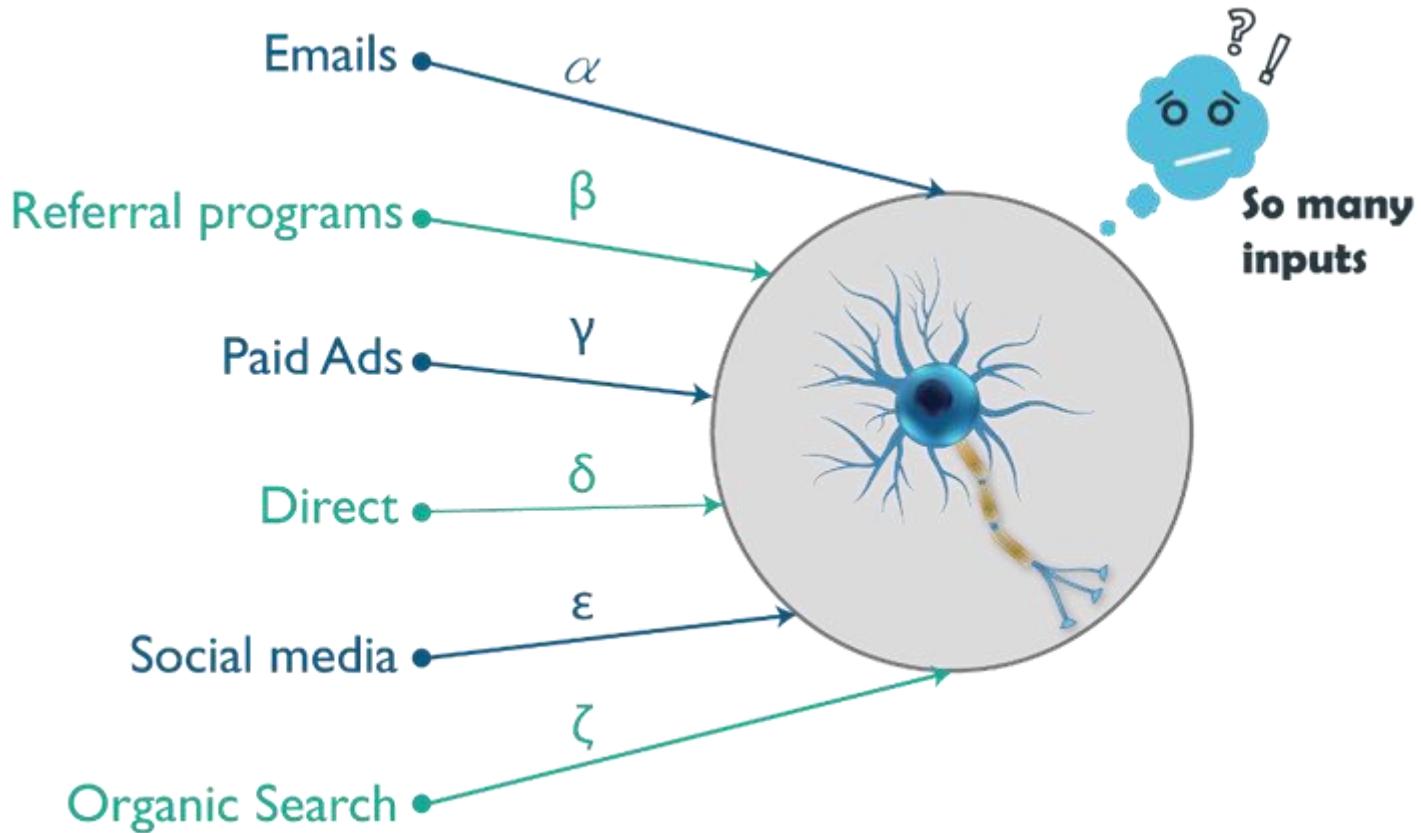
Complex problems, that involve a lot of parameters cannot be solved by Single-Layer Perceptrons:



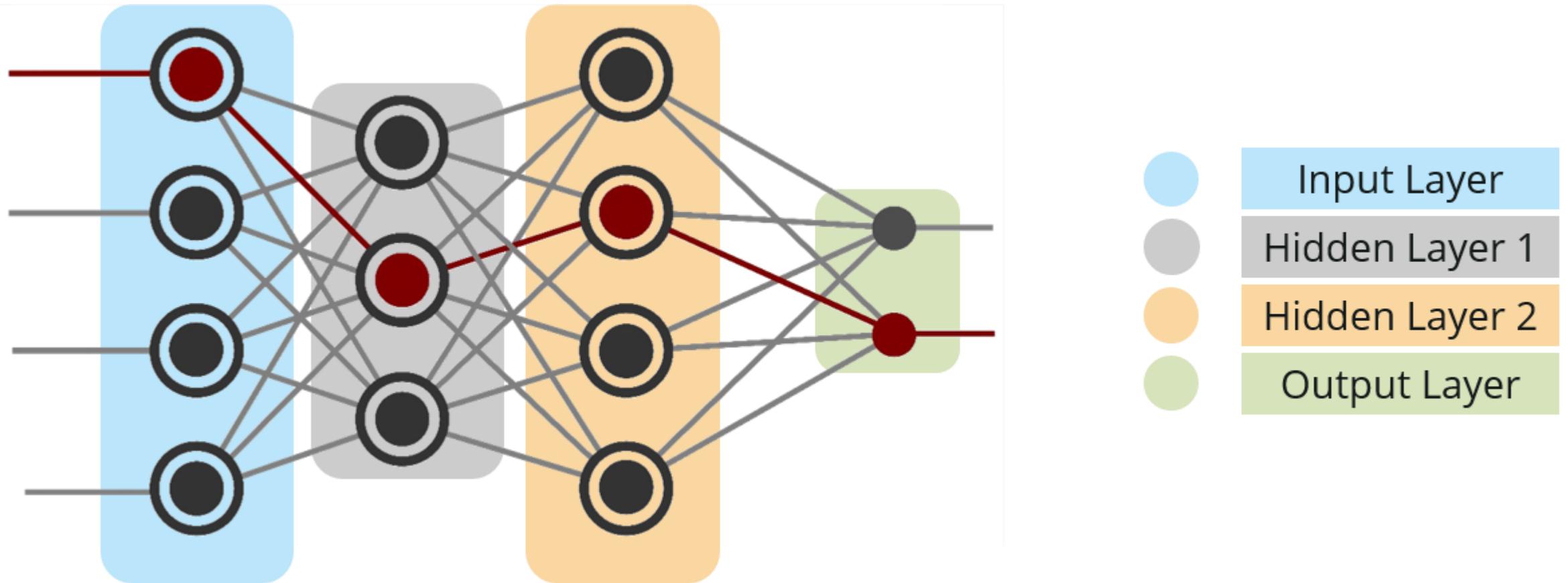
Complex problems, that involve a lot of parameters cannot be solved by Single-Layer Perceptrons:



Complex problems, that involve a lot of parameters cannot be solved by Single-Layer Perceptrons:



What is Multi-Layer Perceptron?



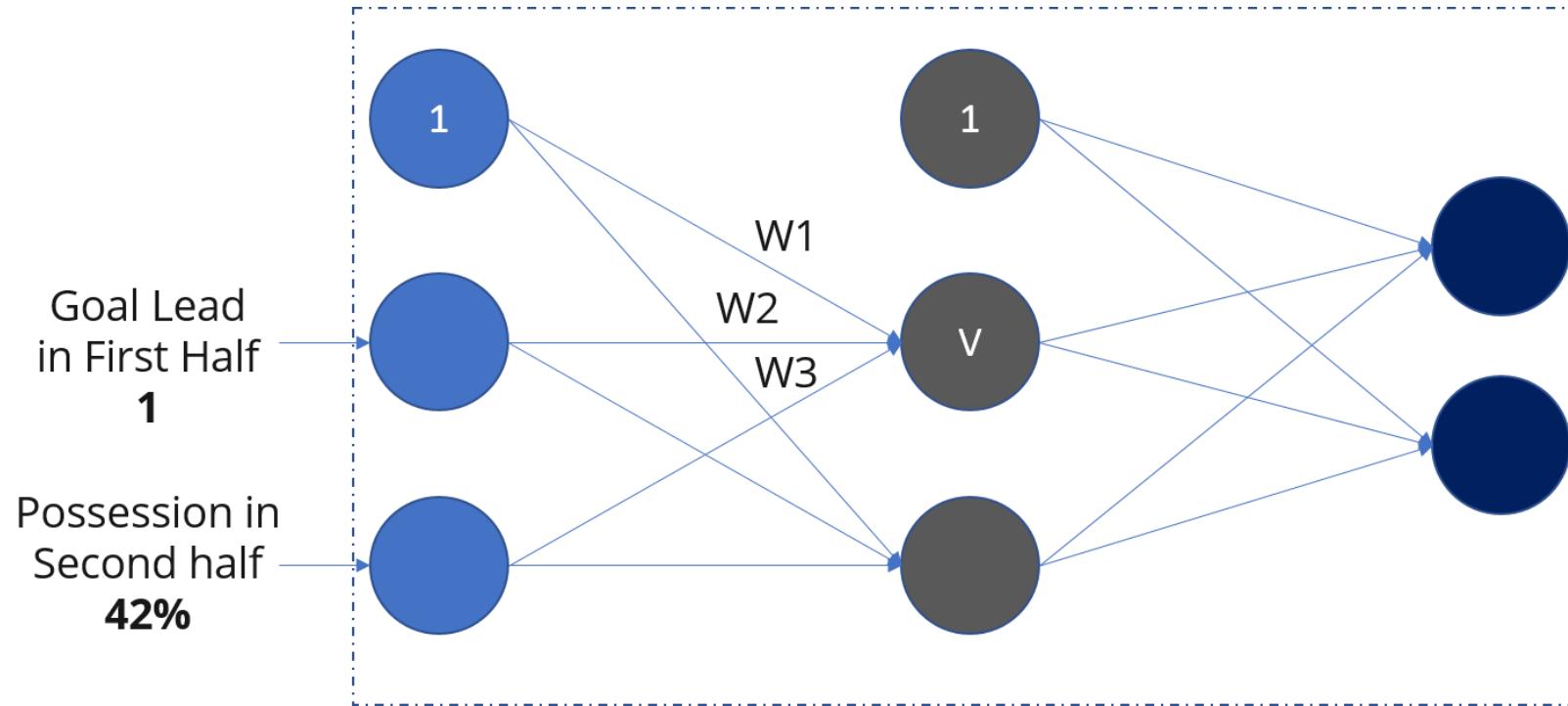
What is Multi-Layer Perceptron?

- Input Nodes – The Input nodes provide information from the outside world to the network and are together referred to as the “Input Layer”. No computation is performed in any of the Input nodes – they just pass on the information to the hidden nodes.
- Hidden Nodes – The Hidden nodes have no direct connection with the outside world (hence the name “hidden”). They perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a “Hidden Layer”. While a network will only have a single input layer and a single output layer, it can have zero or multiple Hidden Layers. A Multi-Layer Perceptron has one or more hidden layers.
- Output Nodes – The Output nodes are collectively referred to as the “Output Layer” and are responsible for computations and transferring information from the network to the outside world.

What is Multi-Layer Perceptron?

Goal Lead in First Half	Possession in Second Half	Won or Lost (1,0)?
0	80%	1
0	35%	0
1	42%	1
2	20%	0
-1	75%	1

- Suppose we have data of a football team, **Chelsea**. The data contains three columns. The last column tells whether Chelsea won the match or they lost it. The other two columns are about, goal lead in the first half and possession in the second half. Possession is the amount of time for which the team has the ball in percentage. So, if I say that a team has 50% possession in one half (45 minutes), it means that, the team had ball for 22.5 minutes out of 45 minutes.



Probability of winning = 0.4 Target = 1

$$\text{Error} = (1-0.4) = 0.6$$

Probability of losing = 0.6 Target = 0

$$\text{Error} = (0-0.6) = -0.6$$

Forward Propagation:

- Here, we will propagate forward, i.e. calculate the weighted sum of the inputs and add bias. In the output layer we will use the softmax function to get the probabilities of Chelsea winning or loosing.
- If you notice the diagram, winning probability is 0.4 and loosing probability is 0.6. But, according to our data, we know that when goal lead in the first half is 1 and possession in the second half is 42% Chelsea will win. Our network has made wrong prediction.
- If we see the error (Comparing the network output with target), it is 0.6 and -0.6.

Backward Propagation and Weight Updation:

- We calculate the total error at the output nodes and propagate these errors back through the network using Backpropagation to calculate the gradients. Then we use an optimization method such as Gradient Descent to ‘adjust’ all weights in the network with an aim of reducing the error at the output layer.
- Let me explain you how the gradient descent optimizer works:
- Step – 1: First we calculate the error, consider the equation below

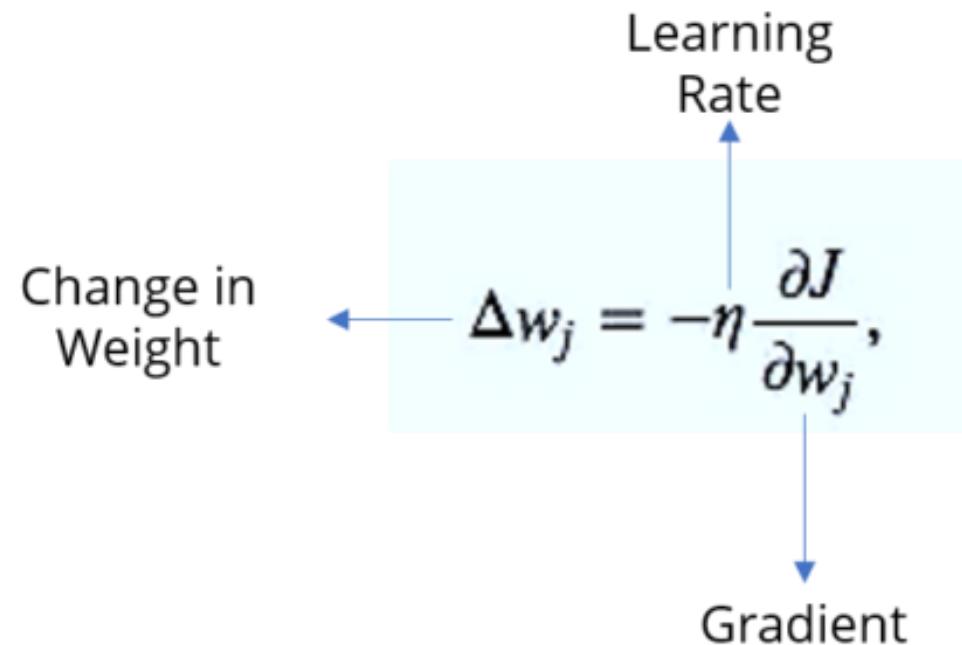
Network Output

$$\text{Error/Loss} \leftarrow J(\mathbf{w}) = \frac{1}{2} \sum_i (\text{target}^{(i)} - \text{output}^{(i)})^2$$

Actual Output

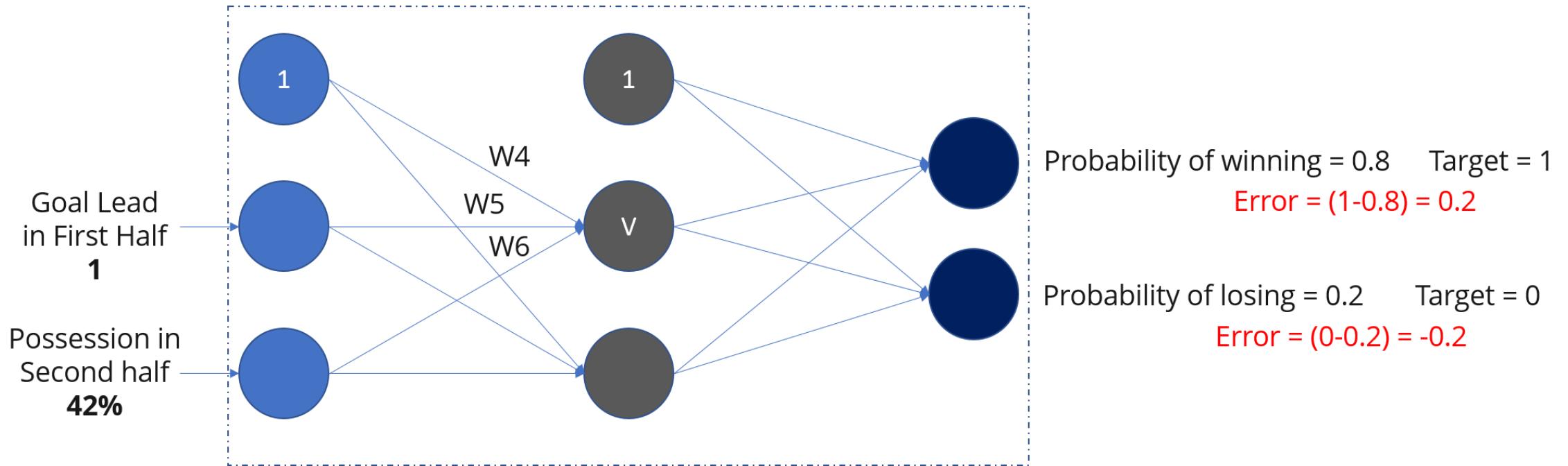
Backward Propagation and Weight Updation

- Step – 2: Based on the error we got, it will calculate the rate of change of error w.r.t change in the weights.



Backward Propagation and Weight Updation

- Step – 3: Now, based on this change in weight, we will calculate the new weight value.
- If we now input the same example to the network again, the network should perform better than before since the weights have now have been adjusted to minimize the error in prediction. Consider the example below, As shown in Figure, the errors at the output nodes now reduce to [0.2, -0.2] as compared to [0.6, -0.4] earlier. This means that our network has learnt to correctly classify our first training example.



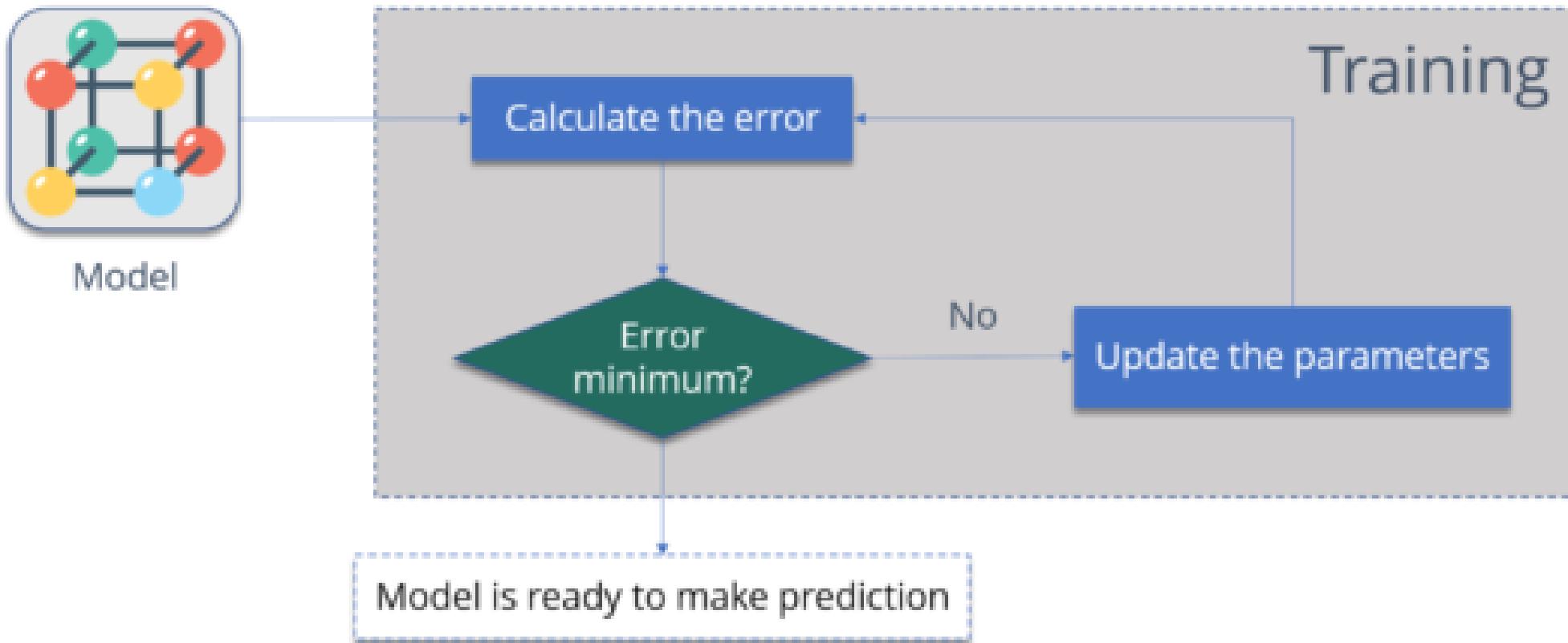
What is Multi-Layer Perceptron?

- Now, suppose, we want to predict whether Chelsea will win the match or not, if the goal lead in the first half is 2 and the possession in the second half is 32%.
- This is a binary classification problem where a multi layer Perceptron can learn from the given examples (training data) and make an informed prediction given a new data point.
- The process by which a Multi Layer Perceptron learns is called the **Backpropagation** algorithm.

Backpropagation:

- Backpropagation is a supervised learning algorithm, for training Multi-layer Perceptrons (Artificial Neural Networks).
- **Why We Need Backpropagation?**
- While designing a Neural Network, in the beginning, we initialize weights with some random values or any variable for that fact.
- Now obviously, we are not *superhuman*. So, it's not necessary that whatever weight values we have selected will be correct, or it fits our model the best.
- Okay, fine, we have selected some weight values in the beginning, but our model output is way different than our actual output i.e. the error value is huge.
- Now, how will you reduce the error?
- Basically, what we need to do, we need to somehow explain the model to change the parameters (weights), such that error becomes minimum.

Backpropagation:



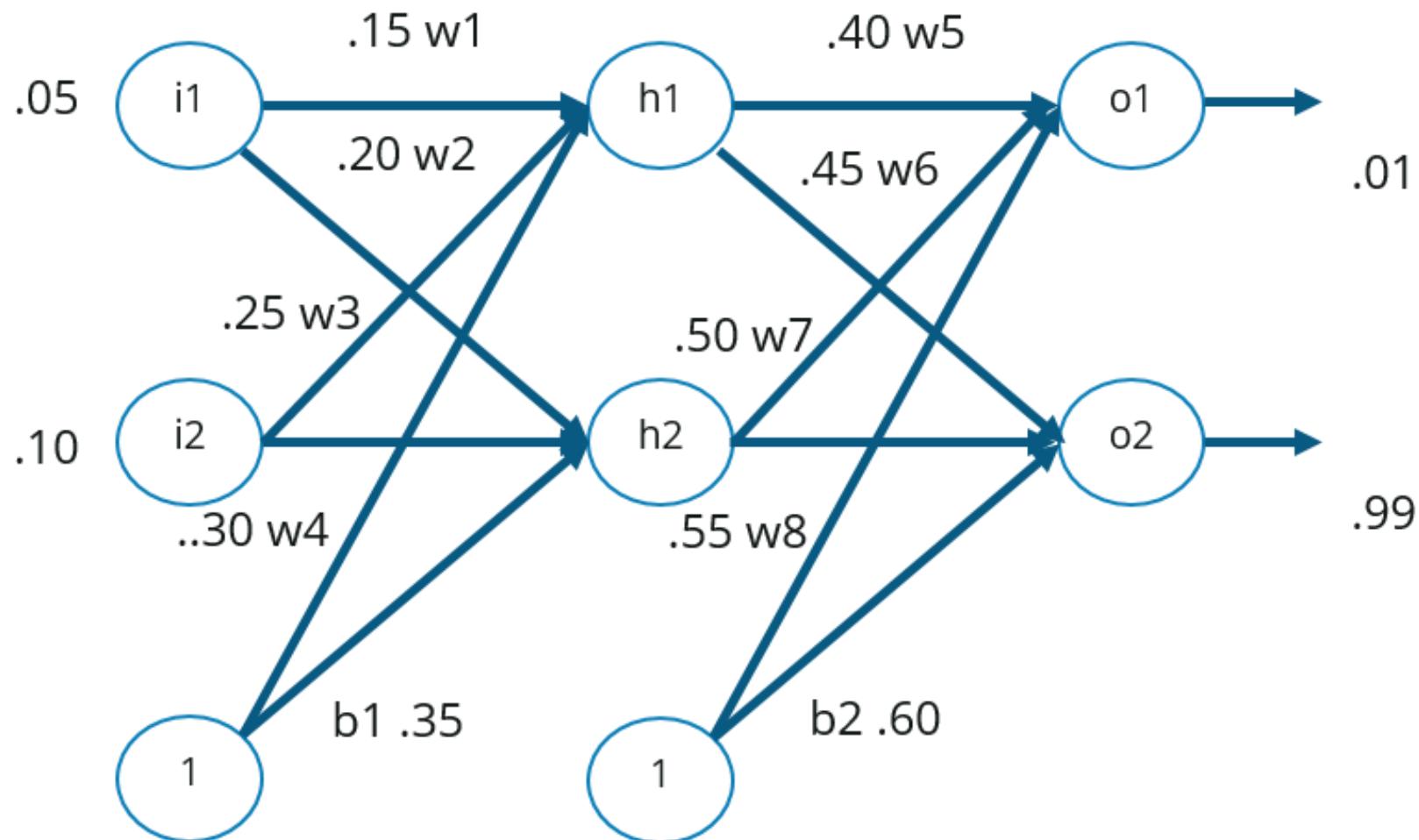
Backpropagation:

- Let me summarize the steps for you:
- Calculate the error – How far is your model output from the actual output.
- Error minimum? – Check whether the error is minimized or not.
- Update the parameters – If the error is huge then, update the parameters (weights and biases). After that again check the error. Repeat the process until the error becomes minimum.
- Model is ready to make a prediction – Once the error becomes minimum, you can feed some inputs to your model and it will produce the output.

Backpropagation Algorithm

- initialize network weights (often small random values)
- do
 - forEach training example named ex
 - prediction = neural-net-output(network, ex) // forward pass
 - actual = teacher-output(ex)
 - compute error (prediction - actual) at the output units
 - compute $\Delta w_{\{h\}}$ for all weights from hidden layer to output layer // backward pass
 - compute $\Delta w_{\{i\}}$ for all weights from input layer to hidden layer // backward pass continued
 - update network weights // input layer not modified by error estimate
- until all examples classified correctly or another stopping criterion satisfied
- return the network

How Backpropagation Works?



Step – 1: Forward Propagation

Net Input For h1:

$$\text{net } h1 = w1*i1 + w2*i2 + b1*1$$

$$\text{net } h1 = 0.15*0.05 + 0.2*0.1 + 0.35*1 = 0.3775$$

Output Of h1:

$$\text{out } h1 = 1/(1+e^{-\text{net } h1})$$

$$1/(1+e^{-.3775}) = 0.593269992$$

Output Of h2:

$$\text{out } h2 = 0.596884378$$

We will repeat this process for the output layer neurons, using the output from the hidden layer neurons as inputs.

Output For o1:

$$\text{net o1} = w5 * \text{out h1} + w6 * \text{out h2} + b2 * 1 \rightarrow 0.4 * 0.593269992 + 0.45 * 0.596884378 + 0.6 * 1 = 1.105905967$$

$$\text{Out o1} = 1 / (1 + e^{-\text{net o1}}) \rightarrow 1 / (1 + e^{-1.105905967}) = 0.75136507$$

Output For o2:

$$\text{Out o2} = 0.772928465$$

Now, let's see what is the value of the error:

Error For o1:

$$E_{o1} = \frac{1}{2}(\text{target} - \text{output})^2$$

$$\frac{1}{2} (0.01 - 0.75136507)^2 = 0.274811083$$

Error For o2:

$$E_{o2} = 0.023560026$$

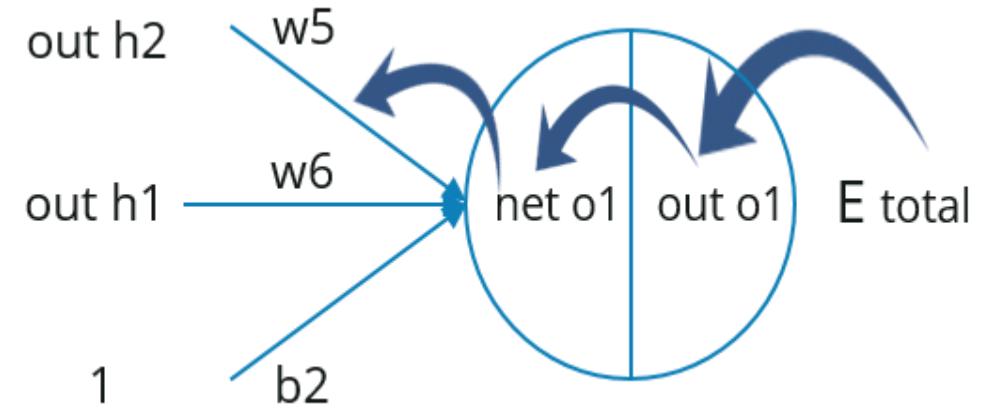
Total Error:

$$E_{\text{total}} = E_{o1} + E_{o2}$$

$$0.274811083 + 0.023560026 = 0.298371109$$

Step – 2: Backward Propagation

$$\frac{\delta E_{total}}{\delta w5} = \frac{\delta E_{total}}{\delta out\ o1} * \frac{\delta out\ o1}{\delta net\ o1} * \frac{\delta net\ o1}{\delta w5}$$



Since we are propagating backwards, first thing we need to do is, calculate the change in total errors w.r.t the output O1 and O2.

$$E_{\text{total}} = 1/2(\text{target o1} - \text{out o1})^2 + 1/2(\text{target o2} - \text{out o2})^2$$

$$\frac{\delta E_{\text{total}}}{\delta \text{out o1}} = -(\text{target o1} - \text{out o1}) = -(0.01 - 0.75136507) = 0.74136507$$

$$\text{out o1} = 1/(1+e^{-net o1})$$

$$\frac{\delta \text{out o1}}{\delta net o1} = \text{out o1} (1 - \text{out o1}) = 0.75136507 (1 - 0.75136507) = 0.186815602$$

Let's see now how much does the total net input of O1 changes w.r.t W5?

$$\text{net o1} = w5 * \text{out h1} + w6 * \text{out h2} + b2 * 1$$

$$\frac{\delta \text{net o1}}{\delta w5} = 1 * \text{out h1 } w5^{(1-1)} + 0 + 0 = 0.593269992$$

Step – 3: Putting all the values together and calculating the updated weight value

$$\frac{\delta E_{total}}{\delta w_5} = \frac{\delta E_{total}}{\delta o_{out 1}} * \frac{\delta o_{out 1}}{\delta n_{net 01}} * \frac{\delta n_{net 01}}{\delta w_5}$$

0.082167041

$$w_5^+ = w_5 - n \frac{\delta E_{total}}{\delta w_5}$$

$$w_5^+ = 0.4 - 0.5 * 0.082167041$$

Updated w_5

0.35891648

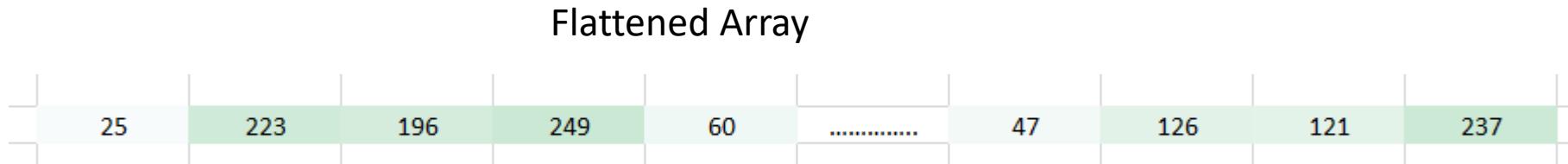
How does a machine look at an image?

- In simple terms, every image is an arrangement of dots (a pixel) arranged in a special order. If you change the order or color of a pixel, the image would change as well.
- The machine will basically break this image into a matrix of pixels and store the color code for each pixel at the representative location.
- In the representation below – number 1 is white and 256 is the darkest shade of green color (I have constrained the example to have only one color for simplicity).

How does a machine look at an image?

25	2	1	44
223	7	6	60
196	8	2	148
249	1	3	40
60	7	1	154
59	1	7	213
214	7	3	163
89	182	219	13
74	146	113	72
89	18	244	85
1	4	8	97
3	4	2	121
2	1	2	131
7	6	8	47
3	5	5	126
7	6	8	121
5	3	1	237

How do we help a neural network to identify images ?



Using Artificial Neural Network, we need to figure out, if the bank notes are real or fake?



4.0127	10.1477	-3.9366	-4.0728	0
2.6606	3.1681	1.9619	0.18662	0
3.931	1.8541	-0.02343	1.2314	0
0.01727	8.693	1.3989	-3.9668	0
3.2414	0.40971	1.4015	1.1952	0
2.2504	3.5757	0.35273	0.2836	0
-1.3971	3.3191	-1.3927	-1.9948	1
0.39012	-0.14279	-0.03199	0.35084	1
-1.6677	-7.1535	7.8929	0.96765	1
-3.8483	-12.8047	15.6824	-1.281	1
-3.5681	-8.213	10.083	0.96765	1
-2.2804	-0.30626	1.3347	1.3763	1
-1.7582	2.7397	-2.5323	-2.234	1
-0.89409	3.1991	-1.8219	-2.9452	1
0.3434	0.12415	-0.28733	0.14654	1

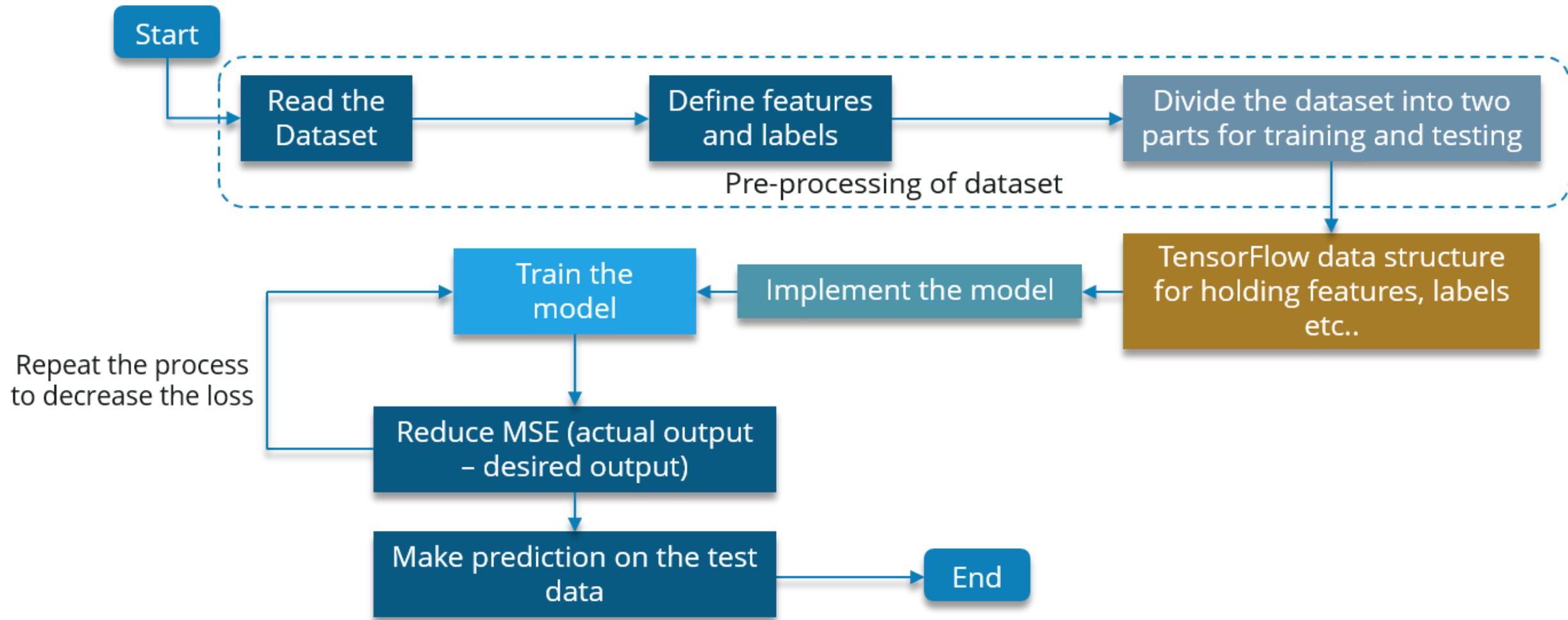
Features

- Variance of Wavelet Transformed image
- Skewness of Wavelet Transformed image
- Kurtosis of Wavelet Transformed image
- Entropy of image

Label

1 - Real, 0 - Fake

Backpropagation



Convolutional Neural Networks

- CNNs, like neural networks, are made up of neurons with learnable weights and biases.
- Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output.
- The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply on CNNs.

Defining a Convolutional Neural Network

- We need three basic components to define a basic convolutional network.
- The convolutional layer
- The Pooling layer[optional]
- The output layer

The Convolution Layer

- Suppose we have an image of size 6*6. We define a weight matrix which extracts certain features from the images

INPUT IMAGE					
18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

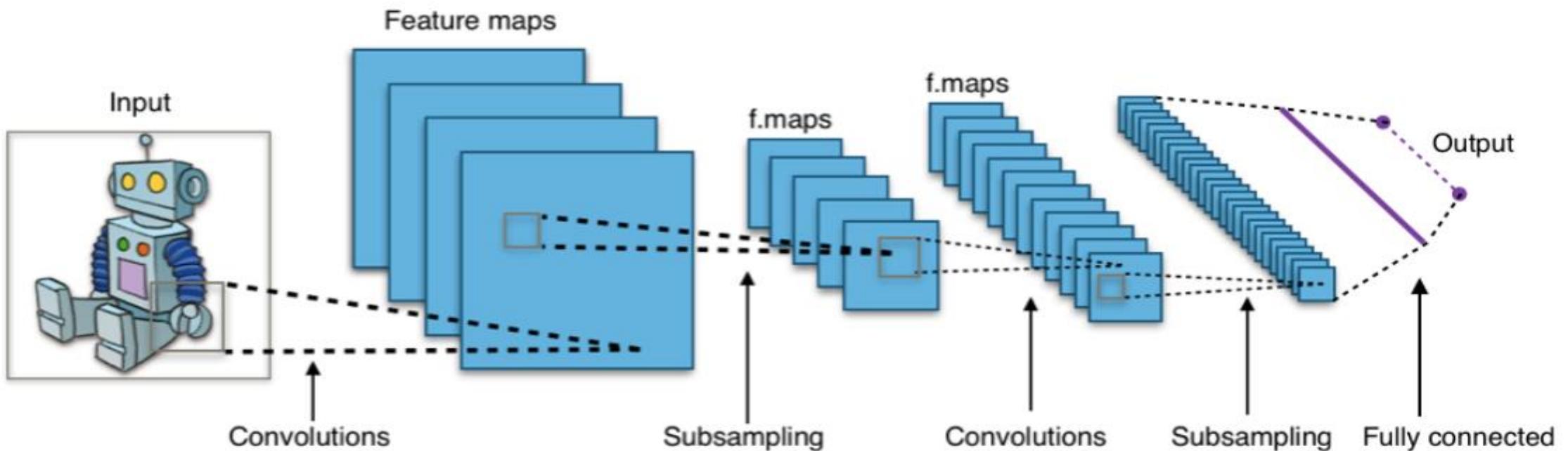
WEIGHT		
1	0	1
0	1	0
1	0	1

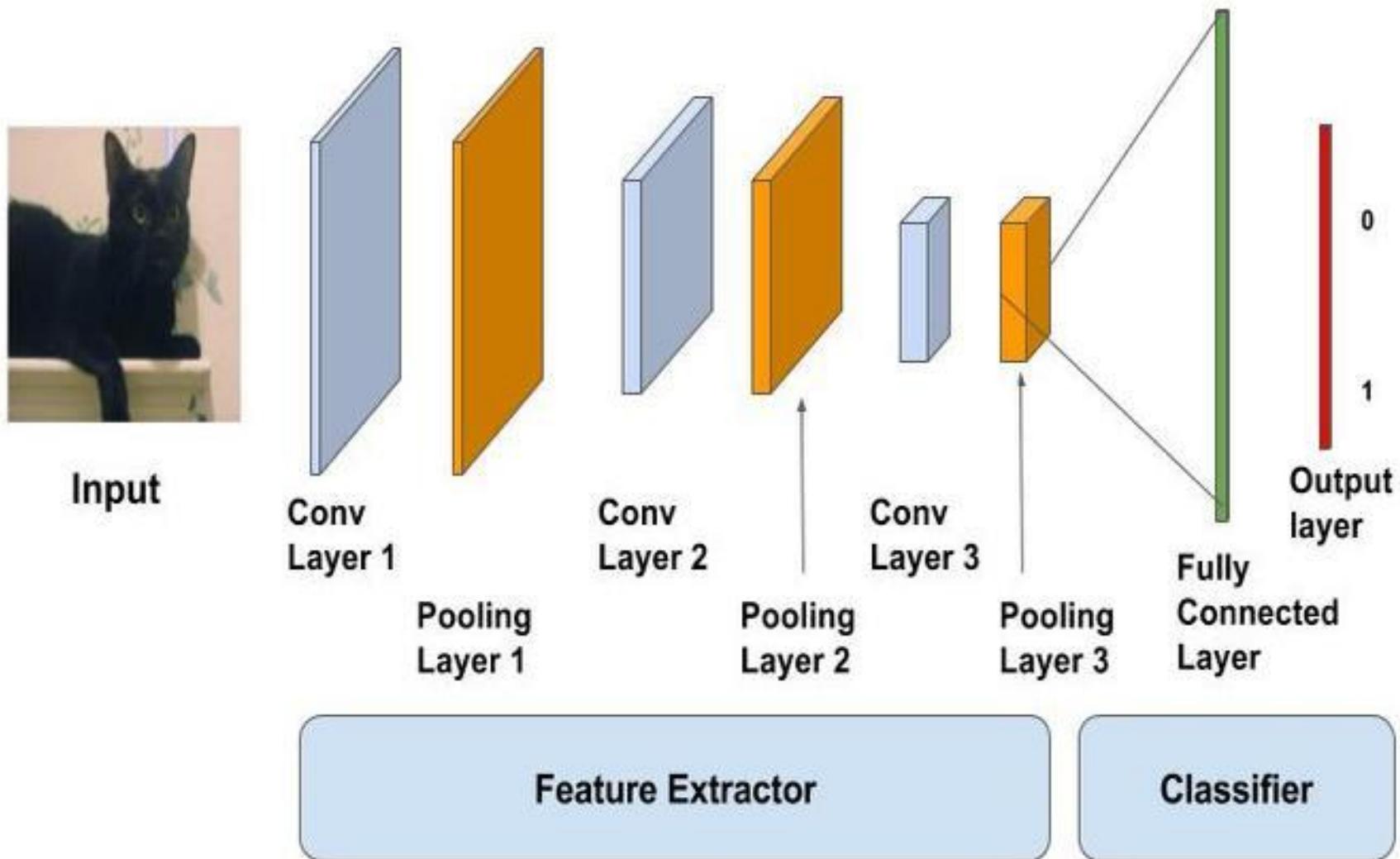
429

INPUT IMAGE					
18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT		
1	0	1
0	1	0
1	0	1

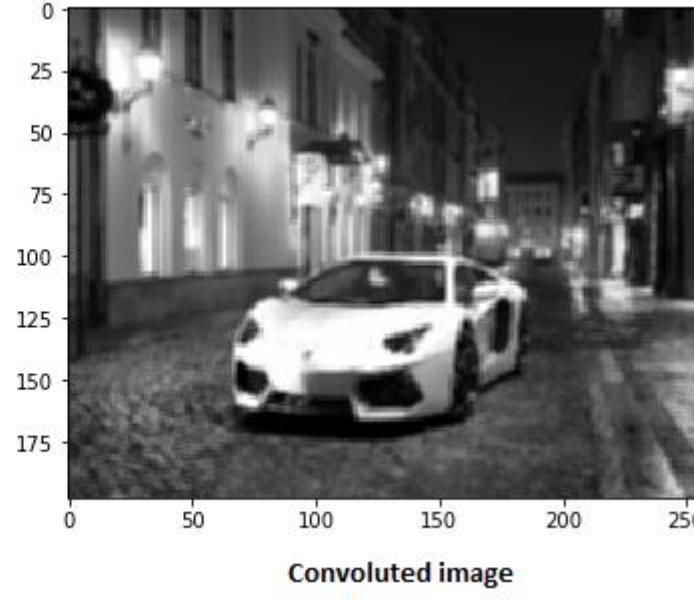
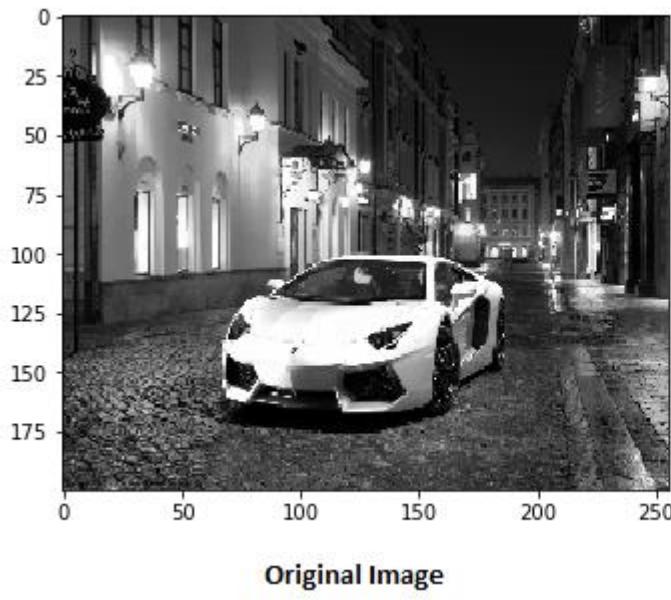
429





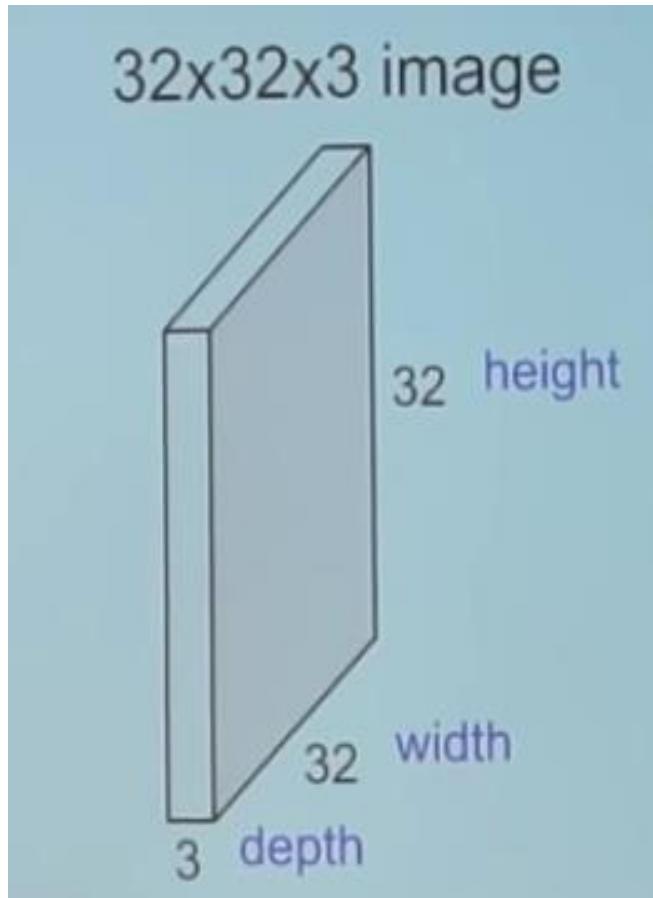
The Convolution Layer

- Suppose we have an image of size $6*6$. We define a weight matrix which extracts certain features from the images



How are Convolutional Neural Networks different than Neural Networks?

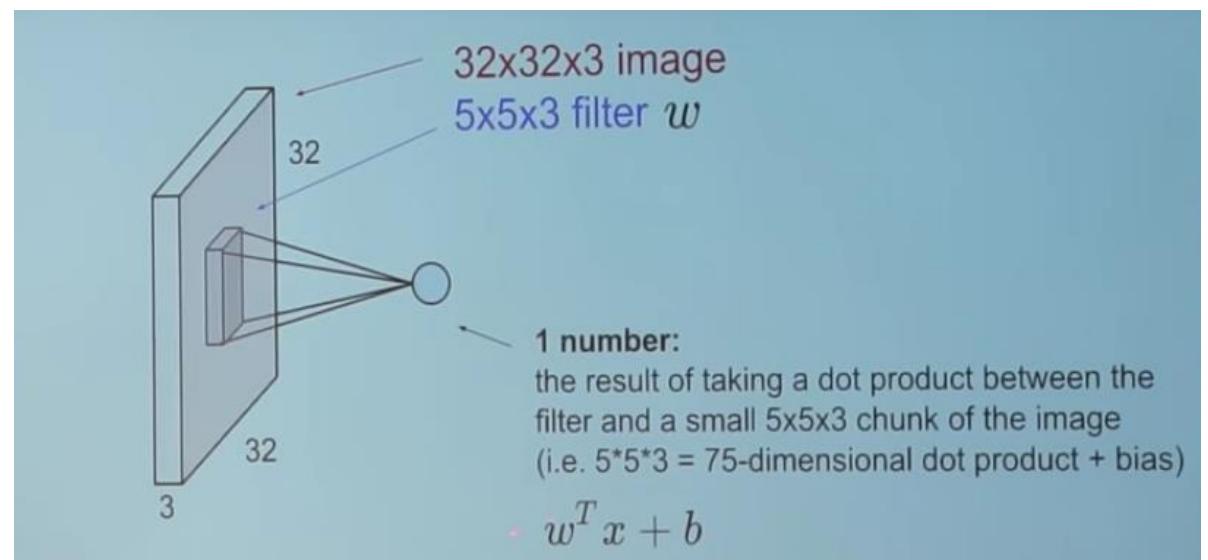
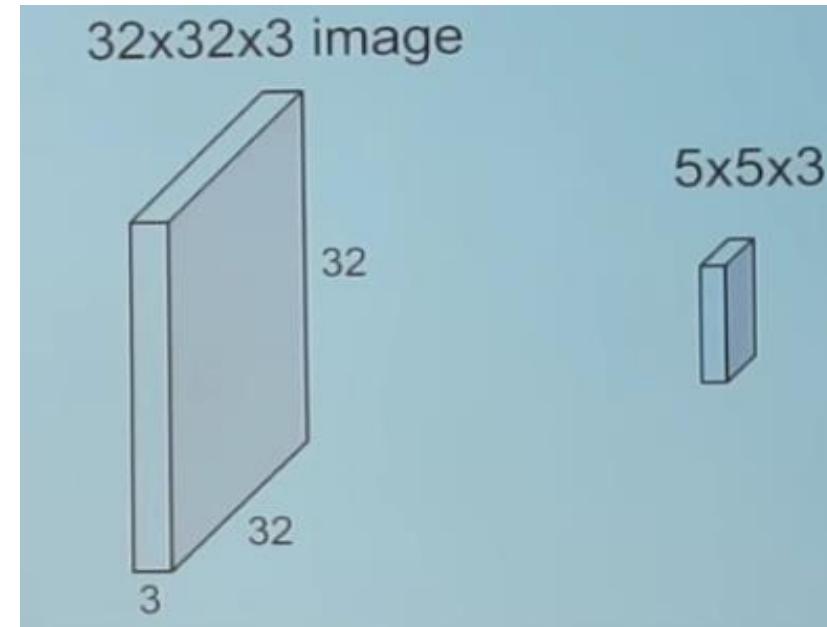
CNNs operate over Volumes !

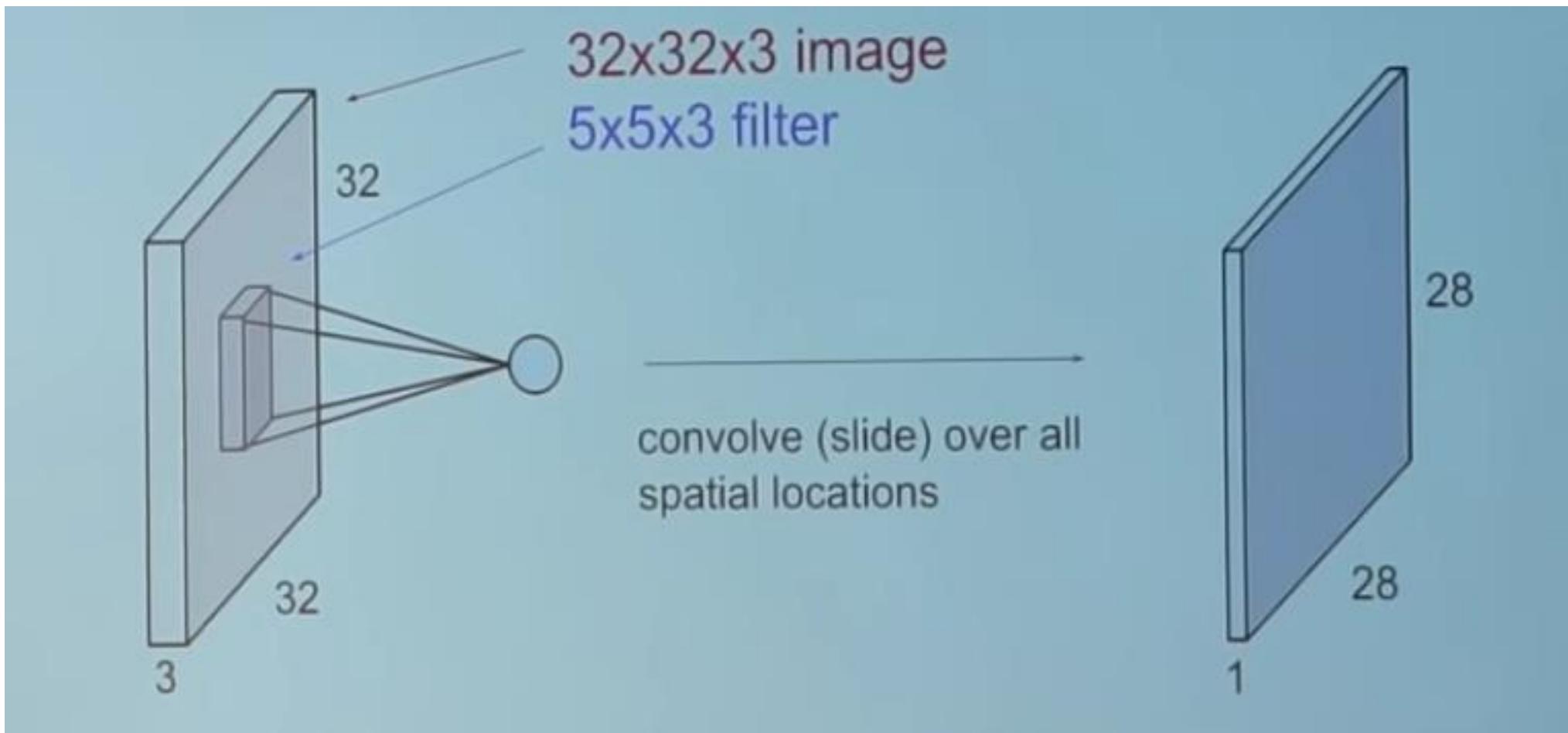


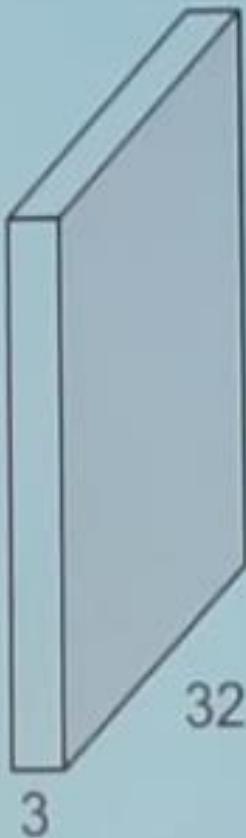
Unlike neural networks, where the input is a vector, here the input is a multi-channeled image (3 channeled in this case).

Convolution

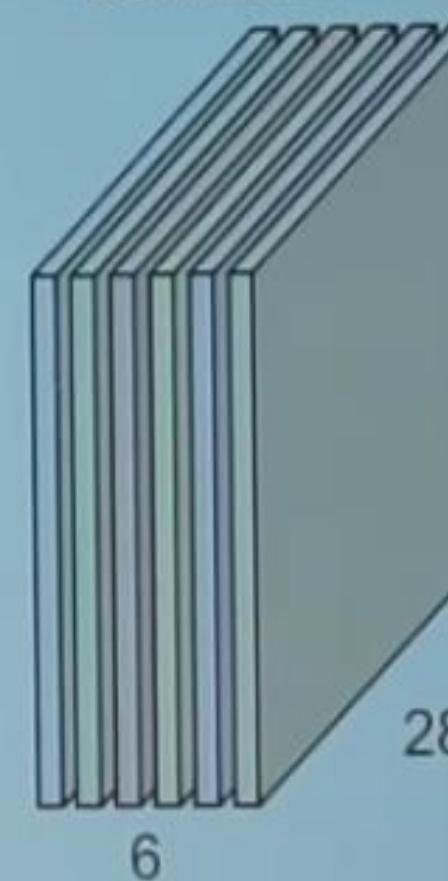
- Convolution is a mathematical way of combining two signals to form a third signal.
- It is the single most important technique in Digital Signal Processing.
- Using the strategy of impulse decomposition, systems are described by a signal called the *impulse response*.
- Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response.





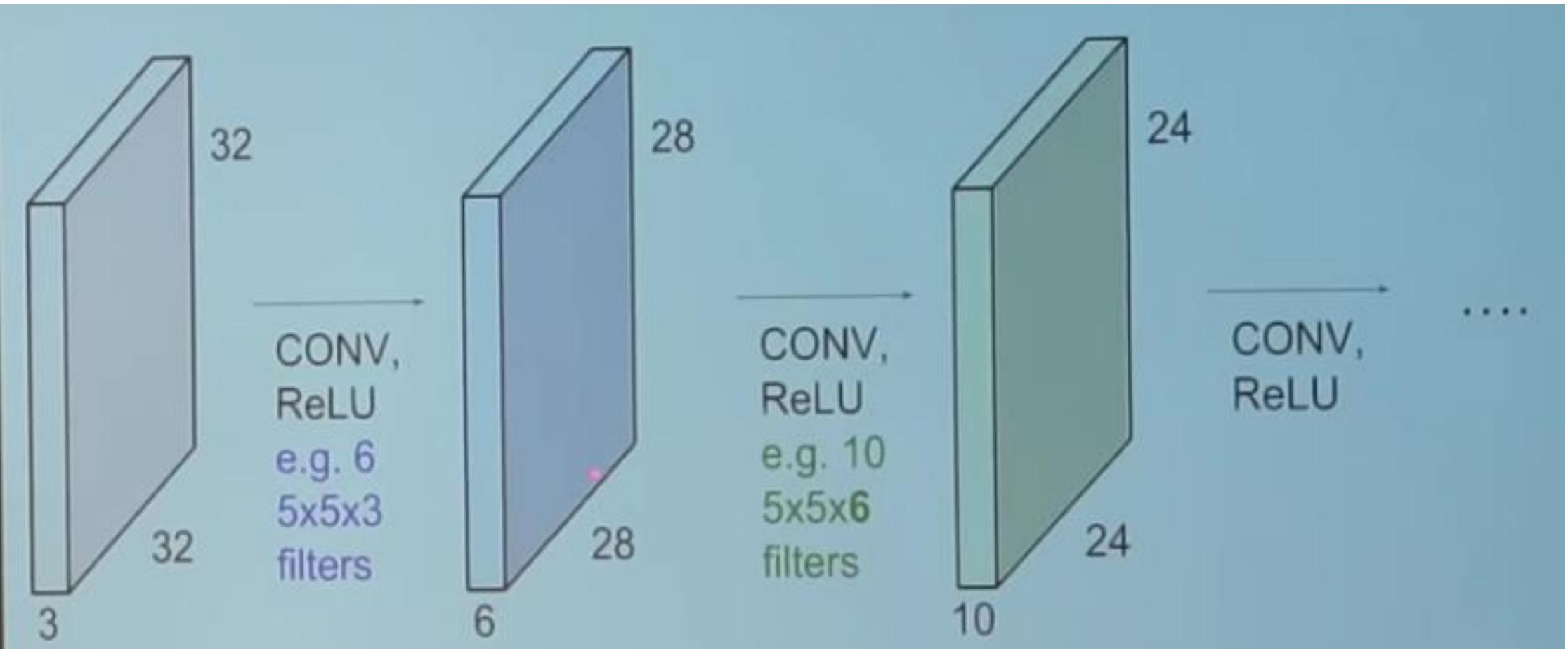


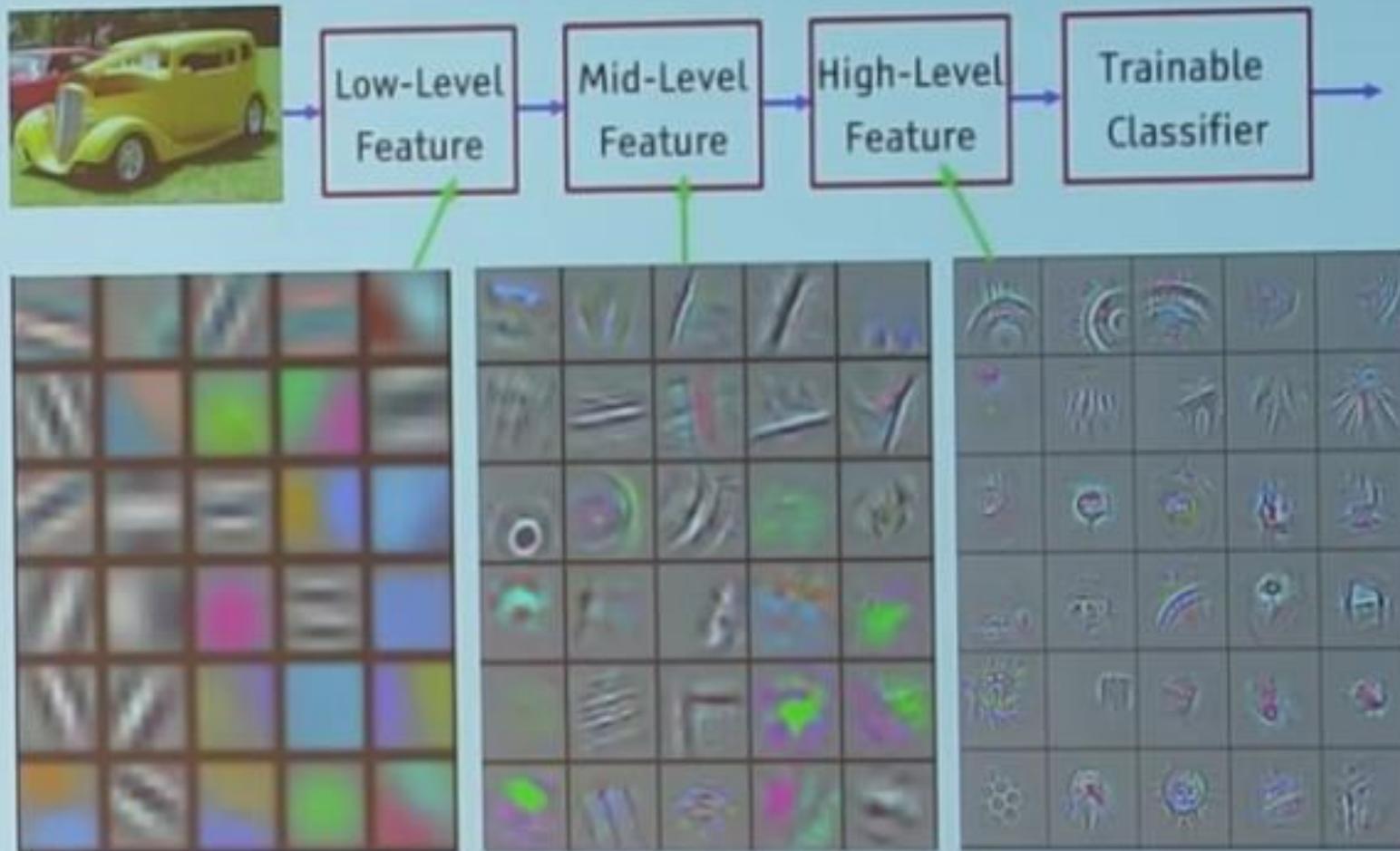
Convolution Layer



activation maps

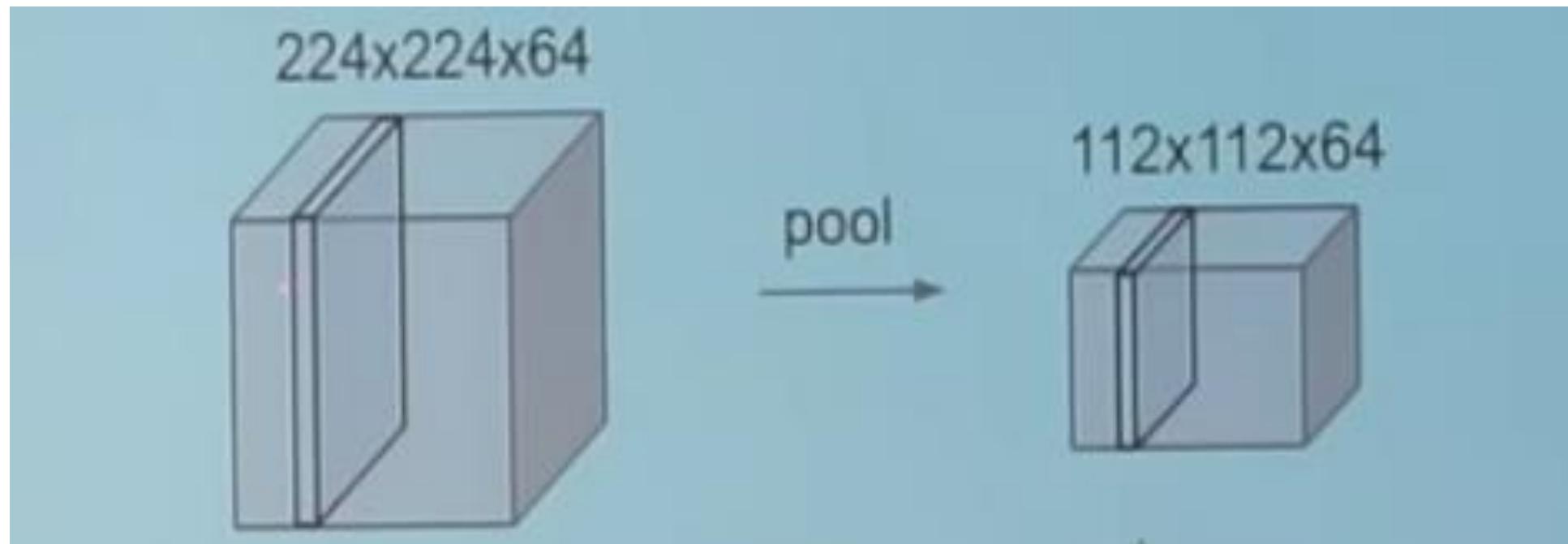
We stack these up to get a “new image” of size $28 \times 28 \times 6$!



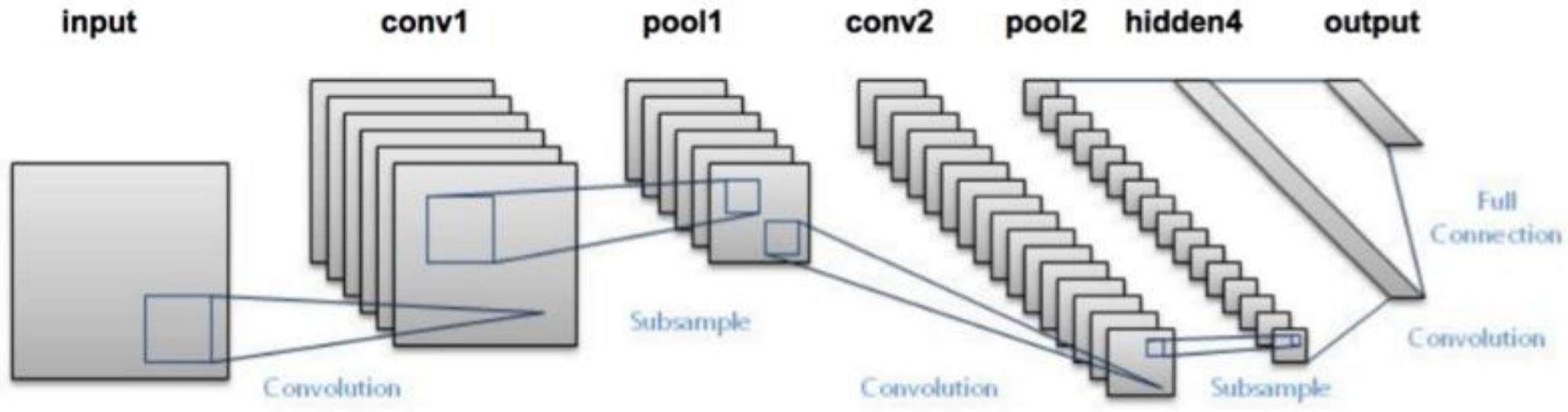


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

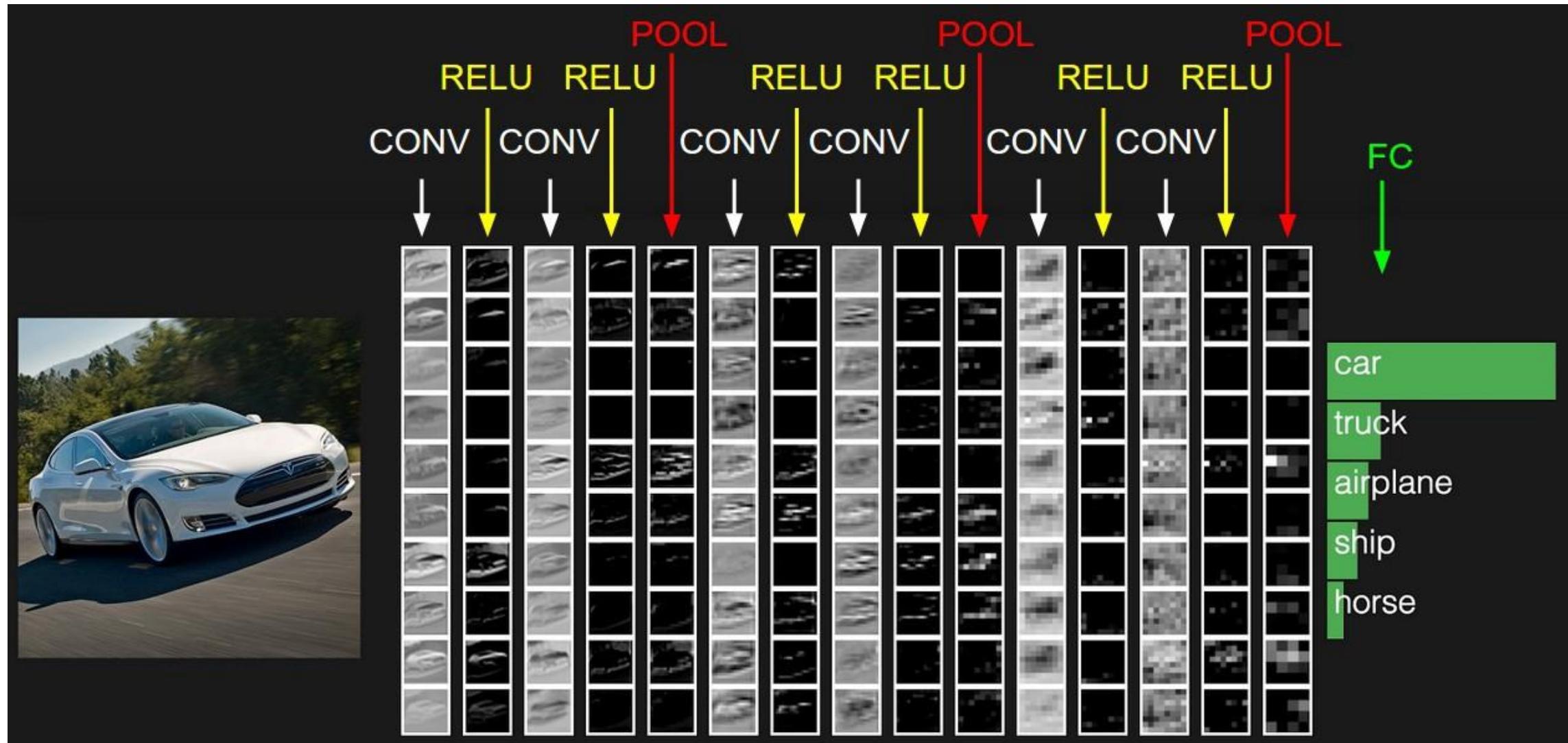
Pooling Layers



Putting it all together – How does the entire network look like ?



Typical Architecture of a CNN



Reinforcement Learning

- Reinforcement Learning focusses on getting machines and software agents to automatically determine the best behaviour within a specific context, in order to maximize performance.
- When the machine takes an action that increases the performance it gets an simple reward feedback, this is known as the reinforcement signal hence the name reinforcement learning.
- In other words, the machine is learning a behaviour based on feedback from the environment, this process can be done once and for all or keep on improving preformance as time goes by.

Armed Bandit problem

- A single armed Bandit process is a machine that is characterized by random sequences, and can be found in different states where some states deliver reward and others a penalty.
- In our example the slot machine changes to different states when engaged and results in a return on investment when we win and no return when we lose.

Multi Armed Bandit problem

- The multi armed Bandit problem is a collection of K single armed Bandit processes and one controller.
- The controller can choose to operate only one machine at a time, the other machines remain unchanged.
- We assume that every machine has a different probability distribution on giving an reward to the controller, in other words each machine has a fixed probability of winning and losing.
- Our goal is to find the machine with the highest probability of winning, machine with the highest average return.
- The problem lies in the fact that we do not know which machine has the highest average return, nor can we discriminate between the machines because they look the same.
- We need a strategy that in each round tells me which machine to try next.

The Upper Confidence Bound Algorithm

- The Upper Confidence Bound Algorithm is a Reinforcement Learning Algorithm that learns an optimal behavior by minimizing regret, following an optimistic strategy, exploring/exploiting and confidence intervals.
- Minimizing regret means to minimize the cost of not knowing which of the K machines is optimal, an optimal machine having the highest average return.
- Regret at a given round t (pink formula) is calculated with the mean distribution of the optimal machine minus that of the current machine.
- The total regret (RT) being the sum of regrets.
- RT divided by T (total rounds) should be approaching zero when T approaches infinite, ensuring that $RT/T \rightarrow 0$ is stronger than maximizing rewards, since in the limit we discover the true optimal machine.

The Upper Confidence Bound Algorithm

$$r_t = \mu^* - \mu_{a_t}$$

$$R_T = \sum_{t=1}^T r_t$$

$$\frac{R_T}{T} \rightarrow 0 \text{ as } T \rightarrow \infty$$

UCB

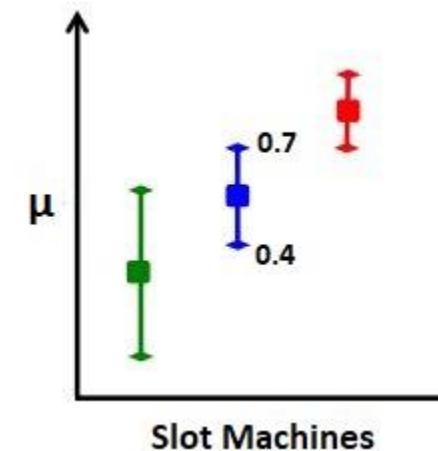
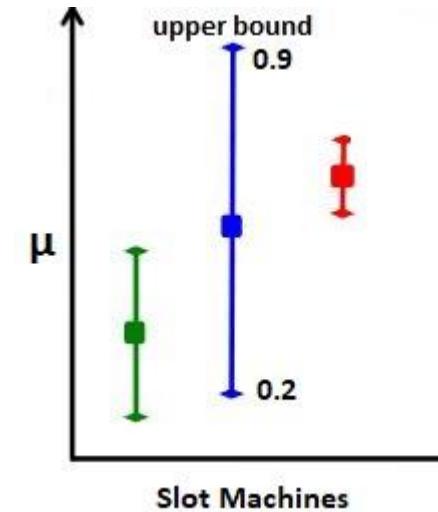
- The UCB follows a optimistic strategy because the algorithm believes it can obtain extra rewards by reaching the unexplored parts of the state space.
- n other words, the UCB will continue exploring other behaviours (or machines) until the true optimal behaviour or sequence of behaviours is found.
- Exploring is about gathering information about the the returns of the K machines while exploitation is about making decisions based on data already gathered.
- There will be a trade off between exploring and exploiting, in the case of the UCB it will keep on exploring until true optimal is found.

UCB

- The UCB algorithm decides which machine to exploit or if it should explore another option using confidence intervals, more specific the highest upper confidence bound.
- A confidence interval is a range of values within which we are sure the mean lies with a certain probability.
- The less we try a machine the less accurate our estimated average return for that machine and so the confidence interval is large.
- Every round we try machine we gather more information and the confidence interval shrinks, narrowing in on the true mean of the slot machine.
- To make it more clear how confidence bounds works let's assume that the mean of the slot machine blue is in within $[0.2, 0.9]$ with probability 0.95

UCB

- After a few rounds trying the blue slot machine, since it has the highest upper bounds, the confidence bound has shrink to [0.4, 0.7].
- During the exploring/exploiting process UBC also tries the red slot machine, since it had at times the highest upper bound and actually is the machine with the highest average return.



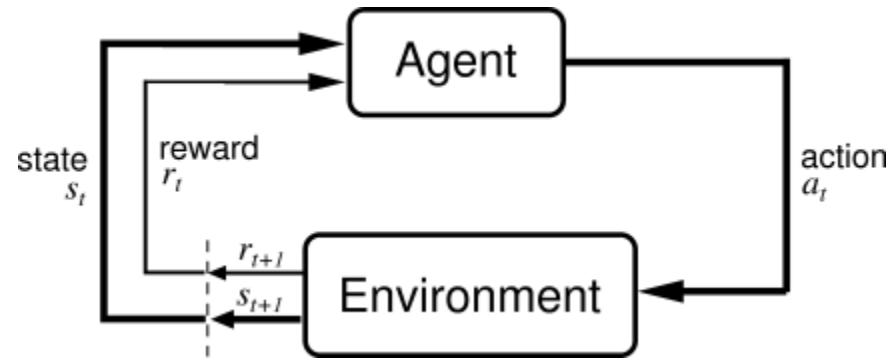
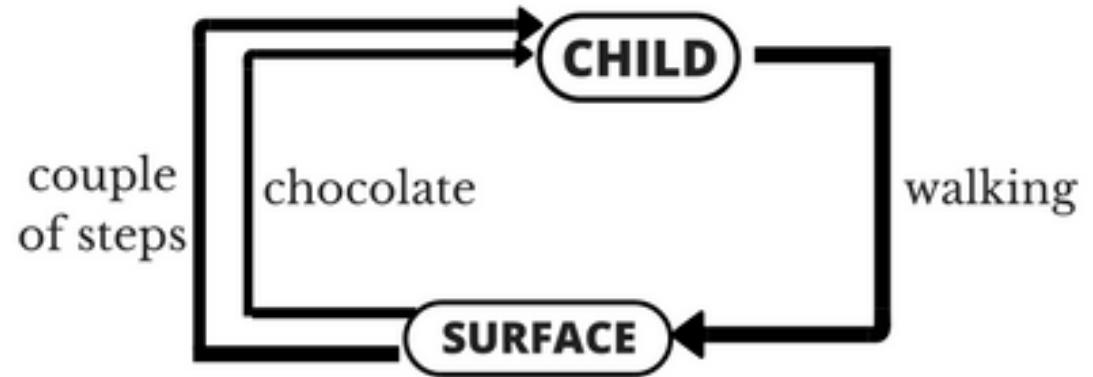
Reinforcement Learning

- Reinforcement Learning is learning what to do and how to map situations to actions.
- The end result is to maximize the numerical reward signal.
- The learner is not told which action to take, but instead must discover which action will yield the maximum reward.
- Let's understand this with a simple example below.

Reinforcement Learning



Reinforcement Learning

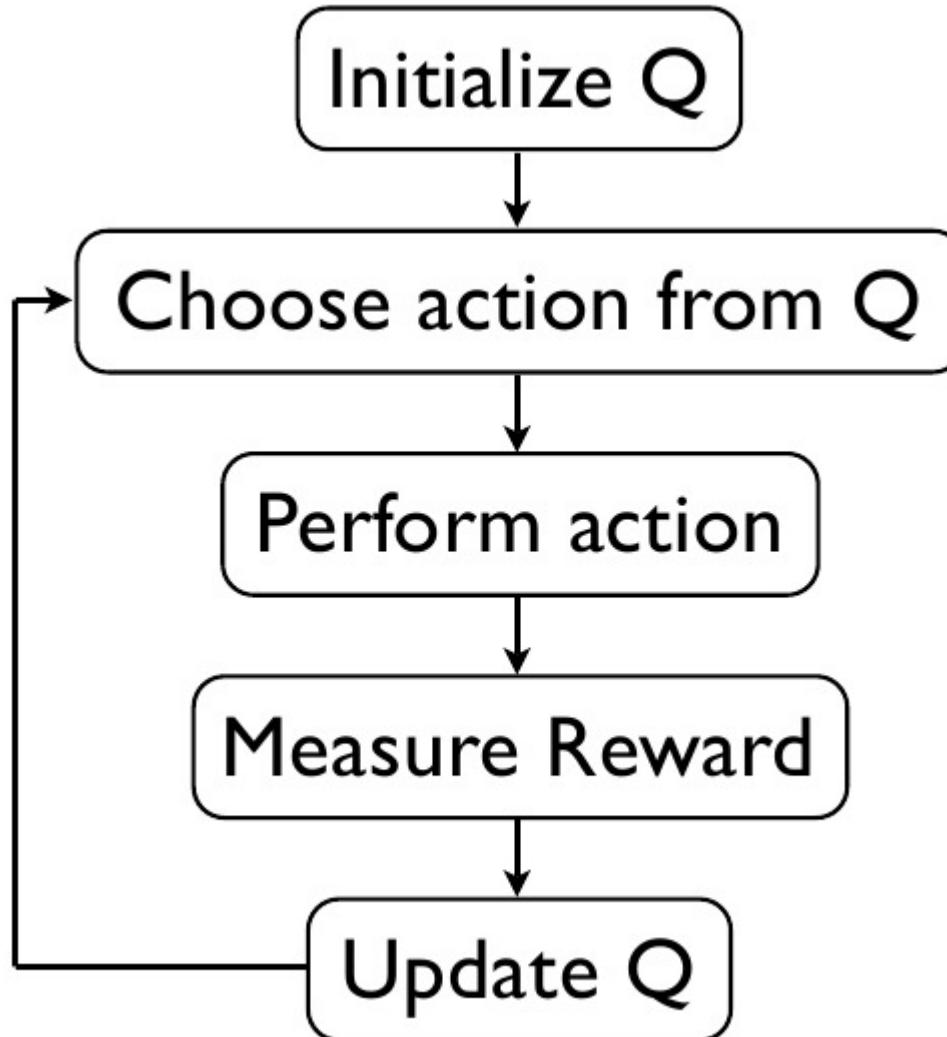


Slot Machines

Now you want to do is get the maximum bonus from the slot machines as fast as possible. What would you do?



An implementation of Reinforcement Learning



Exploration vs Exploitation Dilemma

- **Online** decision making involves a fundamental choice:
 - **Exploitation**: make the **best decision** given current information
 - **Exploration**: gather **more information**
- The best long-term strategy may involve **short-term sacrifices**
- Gather **enough information** to make the best overall decisions

Examples

- Restaurant Selection
 - Exploitation: Go to favorite restaurant
 - Exploration: Try a new restaurant
- Online Banner Advertisements
 - Exploitation: Show the most successful advert
 - Exploration: Show a different advert
- Oil Drilling
 - Exploitation: Drill at the best known location
 - Exploration: Drill at a new location
- Game Playing
 - Exploitation: Play the move you believe is best
 - Exploration: Play an experimental move
- Clinical Trial
 - Exploitation: Choose the best treatment so far
 - Exploration: Try a new treatment

The Bandit Problem

- There are K possible actions.
- Each time we take an action we get a reward between 0 and 1.
- There is a different distribution governing rewards for each action.
- Each distribution doesn't change over time (i.e. it is stationary).
- What actions should we take?

Approaches to the Bandit Problem

- Regret is defined in terms of the average reward.
- So if we can estimate average reward we can minimise regret.
- So let's take the action with the highest average reward directly.
 - Assume two actions.
 - Action 1 has reward of 1 with probability 0.3 and otherwise has reward of 0.
 - Action 2 has reward of 1 with probability 0.7 and otherwise has reward of 0.
 - Play action 1 first, get reward of 1.
 - Play action 2, get reward of 0.
 - Now average reward of action 1 will never drop to 0, so we'll never play action 2.

Exploring and Exploiting

- This illustrates a classic problem, which is the defining characteristic of *decision making*: the trade-off between exploring and exploiting. Exploring means to try new actions to learn their effects. Exploiting means to try what we know has worked in the past.
- The algorithm above does not explore sufficiently.

Optimism

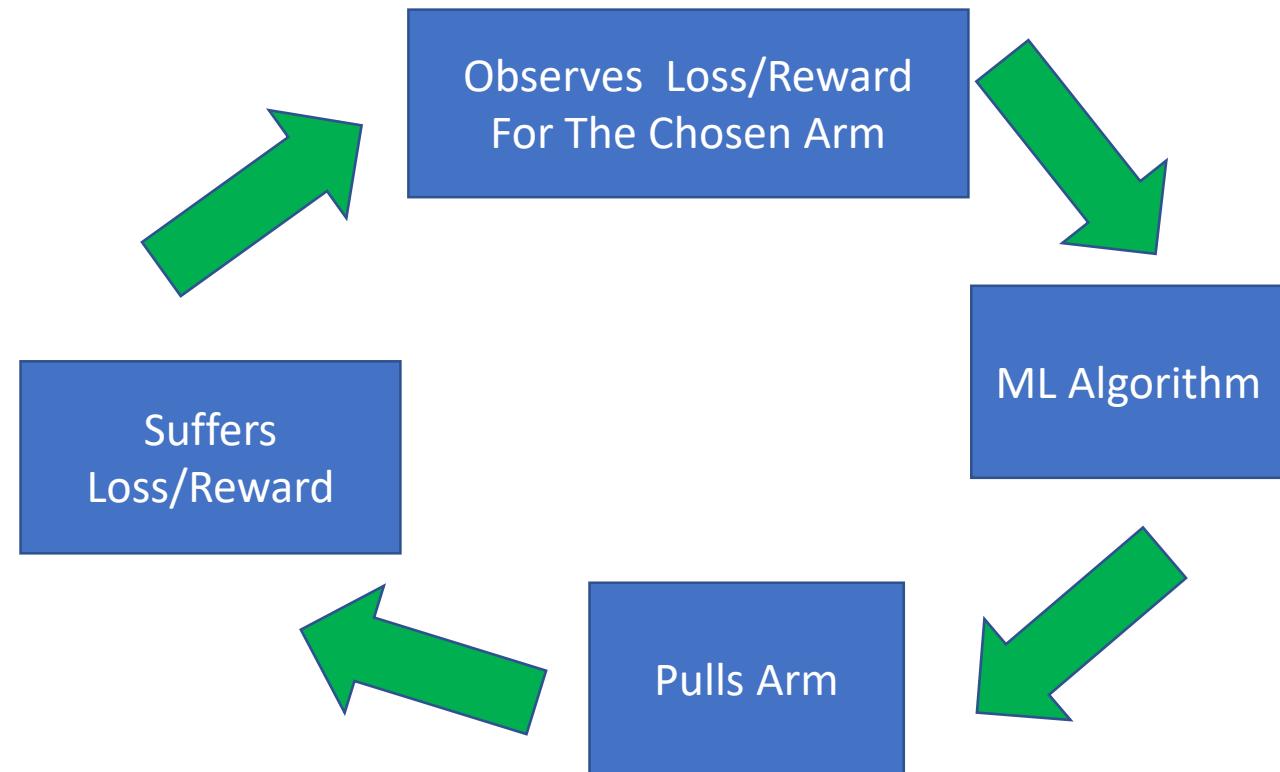
- The key problem with the algorithm above is that we're too certain of our estimates. When we have seen a single reward of 0 we shouldn't conclude the average reward is 0, but rather than it lies within some *confidence interval* that we adjust to account for the information we have received.
- A confidence interval is a range of values within which we are sure the mean lies with a certain probability. E.g. we could believe the mean is within $[0.2, 0.5]$ with probability 0.95.
- If we have tried an action less often, our estimated reward is less accurate so the confidence interval is larger. It shrinks as we get more information (i.e. try the action more often).
- Then, instead of trying the action with the highest mean we can try the action with the highest upper bound on its confidence interval.
- This is called an *optimistic* policy. We believe an action is as good as possible given the available evidence.

Multi-Armed Bandit (MAB) Problem

N arms



Assume
Statistical feedback
Assume
rewards in $[0,1]$
...



Regret in the MAB Setting

- Since we are stochastic, each arm has a true mean
- Do as well as if pulled always pulled arm with highest mean

MABs have a huge number of applications!

- Internet Advertising
- Website Layout
- Clinical Trials
- Robotics
- Education
- Videogames
- Recommender Systems
- ...

Unlike most of Machine Learning, how to ACT,
not just how to predict!

MAB algorithm #1: ϵ -first

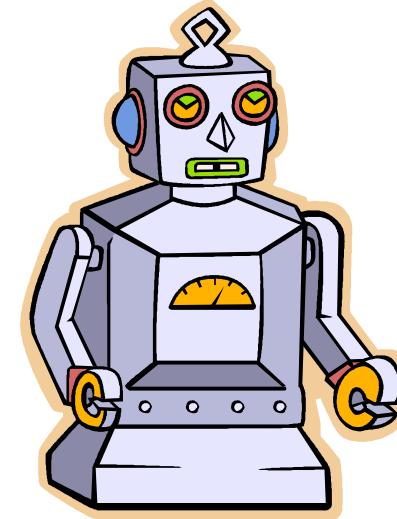
- Phase 1: Explore arms uniformly at random
- Phase 2: Exploit best arm so far
- Only 1 parameter: how long to wait!
- Problem?
- Not very efficient
- Makes statistics easy to run



MAB algorithm #2: ϵ -greedy

- Flip a coin:
- With probability ϵ , explore uniformly
- With probability $1 - \epsilon$, exploit
- Only 1 parameter: ϵ

- Problem?
- Often used in robotics and reinforcement learning



Upper Confidence Bound (UCB)

1. Play each arm once
2. Play arm i that maximizes:

$$\hat{\mu}_i + \sqrt{\frac{2\log(t)}{n_i}}$$

3. Repeat Step 2 forever

UCB Regret Bound

Problem –independent:

$$O(\sqrt{NT \log T})$$

Problem –dependent:

$$O\left(\frac{\log T}{\min_i \mu^* - \mu_i}\right)$$

Can't do much better!

UCB1 Details

- For each action j record the average reward \bar{x}_j and number of times we have tried it n_j . We write n for total number of actions we have tried.
- Try the action that maximises $\bar{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$
- That is all! What is the confidence bound we're using?

UCB1's Explore/Exploit Tradeoff

- From our analysis of the Chernoff-Hoeffding bound above we can see that the confidence bound grows with the total number of actions we have taken but shrinks with the number of times we have tried this particular action. This ensures each action is tried infinitely often but still balances exploration and exploitation.

UCB1 Regret Bound

- The regret for UCB1 grows at a rate of $\ln n$. In particular, after n actions it is at most

$$\sum_{j=1}^K \frac{4\ln n}{\Delta_j} + \left(1 + \frac{\pi^2}{3}\right) \Delta_j \quad (4)$$

where $\Delta_j = \mu^* - \mu_j$.

UCB1-Tuned

- In practice (but not in theory, 'cause it is too hard to analyse) we can improve on UCB1.
- Note that a Bernoulli random variable with $p = 0.5$ is the reward distribution that will give the highest variance (which is $\frac{1}{4}$).
- We can also compute the sample variance σ_j for each action.
- Then use the upper confidence bound for action j of:

$$\sqrt{\frac{\ln n}{n_j} \min \left(\frac{1}{4}, \left(\sigma_j + \frac{2 \ln n}{n_j} \right) \right)} \quad (5)$$

Thompson Sampling

- Like the UCB the Thompson Sampling Algorithm deals with the exploration-exploitation dilemma in the multi-armed bandit problem.
- However it does this in a different manner.
- In contrast to the UCB, which is a deterministic algorithm, Thompson Sampling is a probabilistic algorithm making decisions based on random sampling of prior distributions, whereas the UCB does this strictly with the confidence bounds.

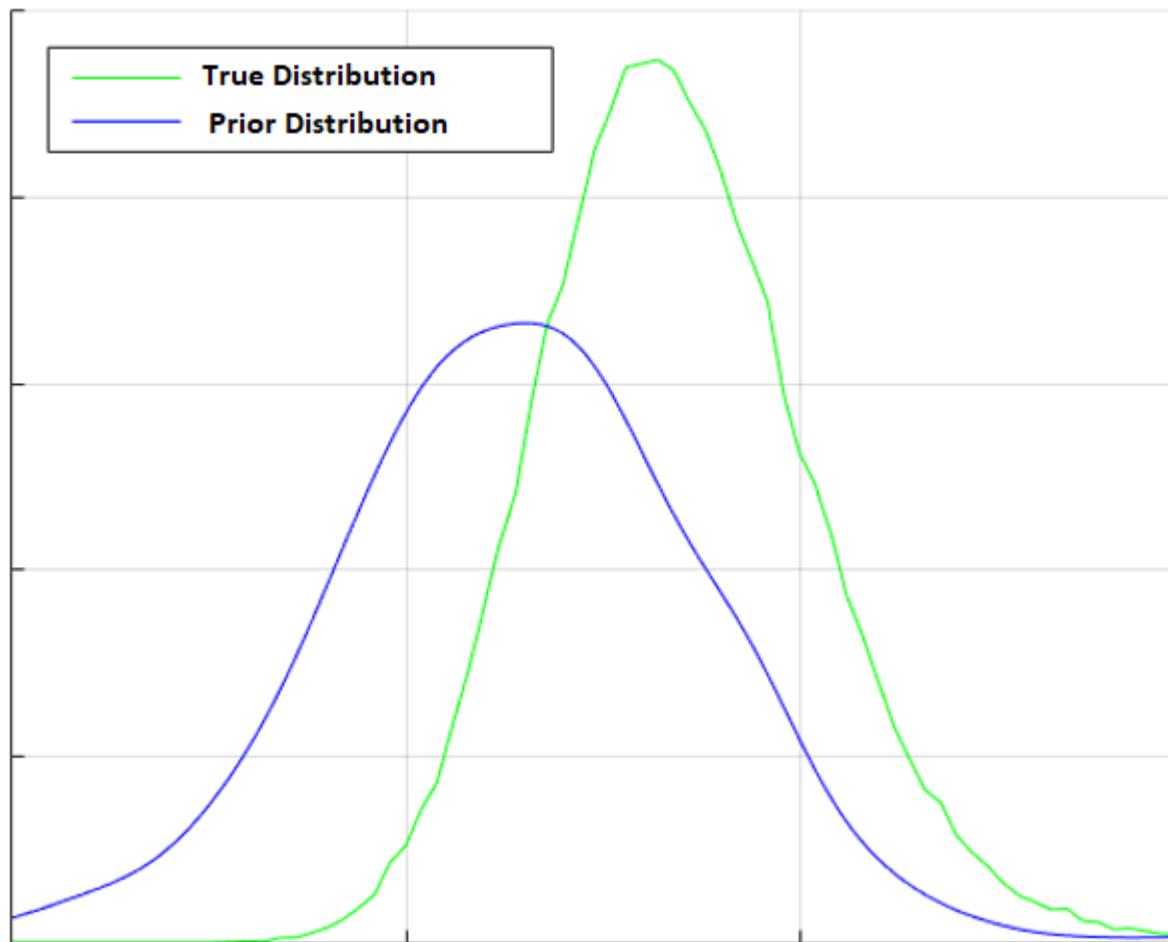
Thompson Sampling

- The TS takes n samples of each of the K armed bandits and creates K distributions based on these samples.
- These initial distributions are not the true distributions of the K bandits, so we call them prior distributions.
- Like the UCB we want to find the bandit with the highest mean, to refer the same example; to find the slot machine with the highest average return.
- The TS takes one value from each K prior distributions and uses these values as the prior mean of each of the K bandits.
- The TS will exploit the bandit with the highest prior mean and uses the received reward to improve the prior distribution it has for this bandit, this process repeats until the optimal bandit is found.

Thompson Sampling

- In other words, the TS algorithm creates a hypothetical set up of the bandits and randomly samples every round their prior means, the bandit with the highest prior mean will be exploited and thus resulting in more information about the true mean of this bandit.
- After plenty of rounds the prior distributions will start resembling the true distributions and thus revealing the optimal bandit.

Thompson Sampling



Thompson Sampling

- Idea:
 - Sample several potential average rewards:
 $R_1(a), \dots, R_k(a) \sim \Pr(R(a)|r_1^a, \dots, r_n^a)$ for each a
 - Estimate empirical average
 $\hat{R}(a) = \frac{1}{k} \sum_{i=1}^k R_i(a)$
 - Execute $\text{argmax}_a \hat{R}(a)$
- Coin example
 - $\Pr(R(a)|r_1^a, \dots, r_n^a) = \text{Beta}(\theta_a; \alpha_a, \beta_a)$
where $\alpha_a - 1 = \#heads$ and $\beta_a - 1 = \#tails$

Thompson Sampling Algorithm Bernoulli Rewards

ThompsonSampling(h)

$V \leftarrow 0$

For $n = 1$ to h

 Sample $R_1(a), \dots, R_k(a) \sim \Pr(R(a)) \quad \forall a$

$\hat{R}(a) \leftarrow \frac{1}{k} \sum_{i=1}^k R_i(a) \quad \forall a$

$a^* \leftarrow \text{argmax}_a \hat{R}(a)$

 Execute a^* and receive r

$V \leftarrow V + r$

 Update $\Pr(R(a^*))$ based on r

Return V

Sample Size

- In Thompson sampling, amount of data n and sample size k regulate amount of exploration
- As n and k increase, $\hat{R}(a)$ becomes less stochastic, which reduces exploration
 - As $n \uparrow$, $\Pr(R(a)|r_1^a \dots r_n^a)$ becomes more peaked
 - As $k \uparrow$, $\hat{R}(a)$ approaches $E[R(a)|r_1^a \dots r_n^a]$
- The stochasticity of $\hat{R}(a)$ ensures that all actions are chosen with some probability

Continuous Rewards

- So far we assumed that $r \in \{0,1\}$
- What about continuous rewards, i.e. $r \in [0,1]$?
 - NB: rewards in $[a,b]$ can be remapped to $[0,1]$ by an affine transformation without changing the problem
- Idea:
 - When we receive a reward r
 - Sample $b \sim \text{Bernoulli}(r)$ s.t. $b \in \{0,1\}$

Thompson Sampling Algorithm Continuous rewards

ThompsonSampling(h)

$V \leftarrow 0$

For $n = 1$ to h

 Sample $R_1(a), \dots, R_k(a) \sim \Pr(R(a)) \quad \forall a$

$\hat{R}(a) \leftarrow \frac{1}{k} \sum_{i=1}^k R_i(a) \quad \forall a$

$a^* \leftarrow \operatorname{argmax}_a \hat{R}(a)$

 Execute a^* and receive r

$V \leftarrow V + r$

 Sample $b \sim \text{Bernoulli}(r)$

 Update $\Pr(R(a^*))$ based on b

Return V

Apriori

- With the quick growth in e-commerce applications, there is an accumulation vast quantity of data in months not in years. Data Mining, also known as Knowledge Discovery in Databases(KDD), to find anomalies, correlations, patterns, and trends to predict outcomes.
- Apriori algorithm is a classical algorithm in data mining. It is used for mining frequent itemsets and relevant association rules. It is devised to operate on a database containing a lot of transactions, for instance, items brought by customers in a store.

Apriori

- It is very important for effective Market Basket Analysis and it helps the customers in purchasing their items with more ease which increases the sales of the markets.
- It has also been used in the field of healthcare for the detection of adverse drug reactions. It produces association rules that indicates what all combinations of medications and patient characteristics lead to ADRs.

Association Rules

- Let I be a set of n attributes called items and T be the set of transactions. It is called database. Every transaction, in T , has a unique transaction ID, and it consists of a subset of itemsets in I .
- A rule can be defined as an implication, where X and Y are subsets of I , and they have no element in common, i.e., $X \cap Y = \emptyset$. X and Y are the antecedent and the consequent of the rule, respectively.

Association Rules

- Let's take an easy example from the supermarket sphere.
- The example that we are considering is quite small and in practical situations, datasets contain millions or billions of transactions.
- The set of itemsets, $=\{\text{Onion}, \text{Burger}, \text{Potato}, \text{Milk}, \text{Beer}\}$ and a database consisting of six transactions. Each transaction is a tuple of 0's and 1's where 0 represents the absence of an item and 1 the presence.
- An example for a rule in this scenario would be $\{\text{Onion}, \text{Potato}\} \Rightarrow \{\text{Burger}\}$, which means that if onion and potato are bought, customers also buy a burger.

Association Rules

Transaction ID	Onion	Potato	Burge r	Milk	Beer
	1	1	1	0	0
	0	1	1	1	0
	0	0	0	1	1
	1	1	0	1	0
	1	1	1	0	1
	1	1	1	1	1

Association Rules

- Support
 - The support of an itemset , is the proportion of transaction in the database in which the item X appears. It signifies the popularity of an itemset.
- Confidence
- Confidence of a rule is defined as follows:
- It signifies the likelihood of item Y being purchased when item X is purchased. So, for the rule {Onion, Potato} => {Burger},

Association Rules

- Lift
- The lift of a rule is defined as:
- This signifies the likelihood of the itemset being purchased when item is purchased while taking into account the popularity of .
- In our example above,
- If the value of lift is greater than 1, it means that the itemset is likely to be bought with itemset , while a value less than 1 implies that itemset is unlikely to be bought if the itemset is bought.

Association Rules

- Conviction
- The conviction of a rule can be defined as:
- For the rule $\{\text{onion, potato}\} \Rightarrow \{\text{burger}\}$
- The conviction value of 1.32 means that the rule $\{\text{onion, potato}\} \Rightarrow \{\text{burger}\}$ would be incorrect 32% more often if the association between them was an accidental chance.

How does Apriori algorithm work?

- A key concept in Apriori algorithm is the anti-monotonicity of the support measure. It assumes that
- All subsets of a frequent itemset must be frequent
- Similarly, for any infrequent itemset, all its supersets must be infrequent too
- Let us now look at the intuitive explanation of the algorithm with the help of the example we used above. Before beginning the process, let us set the support threshold to 50%, i.e. only those items are significant for which support is more than 50%.

How does Apriori algorithm work?

- Step 1: Create a frequency table of all the items that occur in all the transactions. For our case:
- ItemFrequency (No. of transactions)
- Onion(O) 4
- Potato(P) 5
- Burger(B) 4
- Milk(M) 4
- Beer(Be) 2

How does Apriori algorithm work?

- Step 2: We know that only those elements are significant for which the support is greater than or equal to the threshold support. Here, support threshold is 50%, hence only those items are significant which occur in more than three transactions and such items are Onion(O), Potato(P), Burger(B), and Milk(M). Therefore, we are left with:
- Item Frequency (No. of transactions)
- Onion(O) 4
- Potato(P) 5
- Burger(B) 4
- Milk(M) 4

How does Apriori algorithm work?

- **Step 3:** The next step is to make all the possible pairs of the significant items keeping in mind that the order doesn't matter, i.e., AB is same as BA. To do this, take the first item and pair it with all the others such as OP, OB, OM. Similarly, consider the second item and pair it with preceding items, i.e., PB, PM. We are only considering the preceding items because PO (same as OP) already exists. So, all the pairs in our example are OP, OB, OM, PB, PM, BM.

How does Apriori algorithm work?

- Step 4: We will now count the occurrences of each pair in all the transactions.
- Itemset Frequency (No. of transactions)
 - OP 4
 - OB 3
 - OM 2
 - PB 4
 - PM 3
 - BM 2

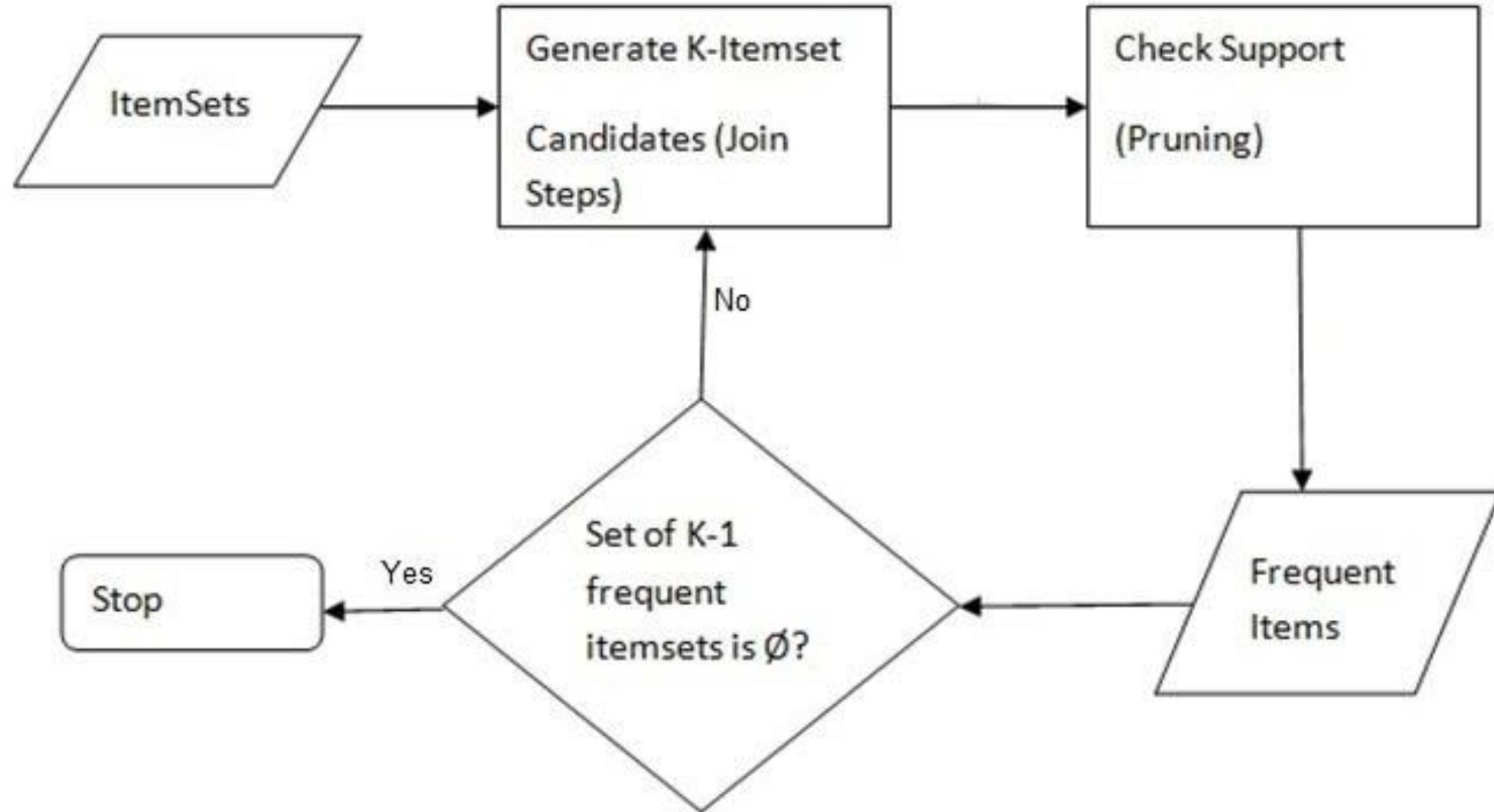
How does Apriori algorithm work?

- Step 5: Again only those itemsets are significant which cross the support threshold, and those are OP, OB, PB, and PM.
- Step 6: Now let's say we would like to look for a set of three items that are purchased together. We will use the itemsets found in step 5 and create a set of 3 items.
- To create a set of 3 items another rule, called self-join is required. It says that from the item pairs OP, OB, PB and PM we look for two pairs with the identical first letter and so we get
 - OP and OB, this gives OPB
 - PB and PM, this gives PBM

How does Apriori algorithm work?

- Next, we find the frequency for these two itemsets.
- Itemset Frequency (No. of transactions)
- OPB 4
- PBM 3

How does Apriori algorithm work?



Pros and Cons

- It is an easy-to-implement and easy-to-understand algorithm.
- It can be used on large itemsets.
- Cons of the Apriori Algorithm
- Sometimes, it may need to find a large number of candidate rules which can be computationally expensive.
- Calculating support is also expensive because it has to go through the entire database.

Association Rule Learning in R

- When we go grocery shopping, we often have a standard list of things to buy. Each shopper has a distinctive list, depending on one's needs and preferences.
- A housewife might buy healthy ingredients for a family dinner, while a bachelor might buy coffee and chips.
- Understanding these buying patterns can help to increase sales in several ways. If there is a pair of items, X and Y, that are frequently bought together:
 -

Association Rule Learning in R

- Both X and Y can be placed on the same shelf, so that buyers of one item would be prompted to buy the other.
- Promotional discounts could be applied to just one out of the two items.
- Advertisements on X could be targeted at buyers who purchase Y.
- X and Y could be combined into a new product, such as having Y in flavors of X.

Association Rule Learning in R

- **Measure 1: Support.** This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears. In Table 1 below, the support of {apple} is 4 out of 8, or 50%. Itemsets can also contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%.

Transaction 1				
Transaction 2				
Transaction 3				
Transaction 4				
Transaction 5				
Transaction 6				
Transaction 7				
Transaction 8				

$$\text{Support } \{\text{apple}\} = \frac{4}{8}$$

Association Rule Learning in R

- If you discover that sales of items beyond a certain proportion tend to have a significant impact on your profits, you might consider using that proportion as your *support threshold*.
- You may then identify item sets with support values above this threshold as significant item sets.

$$\text{Support } \{\text{apple}\} = \frac{4}{8}$$

Association Rule Learning in R

- **Measure 2: Confidence.** This says how likely item Y is purchased when item X is purchased, expressed as $\{X \rightarrow Y\}$. This is measured by the proportion of transactions with item X, in which item Y also appears. In Table 1, the confidence of $\{\text{apple} \rightarrow \text{beer}\}$ is 3 out of 4, or 75%.

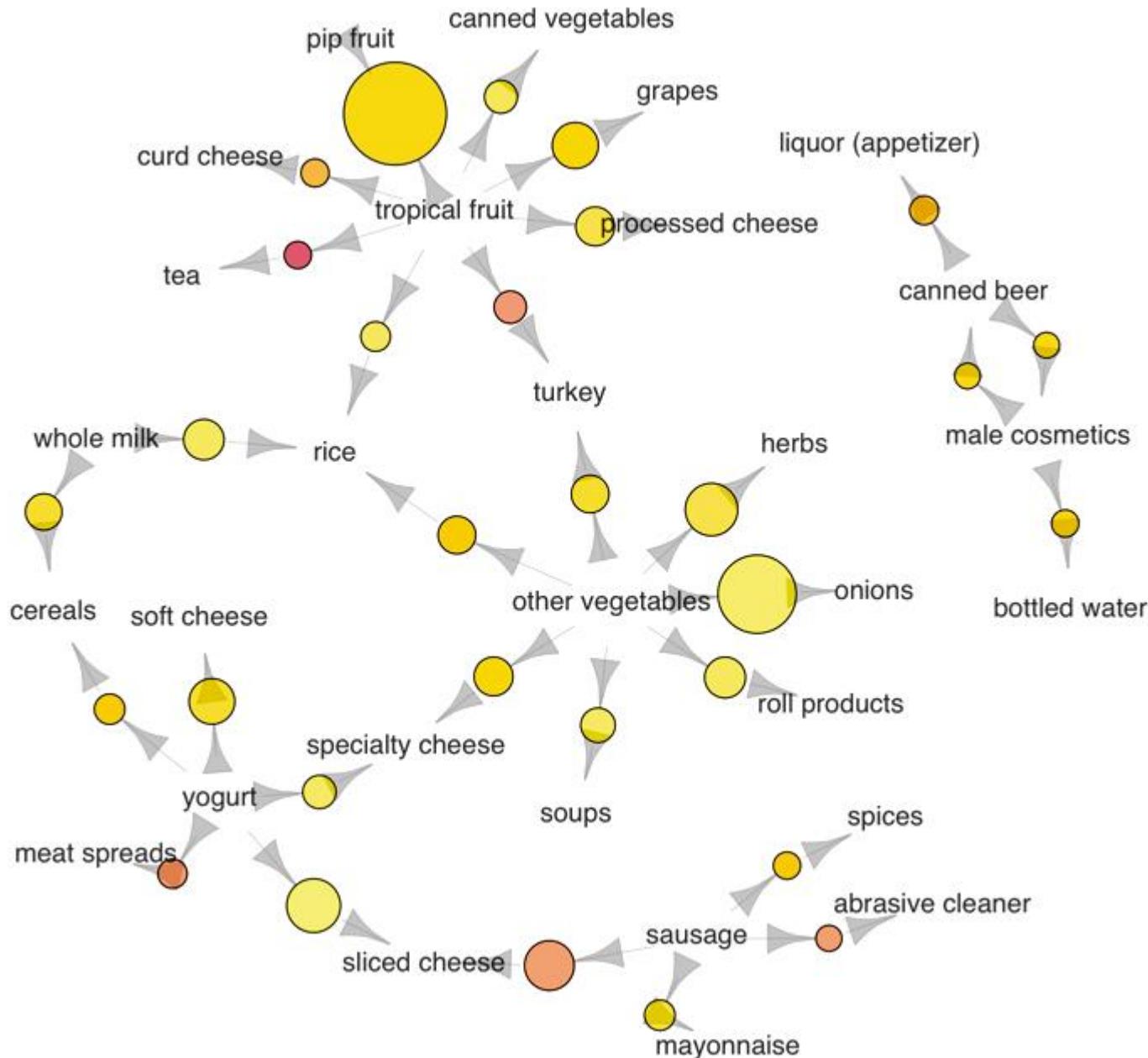
$$\text{Confidence } \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support } \{\text{🍎}, \text{🍺}\}}{\text{Support } \{\text{🍎}\}}$$

Association Rule Learning in R

- **Measure 3: Lift.** This says how likely item Y is purchased when item X is purchased, while controlling for how popular item Y is. In Table 1, the lift of {apple -> beer} is 1, which implies no association between items. A lift value greater than 1 means that item Y is *likely* to be bought if item X is bought, while a value less than 1 means that item Y is *unlikely* to be bought if item X is bought.

$$\text{Lift } \{\text{🍎} \rightarrow \text{🍺}\} = \frac{\text{Support } \{\text{🍎}, \text{🍺}\}}{\text{Support } \{\text{🍎}\} \times \text{Support } \{\text{🍺}\}}$$

Associations between selected items. Visualized using the arulesViz R library.



Transaction	Support	Confidence	Lift
Canned Beer → Soda	1%	20%	1.0
Canned Beer → Berries	0.1%	1%	0.3
Canned Beer → Male Cosmetics	0.1%	1%	2.6

Transaction	Support
Canned Beer	10%
Soda	20%
Berries	3%
Male Cosmetics	0.5%

Apriori Algorithm

- The *apriori principle* can reduce the number of itemsets we need to examine. Put simply, the apriori principle states that if an itemset is infrequent, then all its subsets must also be infrequent.
- This means that if {beer} was found to be infrequent, we can expect {beer, pizza} to be equally or even more infrequent. So in consolidating the list of popular itemsets, we need not consider {beer, pizza}, nor any other itemset configuration that contains beer.

Finding itemsets with high support

- Using the apriori principle, the number of itemsets that have to be examined can be pruned, and the list of popular itemsets can be obtained in these steps:
- **Step 0.** Start with itemsets containing just a single item, such as {apple} and {pear}.
- **Step 1.** Determine the support for itemsets. Keep the itemsets that meet your minimum support threshold, and remove itemsets that do not.
- **Step 2.** Using the itemsets you have kept from Step 1, generate all the possible itemset configurations.
- **Step 3.** Repeat Steps 1 & 2 until there are no more new itemsets.

Limitations

- **Computationally Expensive.** Even though the apriori algorithm reduces the number of candidate itemsets to consider, this number could still be huge when store inventories are large or when the support threshold is low. However, an alternative solution would be to reduce the number of comparisons by using advanced data structures, such as hash tables, to sort candidate itemsets more efficiently.
- **Spurious Associations.** Analysis of large inventories would involve more itemset configurations, and the support threshold might have to be lowered to detect certain associations. However, lowering the support threshold might also increase the number of spurious associations detected. To ensure that identified associations are generalizable, they could first be distilled from a training dataset, before having their support and confidence assessed in a separate test dataset.