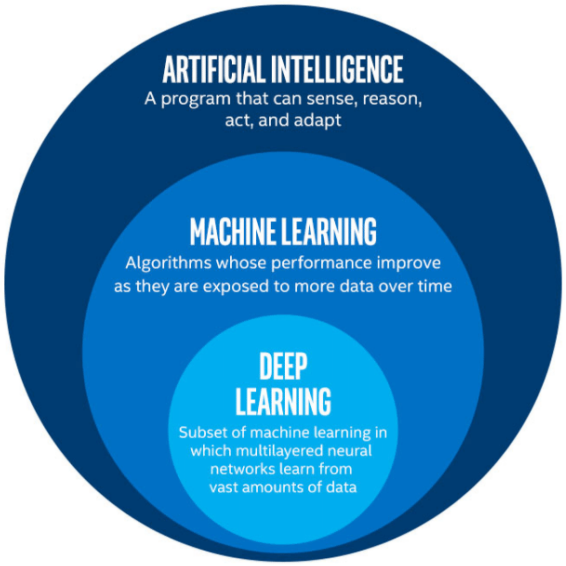
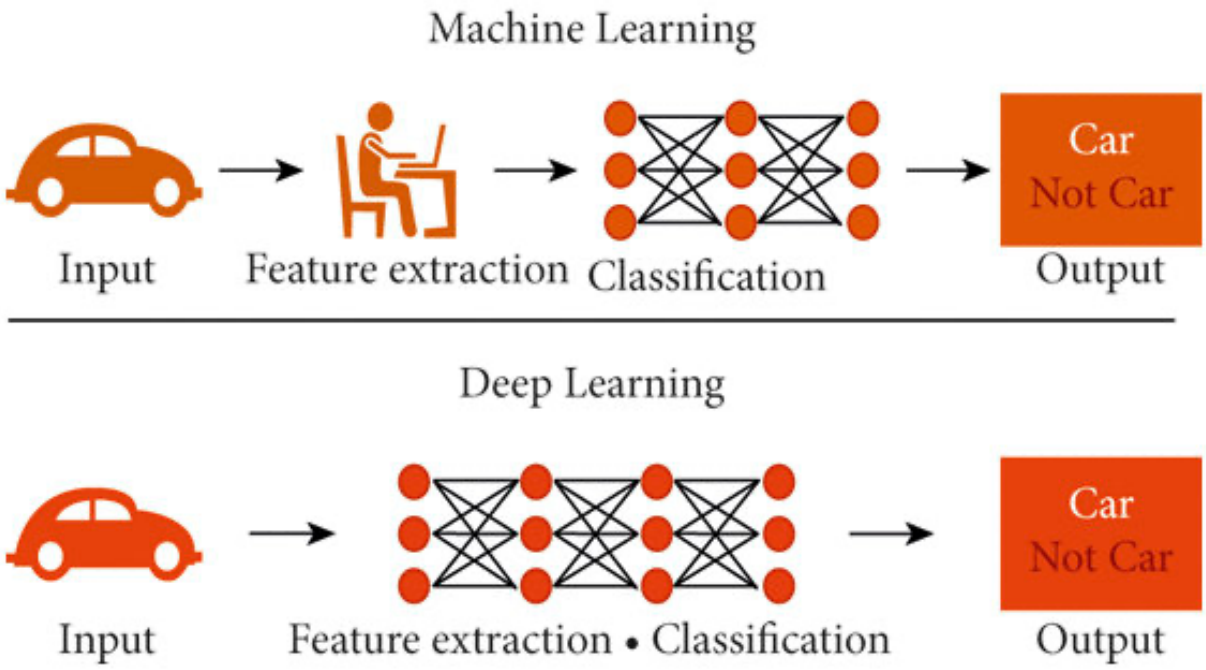
Data Science Advanced

Chapter 1: Intro

AI vs ML





Deep learning → figure out features for themselves. Humans spend more time building the way the model works.

Different types of learning

- Unsupervised → looks at variation = information (standard deviation)
- **Supervised Learning** (focus of this course) → give data and labels
- Reinforcement learning

A short and incomplete overview of Machine Learning

Many different types of ML aims:

- Clustering (unsupervised)
- Computer vision (detecting cancer in images, self-driving cars)
- Natural language processing (Lernout & Hauspi / Alexa / Siri)
- **Classification (binary / multi-class)**
- Recommender systems

Binary classification

- Supervised ML (Labels are known in the learning phase)
- Aims to predict labels of new data
- Matrix format of data (rows X features)

Flower example → see jupyter notebook

- Read in the data → array of RGB colours
- RGB matrix
- Put all pixels and align them after each other (not smart because it is massive, about 400.000 columns)

Multiple things can happen:

1. Performance is good
    1. The model has learned something
    2. Did it learn the data by heart or did it learn the patterns?
        1. Generalised → good
        2. Not generalised → try again
2. Performance is bad
    1. The model didn't learn anything (use of all variables = too many)
    2. Model does not work, try again (a new approach)

STEPS OF PIPELINE:

1. Read → you get multiple matrixes
2. Test/Train split
3. Data exploration
4. Features
5. Build the outcome vector (if not provided)

- **https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15**
- **https://www.itl.nist.gov/div898/handbook/eda/section1/eda11.htm**
- **https://r4ds.had.co.nz/exploratory-data-analysis.html**

Chapter 2: Data Processing & EDA

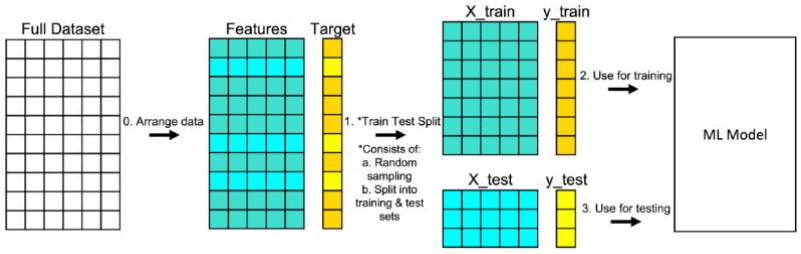
Quiz 1

1. What does EDA stand for?
   - Exploratory Data Analysis
2. If you have a variable, what could you do to see whether there are any outliers? Give 2-3 possibilities.
   - Calculate the mean and standard deviation of the variable, identify any data points that fall outside of a certain number of standard deviations from the mean. For example, data points that fall more than 3 standard deviations from the mean could be considered outliers.
   - Use box plots or scatter plots to visually identify outliers
   - Use the interquartile range (IQR) to identify outliers, which is the difference between the 75th and 25th percentiles. Data points that fall outside of the range of (Q1 - 1.5 * IQR) to (Q3 + 1.5 * IQR) are considered outliers.
3. What is the difference between correlation and covariation?
   - Covariance shows you how the two variables differ, whereas correlation shows you how the two variables are related.

Recap

ML default pipeline:

- Matrix format
- Data splitting → training part and testing part



Why is a model not learning?

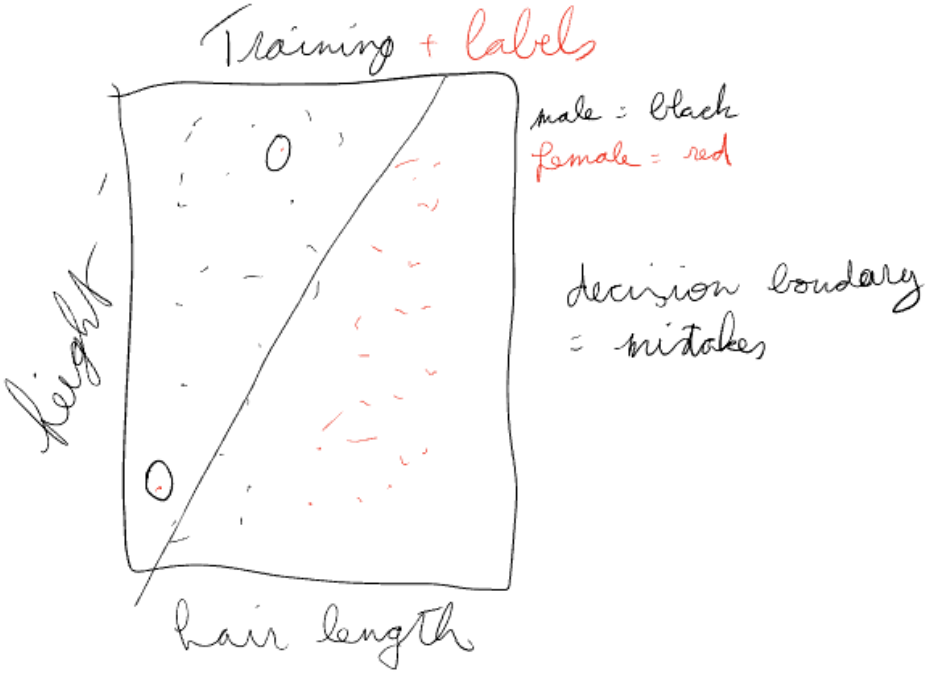
- The problem is too difficult for the model I am using
  - Patterns aren't in the data → clean data
  - Increase model complexity
- Go back to the beginning

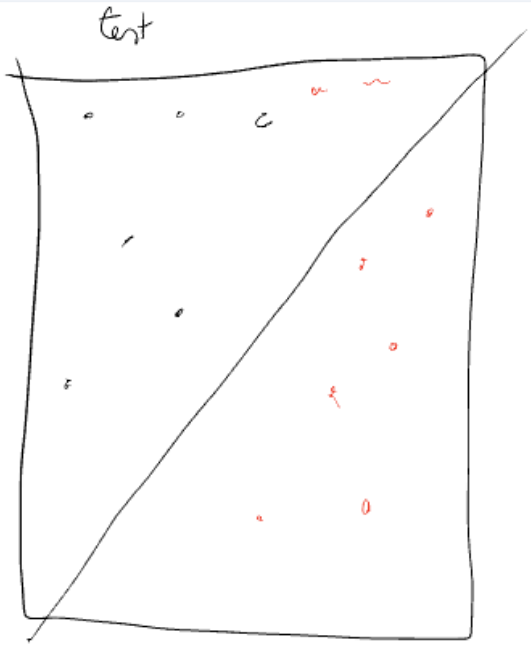
Why's a model very good?

- Overfitting (trained the model by heart)
  - Check by training and test (if it's good on the training data and bad on tests data, it is overfitted)
  - Don't give it the outcome!
- It has actually learned the patterns (the model is generalisable)

NEVER FIT THE TEST DATA → you will shift the decision line to the new data

- Machine learning workflow
- Data format:
  - Matrix format
  - No missing values (some ML algorithms can cope with this)
- Train/Test data
- Decision boundaries





→ if you fit first, you re-lay the decision lines → it will take the new data as decision line

Exploratory Data Analysis (EDA)

a bit like an interrogation:

- You ask a question and the data answers → count (how much of each), shape of the data, correlation,… → what data do we need to answer the question and is the data clean enough
- Ask stupid questions and get stupid answers
- Ask the right questions and you get deeper into the truth

What is the quality of my data? Missing values, faulty data (wrong measurement,..) and so on. → find the good parts

- Goal is to get to the core of the data
- The core is dependent on the question we want to solve
- The core is hidden by problems:
  - Missing values
  - Strange/unknown variables
  - Faulty measurements
  - correlated variables

Measurement types

- Nominal data (is in text: French, Dutch) → dummy coding is a solution: one-hot encoding → Make a column for each (column french, dutch) → 1 if you are dutch, 0 if you are not. But it has a lot of columns
- Ordinal data (isn't numeric, it is text but there is an order (height: short, average, tall) → you can put this in numbers
- Interval data (a bit less quality, also numbers)
- Ratio data (makes sense, if you are 1m and the other 2m, the other one is twice as high)

Time: can be ratio if it is with a chronometer (start from 0) but if it's in hours (13:45) it is interval

Measurement types: processing

Not all measurement/data types van be used "as is: for machine learning. Some need some processing:

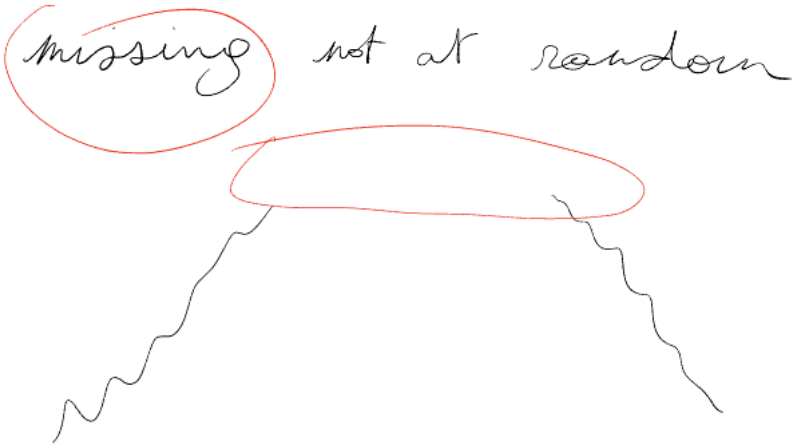
- Nominal data is tricky for most ML algorithms (work internally with numbers). One-hot encoding possible solution.
- Ordinal data can be converted to a number

Interval and Ratio data can often be used to create more data with pre-processing nd feature engineering.

Missing values

Different types of missing data:

- Missing completely at random (MCAR) → an accident
- Missing at random (MAR) → underlying process which allows for more values missing but is still ok
- Missing not at random (MNAR / NMAR) → issue! There is a clear process that stops me from having data there (for example, the range of the sensor can't go so far)



What you can do:

- Carry last value forward → now count how many times you are above the highest value

Missing data strategies

Below are some missing data strategies from the least to the most complex.

- Remove the data (complete case analysis)
  - Remove rows
  - Remove columns
- Impute with a default value:
  - 0 (really basic)
  - Mean/median (better in a certain situation: there are no extremes)
  - NA
  - 999/-999 (If you ever do this I will come and kill you)
- Impute with realistic guess
- Build a machine learning model to impute the values (e.g. missforest)

Chapter 3: Performance Evaluation

Train/Test

Take a part of our data because we don't want it in our model because we want to evaluate our model on data it hasn't seen before to test its generalisability. Training data has produced a model

ML output

In:

Testdata (10 values)

Out:

2 ways it can output (you choose this): 10 labels or 10 numbers between 0 (sure it's false) and 1 (sure it's true) → It calculates the probability

T	0.99	T	0.9
F	0.01	F	0.3
F	0.01	F	0.1
T	0.99	T	0.6
T	0.99	T	0.8

Confusion matrix

		Predicted	
		Positive (1)	Negative (0)
Truth	Positive (1)	True Positive (TP)	False Negative (FN)
	Negative (0)	False Positive (FP)	True Negative (TN)

Table: Confusion matrix outline

model output	by label
H	H
H	H
D	D
H	D
D	D
D	H
H	H

Count each one (predicted & truth)

2 true positive	1 false negative
1 false positive	3 true negative

- sensitivity of model/recall = TP/P = TP/(TP+FN) = 2/2+1 → only takes 2/3rd into account → how many of the ones we want to detect do we detect?
- selectivity/specificity = TN/N = TN/(TN + FP) = 3/3+1 → how specific is my model?
- precision = TP/(TP + FP) = 2/2+1

Example:
10 cancer patients
990 healthy

MODEL 1 → all have cancer:

10 TP	0 FN
990 FP	0 TN

sensitivity:
10/(10+0) = 100%

specificity =
0/(0+990) = 0%

precision =
10/(10+990) = 1%

MODEL 2 → no one has cancer:

0 TP	10 FN
0 FP	990 TN

sensitivity:
0/(0+10) = 0%

specificity =
100%

precision =
0

MODEL 3→ all cancers at the cost of 300 FP:

10 TP	0 FN
300 FP	690 TN

sensitivity:
100%

specificity =
690/(690+300) = 69%

precision =
3.3%

MODEL 4:
1000 cancer patient

600 TP	400 FN
300 FP	700 TN

sensitivity =
600/(600+400) = 60%

specificity =
700/(700+300) = 70%

precision =
600/(600+300) = 66%

Performance metrics
https://en.wikipedia.org/wiki/Confusion_matrix

Performance plots

T	0.99	T	0.9
F	0.01	F	0.3
F	0.01	F	0.1
T	0.99	T	0.6
T	0.99	T	0.8

Quantify this performance:

3 TP	0 FN
0 FP	2 TN

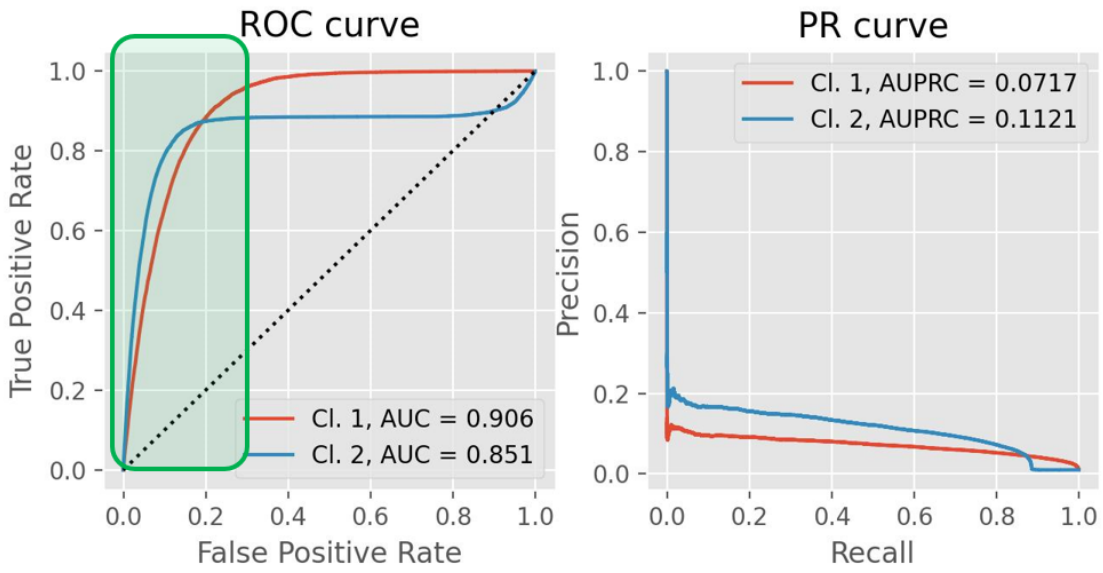
= best model → no mistakes

when bar (decision threshold) goes from 0.5 to 0.75

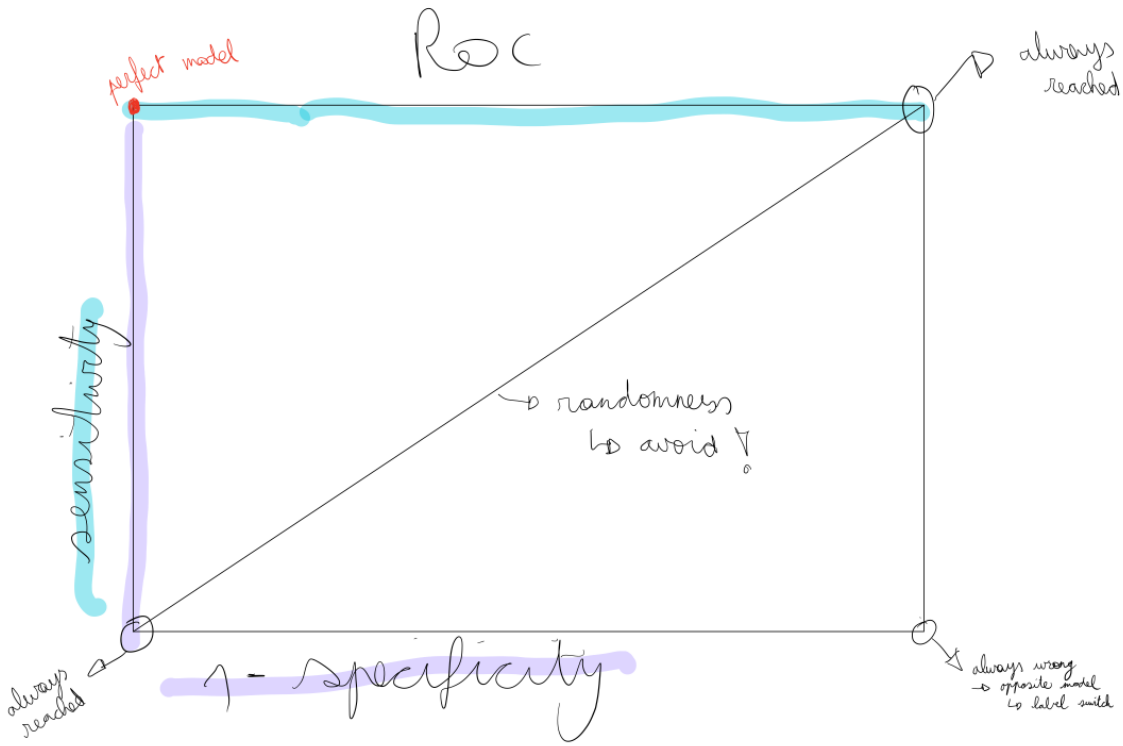
T	0.99	T	0.9
F	0.01	F	0.3
F	0.01	F	0.1
T	0.99	**F**	**0.6**
T	0.99	T	0.8

Start at the threshold at 1 and go up. For each threshold calculate the sensitivity, specificity and precision

ROC vs PR



True positive rate = sensitivity (= recall)
False positive rate = 1-specificity



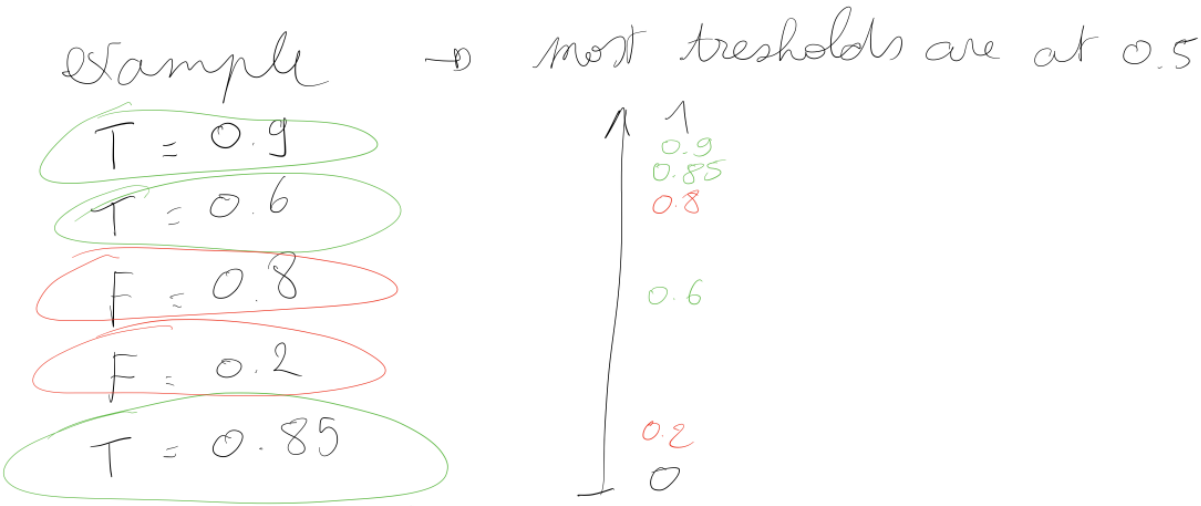
ROC (Receiver operating characteristic) curve:

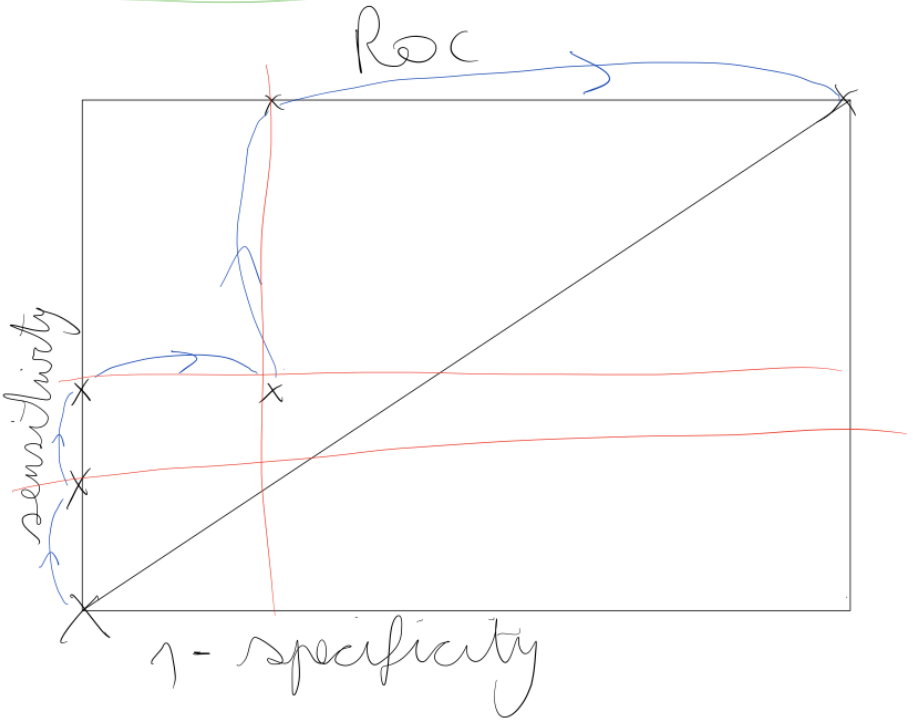
- Sensitivity = how many of the ones that I want to have do I have correct?
- Specificity = how many of the other class do I detect?
- each model gets 1 curve
- AUC = area under roc → 1 AUC is perfect model, useless model = AUC 0.5, AUC 1 reversed model
- average AUC = 0.75

T 0,99
T 0,99
F 0,8 → find treshold
F 0,85
T 0,99
F 0,1

Start threshold from the top value and go down → you'll start at the bottom left in the roc because 1 has 0% sensitivity (all values are below the threshold) and 1 (=100%) specificity

---



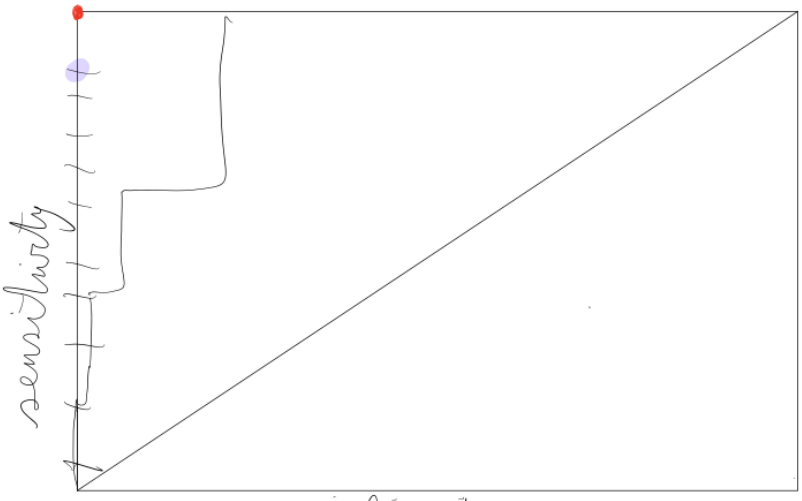


cancer example

10 → cancer

990 → not

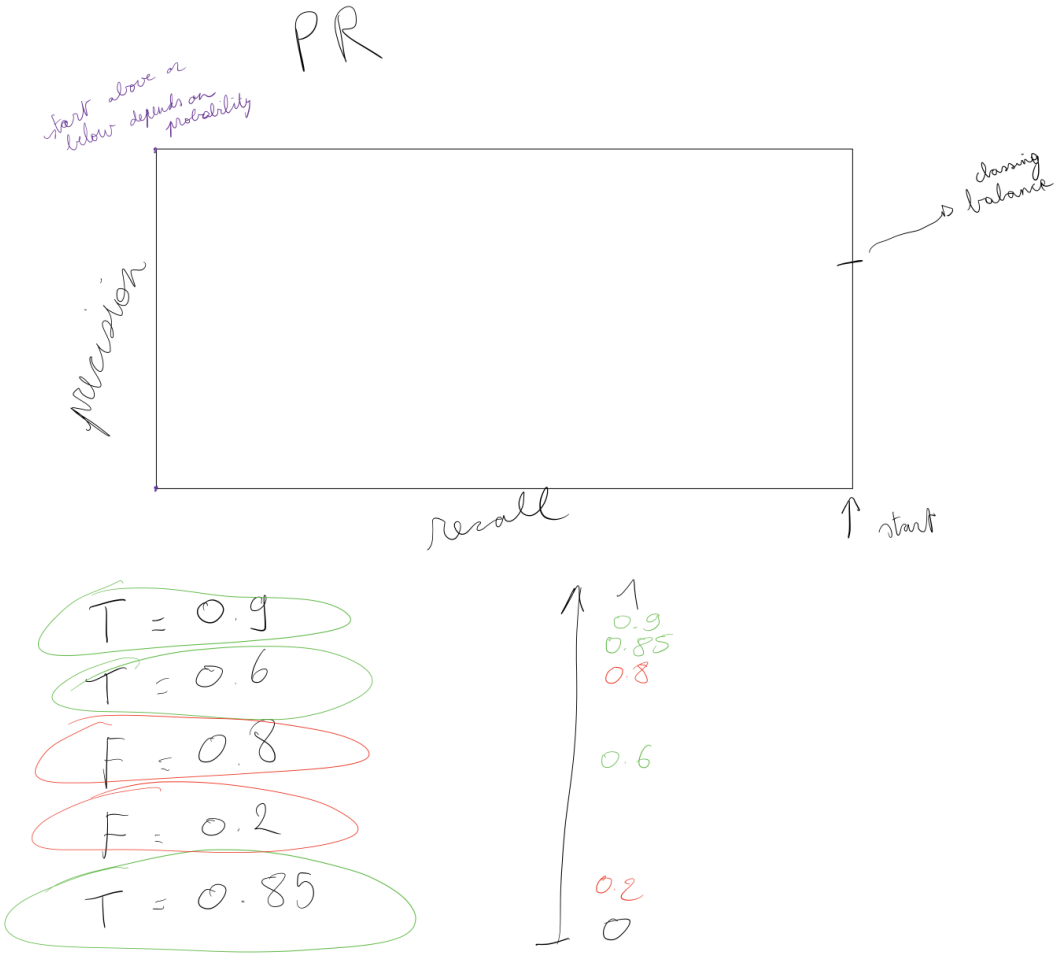
ROC



→ good results because
not many patients w/ cancer
and a lot of other patients

PR (precision recall) curve:

- Recall is the same as the sensitivity → show many of the ones that I want to detect do I detect?
- Precision = of those above the line, how many should there be?
- Classing balance = if you have 10 of class A and 10 of B and you put your threshold at 0, 10 should be above it but the other 10 should not be → 50%

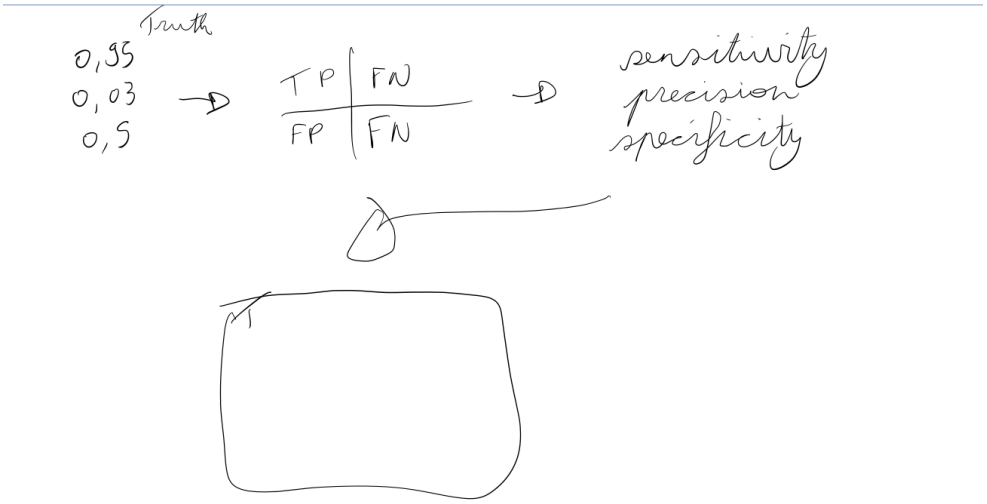


→ plot ROC & PR curve together

- https://en.wikipedia.org/wiki/Receiver_operating_characteristic
- https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
- https://www.youtube.com/watch?v=4jRBRDbJemM
- https://developers.google.com/machine-learning/crash-course/classification/video-lecture

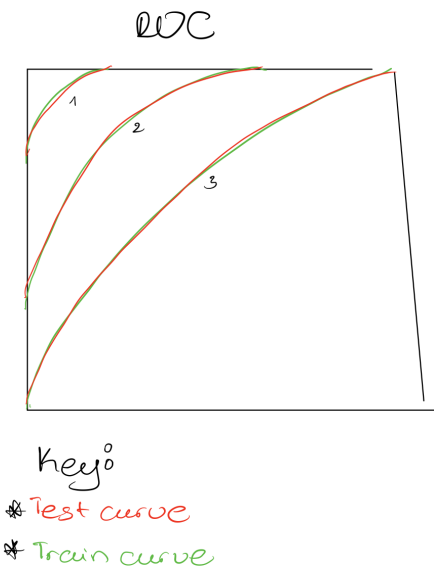
RECAP

You have a trained model → you get 20 labels or 20 values → You get the truth and start at a threshold at either 0 or 1 and at every threshold you calculate the confusion matrix → then calculate metrics: sensitivity, precision, specificity. → then you have a data frame for each threshold. → make plots (ROC/PR curves)



Why is an ROC and PR curve relevant for both training and test dataset?

To see the data is overfilled or it has learned.



- 1st curve has the best performance, the model has learned the patterns.
- 2nd curve has the better performance then the 3rd curve, the model. is overfit and has learned the patterns
- 3rd curve has poor performance, the model is not overfit.

Which one (test/train) will be better in the ROC curve?

If the problem is complex and the model has learned, it is possible it is not perfect. Mostly if a score is perfect, it is overfitted. At most, you are expecting test and train result to be the same.

Cross-Validation (CV)

- Multiple train/test splits (multiple models?)
- Optimal use of data
- k-fold (Stratified), Leave-one-out (LOO)
- https://www.youtube.com/watch?v=fSytzGwwBVw
- https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85
- https://medium.com/@analyttica/what-is-meant-by-stratified-split-289a8a986a90

If you have a dataset and want to build a model and evaluate the performance:

- Split the dataset into test and train data
  - Split in 80/20 (Pareto)
  - 70/30
  - Whatever split you take, which one would you choose? → **Stratified split** = a split which is representative for the true distribution of classes.
    - fe. dataset with 20 rows (10 of each class):
      - 18(train)/2(test)
        - Could be difficult to start seeing a pattern
        - Solution: cross-validation

Cross-validation

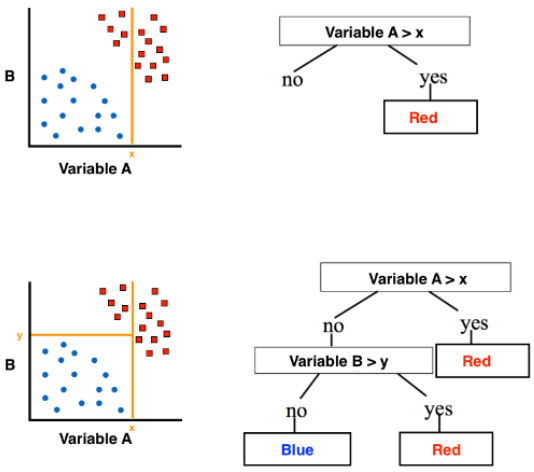
- dividing data not in train/test set but in folds → fe. K-Fold cross-validation, K = 10 → divide data in 10 equal parts
- In each fold should be stratified, a distribution similar to the overall distribution
  - Train first model: use folds 1 to 9 and the 10th fold is the test fold
  - Next time: train folds 2 - 10 and use 1st fold as test
  - And so on.
  - You get the probabilities per fold but you're gonna wait for all folds and at the end for each row you have a probability output from a model that has not seen the data → make a really performative ROC curve.
  - Which model to use? → make 1 model of it (if performance is good)
  - Advantages:
    - Optimally using your dataset
    - Less dependent on sampling → how many % for test/train
  - Disadvantages:
    - Problem, you have 10 models that are slightly different

Chapter 4: Random Forest & Feature Selection

Random Forest

- How does a decision tree work? → decision trees overfit by default.
  - gets data and label, decision tree tries to use these variables to split the data
- How does a random forest work? (building process, decision process, output)
- What are the main parameters and what do they do?

Decision Tree



1. Select the feature with the best separation between classes (how to quantify 'best'?)
2. Use this feature to make the next nodes (leaves) in the tree
3. Repeat process in each node until all samples have been classified (only one class at each endpoint)

Calculate separation quality

- quantify the Gini impurity in each node:
  - $i = 1 - p_P^2 - p_N^2$
- Pp is the proportion (fraction) of positives and vice versa for the negatives
- Use a feature to determine a split in the node and calculate
- the decrease in Gini impurity δi:
  - $\delta i = i_{parent} - i_{child1} * f_{child1} - i_{child2} * f_{child2}$
- f is the fraction of parent samples in that child

Gini impurity

- Lower Gini impurity signifies better separation between classes
- If only a single class occurs and the other is absent (perfect separation), the Gini impurity i will be 0 (= 1 - 1 - 0)
- δi will be 0 if there is no improvement in the separation

0.5 is the highest impurity you can have.

Decision Tree example

Heart	Smoking	Gender
TRUE	TRUE	M
FALSE	FALSE	F
TRUE	FALSE	M
TRUE	TRUE	F
FALSE	FALSE	M
FALSE	TRUE	F
TRUE	TRUE	M
FALSE	FALSE	F
TRUE	TRUE	M
TRUE	TRUE	F

1. Make tree to classify heart disease
2. First node in tree: Smoking or Gender?

Decision Tree (dis)advantages

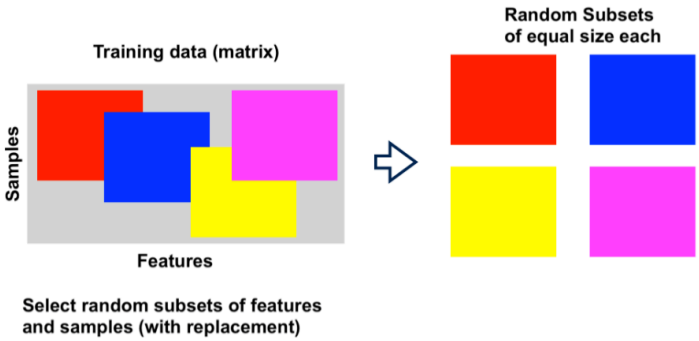
- + Intuitive approach, easy to implement and understand
- + Easy to check why a given sample is classified a certain way (so not a black box)
- - Approach is 'greedy' and will continue until all samples are correctly classified (long and overfit trees)
- - Tend to be very dependent on the input data and small changes can lead to very different trees

Random Forest

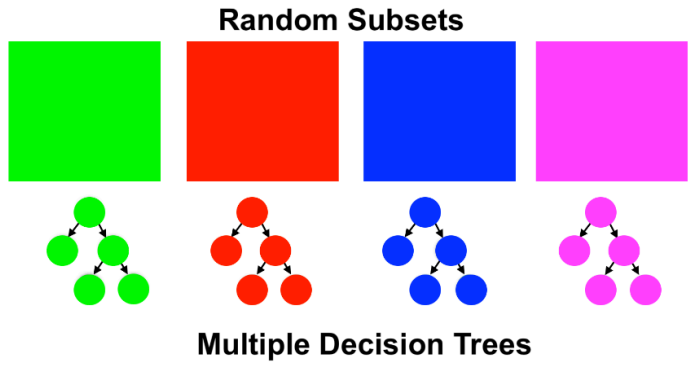
3 crucial concepts:

- Random subsets of data
- Multiple trees
- Voting to get result

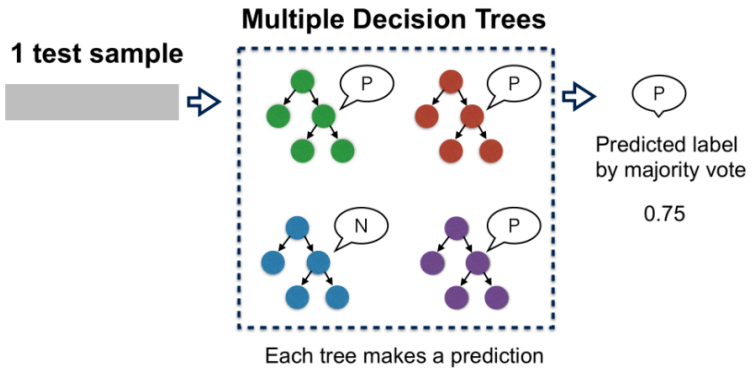
Random Forest step 1



Random Forest step 2



Random Forest step 3



Random Forest sources

- https://towardsdatascience.com/why-random-forests-outperform-decision-trees-1b0f175a0b5
- https://towardsdatascience.com/from-a-single-decision-tree-to-a-random-forest-b9523be65147

Feature Selection

- Manual feature selection
- Statistical feature selection
- Model-based feature selection
- Boruta feature selection

Boruta Source

- https://towardsdatascience.com/boruta-explained-the-way-i-wish-someone-explained-it-to-me-4489d70e154a

Chapter 5: Data Leakage

Data Leakage (information leak)

There is a leak in your pipeline → can affect the data → mistake you made

Data leakage happens when there is information leaking into the model which should not be there. The result ranges from overly-optimistic performance results to a model which simply will not work in the real-world.

There are 2 main types of data leakage:
1. Target leakage
2. Train/test contamination

Target Leakage → most obvious

Target leakage is the type of data leakage with the most obvious effect on performance. It can really destroy a model's performance. This also makes it easier to identify.

Problem	AUC	Real performance
Target (outcome) variable included in training phase	Perfect	Error → variable not found
Variables only known after event occurrence are included (chronology)	Perfect / Good	No error but low performance
Time series values after event are included (windmill failure detection)	Problem specific	Model too slow (late predictions)

Q: What happens when there is a variable which is highly correlated to the outcome variable, but which is known at the time of prediction (so no chronology issues). Is there data leakage? Do you include it yes/no? Investigate why this value is correlated.

Train/Test Contamination

This type is more subtle, it causes **smaller problems** but these are therefore harder to detect. Consequently, these occur very often in the real-world. It happens most often in the preprocessing phase. Some examples:

Problem	AUC	Real performance
Bad train/test split (or fold selection). There is test set information in the training data (patients/windmills) → for example, windmill that was build first, fails often → model will always say this one (this year) will be breaking (got into training set)	Optimistic	Lower
Feature selection before train/test split → the model has "peeked" into the validation set even before predicting on it.	Optimistic	Lower
Imputation/scaling before train/test split → can't see the difference anymore, all features become equally important. fe. shoe size: we have scaled the data (0 = size 35, 1 = size 43) so it will always take that 1 into accountant if it learned correctly it will also take 1.1 into account	Optimistic	(Slightly) lower

How To Detect

Sometimes there are signs that you had a data/information leak in your pipeline. Below are some warnings and red flags:

1. Model gives error in production (missing a feature)
2. (Near) perfect performance for test set (95% AUC be very cautious)
3. Highly/perfectly correlated features (check if chronology is ok)
4. (Small) difference in performance between test/validation and real-world implementation

Validation set

Often in data science courses/trainings they recommend using a train/test/validation setup. Whereby you take a validation dataset at the very beginning which you only use at the very end before going into production.

This obviously has some drawbacks:

A. You lose part of your data
B. This might not detect certain data leaks:
   - Chronology problems with variables
   - Chronology problems with time series
   - Bad split e.g. with patients or windmills being in this set and in the training set

A good validation dataset: when you build your model with all the data and cross-validation etc, and before you go into production, you talk to the people there, and you say, “hey can I run my model in the background for a week?”

Sources & Materials

- https://www.analyticsvidhya.com/blog/2021/07/data-leakage-and-its-effect-on-the-performance-of-an-ml-model
- https://machinelearningmastery.com/data-leakage-machine-learning

Chapter 6: Time-Series Forecasting

Intro

Time-Series Forecasting is widely used in business contexts for predictions purposes

- sales
- customers
- stock
- server load
- anomaly detection
- etc

Difference with ML

ML:

- NxM matrix format data + outcome vector y (length N).
- we use M variables to predict the outcome vector y for unknown rows N*

Time-series forecasting:

- we have a time series y(t), which stops at time t = 0
- we want to predict y(t > 0) and for this we use y(t ≤ 0)
- i.e. the variable we want to predict is also the one we use to fit (train in ML)

The problem

Time-Series Forecasting is difficult. And people with good forecasting skills are hard to find.

The reason is mostly down to the algorithms and theory behind it being rather difficult to automate, tune or even set up.

The result is that in many companies the need for forecasts greatly outweighs the rate at which they can be produced.

Quiz 2

1. What is the difference on a data level between time-series forecasting and machine learning?
   - Time-series doesn't have factors in the prediction modelling.
2. An additive prophet model looks something like this:
   1. y(t) = g(t) + s(t) + h(t) + e
   2. What are g, s, h and e?
   - g = trends, s = periodic changes, h = holiday effect, e = errors
3. Prophet does not worker all types of time-series data/problems (e.g. not for stocks etc.), or at least that is not what was designed for. What is the type of data/problem for which it van be used?
   - data with non-linear trend. It takes into account the daily, weekly and yearly seasonalities.

Facebook Prophet

The Prophet algorithm was developed by Facebook (https://facebook.github.io/prophet/) so that their data scientist could do many of the forecasts themselves, without needing to go to a forecasting expert every time.

The aim of the algorithm is not to solve each forecasting problem, rather it can be used for specific problems but with great flexibility. Some problems for which it can work (spot the common theme):

- predicting airline passengers over time
- predicting crowds over time
- predicting sales over time
- predicting server load over time

The common theme is really people interacting with a certain service. So Facebook Prophet is really only suited for predictions in a business context.

It works well for data with strong cycles or seasonalities (daily/weekly/yearly), with certain outlier events known in advance (holiday events, new product launches), and natural growth limits (e.g. product market saturation).

Note that forecasting experts will generally produce better forecasting results with the classical tools, but this takes time.

How does it work

The time-series forecasting problem is reduced from a problem whereby time is explicitly modelled to a simply a fitting problem. The following function is fit to the data y(t):

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

With *g* the trend component, *s* the periodic changes (seasonality), *h* the holiday effects and *E* the error component.

Important to note is that this model is additive, i.e. the components are simply added together and can thus be interpreted independently. But also, more can be added if needed (extra seasonality) but more on that later.

The Trend Component

The trend component is fitted automatically. There are 2 trend types available (problem specific):

- piecewise linear (changepoints selected manually or automatic)
- saturating growth model (product market saturation)

Seasonality

There are different seasonalities available. This is useful because patterns can occur at multiple scales. For example:

- **yearly seasonality** are patterns which occur on a yearly repeating scale but not necessarily on the same date e.g. holiday periods like easter of the start of the new school year.
- weekly seasonalities capture patterns which occur every week scale e.g. weekday-weekend differences
- (Optionally) daily seasonalities captures pattern occurring each day. This becomes tricky as this is on top of the weekly patterns and should therefor be somewhat consistent for each consecutive day.

These seasonalities are fitted with a Fourier Series (https://en.wikipedia.org/wiki/Fourier_series)

Holidays

Holidays are things which can have a tremendous effect on business time series. Why?

Holidays and events sometimes do not occur every year at the same time (i.e. every 76th day of the year), but are determined by lunar/religious/etc rules. The assumption Prophet makes is that the effect of the event is usually the same year over year, but the date is not.

Sources

Paper: Forecasting at Scale by SJ Taylor and B Letham
https://facebook.github.io/prophet/
https://www.youtube.com/watch?v=pOYAXv15r3A
By next lesson:

- Watch the vid
- quiz on FB Prophet (begin of next lesson)
- find a time series dataset you want to predict and set it up in a notebook (load it in and visualise the signal).

Multiplicative or Additive

- Additive: → default, you add all components together

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t$$

- Multiplicative: → add trend * seasonality & add trend * holidays & add error → sometimes there are different patterns

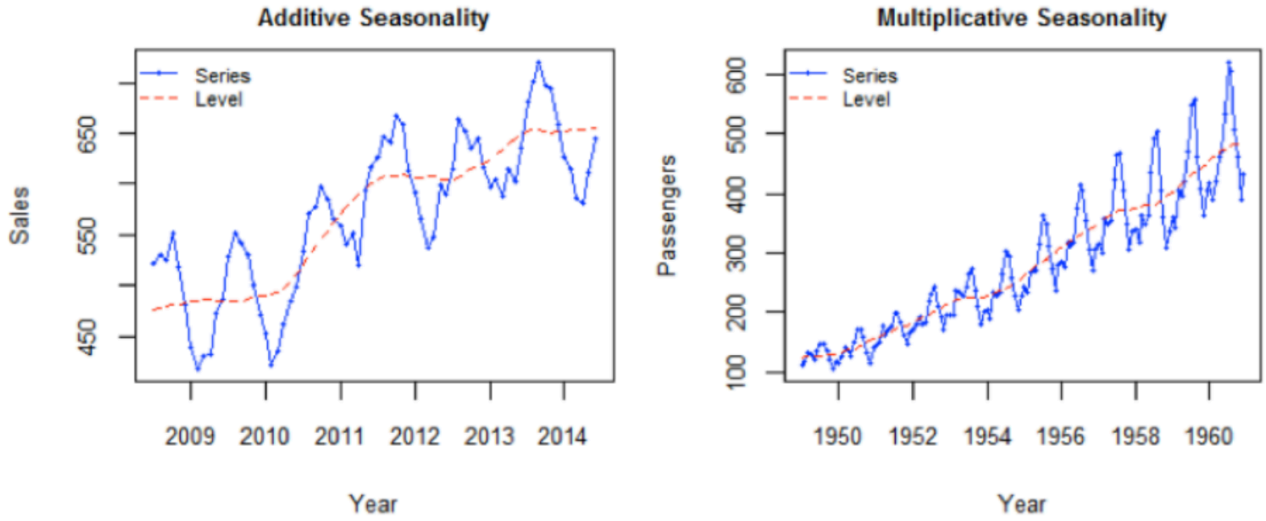
$$\hat{y} = g(t) + g(t) \cdot s(t) + g(t) \cdot h(t) + \epsilon_t$$

- or:

$$\hat{y} = g(t) \cdot (1 + s(t) + h(t)) + \epsilon_t$$

Again with *g* the trend component, *s* the periodic changes (seasonality), *h* the holiday effects and the *E* error component.

Multiplicative or Additive: spot the difference



Performance evaluation

How good is my forecast?

Different data to be used (think of ML):

1. metrics calculated on the training data performance (goodness of fit)
2. metrics calculated on a test set (end of data, remember data leakage)

Performance evaluation: Training data

by checking the performance on the training data we can evaluate a number of things:

- get a feel for the model: how do the model components look like, do these make sense or are they overfitting?
- are we missing things: by looking at the residuals we can check for missed (periodic) patterns

if you see that errors go up every time → maybe include a holiday parameter or something.

Performance evaluation: Test data

The best quantification whether the model is actually able to predict things is by looking at the predictions on the test data.

- Root Mean Square Error/Deviation:

$$RMSE/RMSD = \sqrt{\frac{\sum_i^N (\hat{y}_i - y_i)^2}{N}}$$

→ ŷ = predictions - truth / how many you have
→ looks like standard deviation

- Mean absolute error:

$$MAE = \frac{\sum_i^N |\hat{y}_i - y_i|}{N}$$

→ in the RMSE the larger ones contribute more so you penalise more for larger errors and less for smaller. Here everything is added on the same pace

- Mean Absolute Percentage Error/Deviation:

$$MAPE/MAPD = \frac{1}{N}\sum_i^N \left|\frac{\hat{y}_i - y_i}{y_i}\right|$$

→ MAE in percentage: % error of how much you are off.

Example shop:
predict next 3 days in shop → evaluate model with last 3 days off (we want to predict those).
3 actual values: 1100, 1000, 500
predictions: 1000, 1100, 500
RMSE: -100 off / +100 off / - → don't add them together but square them (makes everything positive) and divide by amount of predictions (to get an idea for how much you're off every time) take root again to balance it out again.
MAE: make numbers absolute: 100 + 100 / 3 in this case
MAPE: divide by actual value fe. 1100

Besides the default performance metrics there are also more visual variants:

- Plot of data vs predictions
- Distribution plot of residuals
- Correlation plot → if you don't care about time
- Rolling correlation plot (N-day or monthly): time series of the correlation over time.

DIY

Use the Auckland cycle data and do the following:

1. load in the data and select only the cyclists on Tamaki Drive (choose one way for your convenience).
2. Select a test set of a full year
3. train a basic model and evaluate the performance both numerically and visually
4. add holidays and evaluate the performance
5. add weather and evaluate the performance

Chapter 7: Neural networks - the (possible) future of ML & AI

Intro

Machine Learning

1. pre-process data
2. (build features)
3. build model
4. evaluate performance

How do you see the world?

https://www.theguardian.com/education/2012/nov/12/improbable-research-seeing-upside-down

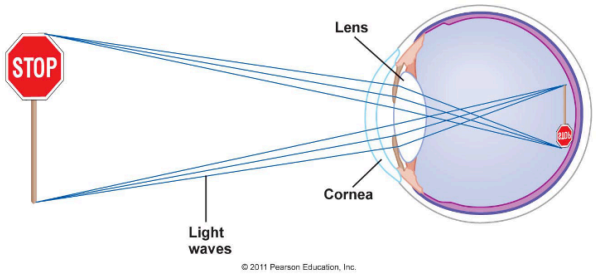


Figure: You already see the world upside down.

(Artificial) Neurons

The Brain

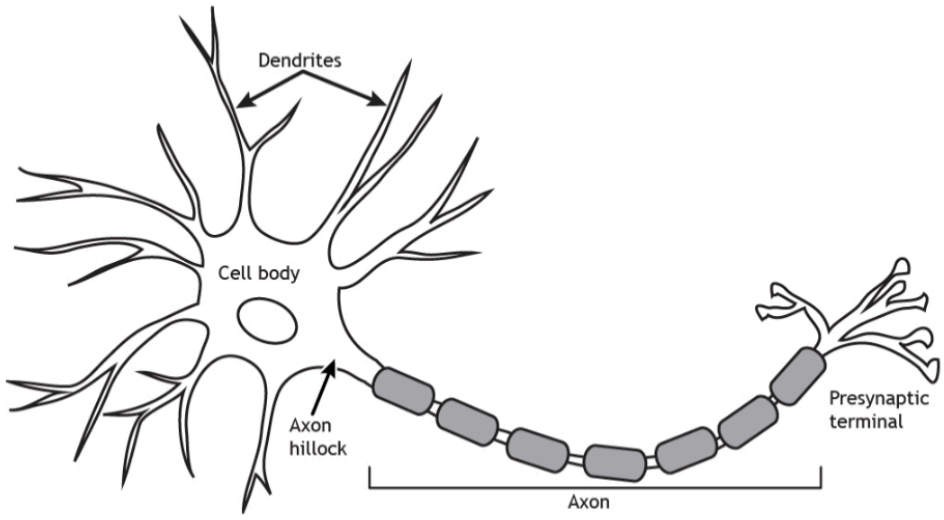
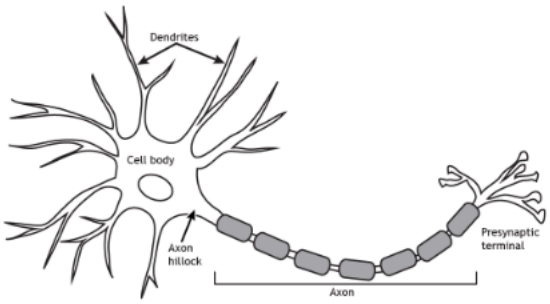
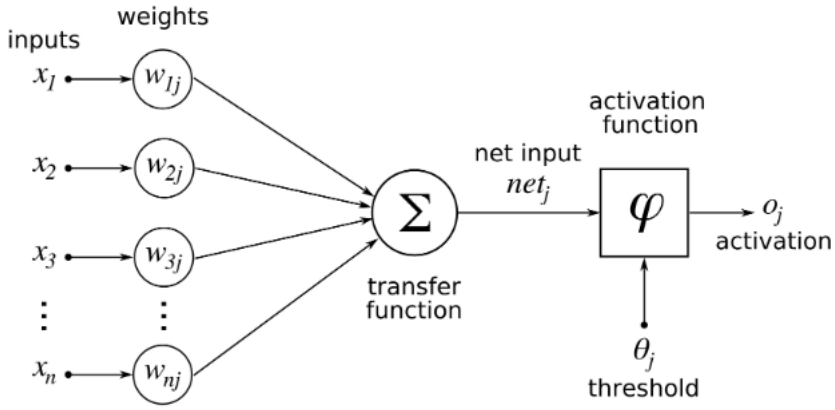
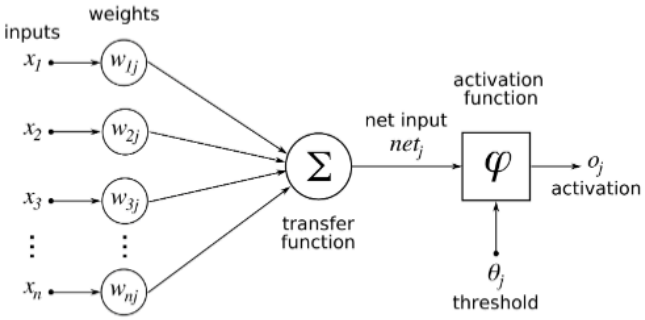


Figure: 'Neuron' by Casey Henley

Artificial Neuron



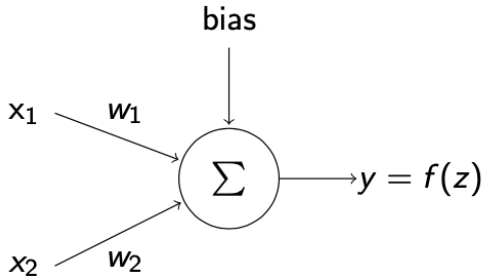




- inputs = features
- transfer function $= \sum x_i \cdot w_i$
- activation: net input $\geq$ threshold

Artificial Neuron in 2D feature space

Example, data with 2 features x_1 and x_2:



Activation functions:

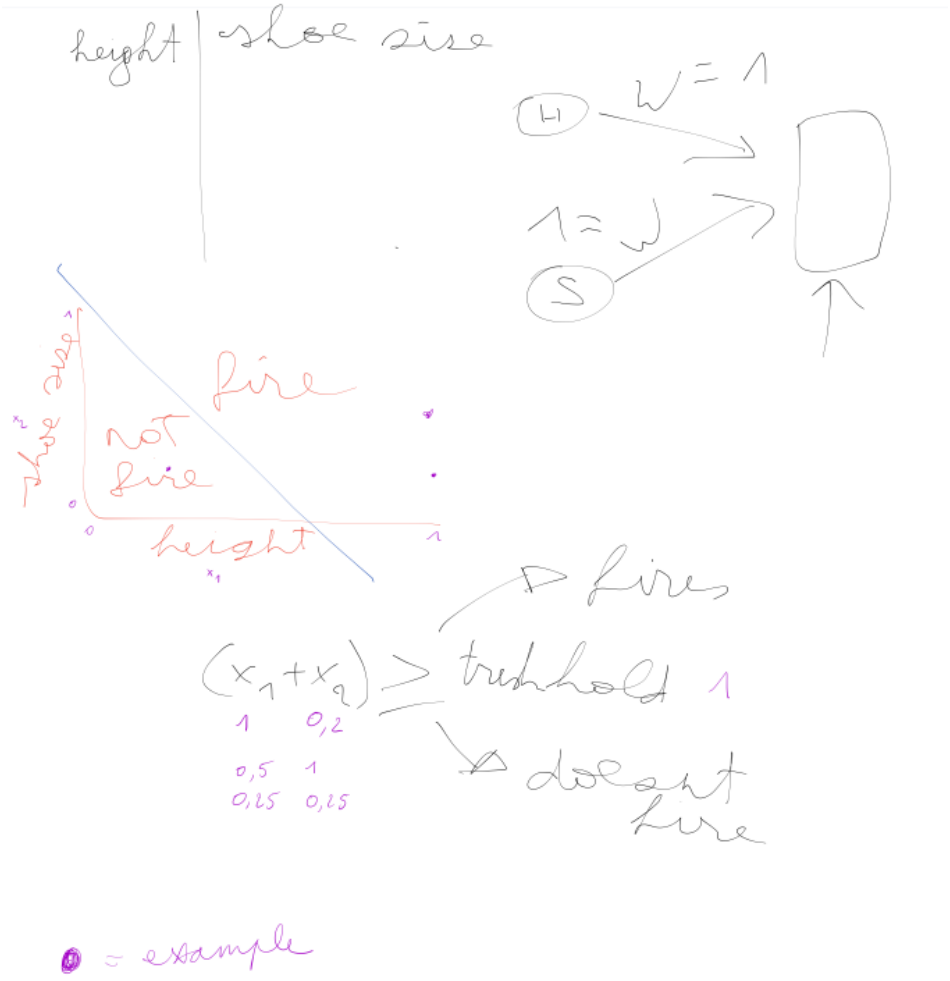
$f(z) = \sigma(z) = \frac{1}{1+e^{-x}}$

$f(z) = ReLU = max(0, z)$

$$z = bias + \sum_{i=1}^{2} x_i \cdot w_i$$

This boils down to a decision line in 2D feature space!

Example height and shoe size:



x1 = heigh
x2 = shoe size
→ x1 + x2 if that is bigger than a threshold, then it fires, and if it's lower, it doesn't fire. → it is a classifier.

Activation function is something special: it used to be a sigmoid:

→ range of input values (sum of x1 + x2 + bias)
→ sigmoid makes inputs (extremely positive or negative) to what the neuron can do: fire or not fire. In-between is a linear thing.
→ problem: not easy to calculate
→ relU matches the input and is much easier to calculate.

bias can be defined as the constant which is added to the product of features and weights. It is used to offset the result. It helps the models to shift the activation function towards the positive or negative side.

Neural Network: introductory problem

MNIST dataset

How would you train a model for this?

- pre-processing? → process images so that certain features are calculated (think about flowers)
- model input/features?

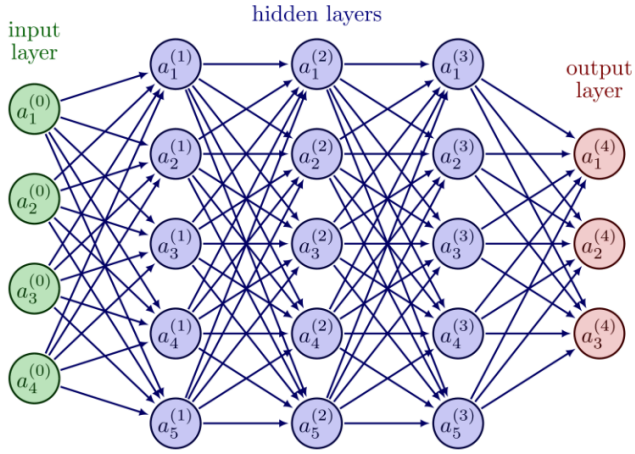


Figure: Note this is a specific type of neural network called a fully connected neural network.

→ hidden layers: model can do whatever they want (we also don't control the weights)

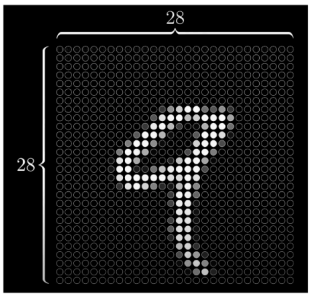


Figure: Screenshot from 3Blue1Brown (see sources).

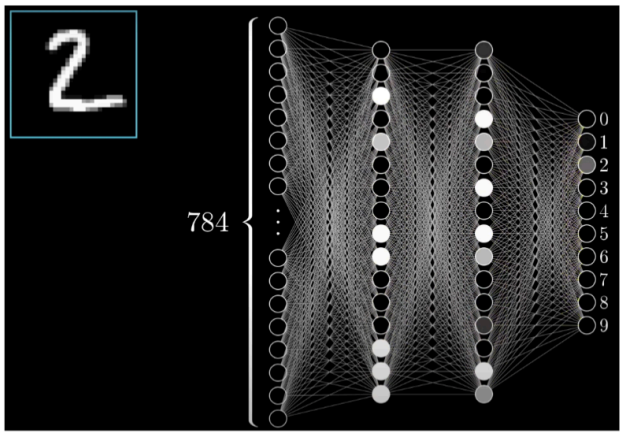


Figure: Screenshot from 3Blue1Brown (see sources).

Neural Network: layers and parameters

MINST example with shown network:

- 4 layers:
  - 1 input layer with 784 input nodes (grayscale pixels) → 28x28 pixels
  - 2 hidden layers with 16 nodes each (arbitrary number in this case)
  - 1 output layer with as many nodes as there are classes
- 784 weights for each node in the second layer
- 1 bias per node
- node activation is determined by:
  - input nodes
  - weights → values multiplied by input → * 0 = 0, * 1 = everything lights up. Set at random → tell the model how bad it is.
  - bias
  - activation function (sigmoid, ReLu, step, etc.)

Neural Network: training

1. Give the model a training instance (image, …)
2. Let the model give the output
3. **Tell the model how bad it is**
4. Give an other instance

How to tell the model how bad it is

Definition of Loss and Cost function.

- The loss function determines the difference between what it should be (the truth) and what the model outputs for one instance. → how bad the model is (high when model is off) → square the difference to get the absolute difference.
- The cost function summarises the loss over the full training set (optionally with a model complexity penalty). → sum of all losses

Objective is to minimise the Cost, and thus the Loss. This is done in an optimisation procedure. Often this is Gradient Descent but many others exist.

Gradient Descent

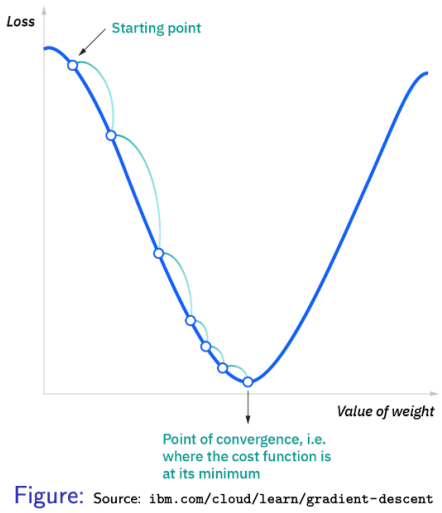


Figure: Source: ibm.com/cloud/learn/gradient-descent

→ after first random weights, we alter the weights a tiny bit, some higher, some lower. Calculate cost again. Try again and again until you find the point of convergence

→ might not reach the global optimum.
→ we all start at a random point, so everyone gets a different local minimum.

Software 2.0

Introduced by Andrej Karpathy
https://karpathy.medium.com/software-2-0-a64152b37c35

Figure: `https://xkcd.com/1838/`

Artificial Neural Networks (and accompanying deep learning) are not just other algorithms which you choose instead of e.g. a Random Forest. This is possible of course, but the possible disruptive nature of this type of learning is much more than that.

- provide data (Labels, augmentation)
- define learning framework/architecture (layers, size, etc) determined by available performance
- train model
- evaluate
- deploy

Software 2.0 VS ML

Difference with ML as we have seen?

- A lot more data needed:
  - data augmentation (make more data from the data you have)
  - gathering of labels
- Model determines what is important, not humans (features are not constructed by people but by computers)
- Humans provide a learning framework (Data, model, architecture, computer power)
- (Sometimes) low model explainability
- More compute power necessary

Example of the future today

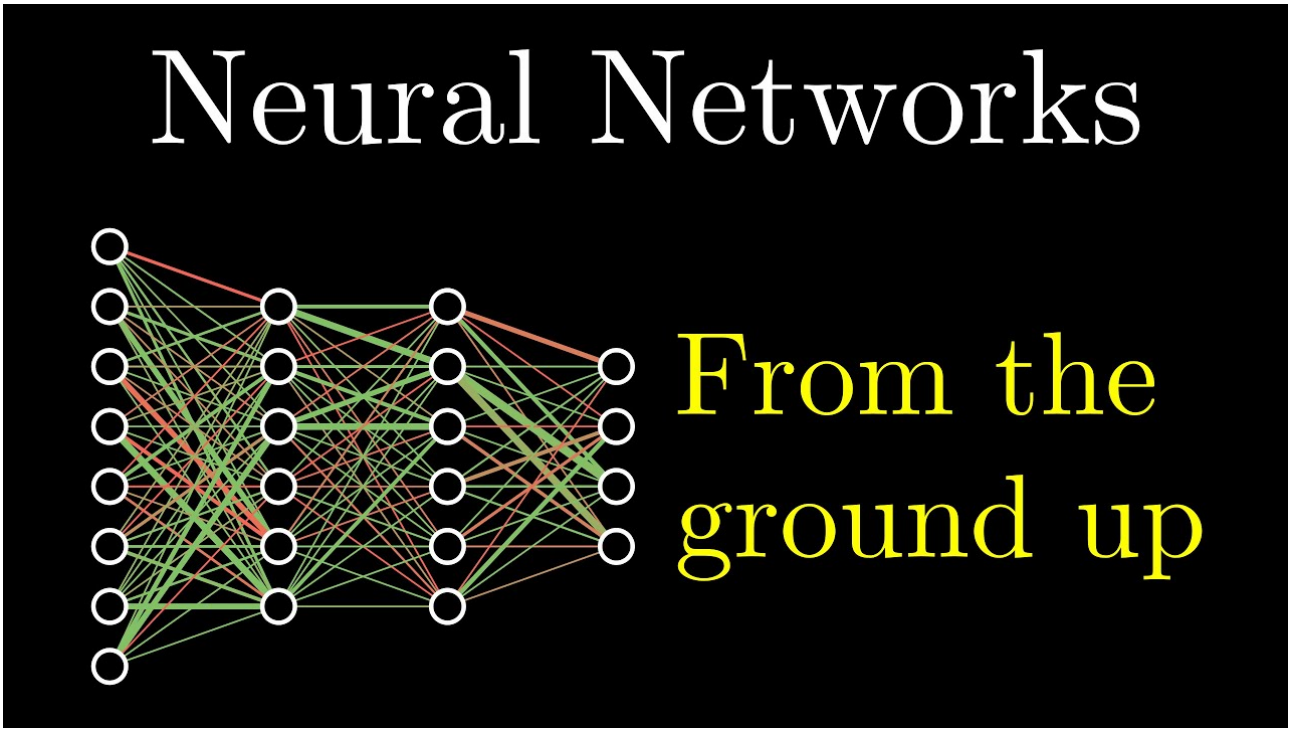
Mostly Google (DeepMind) and OpenAI

- ChatGP (https://chat.openai.com)
- DALL.E (https://chat.openai.com/blog/dall-e/)
- Tesla/Google/others autonomous driving
- GitHub Copilot
- Google Deepmind research (https://deepmind.com/reserach)

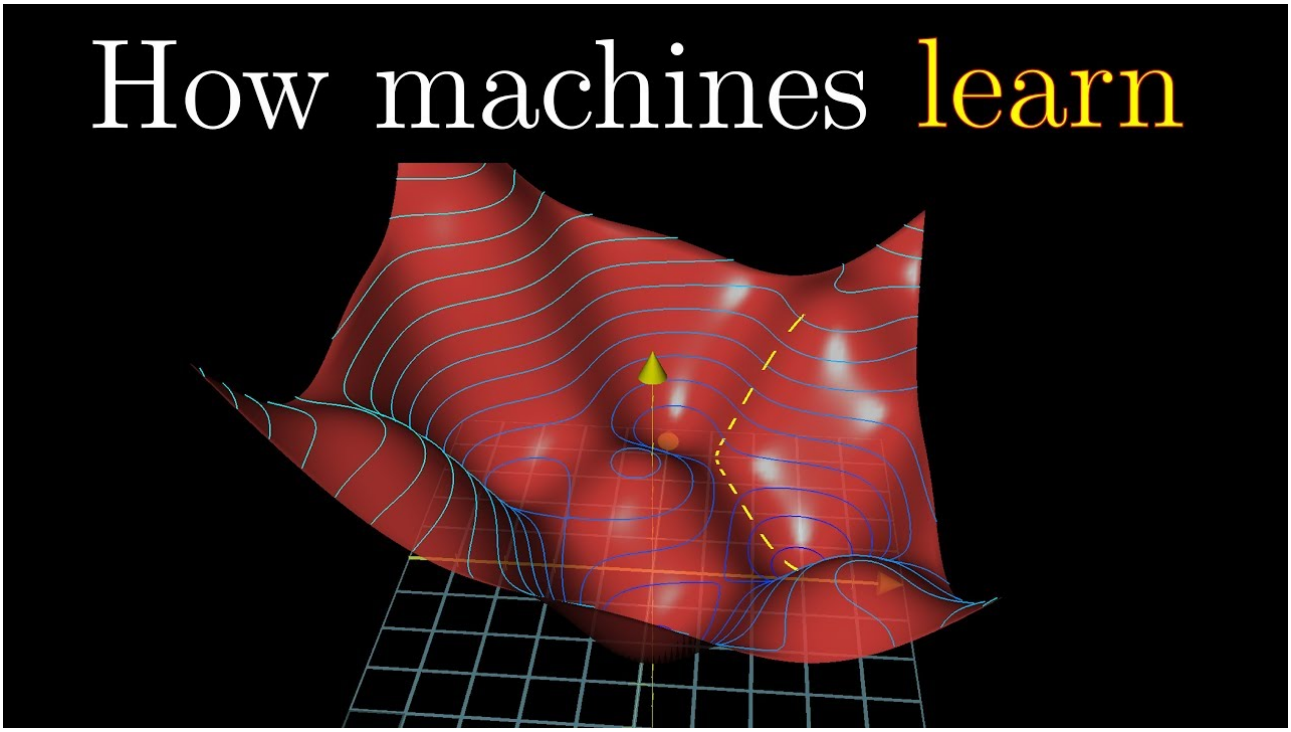
Sources & Materials

Included in the stuff you need to know for the exam:

- But what is a neural network?

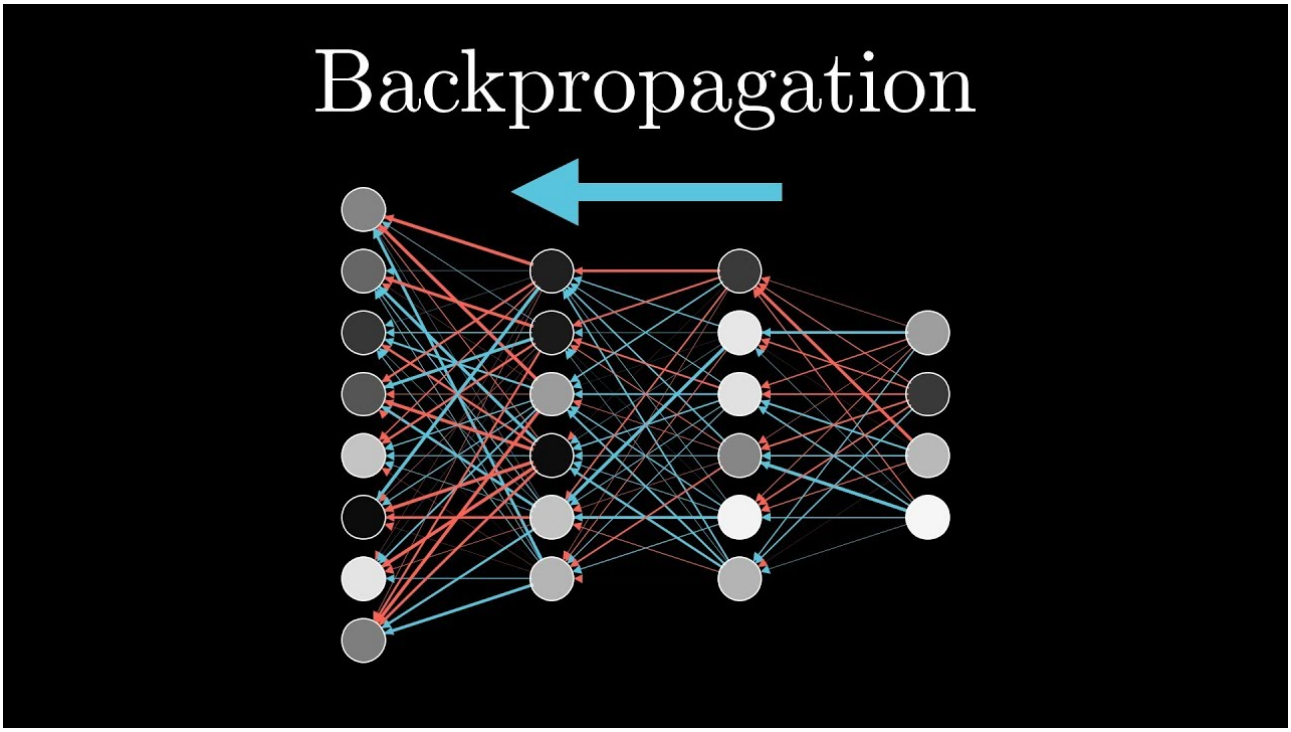


- Gradient descent, how neural networks learn



Extra: only if you are interested:

- What is backpropagation really doing?



- Andrej Karpathy: Tesla AI, Self-Driving, Optimus, Aliens, and AGI

Exam → question can be: **What is gradient descent, when does that happen, what's the problem and what does it do?** Need to be able to say what it does, what it solves and where it is used, show that the concept is clear.