Introduction to Machine Learning

Kristina Kislyakova
21.01.2026

Some announcements

Mini-Projects

- Mini-projects are online
- The deadline for submission is February, 15th, 2026.
- You can upload your solution to Github.

Exams

- 28th of January and 4th of February
- Please register by tonight.

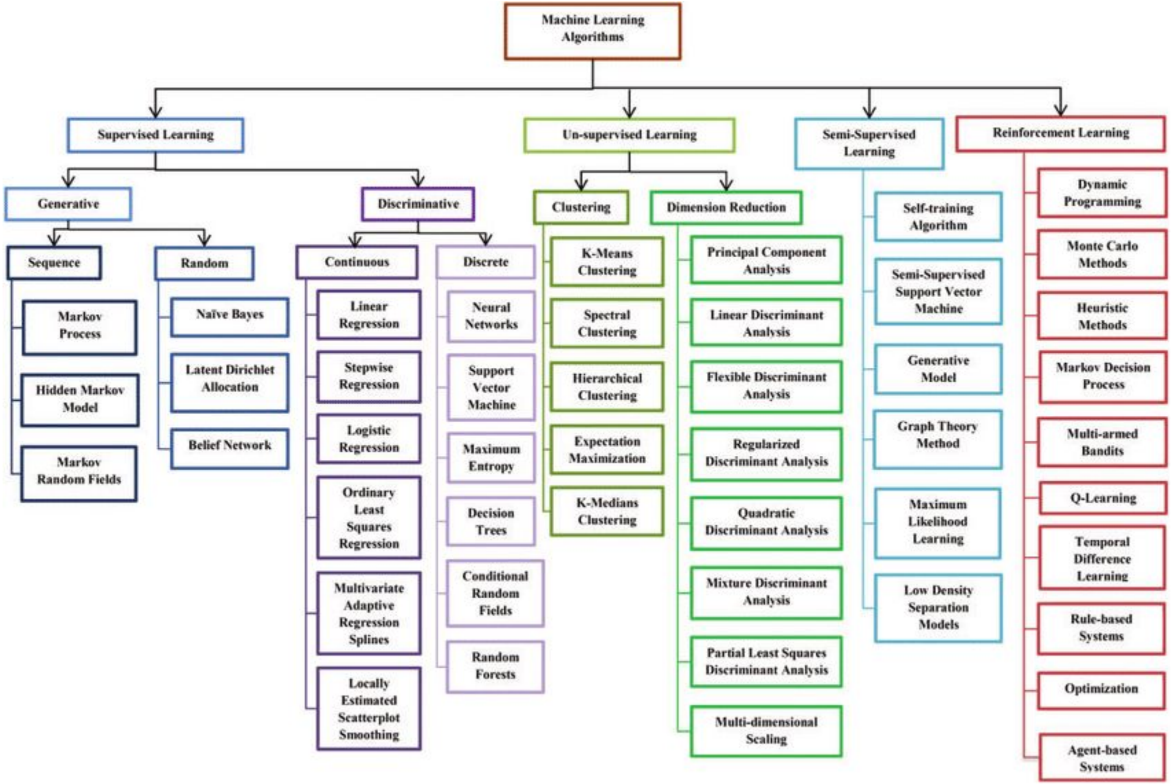
Outline

- **Numerical simulations**
- **Dimensionality reduction**
- **Simple machine learning examples**
    - Unsupervised
    - Supervised
    - Real world examples (*and problems*)
- **Neural Networks**
    - Architecture
    - Training and Validation
    - Caveats and Warnings
    - Astrophysical examples (and cautionary tales!)

What is meant by AI/ML

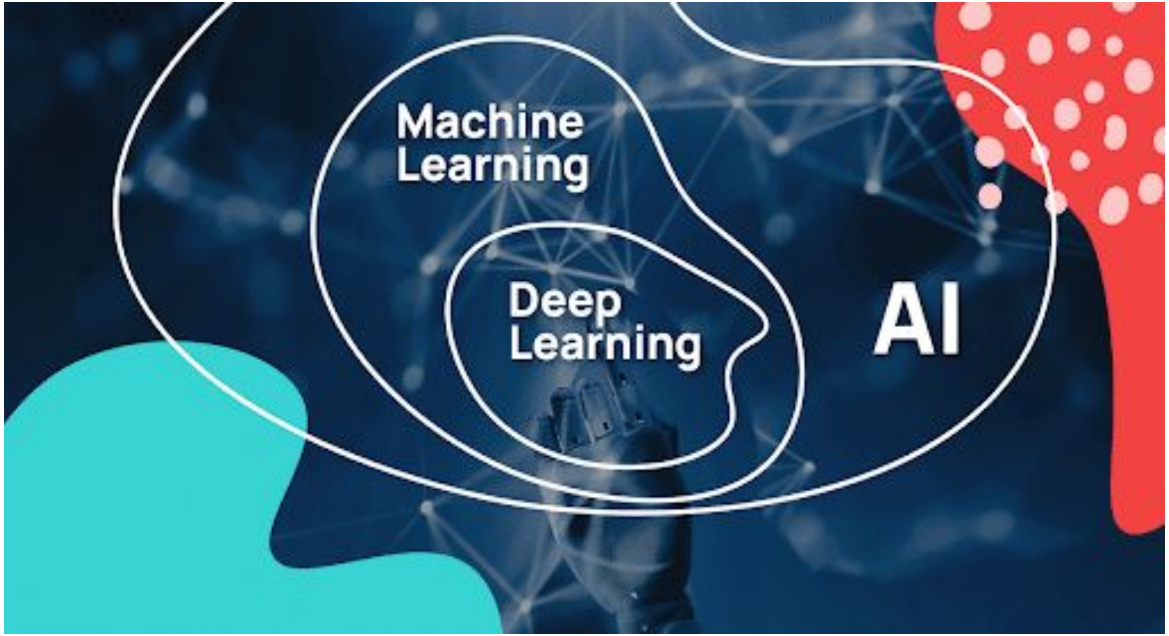
A lot of the buzzwords in this field have overlap/ambiguity between what they mean in practice, depending on who you talk to

machine learning…
artificial intelligence…
deep learning…
adversarial networks…
random forests…
support vectors…



What is meant by AI/ML

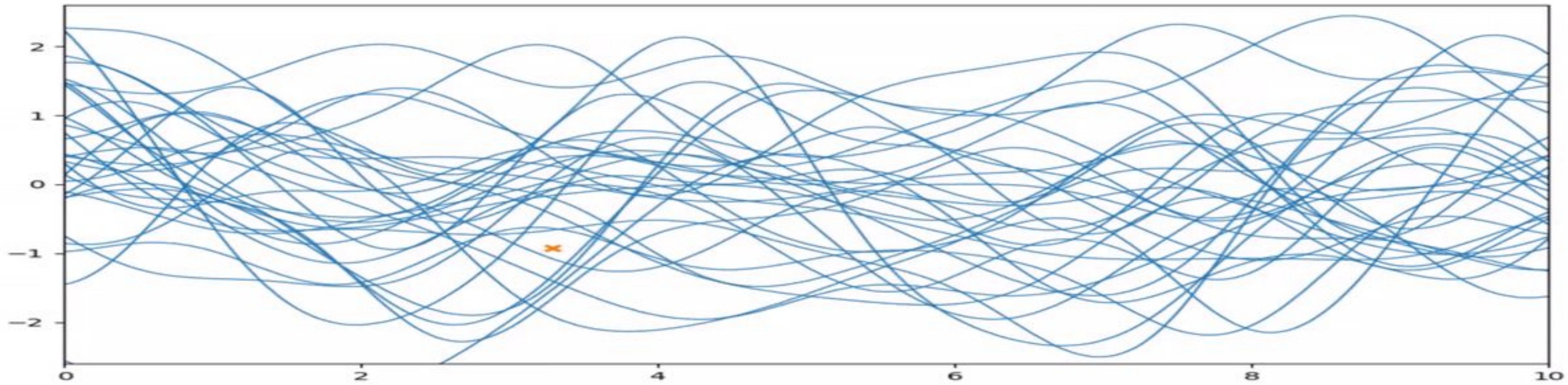
But in general we are thinking of algorithms which can study large datasets together with a model architecture which uses training sets and validation steps to help develop rules so that it can predict/classify/optimize outcomes of some problem.



What is meant by AI/ML

ML can be familiar and simple…

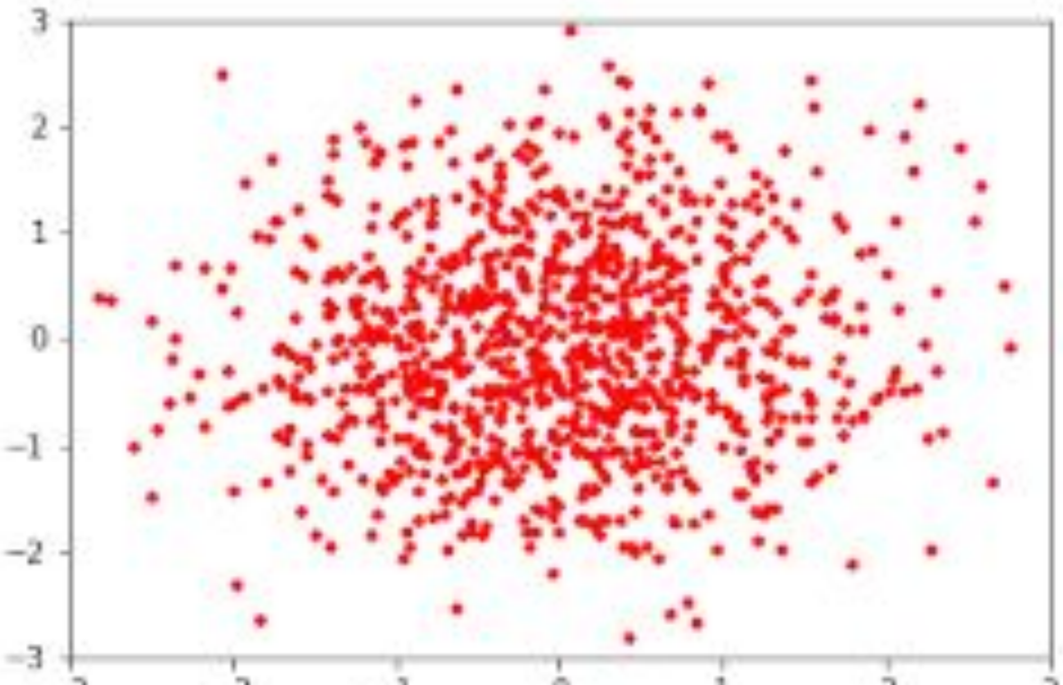
- *Interpolation in a flexible way (Gaussian Process Regression)*



What is meant by AI/ML

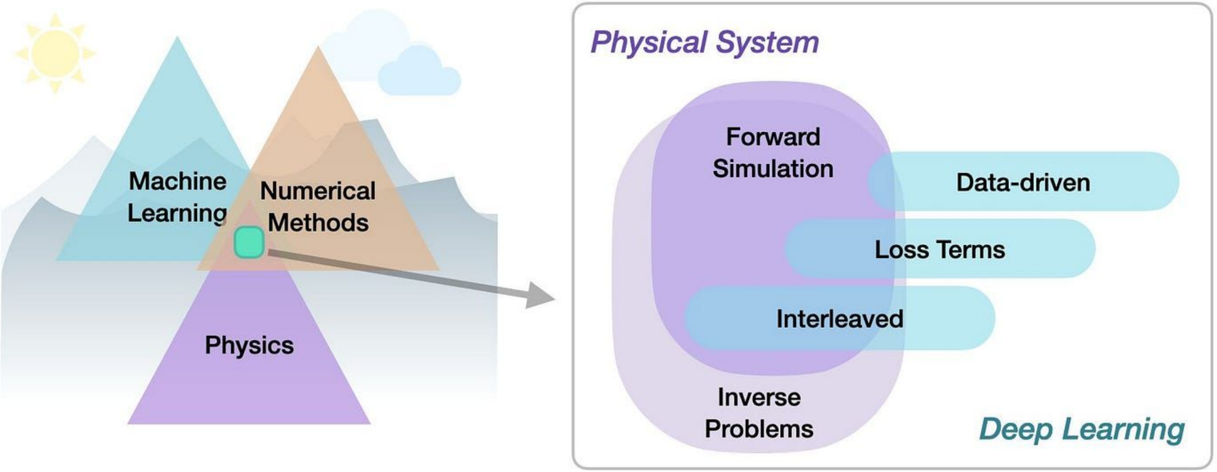
…
or less familiar and more abstract

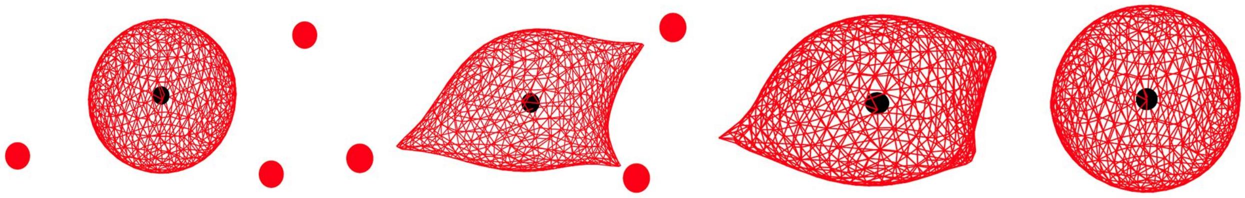
- *Normalizing flows to reconstruct multi-dimensional distribution functions*



What is meant by AI/ML

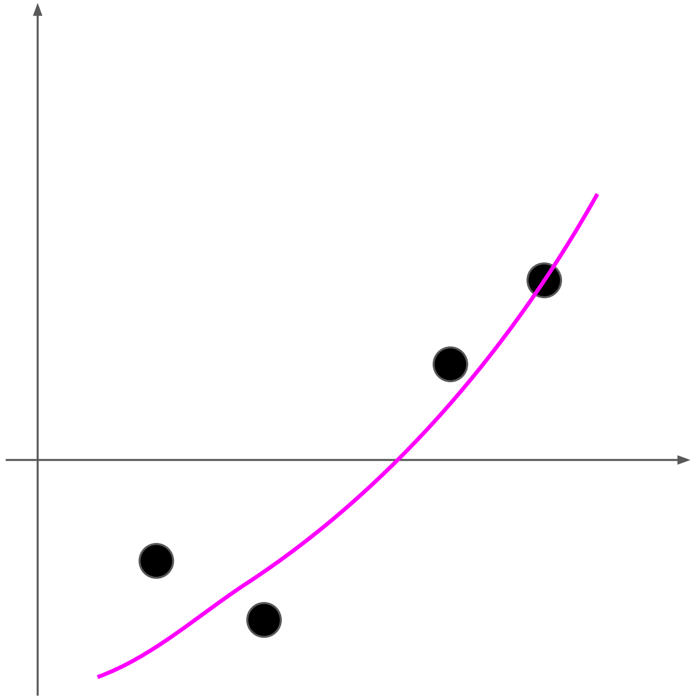
In any case, the usage, goal and approach may be different - but several of the numerical methods (*sampling, model fitting, optimization*) we discussed will be relevant when implementing any of them.





Interpolation, approximation and regression

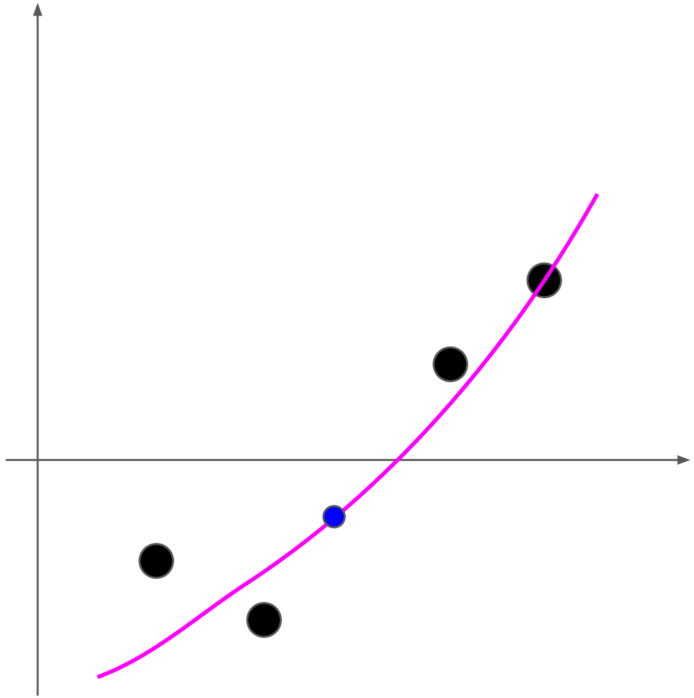
- We have looked at this for simple data sets you may want to describe the trends within the data.
- While we described these differently, you may hear them still described a a form of low-level machine learning, in that they are predicting...



Interpolation, approximation and regression

- We have looked at this for simple data sets you may want to describe the trends within the data.
- While we described these differently, you may hear them still described a a form of low-level machine learning, in that they are predicting...

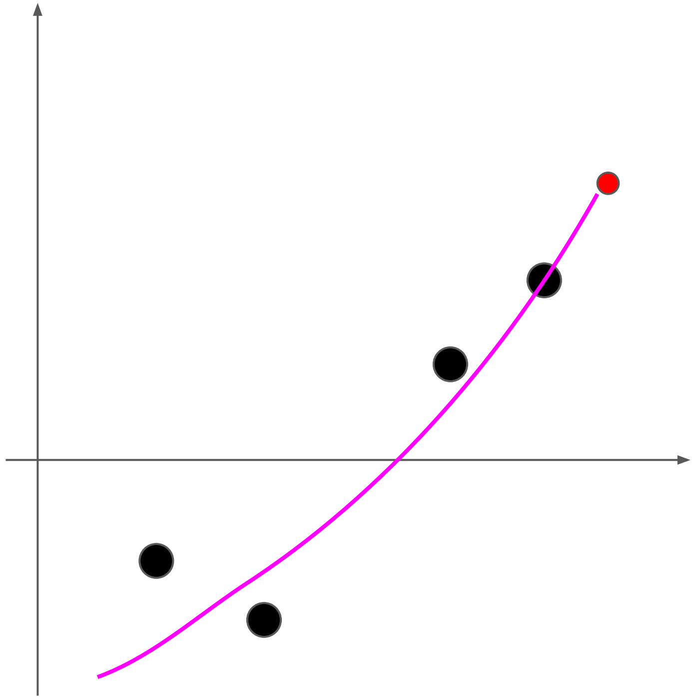
- Values in between known data (*interpolation*)



Interpolation, approximation and regression

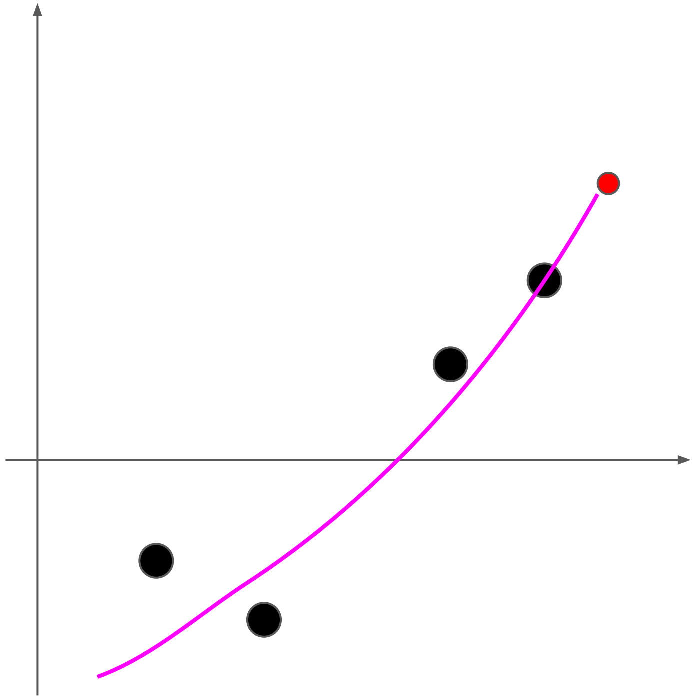
- We have looked at this for simple data sets you may want to describe the trends within the data.
- While we described these differently, you may hear them still described a a form of low-level machine learning, in that they are predicting...

- Values beyond known data (*extrapolation*)



Interpolation, approximation and regression

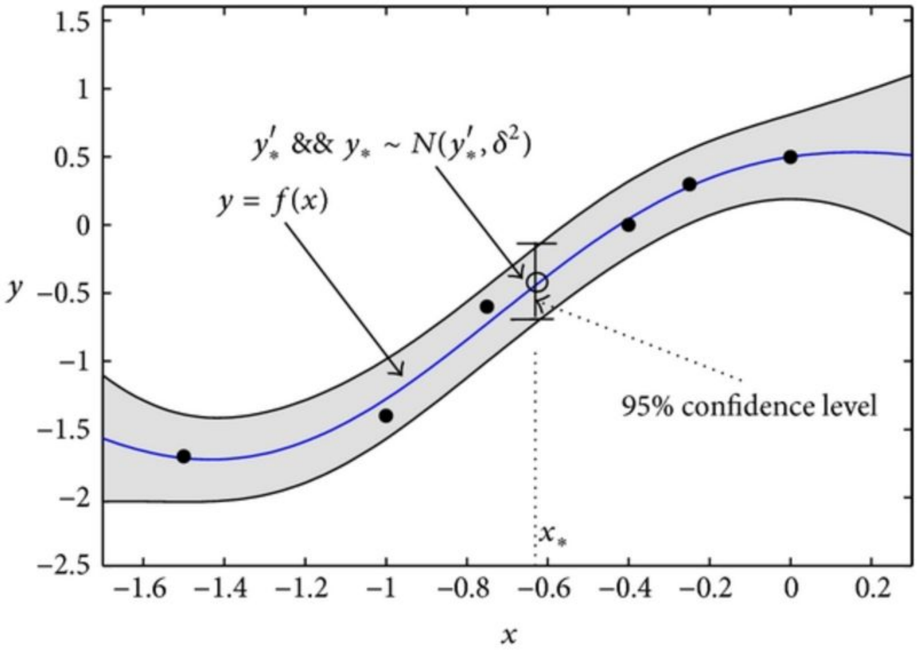
- In both cases you are using known data (*~ training set*) to define a regression or interpolating function (*~ network layers/neurons*) and then using that to infer patterns or predictions (*~ output layer*)



Interpolation, approximation and regression

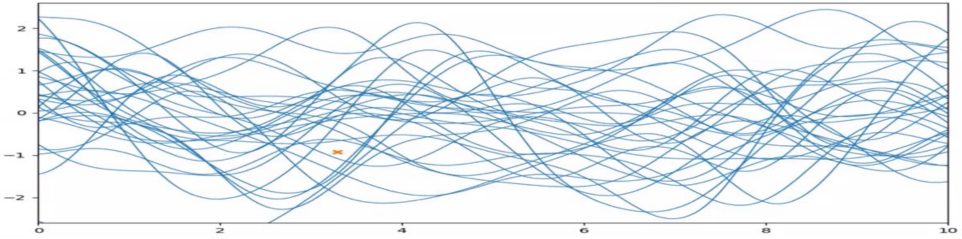
- In both cases you are using known data (*~ training set*) to define a regression or interpolating function (*~ network layers/neurons*) and then using that to infer patterns or predictions (*~ output layer*)

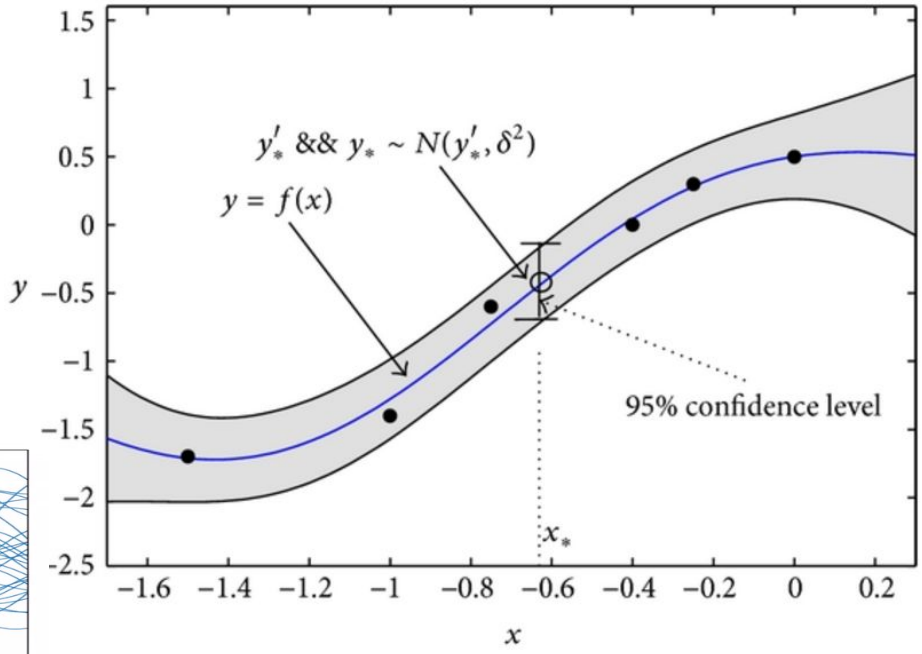
- A flexible version of this is Gaussian Process Regression



Interpolation, approximation and regression

- Your model assumes the data are being generated from a Gaussian process at every measured (and in-between) x-values.



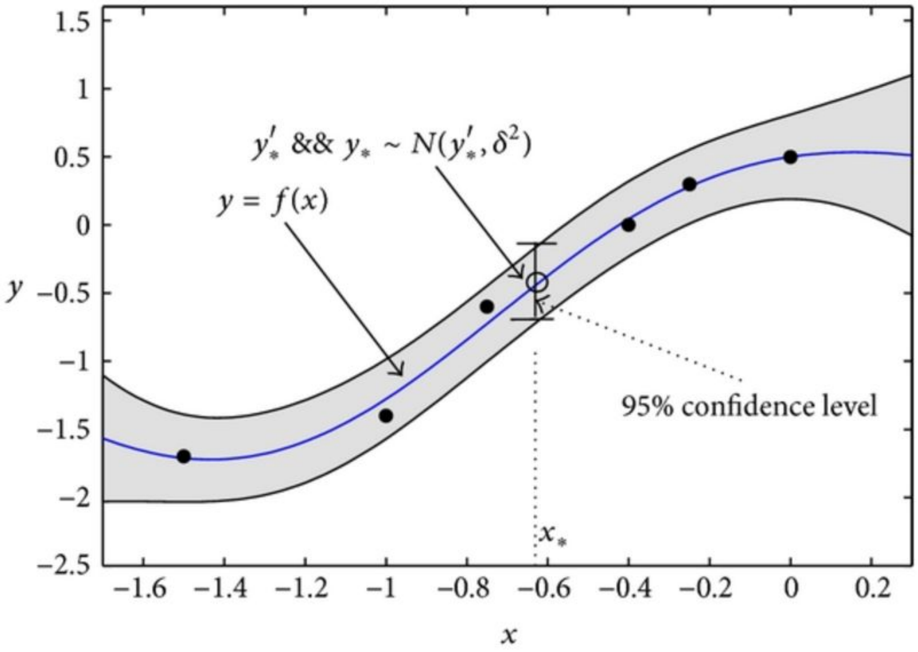


Interpolation, approximation and regression

- Your model assumes the data are being generated from a Gaussian process at every measured (and in-between) x-values.

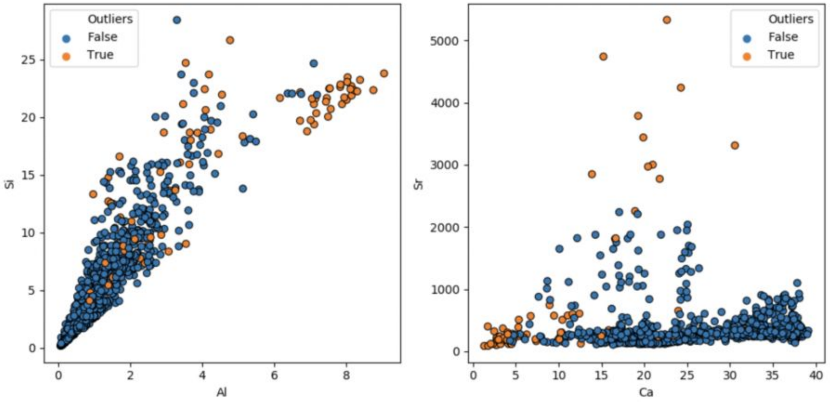
- Allows for non-parametric or general regression and prediction from data with errors.

http://www.tmpl.fi/gp/

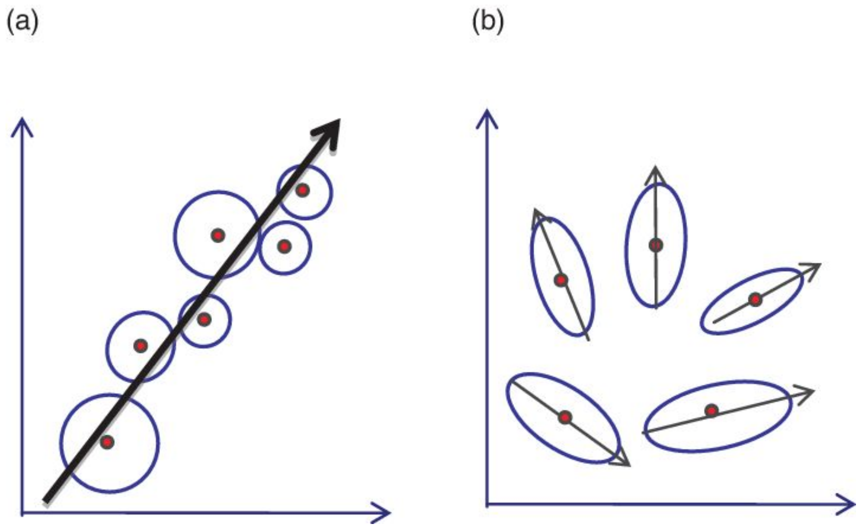


Dimensionality reduction

Running unsupervised machine learning (k-means clustering) to separate groups of data can be difficult if the variables are correlated.



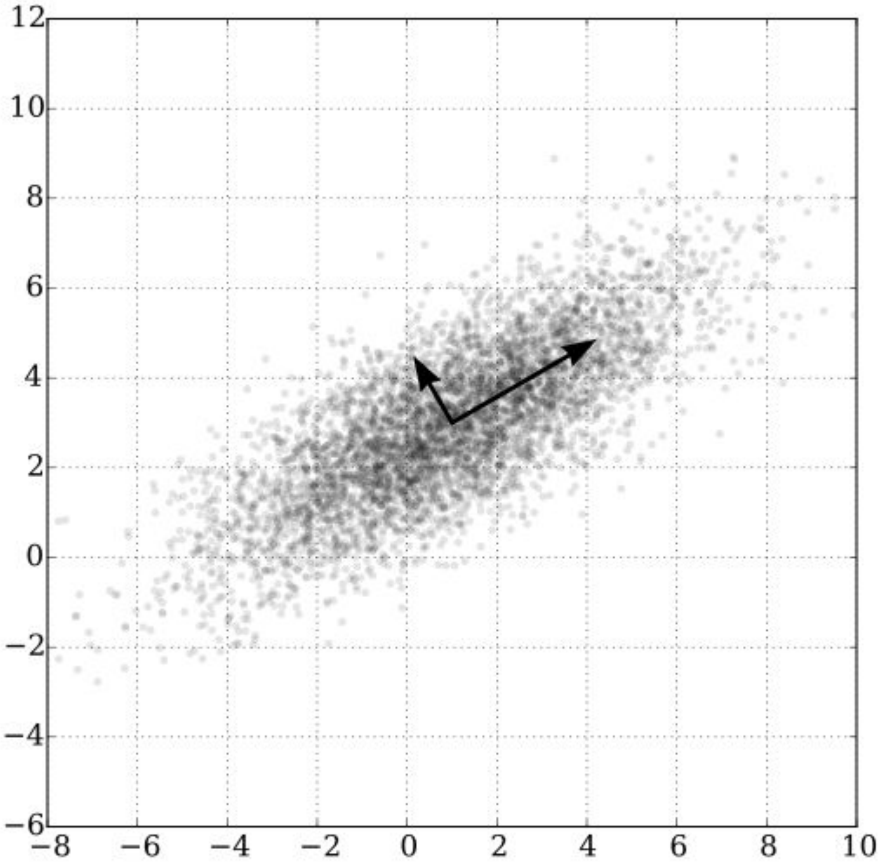
By transforming to a new basis set with PCA (Principal Component Analysis) / LDA (Linear Discriminant analysis) first, sometimes this type of cluster separation can be improved.



Dimensionality reduction

Principal component analysis (PCA)

Principal components analysis is a simple, non-parametric method for describing complex data sets.
It provides a roadmap for how to *reduce a complex data set to a lower dimension* to reveal the sometimes hidden, simplified dynamics that often underlie it. Principal component analysis does this by computing the most meaningful *basis* to re-express a noisy, garbled data set.



Dimensionality reduction

Principal component analysis (PCA)

For high dimensional data sets, PCA can help us figure out where most of the information is contained which describes the entire data - but with minimal loss of information.

To do this it tries to find a ‘few’ new variables to describe the data which don’t lose too much information

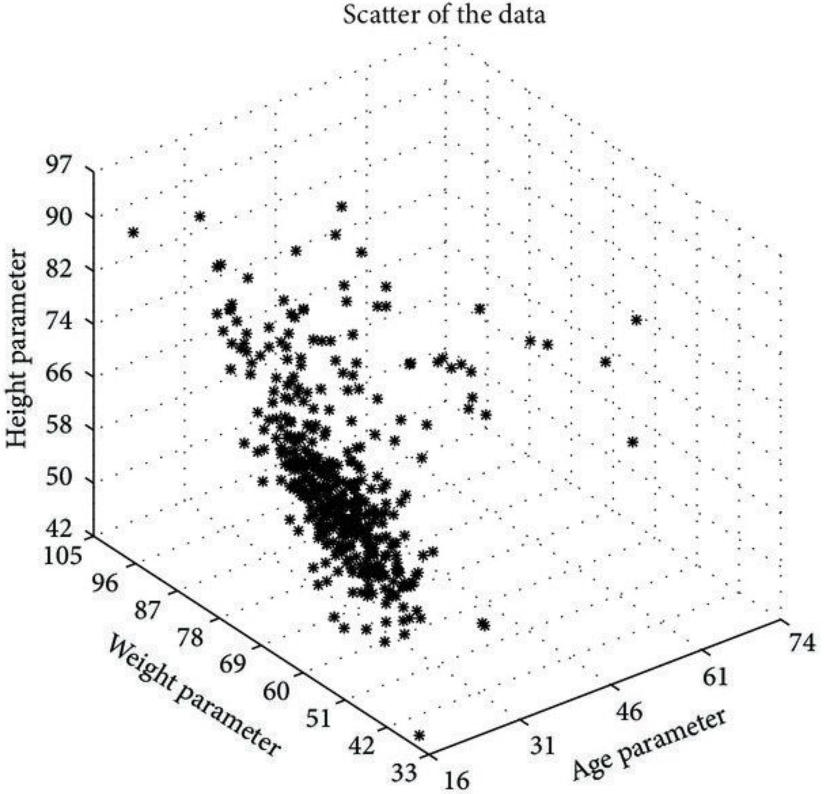
E.g., describing a 2D shadow instead of the full 3D object

Dimensionality reduction

Principal component analysis (PCA)

If we have a data set with say 3 variables (height, weight, age), these are not necessarily orthogonal like we think of X-Y-Z data points.

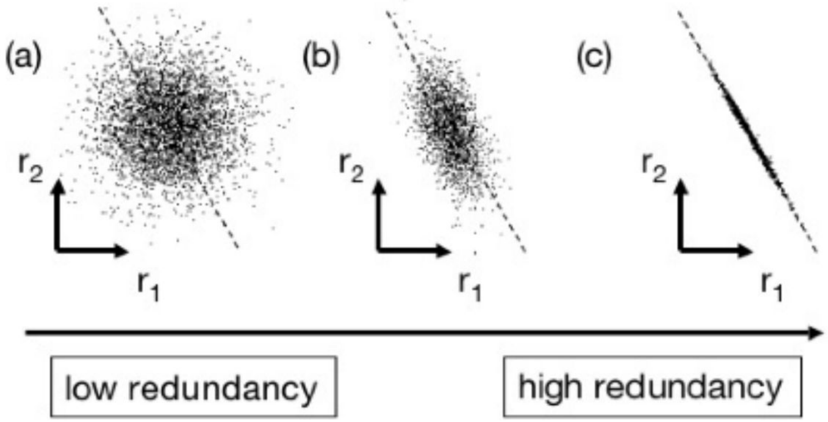
Height and weight may be tightly correlated, and at some point also mildly correlate with age

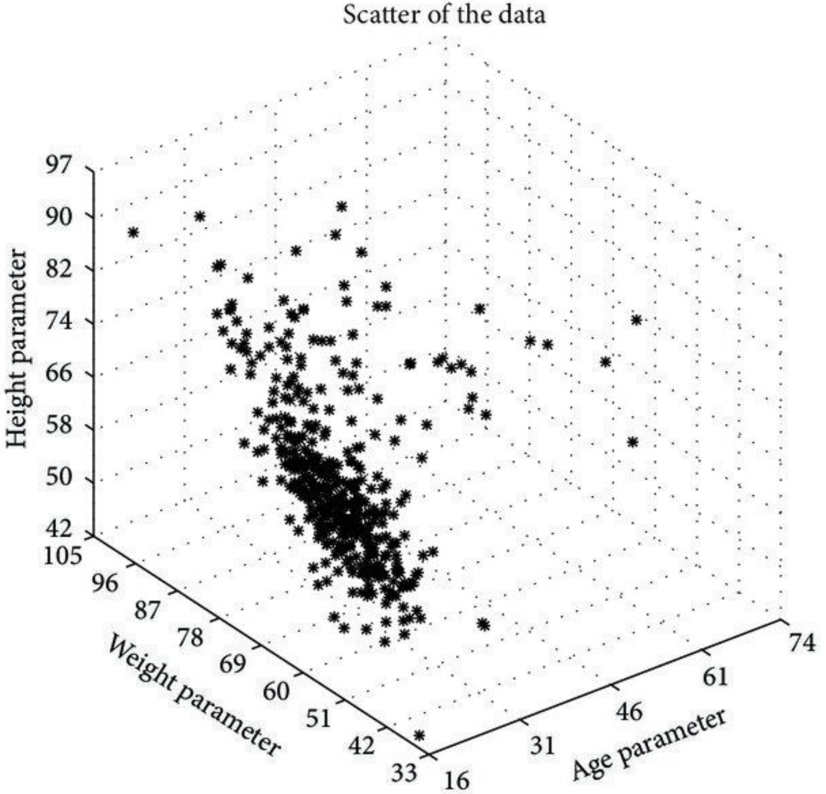


Dimensionality reduction

Principal component analysis (PCA)

When one or more of the variables are highly correlated, movement in that common direction isn't giving you much more information about the other correlated variable.

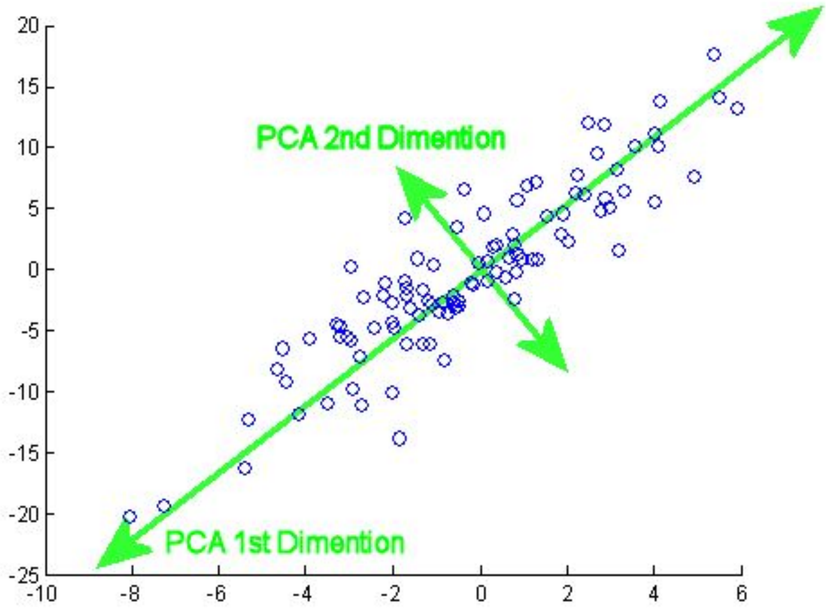




Dimensionality reduction

Principal component analysis (PCA)

So we want to find new ‘coordinates’ (which are going to be some linear combination of our original variables) that are going to be closely related to the amount of correlation between the original variables.

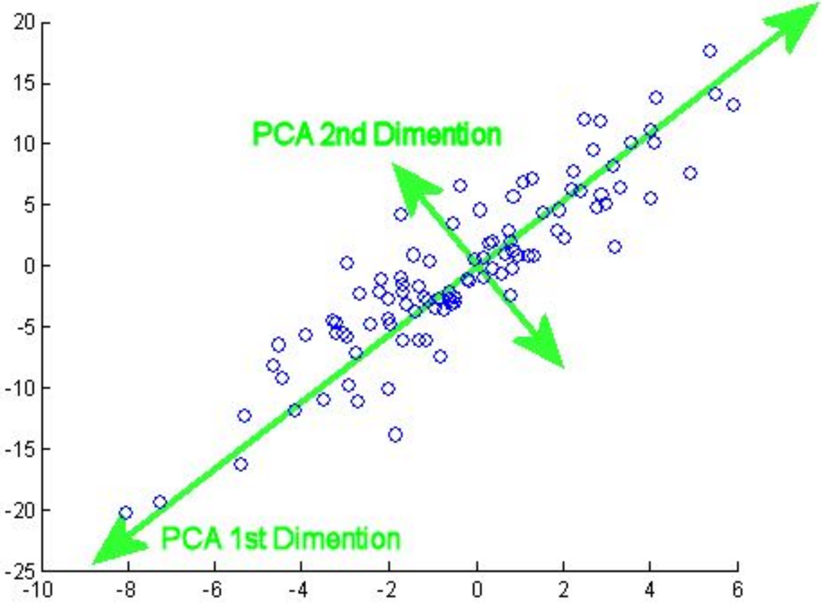


Dimensionality reduction

Principal component analysis (PCA)

We won't go into details here, but in practice we would take the following steps:

1) Standardize the data in our original variables
2) Compute the covariance matrix of the standardized variables
3) The eigenvalues and eigenvectors of the covariance matrix then give you an indication of 'directions' of the data in your original variable space where most of the information is contained



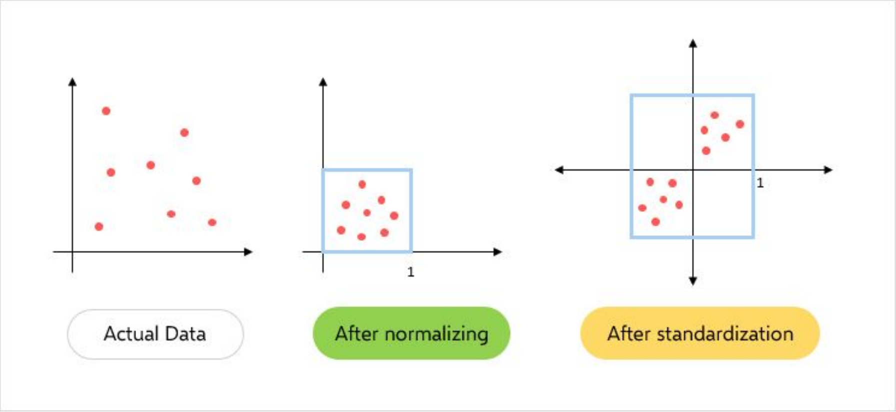
Dimensionality reduction

Principal component analysis (PCA)

We won't go into details here, but in practice we would take the following steps:

1) **Standardize the data in our original variables**
2) Compute the covariance matrix of the standardized variables
3) The eigenvalues and eigenvectors of the covariance matrix then give you an indication of 'directions' of the data in your original variable space where most of the information is contained

$$Z = \frac{VALUE - MEAN}{STANDARD\ DEVIATION}$$



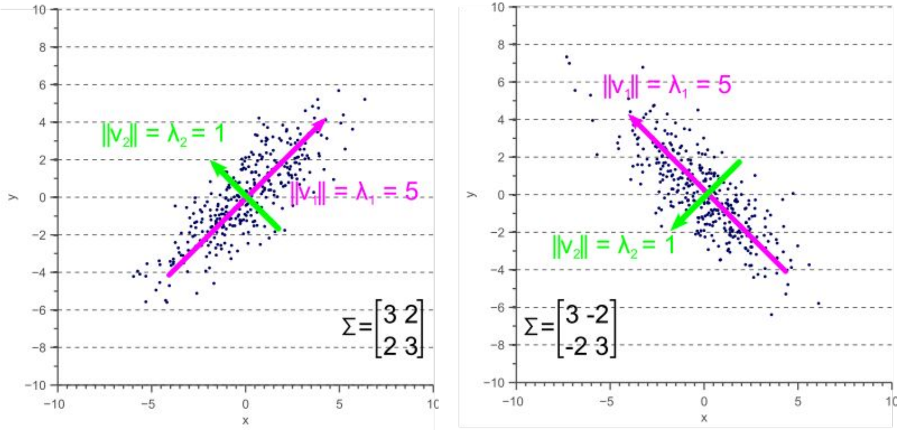
Dimensionality reduction

Principal component analysis (PCA)

We won't go into details here, but in practice we would take the following steps:

1) Standardize the data in our original variables
2) **Compute the covariance matrix of the standardized variables**
3) The eigenvalues and eigenvectors of the covariance matrix then give you an indication of 'directions' of the data in your original variable space where most of the information is contained

$$\text{Covariance Matrix} = \begin{bmatrix} COV\ (X,X) & COV\ (X,Y) \\ COV\ (Y,X) & COV\ (Y,Y) \end{bmatrix}$$



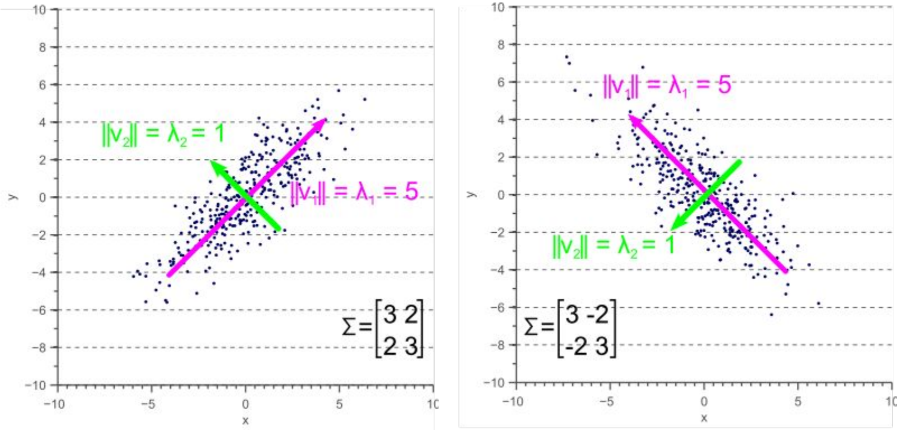
Dimensionality reduction

Principal component analysis (PCA)

We won't go into details here, but in practice we would take the following steps:

1) Standardize the data in our original variables
2) Compute the covariance matrix of the standardized variables
3) **The eigenvalues and eigenvectors of the covariance matrix then give you an indication of 'directions' of the data in your original variable space where most of the information is contained**

$$\text{Covariance Matrix} = \begin{bmatrix} COV\ (X,X) & COV\ (X,Y) \\ COV\ (Y,X) & COV\ (Y,Y) \end{bmatrix}$$

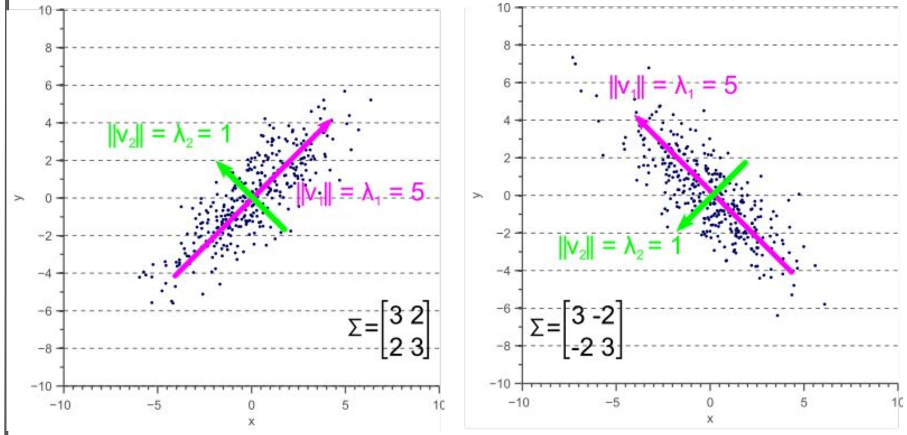


Dimensionality reduction

Principal component analysis (PCA)

The *covariance* of two random variables is a measure of the information they share. The covariances of random variables X_i, each one modelling a measurement type, can be estimated by their covariance matrix.

$$\text{Covariance Matrix} = \begin{bmatrix} COV\,(X,X) & COV\,(X,Y) \\ COV\,(Y,X) & COV\,(Y,Y) \end{bmatrix}$$



Dimensionality reduction

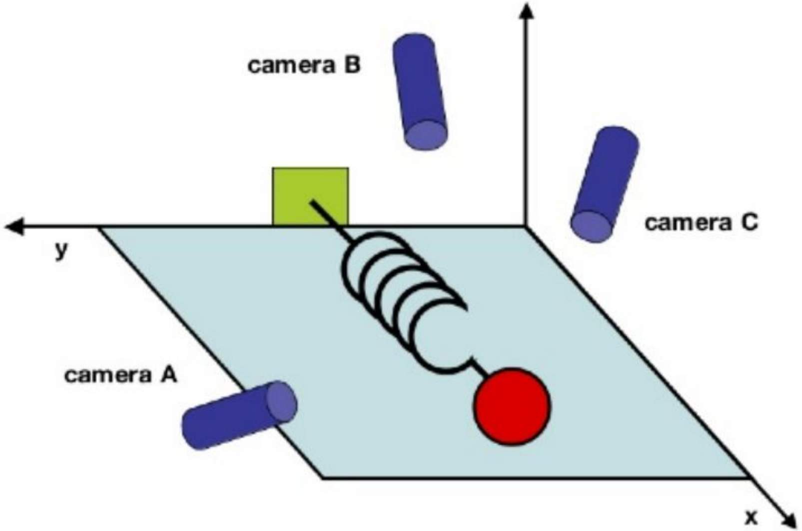
Example - PCA

Toy model: mass attached to an ideal (massless, frictionless) spring that oscillates along the *x* - axis about its equilibrium position at a given frequency.

Dimensionality reduction

Example - PCA

Toy model: mass attached to an ideal (massless, frictionless) spring that oscillates along the *x* - axis about its equilibrium position at a given frequency. Let's pretend that we do not know anything about the motion of this spring. We do not know which, let alone how many, axes and dimensions are important to measure, and therefore we decide to place three movie cameras, placed at random angles, each providing the two dimensional position of the mass at regular time intervals.



Dimensionality reduction

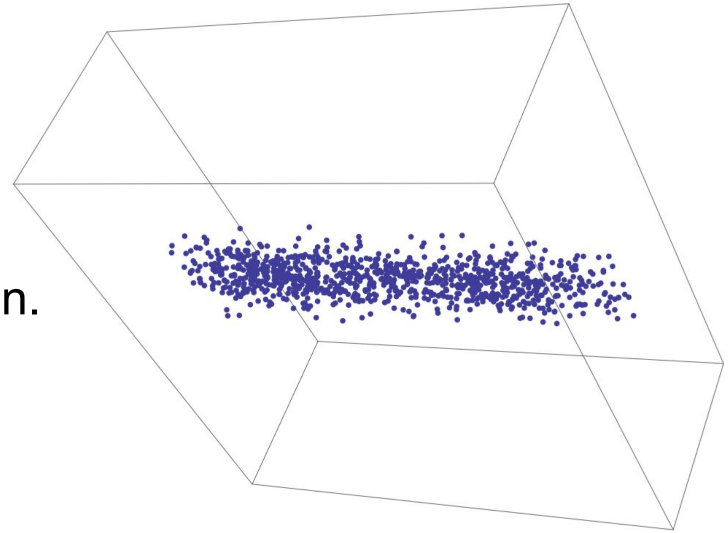
Example - PCA

The data we collect with the cameras are 6D vectors for each time step, each having the form: *$(x_A, y_A, x_B, y_B, x_C, y_C)$*, with each camera contributing a 2D projection of the mass position.

Dimensionality reduction

Example - PCA

The data we collect with the cameras are 6D vectors for each time step, each having the form: $(x_A, y_A, x_B, y_B, x_C, y_C)$, with each camera contributing a 2D projection of the mass position.

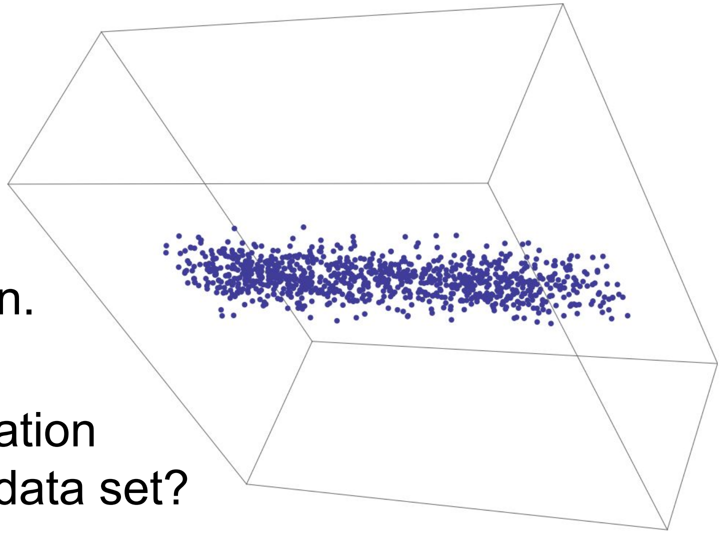


Dimensionality reduction

Example - PCA

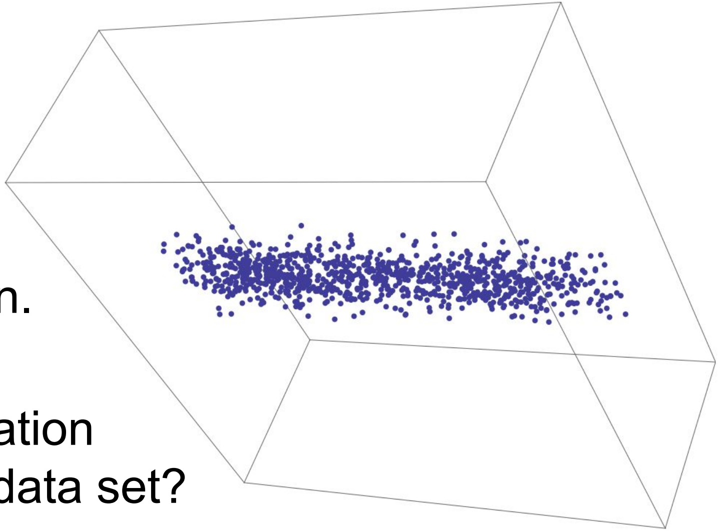
The data we collect with the cameras are 6D vectors for each time step, each having the form: $(x_A, y_A, x_B, y_B, x_C, y_C)$, with each camera contributing a 2D projection of the mass position.

PCA is concerned with the following question:
is there another basis, which is a linear combination of the original basis, that "best" represents our data set?



Dimensionality reduction

Example - PCA



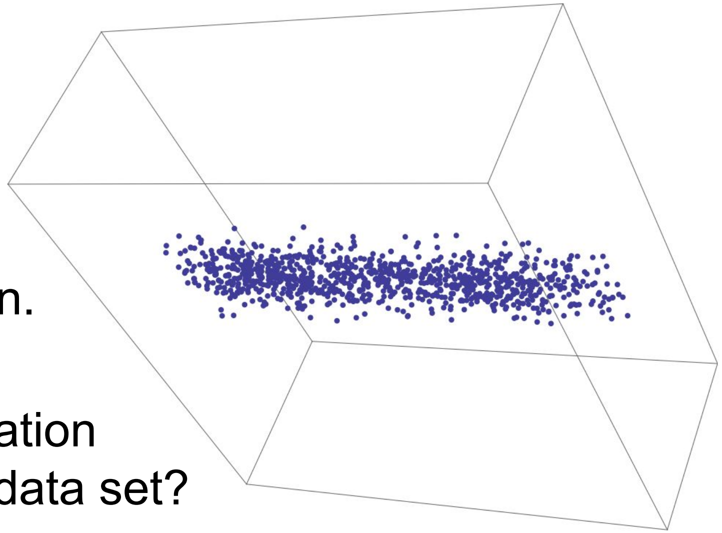
The data we collect with the cameras are 6D vectors for each time step, each having the form: $(x_{A}, y_{A}, x_{B}, y_{B}, x_{C}, y_{C})$, with each camera contributing a 2D projection of the mass position.

PCA is concerned with the following question:
is there another basis, which is a linear combination of the original basis, that "best" represents our data set?

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be $m \times n$ matrices related by a linear transformation $\boldsymbol{P}$.
$\boldsymbol{X}$ is the original recorded data set and $\boldsymbol{Y}$ is the re-representation of the data set: $\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{X}$. $\boldsymbol{P}$ is a $m \times m$ matrix having as rows a set of new basis vector.

Dimensionality reduction

Example - PCA



The data we collect with the cameras are 6D vectors for each time step, each having the form: $(x_A, y_A, x_B, y_B, x_C, y_C)$, with each camera contributing a 2D projection of the mass position.

PCA is concerned with the following question:
is there another basis, which is a linear combination of the original basis, that "best" represents our data set?

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be $m \times n$ matrices related by a linear transformation $\boldsymbol{P}$.
$\boldsymbol{X}$ is the original recorded data set and $\boldsymbol{Y}$ is the re-representation of the data set: $\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{X}$. $\boldsymbol{P}$ is a $m \times m$ matrix having as rows a set of new basis vector.

The original question can be re-expressed as follows: what is the best way to re-express the dataset $\boldsymbol{X}$? Or, equivalently: what is the best choice of $\boldsymbol{P}$?

Dimensionality reduction

Principal component analysis (PCA)

You need to look for redundancy

Consider two arbitrary measurement types r_1 and r_2.

We can have different situations:

Dimensionality reduction

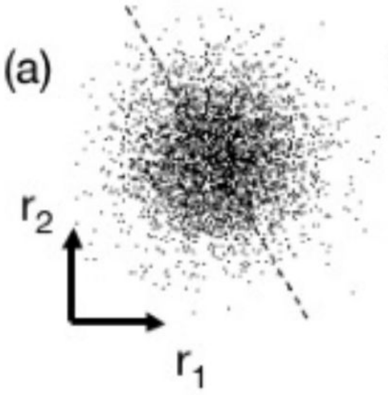
Principal component analysis (PCA)

You need to look for redundancy

Consider two arbitrary measurement types r_1 and r_2.

We can have different situations:

(a) - knowledge of one of the two measurements does not tell us almost anything about the other;



Dimensionality reduction

Principal component analysis (PCA)

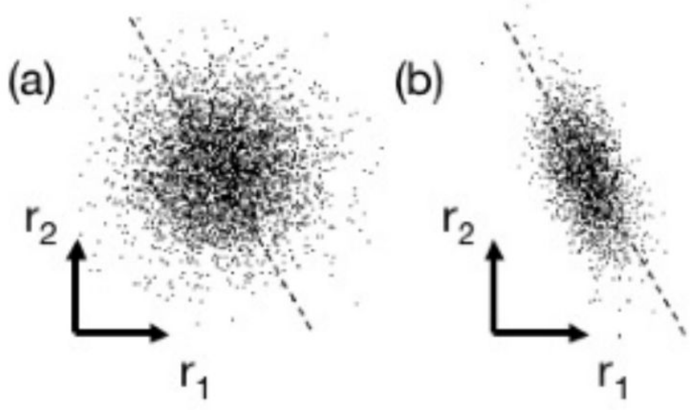
You need to look for redundancy

Consider two arbitrary measurement types r_1 and r_2.

We can have different situations:

(a) - knowledge of one of the two measurements does not tell us almost anything about the other;

(b) - the existence of some correlation between r_1 and r_2 is more evident;



Dimensionality reduction

Principal component analysis (PCA)

You need to look for redundancy

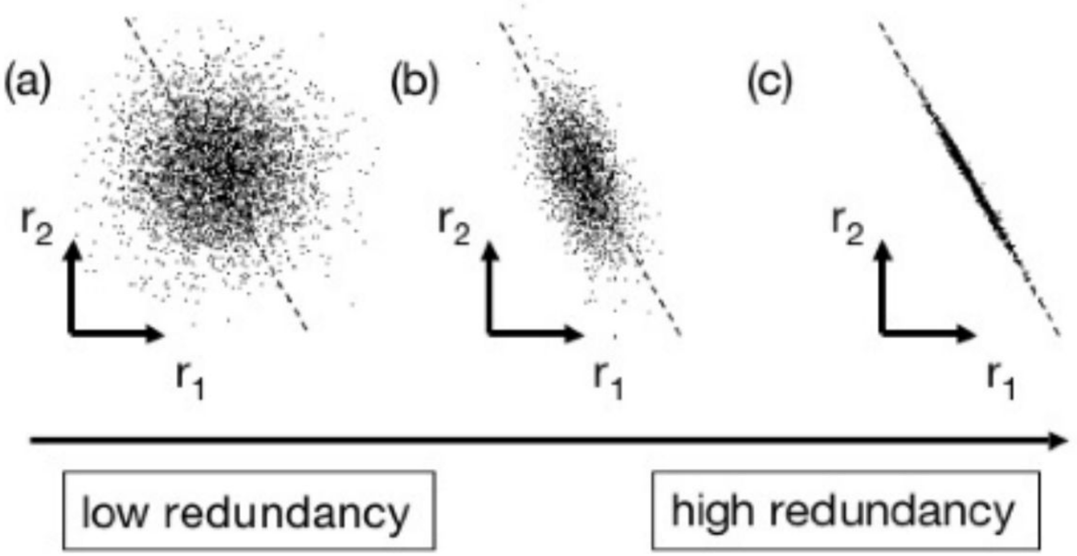
Consider two arbitrary measurement types r_1 and r_2.

We can have different situations:

(a) - knowledge of one of the two measurements does not tell us almost anything about the other;

(b) - the existence of some correlation between r_1 and r_2 is more evident;

(c) - it is clear that there is some kind of relation between the two measurements types.



Dimensionality reduction

Principal component analysis (PCA)

For high dimensional data you can focus on representations of the data using smaller numbers of these new principal components which describe most of the variation in the data. This can reduce the dimensionality of your data set for some applications.

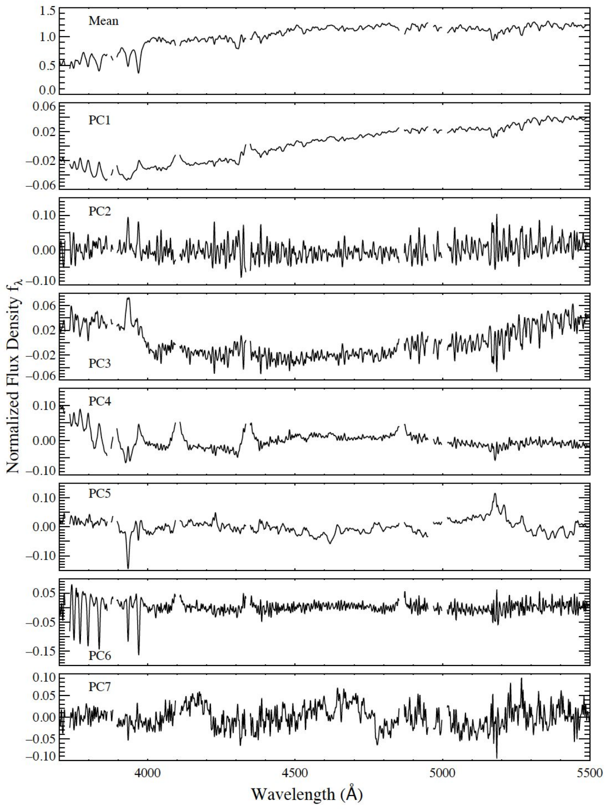


Figure 2: From top to bottom: the mean spectrum of the model library followed by the first to the seventh eigenspectra.

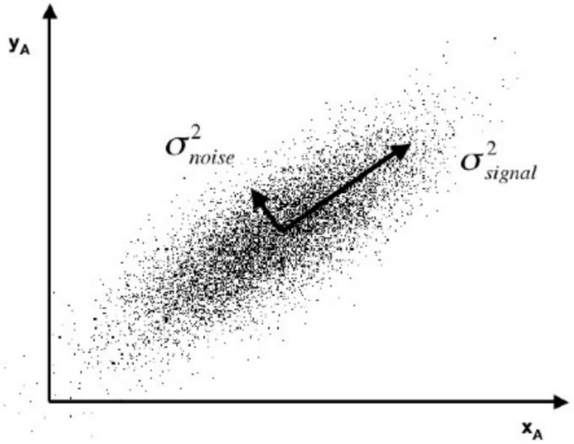
Dimensionality reduction

CAREFUL!

Dimensionality reduction

CAREFUL!

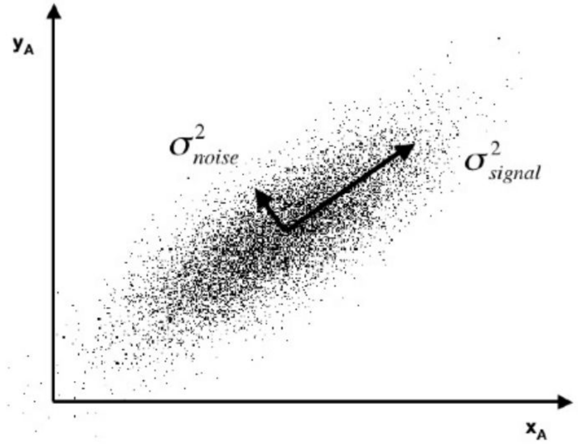
Noise - PCA is not reliable if data are "too noisy". In fact, if this is the case it will be very hard to tell "informative" directions apart. There is no absolute scale for noise: a common measure is the signal-to-noise ratio (SNR).



Dimensionality reduction

CAREFUL!

Noise - PCA is not reliable if data are "too noisy". In fact, if this is the case it will be very hard to tell "informative" directions apart. There is no absolute scale for noise: a common measure is the signal-to-noise ratio (SNR).

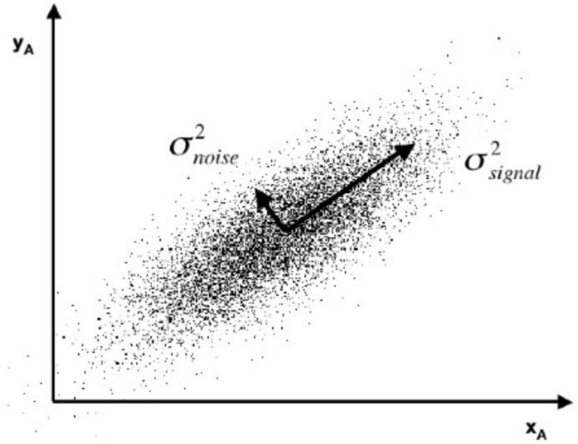


Gaussianity - We implicitly assumed that mean and covariance matrix of the measurements provided a complete account of the data distribution. BUT this works only for Gaussian distributions!

Dimensionality reduction

CAREFUL!

Noise - PCA is not reliable if data are "too noisy". In fact, if this is the case it will be very hard to tell "informative" directions apart. There is no absolute scale for noise: a common measure is the signal-to-noise ratio (SNR).



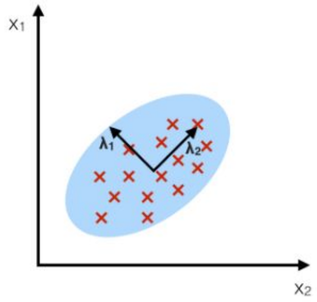
Gaussianity - We implicitly assumed that mean and covariance matrix of the measurements provided a complete account of the data distribution. BUT this works only for Gaussian distributions!

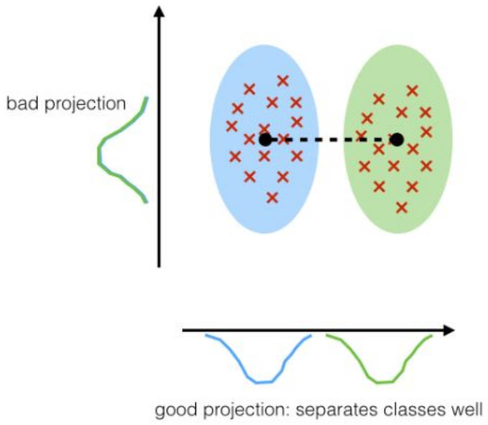
Linearity - This assumption frames the problem as a mere change of basis. The idea of Pearson (who invented PCA in 1901) was to find the line that "best fits the data". So "linear" ➞ linear regression: there is some relation between some independent variables X_i and the response variable Y, and this relation is linear in the parameters defining it.

Dimensionality reduction

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications.
The goal is to project a dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting and also reduce computational costs.



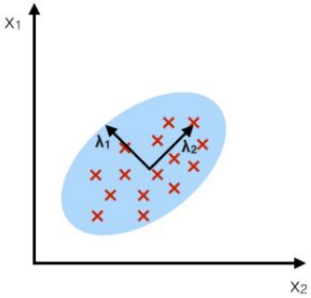


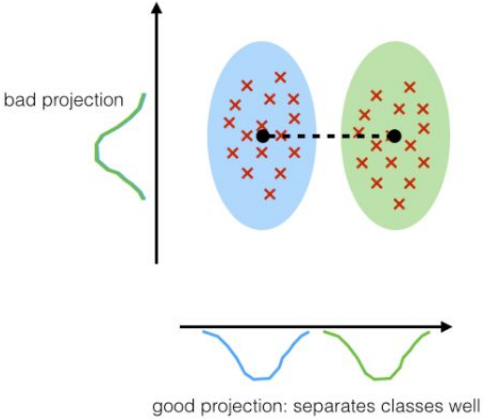
Dimensionality reduction

Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is most commonly used as dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications.
The goal is to project a dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting and also reduce computational costs.

PCA:
component axes that maximize covariance





LDA:
maximizing the component axes for class-separation

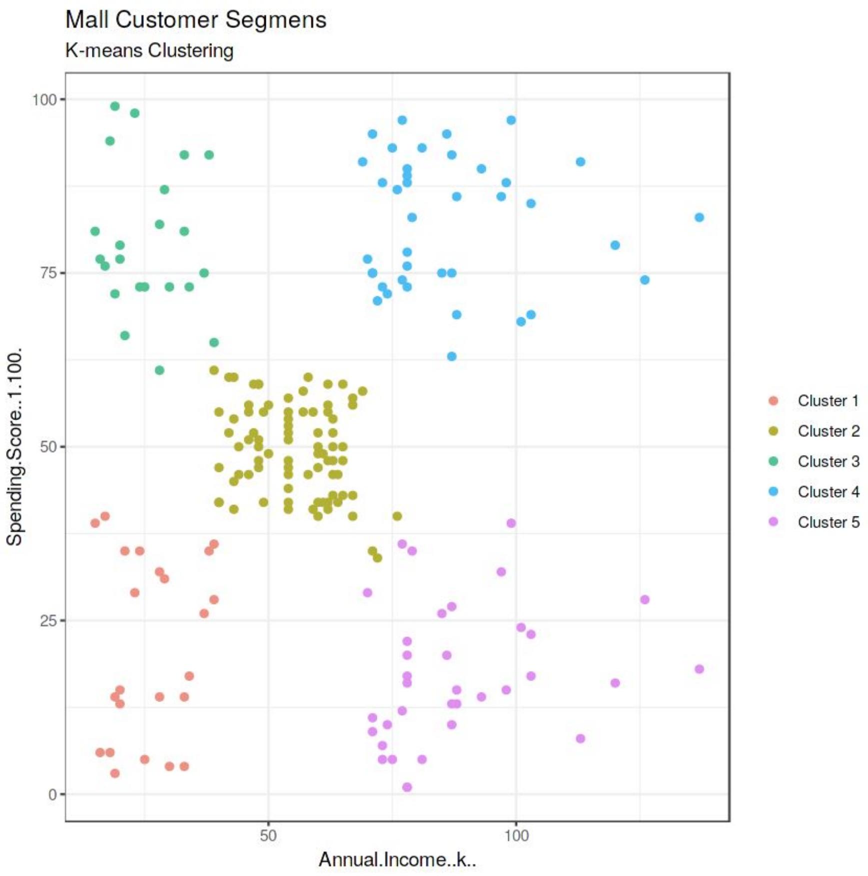
Machine Learning: Unsupervised

The most common task:

- Unsupervised learning for **clustering/exploration of data.**

  Here you may or may not have classes or attributes, but want to find out the structure, groupings or patterns within data without needing an end-to-end model

  E.g., patterns within consumer shopping choices → predict ads for what they will likely buy from a list of new products



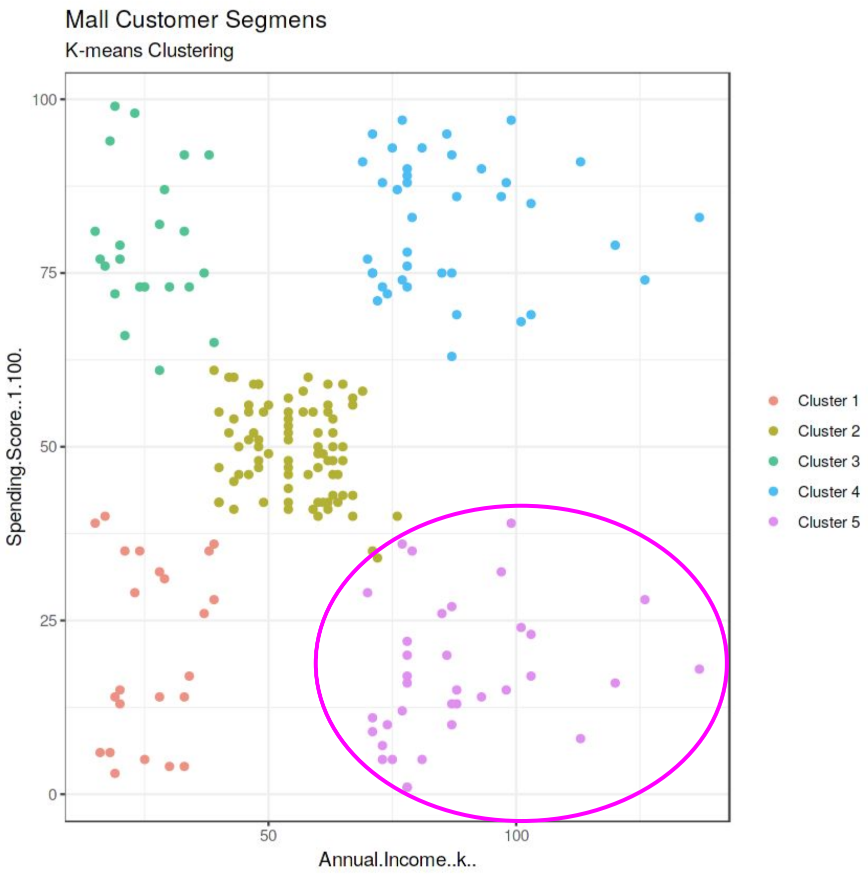
Machine Learning: Unsupervised

The most common task:

- Unsupervised learning for **clustering/exploration of data.**

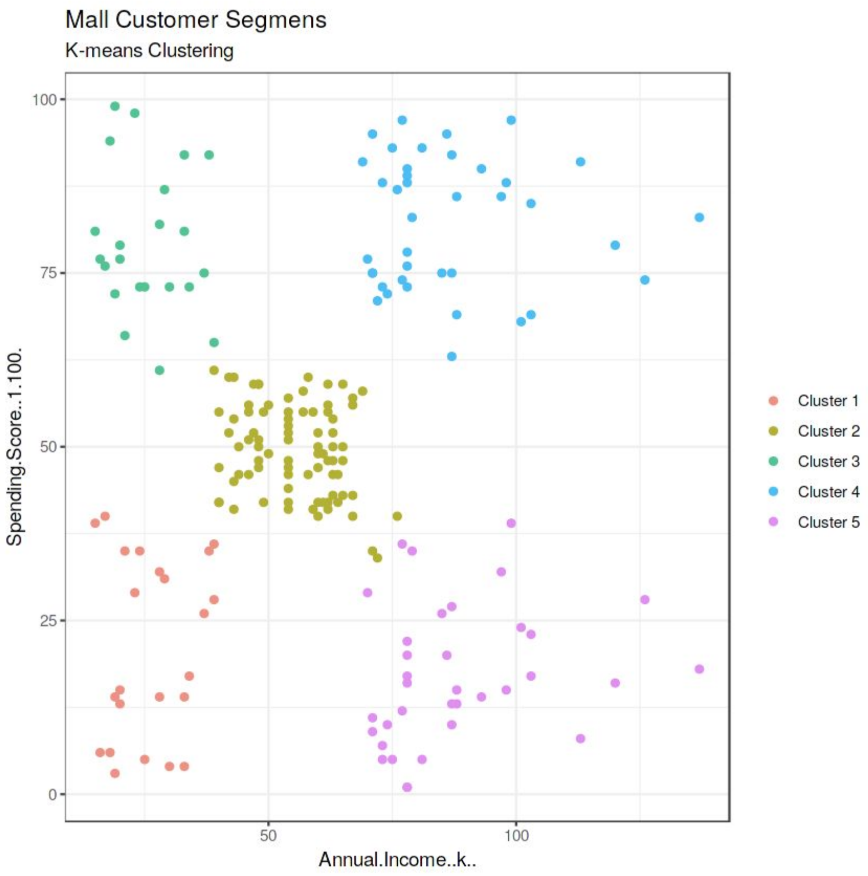
  Here you may or may not have classes or attributes, but want to find out the structure, groupings or patterns within data without needing an end-to-end model

  E.g., patterns within consumer shopping choices → predict ads for what they will likely buy from a list of new products



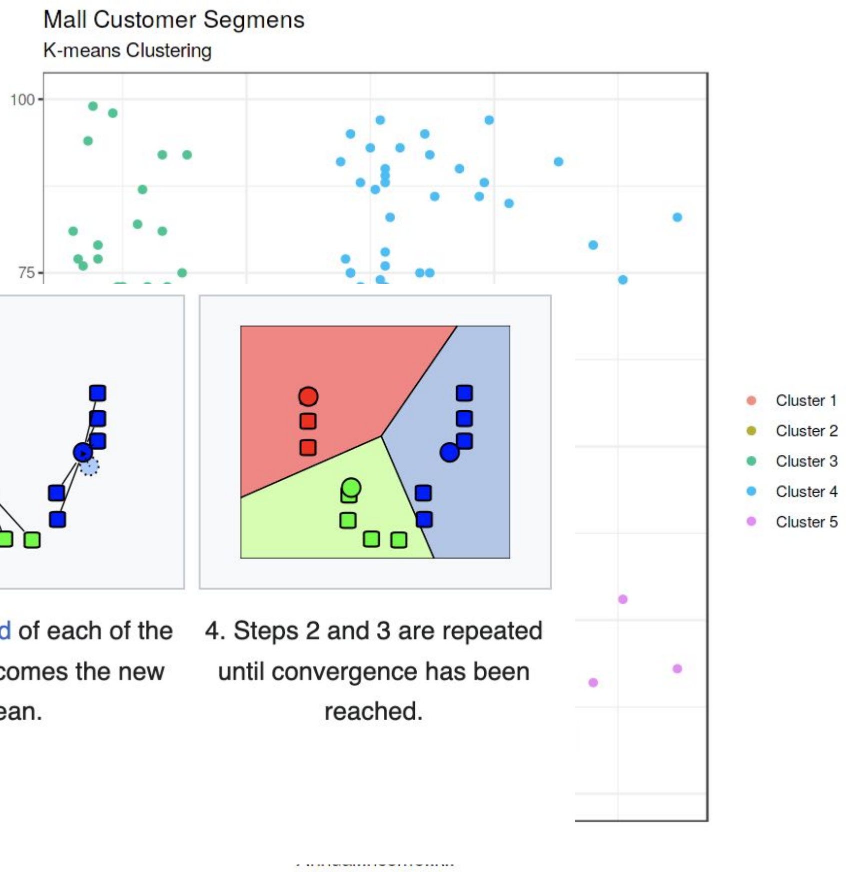
Machine Learning: Unsupervised

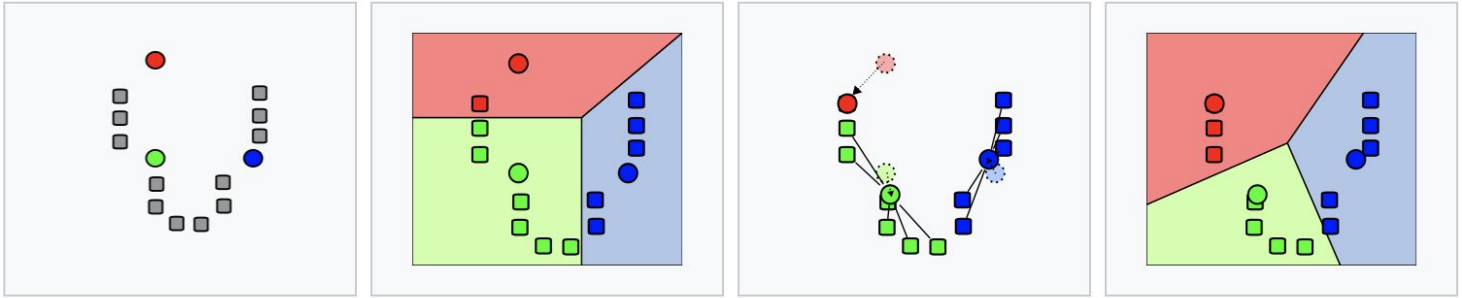
- A common tool for clustering is ***k-means clustering***.
- This can be implemented in most languages (python, IDL, R) easily and allows for multi-dimensional clustering analysis
- Has mathematical link to Principal Component Analysis and also can be thought of as a useful tool for dimensionality reduction



Machine Learning: Unsupervised

- A common tool for clustering is ***k-means clustering***.





1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

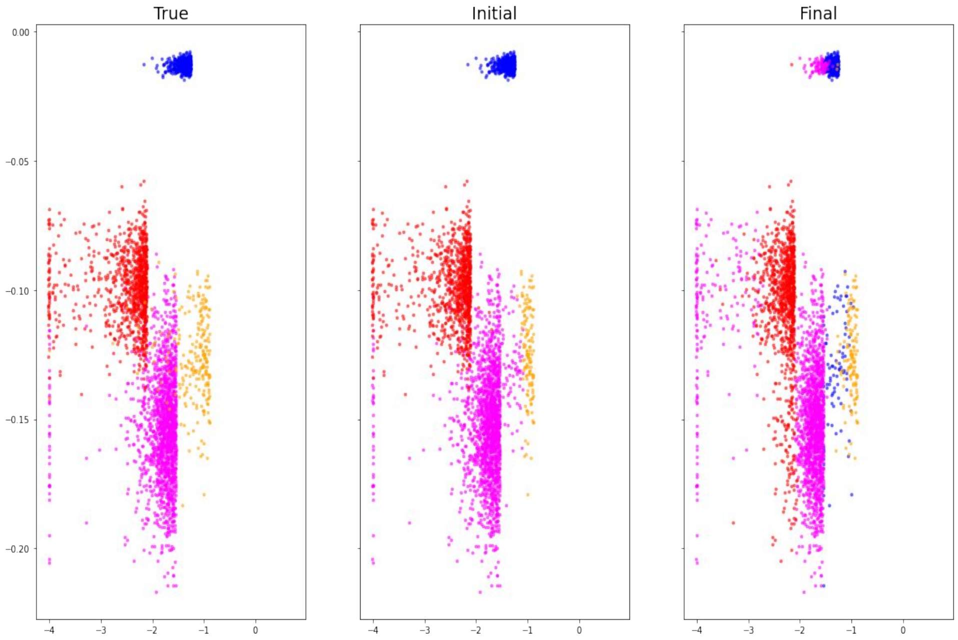
2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

Machine Learning: Unsupervised

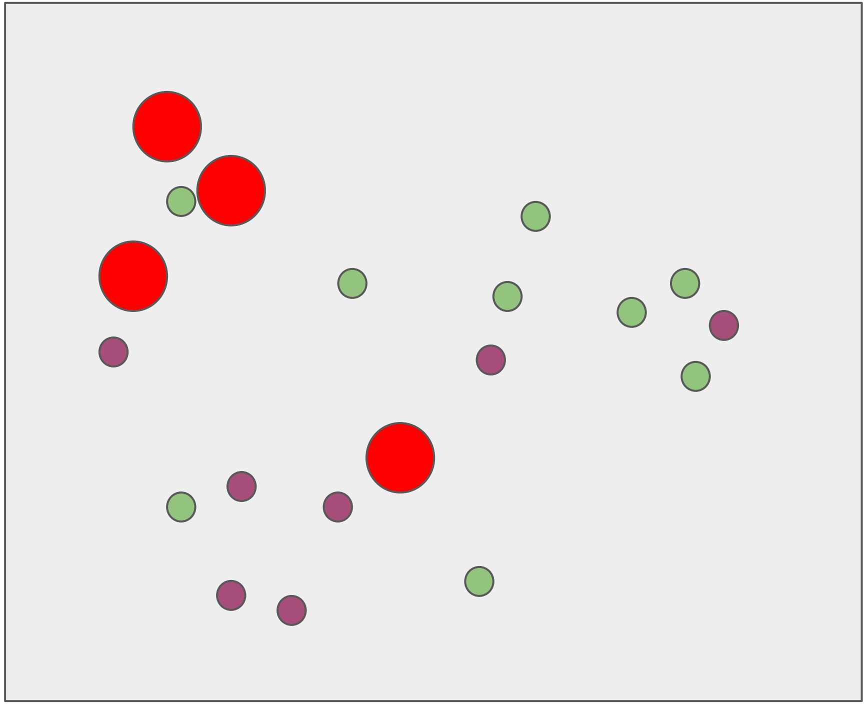
- A common tool for clustering is ***k-means clustering*.**
-
- *Note:* This algorithm needs be done in a highly iterative (changing number of clusters, stochastic perturbations) way and is not guaranteed to converge to a global optimum separation



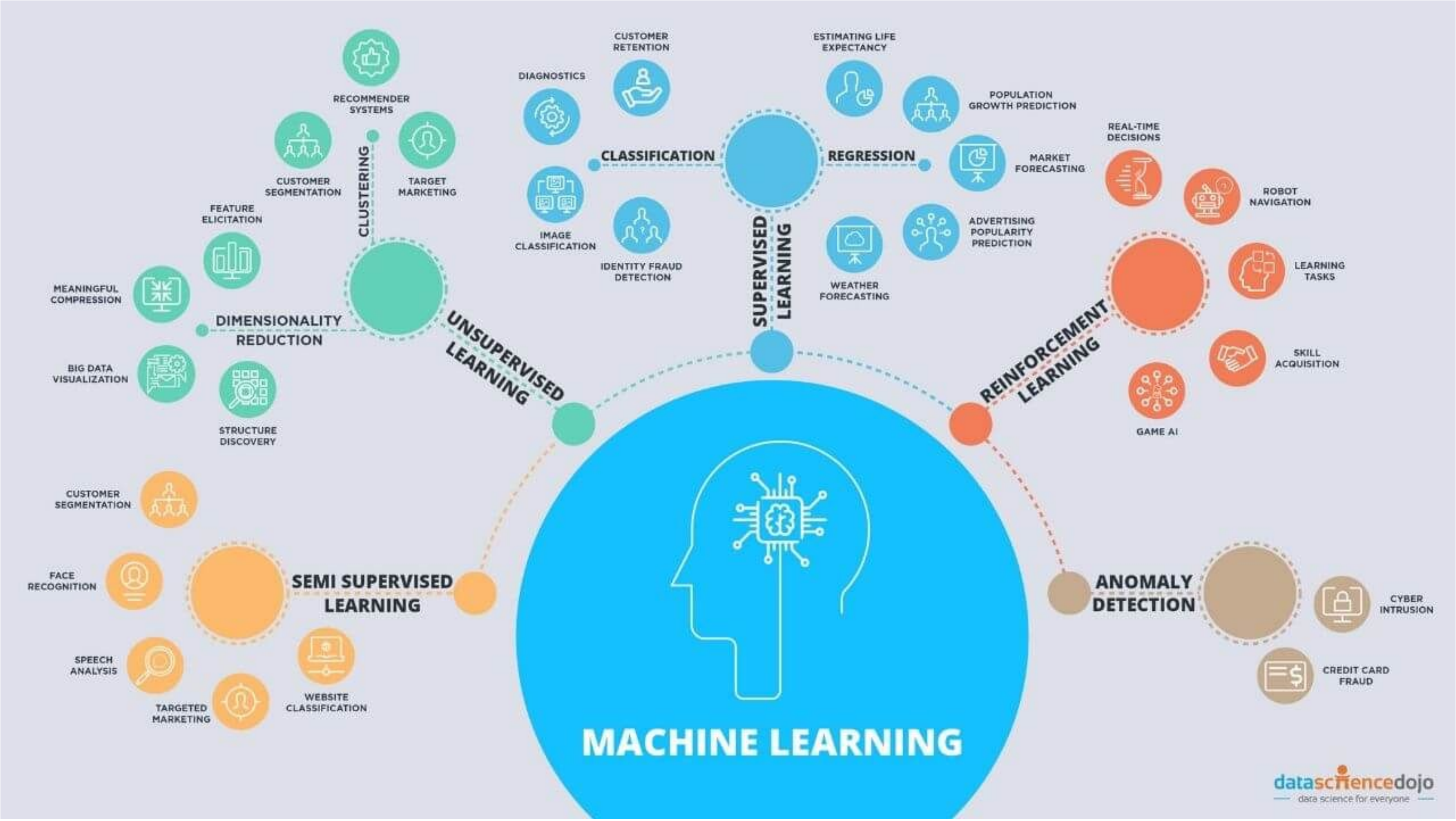
Machine Learning: Supervised

- Supervised learning for **classification.**

  Here you have ***known true*** information about attributes of a subset of objects and want to use that to classify/extract them from larger data sets automatically by training a set of functions in a network to map attributes to classes.
  E.g., (round, red, large) → Apples
  (round, purple, green, small) → Grapes



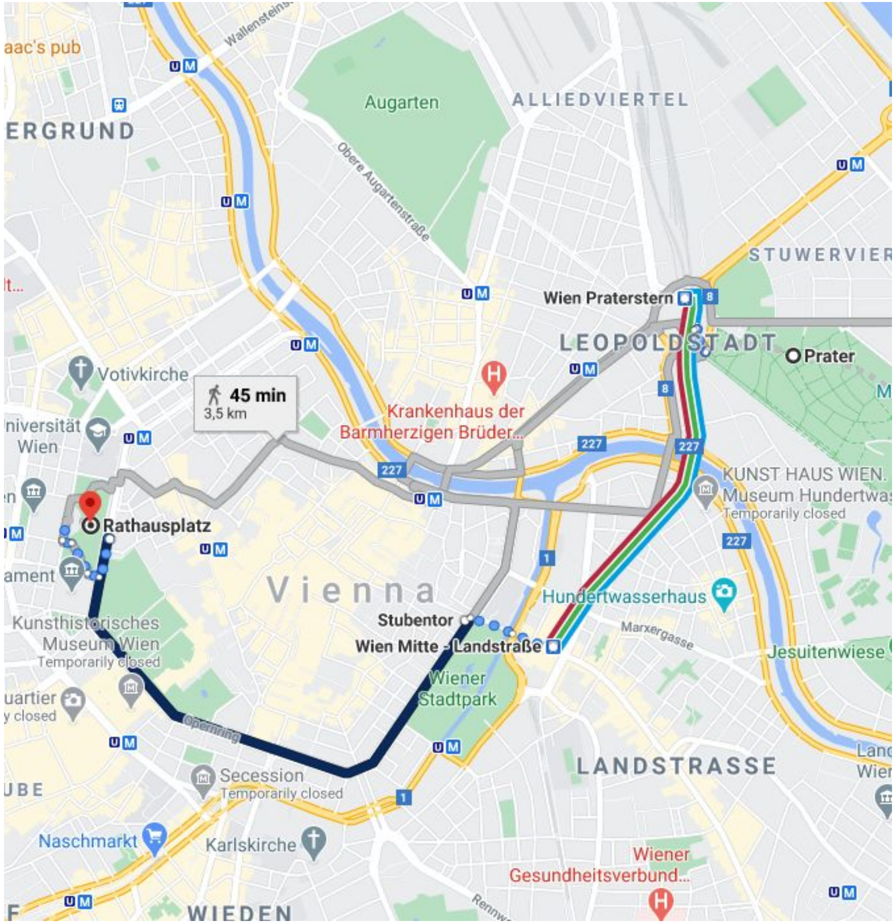
Machine Learning: real world examples



Machine Learning: real world examples

Traffic Alerts

When you look for a route on Google Maps, the result you get is obtained by taking into account how many people are currently using Google Maps, historic data of that route, and some real-time techniques.
The app uses your position and travelling speed, considers day and time (is this a specific occasion?) and gives you exact information by using artificial intelligence and machine learning algorithms.



Machine Learning: real world examples

Image recognition

Using *pattern recognition* and machine learning algorithms computers can recognize any form of visual.
Example: the camera in your phone recognizes 80 nodal points on a human face + machine learning technologies recognize the face
→ you can unlock your phone just by looking at it!

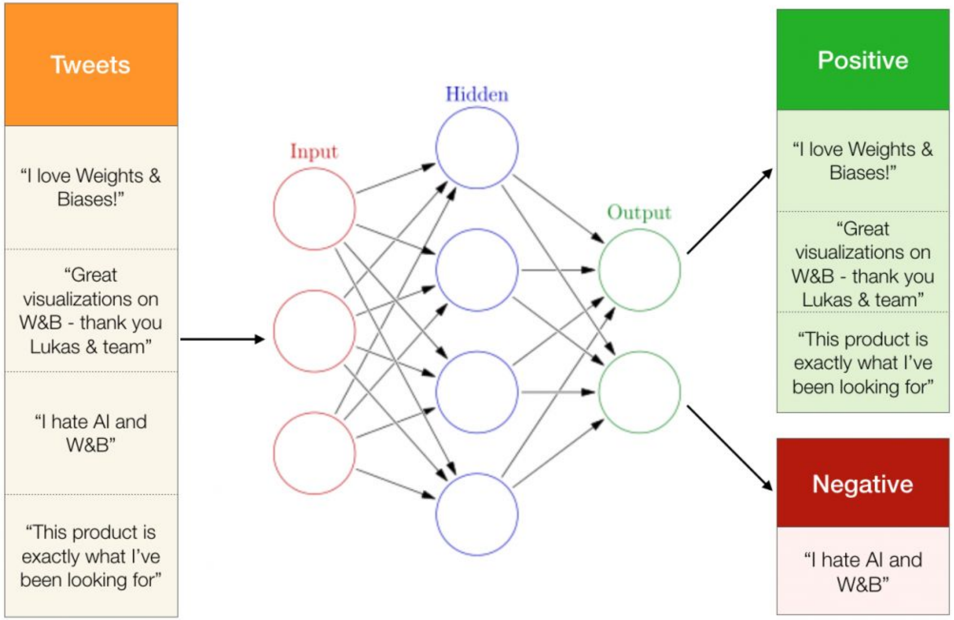
Video surveillance

Used to prevent theft, to detect abnormal events, to monitor operations and traffic, to protect facilities, to study shopping patterns...

Machine Learning: real world examples

Sentiment analysis

Machine learning application, used to *classify* sentiment, to mine opinions and to analyse emotions. This is done by using *words*: algorithms are able to identify positive and negative words, and to assess their magnitude. *Supervised and unsupervised machine learning algorithms + Natural Language Processing* (= subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language)



→ Used by companies to monitor clients reactions and deal with negative feedback

Machine Learning: real world examples

Google translate

It uses Google Neural Machine Translation algorithm.
Neural machine translation (NMT) is an approach to machine translation that uses a large language model to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model.
LMM = CNN + RNN + ANN +...

Virtual Professional Assistants

The accuracy, speed, and contextual abilities of Alexa, Google Assistant, and Siri are all because of machine learning algorithms and servers owned by their developing companies. They all work in a similar manner, the only difference being in their protocols and data privacy details. When you make a request, it is immediately sent to the relative server for a response. There, the words and tone of your request are analyzed by a set of algorithms, which are then matched with the command that they think you asked.

Machine Learning: real world examples

Product recommendation

The suggestions you get are the outcomes of advanced machine learning training where the system learns individual *patterns* of users and suggests additional products to buy.

Online video streaming applications

The apps capture the data of user's activities and provide suggestions. They monitor several types of data: day and time of watching content, type of content you prefer to watch, whether you pause/rewind/fast forward, browsing pattern, trailers you watch before actually watching the movie/show...

Social media

What kind of content users prefer to read/watch depending upon their age, gender, and location?

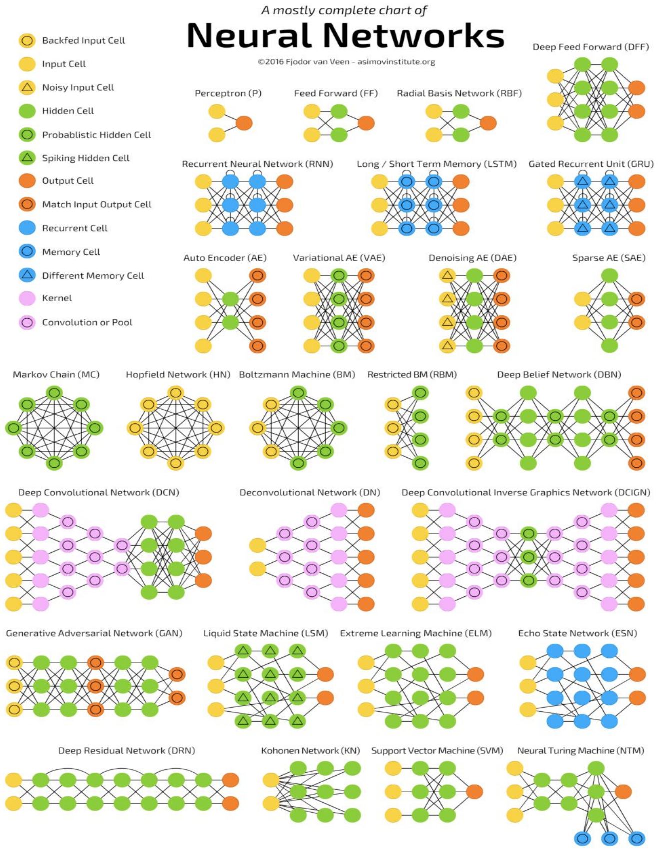
Neural Networks

There are many classes of AI/machine learning algorithms, and even within one class, there are new algorithms coming out every month as this explosion of algorithms fuels research and industry exploration of large data sets.

Neural Networks

- Let's look at one class of machine learning algorithms - **neural networks**

- These are classes of 'deep learning' algorithms which generally fall under the umbrella of supervised machine learning and artificial intelligence and have some common architectures and concepts.

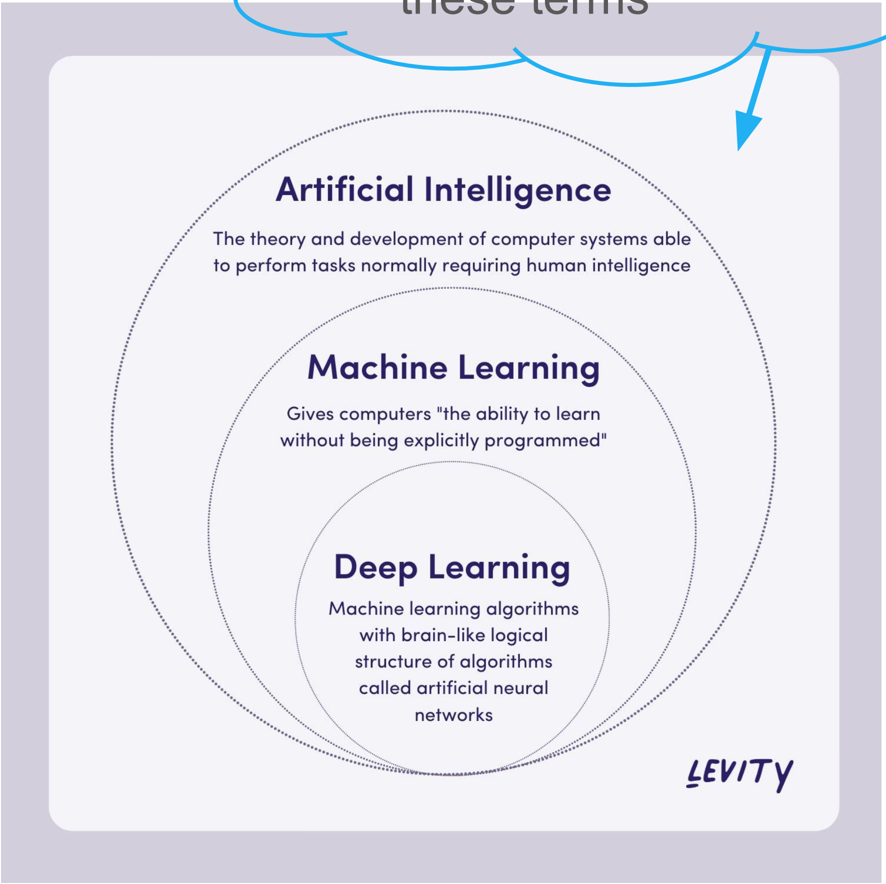


Neural Networks

Take with a grain of salt… different people interchange these terms

- Let's look at one class of machine learning algorithms - **neural networks**

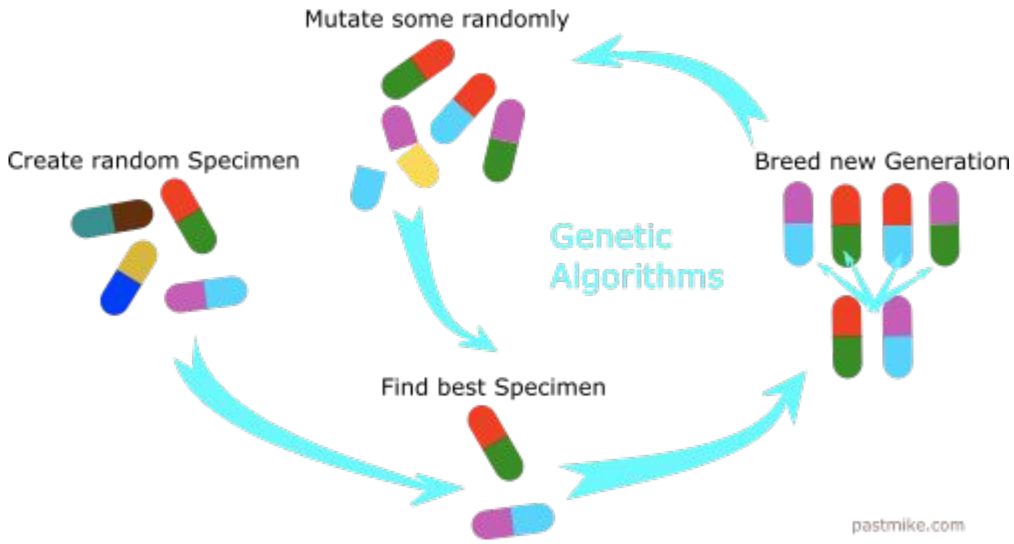
- These are classes of 'deep learning' algorithms which generally fall under the umbrella of supervised machine learning and artificial intelligence and have some common architectures and concepts.



Neural Networks

- These are classes of algorithms which generally fall under the umbrella of supervised machine learning or artificial intelligence and have some common architectures and concepts.

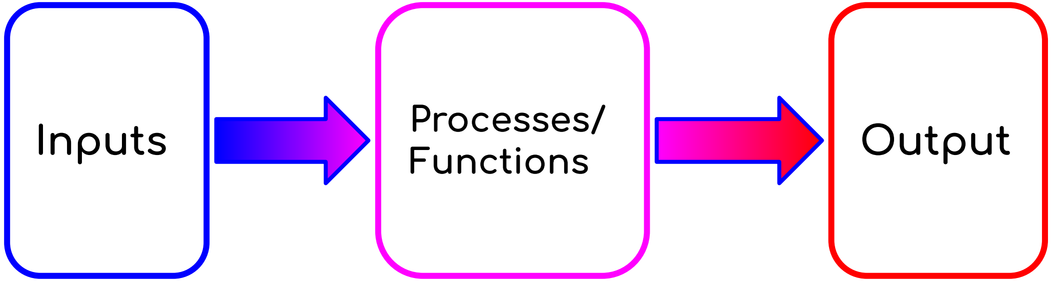
- They have similarities to past algorithms for optimization we have explored (*genetic algorithms*) in that they sometimes use past known information to achieve a task



Neural Networks: Basic aspects

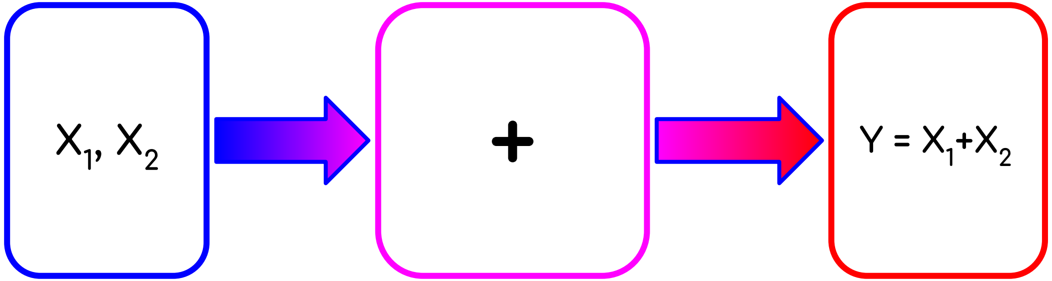
- You want to have a set of processes (mathematical functions, logical functions, etc.) which given some input data, can predict some output quantity.

- Key point is that you allow the processes to be implemented with different weights in order to optimize the behaviour of your network.
- This is typically done through training it on subsets of data where you know the answer.



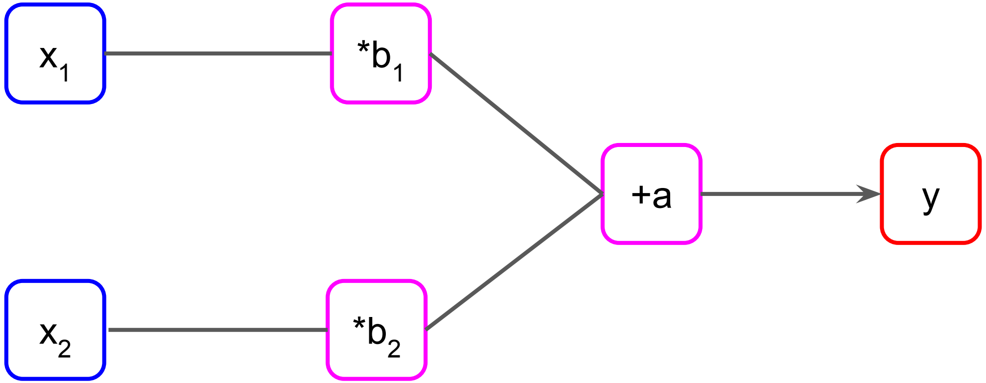
Neural Networks: Basic aspects

- In the loosest sense we could call any function a simple neural network….



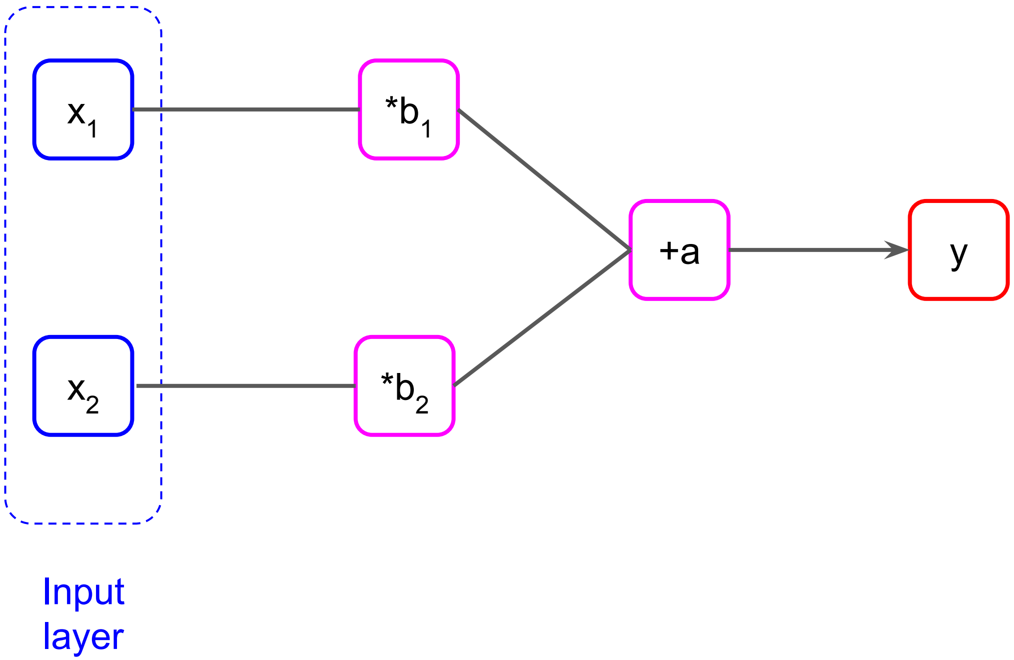
Neural Networks: Basic aspects

- Name comes because you:
  1) layer consecutive operations, and



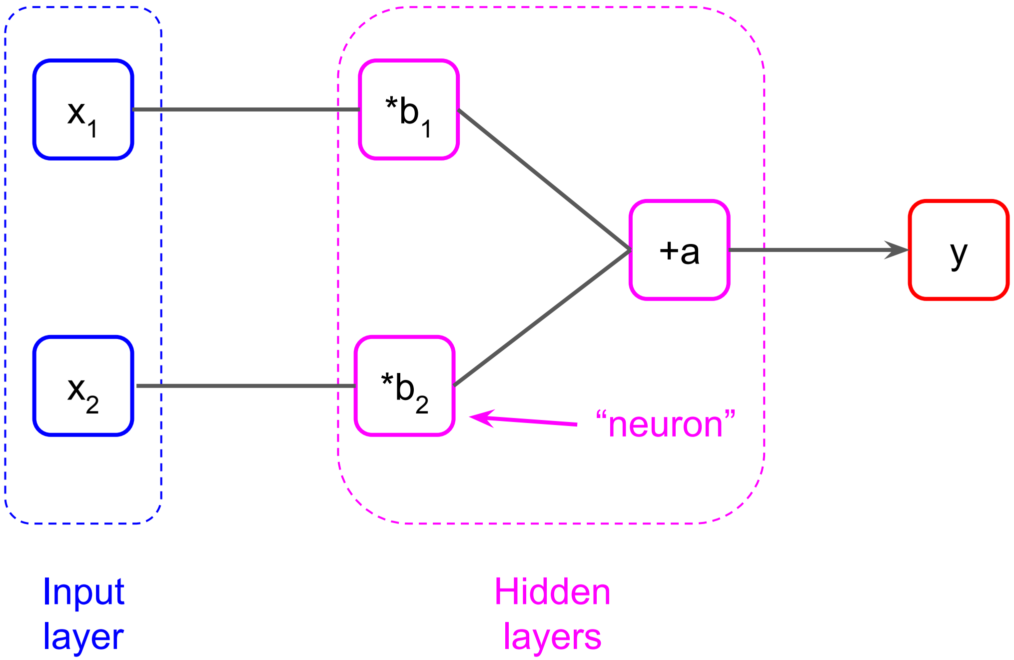
Neural Networks: Basic aspects

- Name comes because you:
  1) layer consecutive operations, and



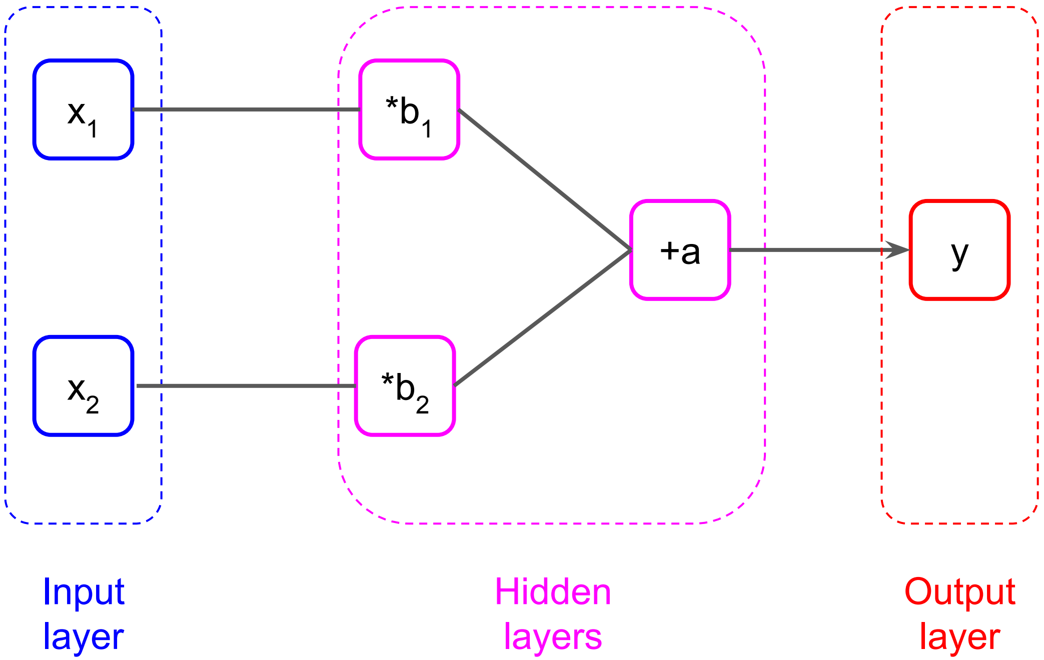
Neural Networks: Basic aspects

- Name comes because you:
  1) layer consecutive operations, and



Neural Networks: Basic aspects

- Name comes because you:
  1) layer consecutive
  operations, and

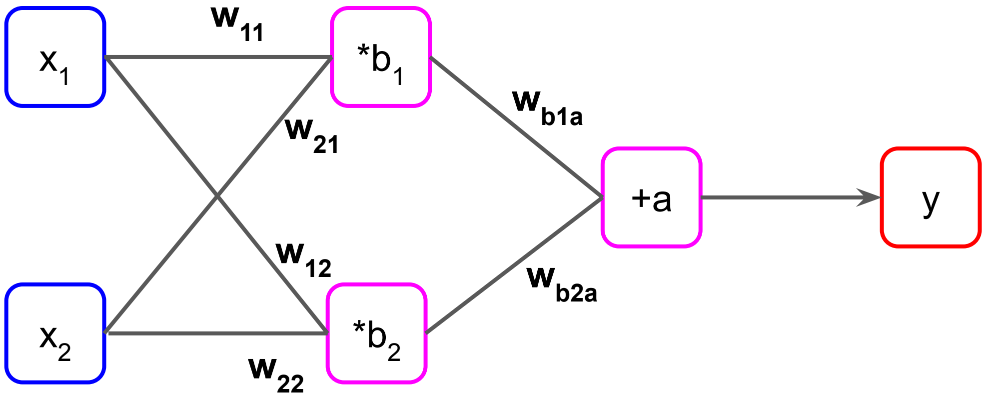


Neural Networks: Basic aspects

- Name comes because you:
  1) layer consecutive operations, and

  2) allow the neurons (processes/functions) within each layer to operate with some weighting

- Keep in mind what each neuron does could be much more complex than a simple addition. Think, rotation, scaling, sub-sampling, convolution..



Neural Networks: Basic aspects

- May not be trivial with complex network architectures...

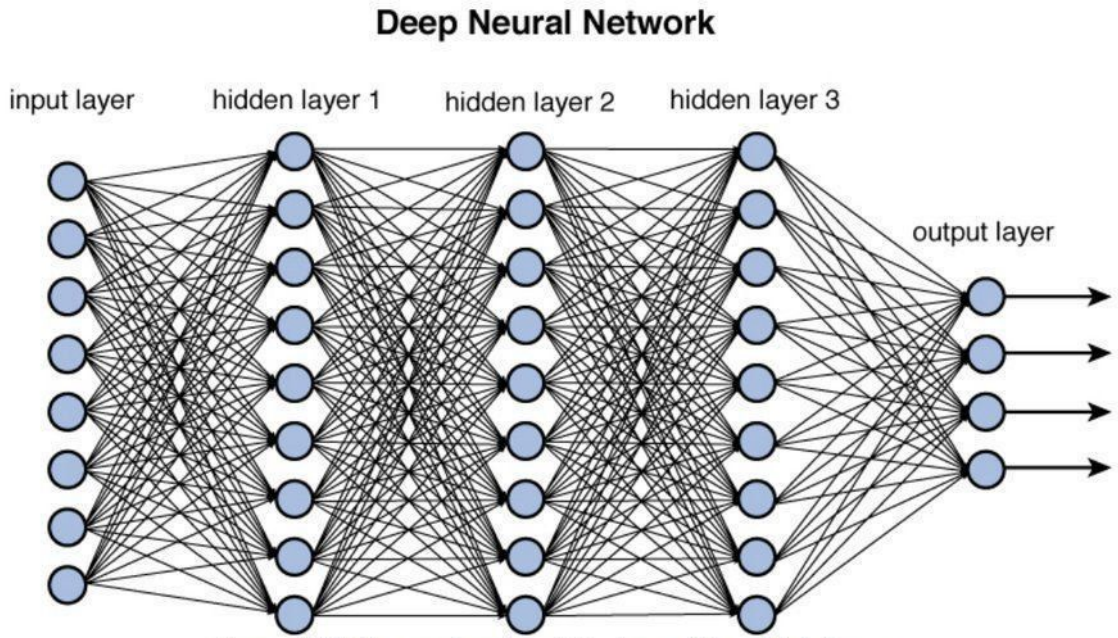
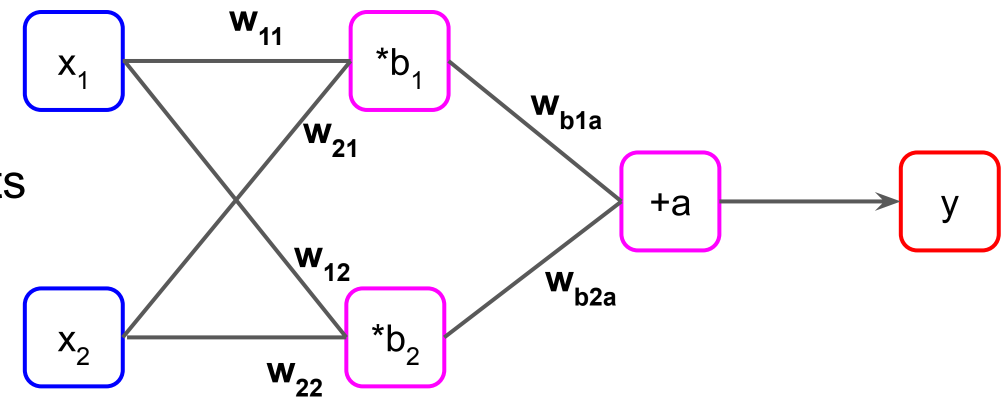


Figure 12.2 Deep network architecture with multiple layers.

Neural Networks: Training

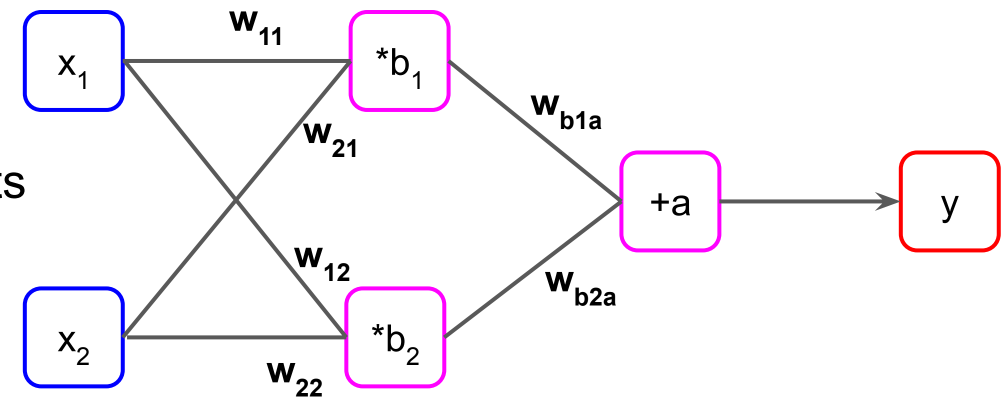
- How do we set the weights each neuron gets?

- Provide **training set** for your network and allow the weights to optimize some **cost/loss/fitness** measure.



Neural Networks: Training

- How do we set the weights each neuron gets?
- Provide **training set** for your network and allow the weights to optimize some **cost/loss/fitness** measure.
- E.g., given $y_{i,\,true}$ for each observation ($[x_1,x_2]_i$), which combinations of weights minimizes something like:



$$n^{-1}\sum_{i=1}^{n}(y_{true}-y_{network})^2$$

Neural Networks: Training

- Typically use a subset of data with known classifications and (hopefully) a wide variety of attributes.

- Why is this important? You are optimizing the network to **predict an output** based on some attributes.

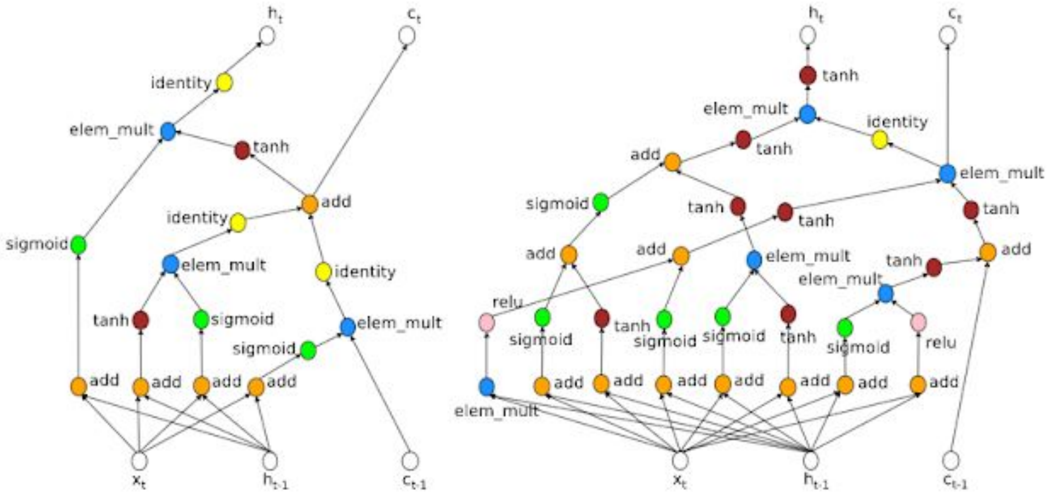
- If there are no past occurrences of an input with particular attributes, it will likely be misclassified. So despite descriptions in popular media:

  1) Neural networks are not always going to predict/discover ‘new things’, as by definition they classify based on past occurrences

  2) Neural networks should have diverse, large training sets to prevent misclassification

Neural Networks: Validation

- In addition to training sets used to find the optimal weights of a neural network. One often will run validation tests to assess how robust the classification results are to the network architecture.
- The results can depend on the number of hidden layers/neurons and their two-way connections. Changing this in a validation stage is an important part of assessing how robust a network is.



Neural Networks: Validation

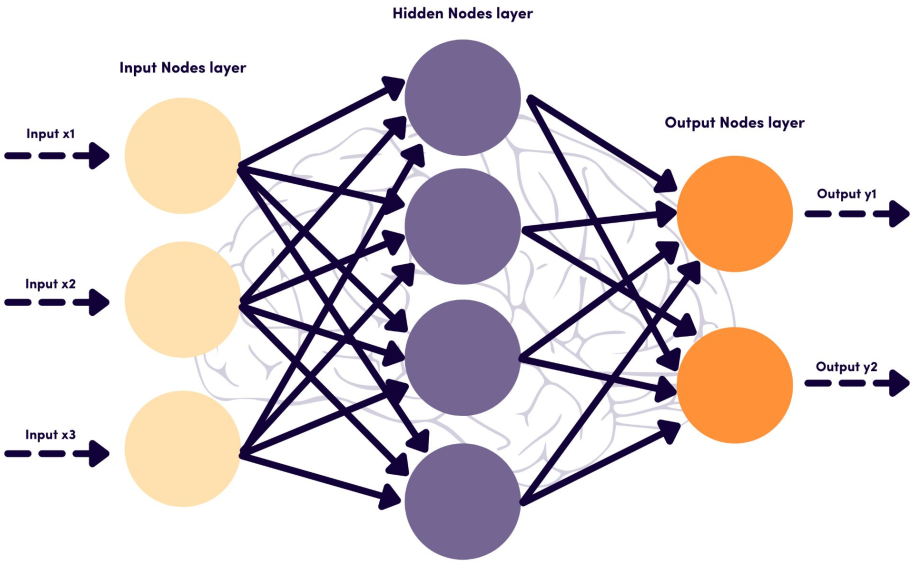
- Consider the language translation example again. Google translate uses an algorithm which can be classed as a recurrent neural network (RNN), while DeepL uses convolutional neural networks (CNN).
- Even within these two individual classes there will be differences due to training sets and network architecture differences.
- However your German-language-deficient lecturers have lots of experience with how these small changes can impact communications!

Neural Networks: Basic aspects

- Some example sub-classes of neural networks…

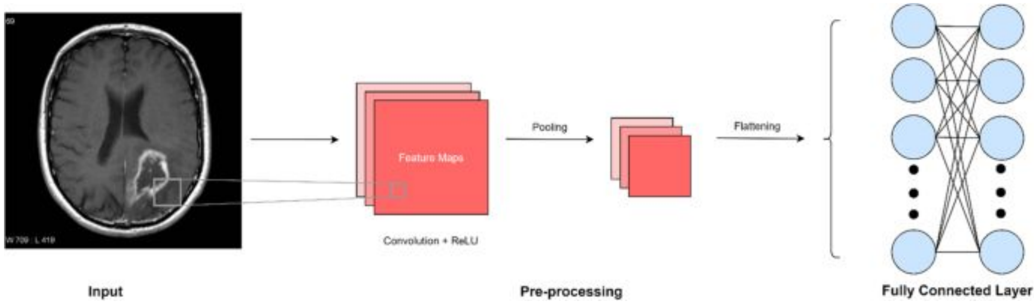
Neural Networks: Basic aspects

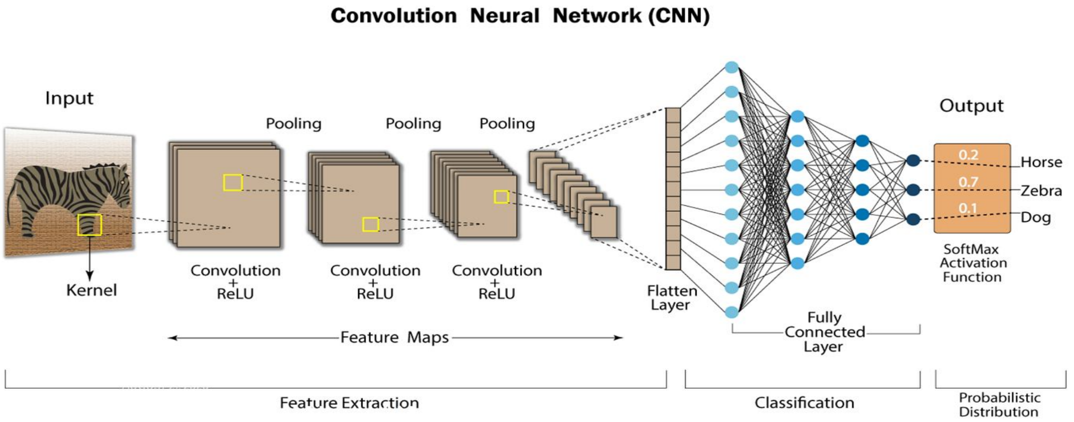
- *Artificial Neural Networks (ANNs)*
- Typically the input layer and hidden layers are organized so that a single ‘neuron’ in a hidden layer only deals with one aspect of the data. E.g., everything is passed forward to a subsequent layer.
- Can be used for **pattern recognition problems** (*facial recognition, text or speech recognition*)



Neural Networks: Basic aspects

- *Convolutional Neural Networks (CNNs)*
- These have a subset of layers which may perform independent operations like ANNs, but then a fully connected layer where operations are more intertwined
- Used for **computer vision problems** like automated *identification of objects/people/medical concerns* in videos/images

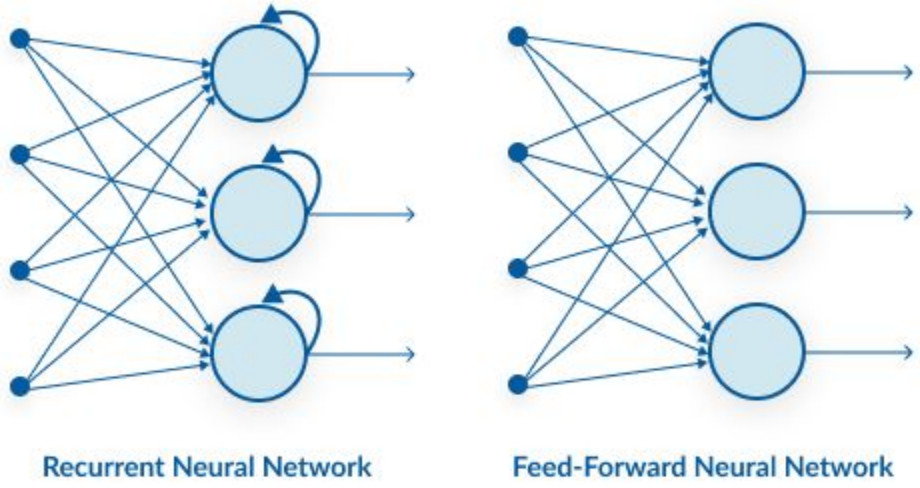




Neural Networks: Basic aspects

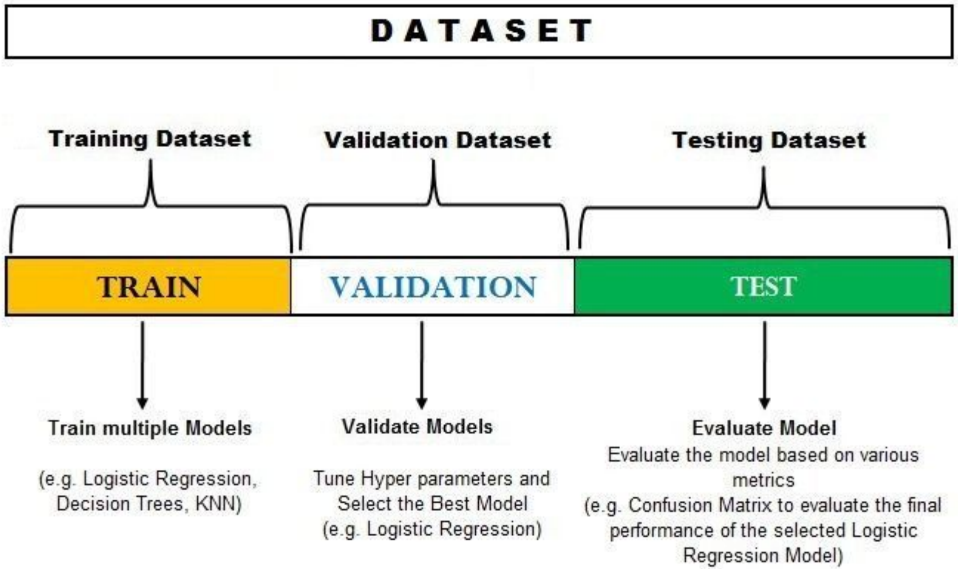
- *Recurrent Neural Networks (RNNs)*
- Similar to the previous, but with more dynamic architecture and training sets to incorporate a memory of past predictive outputs.

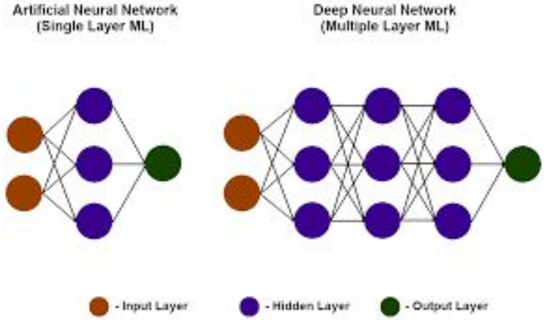
- Can be used for problems on **predictive behaviour/speech** (*e.g., Siri, Alexa for speech-based user interactions*).

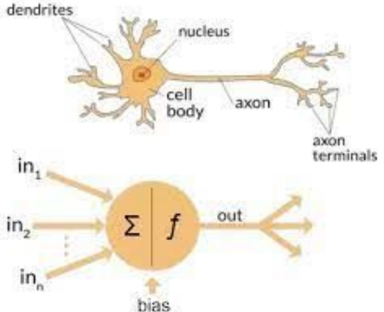


Neural Networks: Caveats and Warnings

- As mentioned, subjective choices in training sets, network architecture and other aspects are still very present in these techniques - the same as others we have covered in the course.
- Validation and representative training sets are key to avoid over-interpretation of results and biases which can have serious real-world consequences.
- Also important to try different architectures.



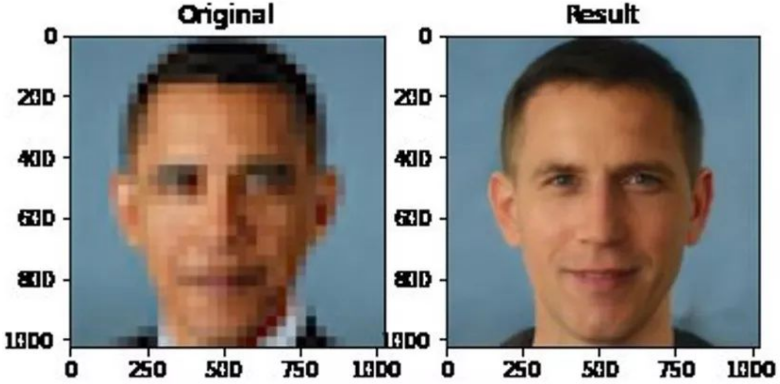




Neural Networks: Caveats and Warnings

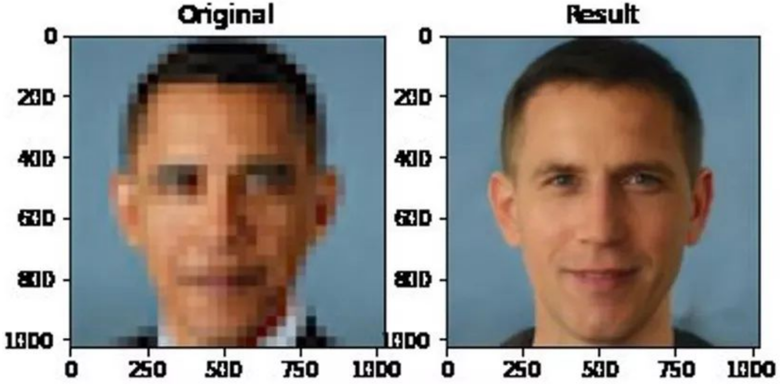
Neural Networks: Caveats and Warnings

PULSE algorithm upscaling technique to process visual data. Upscaling is a “zoom and enhance” feature: real software can’t just generate new data from nothing, so in order to turn a low-resolution image into a high-resolution one, the software has to fill in the blanks using machine learning. When using the algorithm to scale up pixelated images, the algorithm more often generates faces with Caucasian features.



Neural Networks: Caveats and Warnings

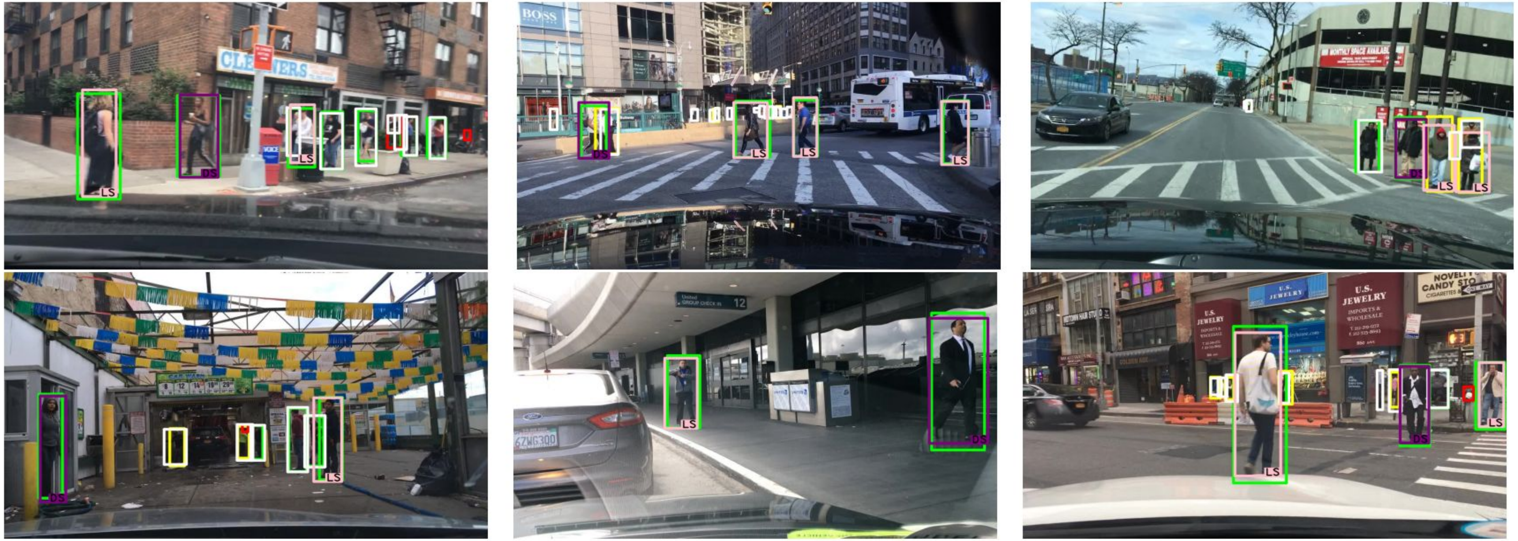
PULSE algorithm upscaling technique to process visual data. Upscaling is a “zoom and enhance” feature: real software can’t just generate new data from nothing, so in order to turn a low-resolution image into a high-resolution one, the software has to fill in the blanks using machine learning. When using the algorithm to scale up pixelated images, the algorithm more often generates faces with Caucasian features.



https://thispersondoesnotexist.com/
https://thesecatsdonotexist.com/

Neural Networks: Caveats and Warnings

Predictive inequity in detecting pedestrians of different skin tones in object detection systems (application: self-driving cars). Authors show that standard models for object detection, trained on standard datasets, exhibit higher precision on individuals with lighter skin types than darker skin types. Simple changes during learning (i.e., reweighting the terms in the loss function) partially mitigate this disparity.
Other problems: small pedestrians and occluded pedestrians.

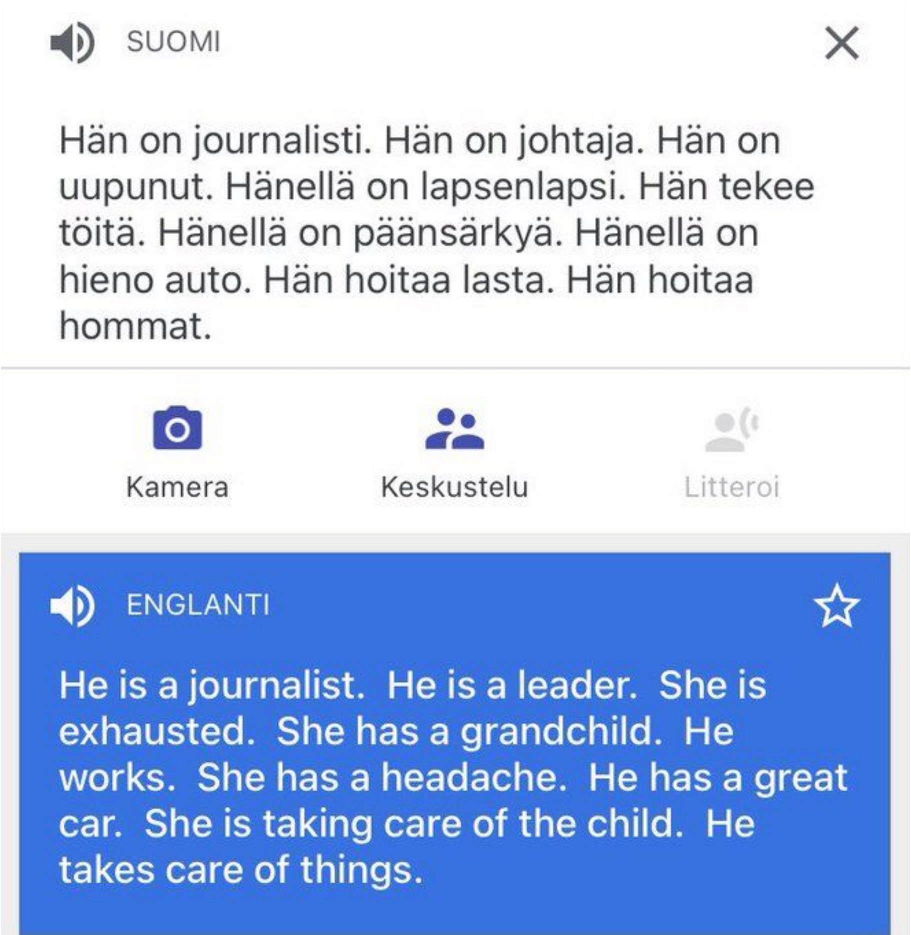


Wilson et al. 2019

Neural Networks: Caveats and Warnings

Not only images have problems: words should be considered carefully too!

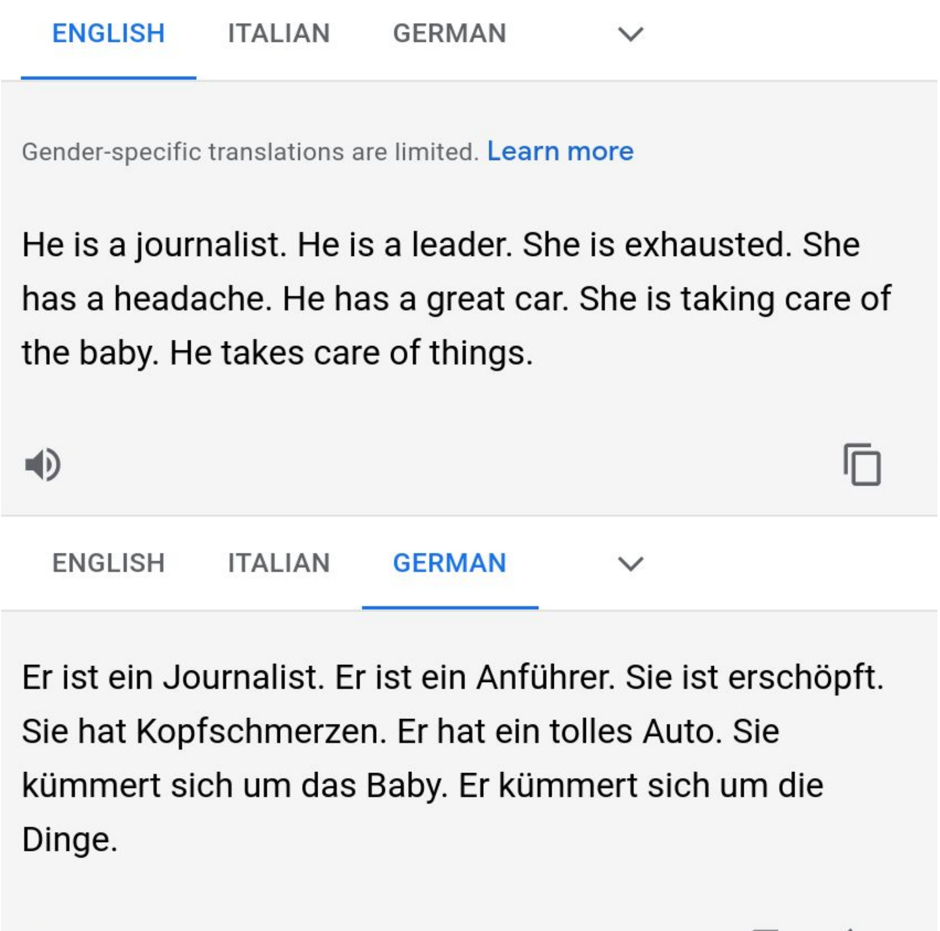
An example was recently shown on twitter by a user who pointed out that:
"In Finnish we have only one pronoun for third person regardless of the gender. If you copy-paste the sentence [...] to google translate [...], you see how the algorithm has learnt to be sexist."



Neural Networks: Caveats and Warnings

Not only images have problems: words should be considered carefully too!

An example was recently shown on twitter by a user who pointed out that:
“In Finnish we have only one pronoun for third person regardless of the gender. If you copy-paste the sentence [...] to google translate [...], you see how the algorithm has learnt to be sexist.”

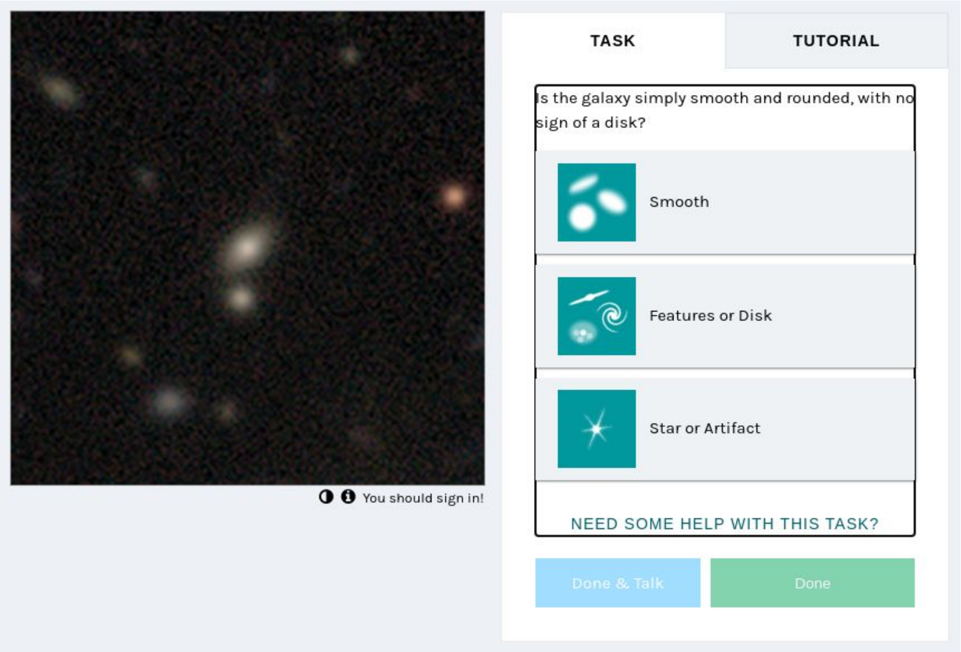


Zooniverse

Galaxy Zoo

"Goal: explore galaxies near and far, sampling a fraction of the roughly one hundred billion that are scattered throughout the observable Universe. Each one of the systems has had a unique life, interacting with its surroundings and with other galaxies in many different ways; the aim of the Galaxy Zoo team is to try and understand these processes, and to work out what galaxies can tell us about the past, present and future of the Universe as a whole."

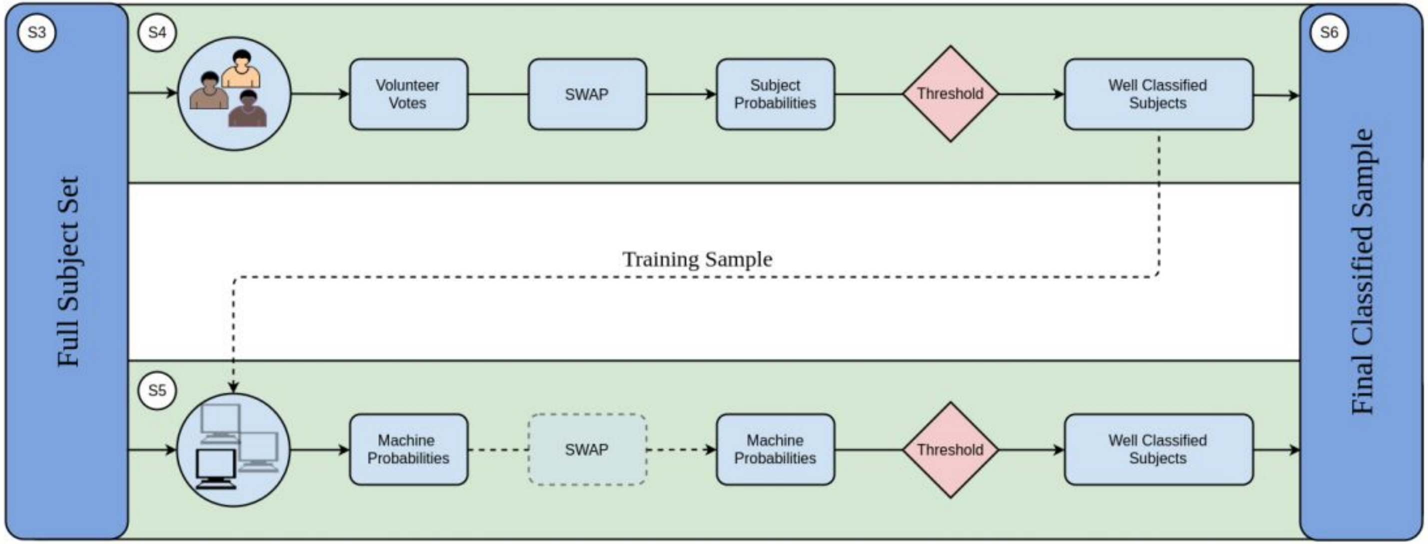
Galaxy Zoo



Zooniverse

Galaxy Zoo

The strategy is based on the fact that you can tell a lot about a galaxy just from its shape. To do this, therefore, it is necessary to classify galaxies. Combination of humans + Random Forest machine learning algorithm (which trains on non-parametric morphology indicators) is very effective! (Beck et al. 2018)



Some fun web-based neural network examples:

- There are a large variety of examples online if you want to explore classification based on neural networks.

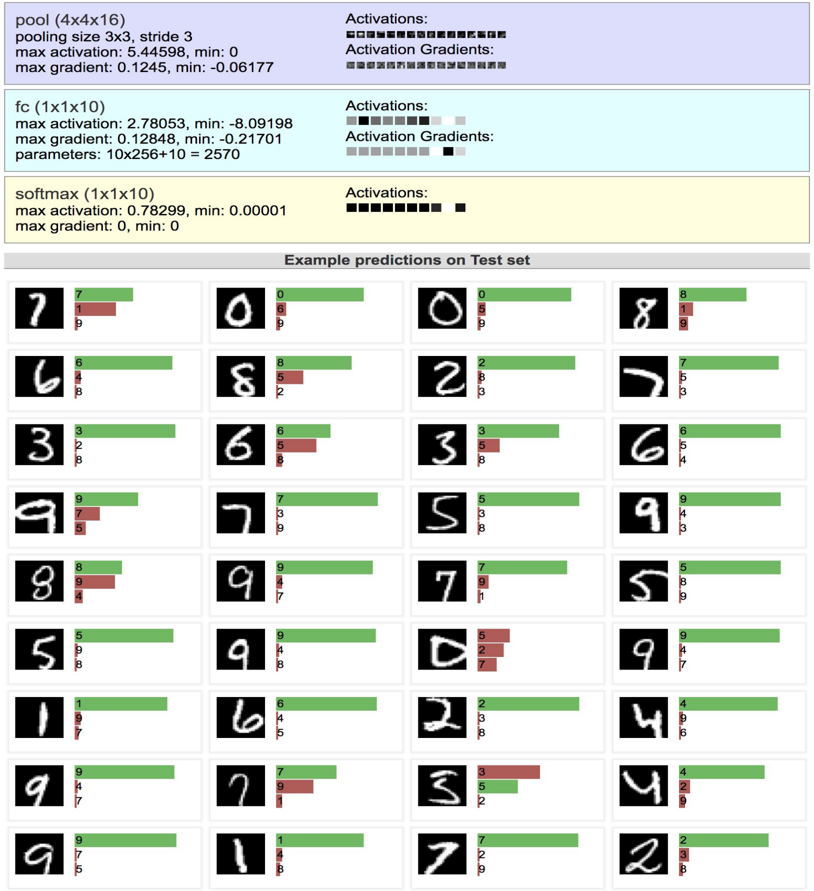
- These range from extremely sophisticated/abstract (see TensorFlow playground) - to more amusing and conceptually intuitive examples.

- For example ....

Some fun web-based neural network examples:

- **Handwritten number identification:**
  A clear example webpage where you can see the algorithms performance as more training sets are utilized.

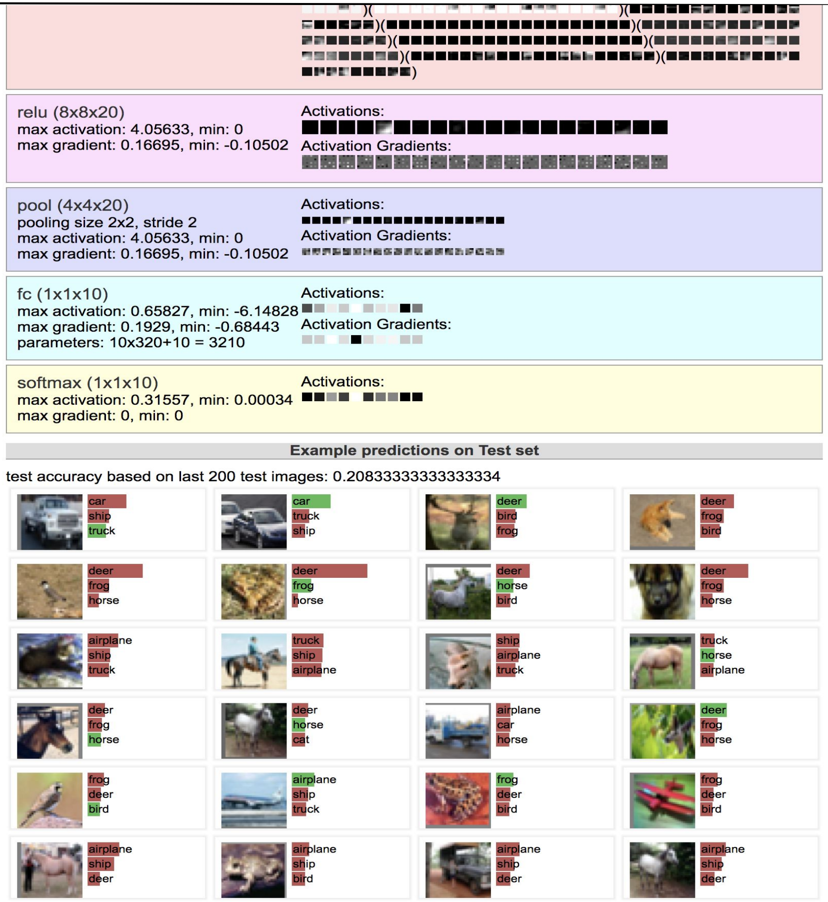
  https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html



Some fun web-based neural network examples:

- **Image recognition:**
  A clear example webpage where you can see the algorithms performance as more training sets are utilized.

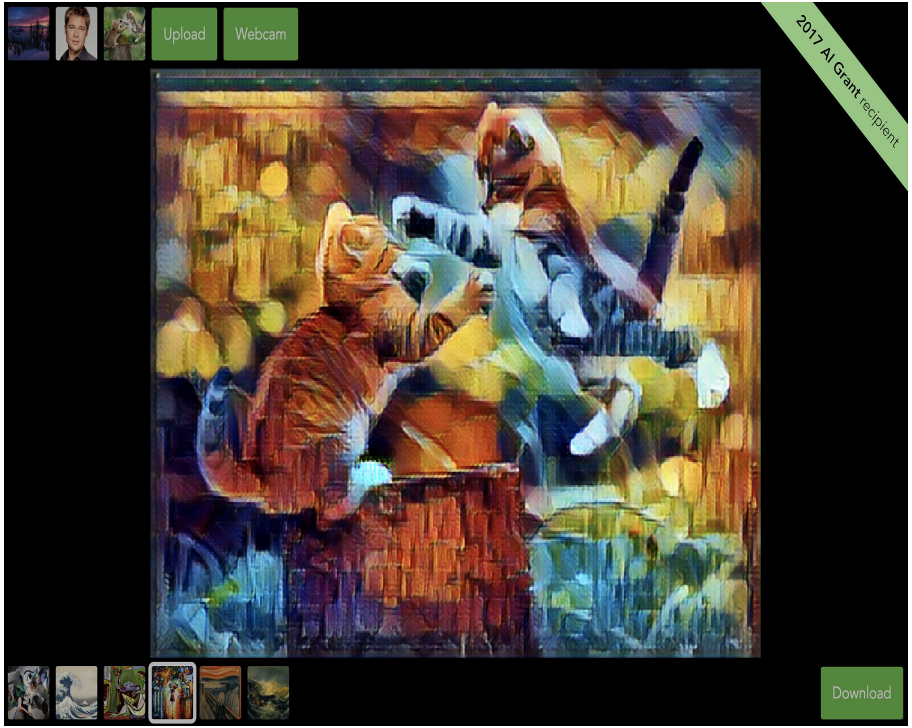
  https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html



Some fun web-based neural network examples:

- **Painting re-styling:**
  Algorithm trained on classical painting styles (cubist, realist, abstract etc.) can re-style an image in that genre

  https://tenso.rs/demos/fast-neural-style/

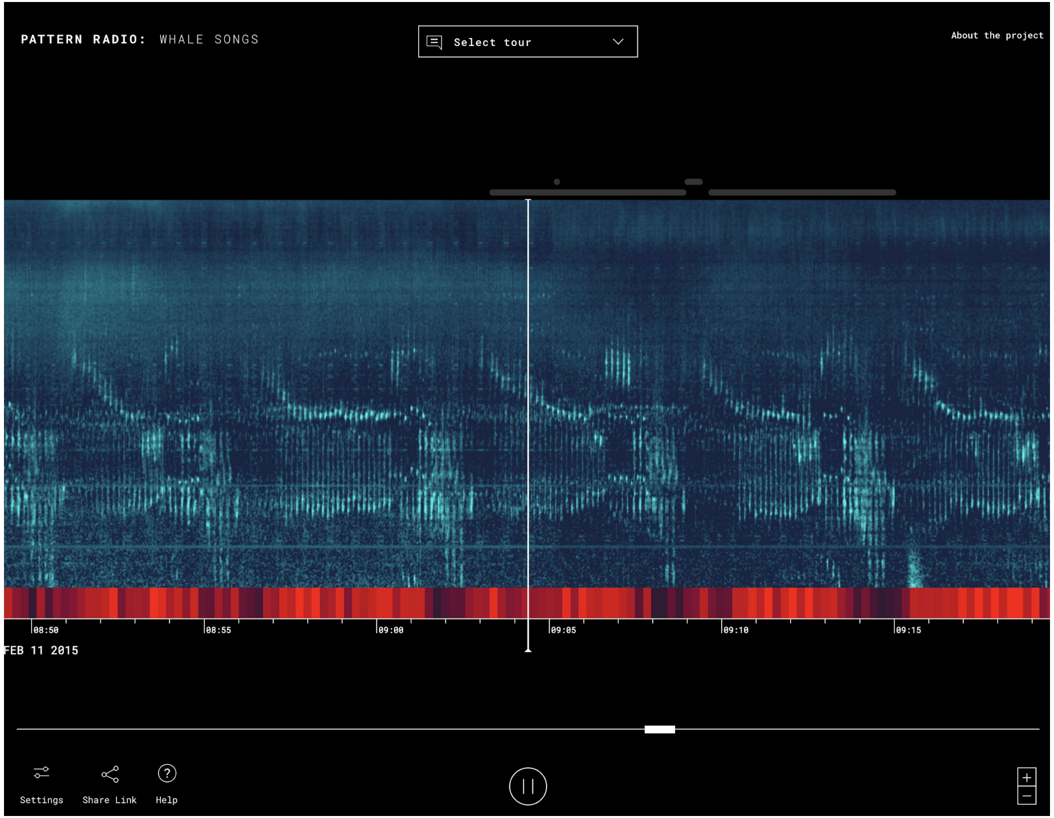


Some fun web-based neural network examples:

- **Real time classification of...**

  Drawings
  https://quickdraw.withgoogle.com/

  Whales!
  https://patternradio.withgoogle.com/

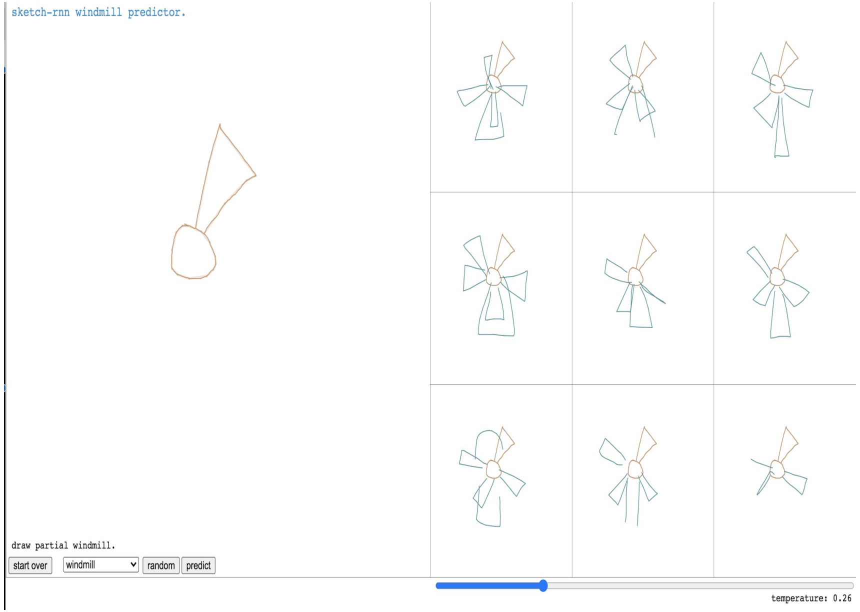


Some fun web-based neural network examples:

- **Real time ‘prediction’ of...**

Text
https://play.aidungeon.io/main/landing

Drawings
https://magenta.tensorflow.org/assets/sketch_rnn_demo/multi_predict.html



I can also recommend the videos on the **3Blue1Brown** YouTube channel as a nice introduction about the topic (the video about LLMs is great!)

Summary

- The numerical method techniques we have learned are required and important aspects of cutting edge-simulations in research in meteorology and astrophysics

- Machine Learning algorithms are increasingly popular as flexible ways to search for patterns and classify large data sets.

- Key to their use is the ability to refine and improve their outputs based on more complete training sets or network architecture.

- Like any algorithm, training, testing and assessment of the networks is crucial (especially for important real-world applications).

Existing solutions in Python

- TensorFlow, PyTorch, Keras for Neural networks
- scikit-learn, statsmodels libraries for *linear regression*
- Scikit-learn library for logistical regression (multi-class classification)
  (contains: k-means clustering, random forests, PCA analysis…)