

Lets go deeper!

Simon Kluettermann

Is9 tu Dortmund

21. November 2022

*Simon Kluettermann*

- No Case Study next week
  - neither Tuesday (29.11) nor Thursday (01.12)
  - if you need help: just write me an email!
- In two weeks: Case Study switched
  - Q+A Tuesday (6.12, 14:00) online only
  - Case Study Meeting Thursday (08.12, 14:00-16:00), in OH12 Room 3.032

# Big Picture

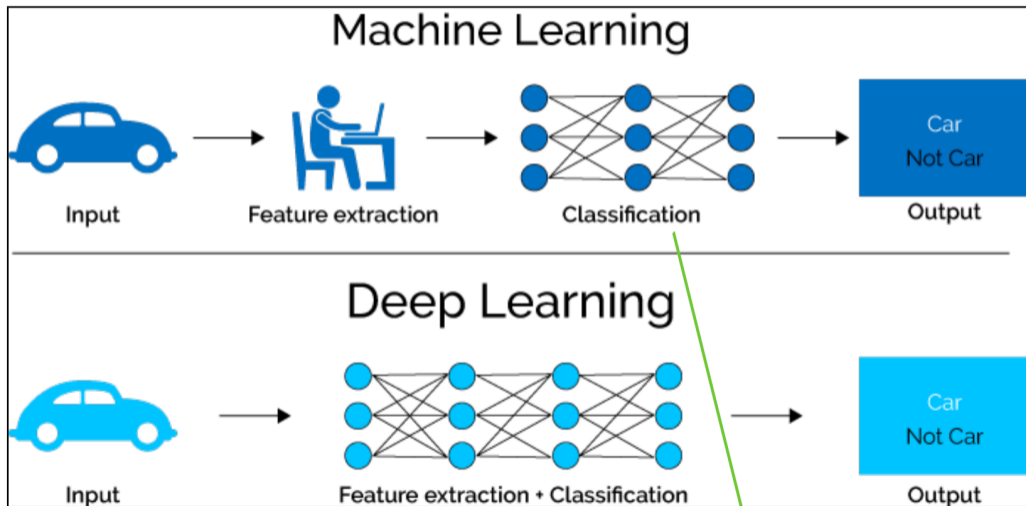
- Goal for this case study: Have better hyperparameters than pyod!
- So each of you: Gets assigned two algorithms!
  - One fairly simple before
  - One more complicated one today
- Try to find the best possible hyperparameters for your algorithms
  - Try to be clever (for example: PCA:  $n_{components} < n_{features}$ . Maybe  $\frac{n_{components}}{n_{features}}$  constant?)
- Afterwards
  - Write down your findings into a simple function (given data, what are my best hyperparameters)
  - Write down your finding into a report (together, double column. max 6 Pages per student, plus comparison of algorithms to each other)
  - One final presentation together in front of my colleagues. About 10min per student.

# Evaluating your hyperparameter

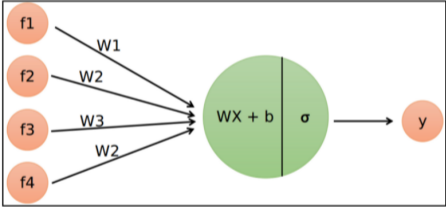
- My suggestion: Compare to normal parameters.
- This means you get two lists of AUC scores
- Your params: [0.80,0.75,0.73,.....,0.95]
- Pyod params: [0.82,0.71,0.48,.....,0.95]
- look at two values
- $\sum_i your_i - pyod_i$ 
  - Total improvement. If positive, then your parameters help;)
  - But hard to see if this is significant
- Fraction of  $your_i > pyod_i$ 
  - Quantised, so does not care about improving your parameters further
  - But easy to see if this is significant
    - 0.5  $\Rightarrow$  Probably just random
    - 0.9  $\Rightarrow$  Probably quite significant

## How to continue

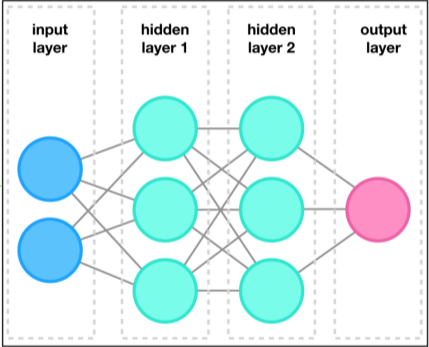
- See how far you can improve this?
- Treat this as a supervised optimisation problem: Given this dataset, find the best hyperparameters
- Might be useful to look at more input parameters
- Might help to formulate your parameters differently
- But be aware of **overfitting!**



# Intro to Deep Learning



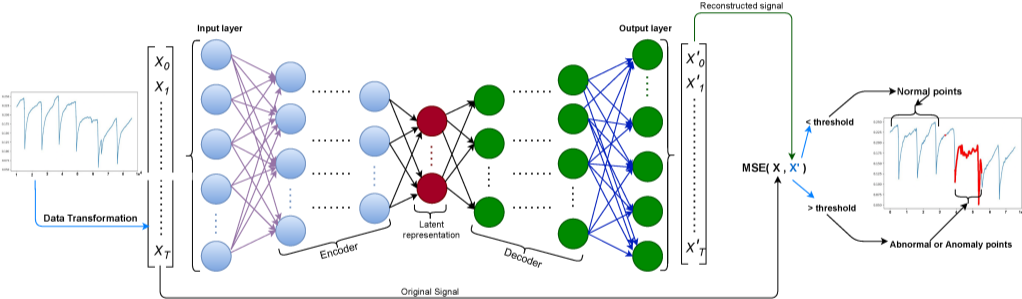
Logistic Regression single neuron



- The idea is always the same:
  - Define complicated model to learn (often millions of parameters)
  - Define loss function that this model should minimize (example:  $\sum_i (y_i - f(x_i))^2$ )
  - Find parameters that minimize the loss ( $\Rightarrow$ Backpropagation)
- Usually Neural Networks:
  - $f(x) = f_n(x) = \text{activation}(A_n \cdot f_{n-1}(x) + b_n)$
  - $f_0(x) = x$
- Powerful, as you can show that when there are 3 Layers+ (and infinitely sized matrices), you can approximate any function
- $\Rightarrow$ So a model becomes a loss function



# Autoencoder



# Autoencoder

- Lets look at some of its Hyperparameters
- Autoencoder Specific
  - Compression factor (Latent space size)
  - Loss function (mse?)
- Neural Network architecture
  - Number of layers
  - Number of neurons in each layer (Shape of the matrices  $A_n$ )
- Optimisation parameters
  - Learning Rate
    - Controls how fast the parameters are found
    - To high value makes the training unstable
  - Batch size
    - Controls how many samples are averaged together.
    - Lower values make the training more stable, but also the result less optimal

## For next time

- (if you have not finished finding good parameters for your old algorithm, continue searching for them)
- Take a look at your new algorithm
- Run it once on cardio, take a look at which parameters you have
- Prepare a similar presentation to last time (include your cardio result)