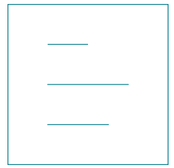
datahouse

Artificial Neural Networks

by Daniel Meister, CTO at Datahouse



Artificial Neural Networks

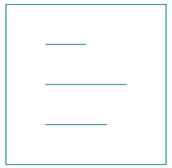
Agenda

Motivation & History, Introduction
Why is it called «Artificial Neural Network» incl. short theory introduction

Classification Problems
Some examples for classification problems; parameter tuning

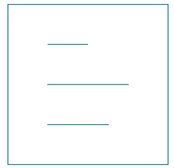
Regression Problems
How to make continuous predictions with artificial neural networks

Deep Learning & Generative Adversial Networks (GANs)
Some current applications of neural networks



Material and Sources

- obviously this presentation takes inspiration and material from various sources
- many of them are itself interesting articles and papers to read
- at the end of the slide set there will be a collection of links for further reading



Organizational Points

- the slides and the accompanying lecture contain all the main ideas
- you should try to follow the more practical examples by executing the R code yourself
- there are also a few ideas for small projects/tasks to do more detailed work
  - if you complete one of those and would like some feedback feel free to send it to me
  - preferably as an RMarkdown file in a Git repository (but I can also handle Word files via e-mail)
- this is a fairly new lecture so there is (not yet) much additional material
  - also if you have feedback to share please let me know!

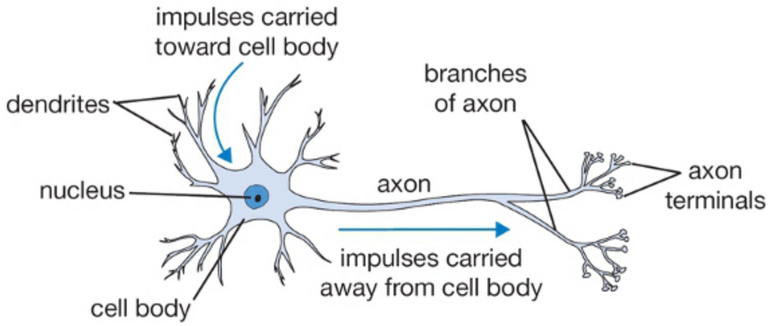
Technical Notes

Motivation & History, Introduction

Motivation & History, Introduction

Original Motivation

Model a program that works similar to the Human Brain



- built of different neurons
  (~ 100 billion - 10^9 in the human brain)
- connected by sending electrical signals
- type and strength of connection influence the flow of those signals

Motivation & History, Introduction

What are Artificial Neural Networks?

Definition on Wikipedia (slightly shortened)

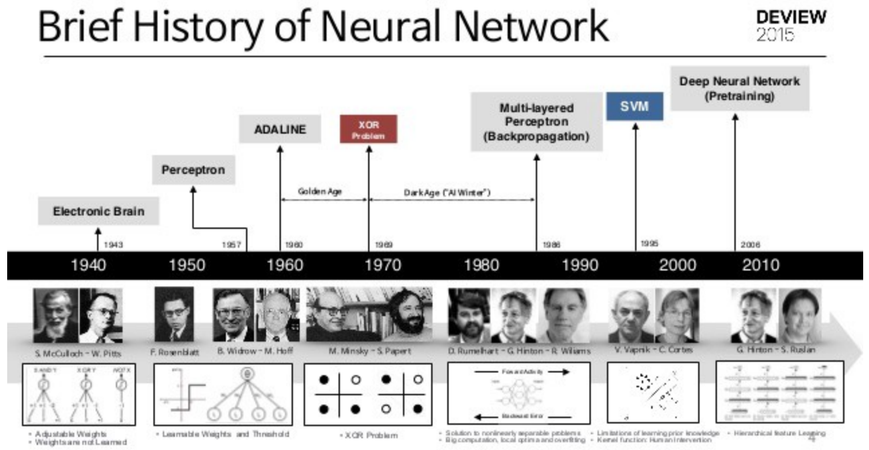
Artificial neural networks (ANNs), usually simply called neural networks (NNs), are computing systems vaguely inspired by the biological neural networks that constitute animal brains.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron that receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. [...] Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer), to the last layer (the output layer), possibly after traversing the layers multiple times.

Motivation & History, Introduction

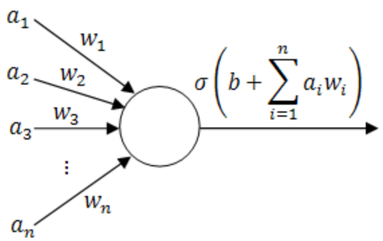
A short History

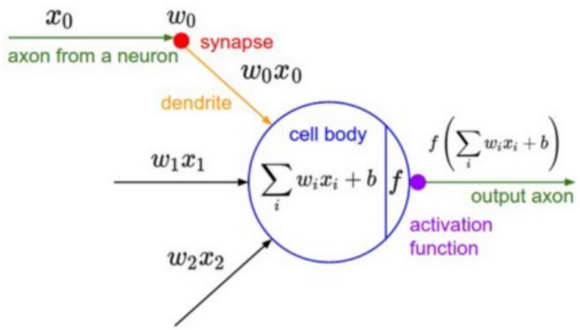
- some early successes until the 1960
- then not much happening until the 1990
- accelerating development over the last 30 years
  - backpropagation algorithm for training
  - application in games (e.g. Backgammon) using reinforcement learning
  - Deep Learning
  - Generative Networks
  - etc.



Motivation & History, Introduction

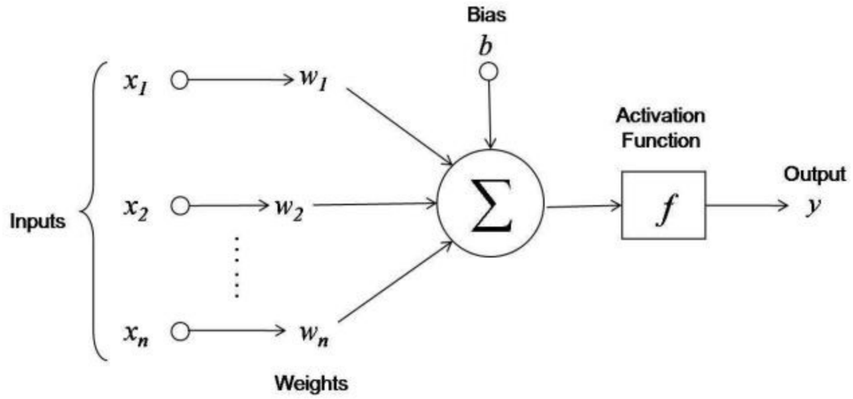
Basic Building Block: The Perceptron



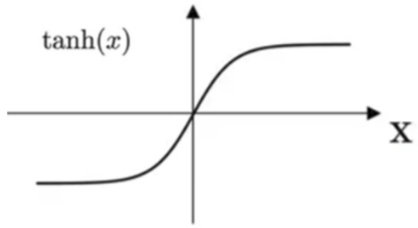


Motivation & History, Introduction

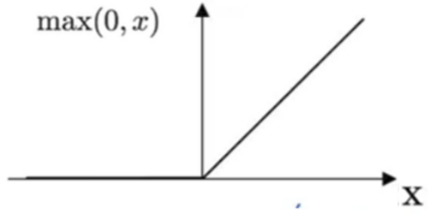
Network Architecture: Activation (Function)



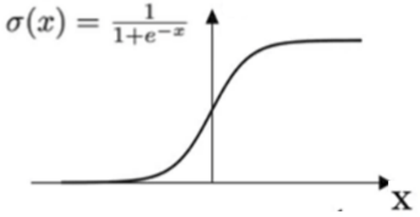
Hyper Tangent Function



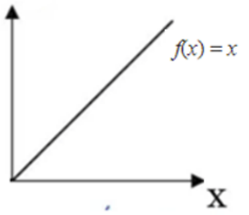
ReLU Function



Sigmoid Function

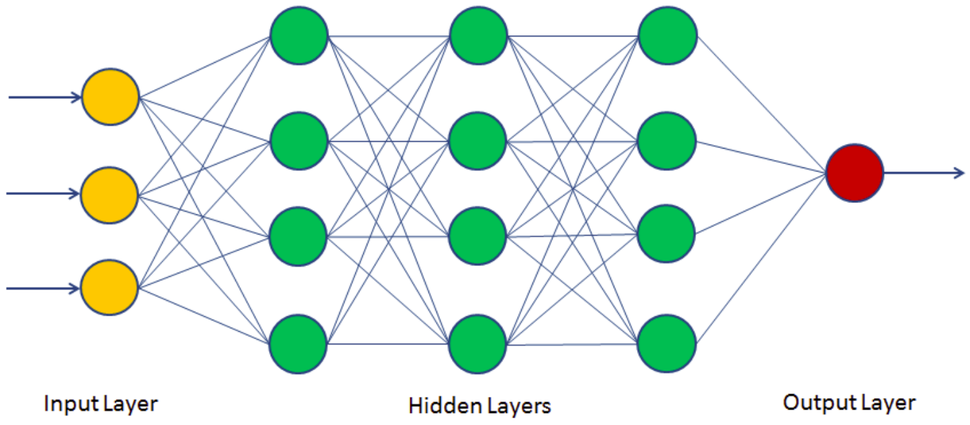


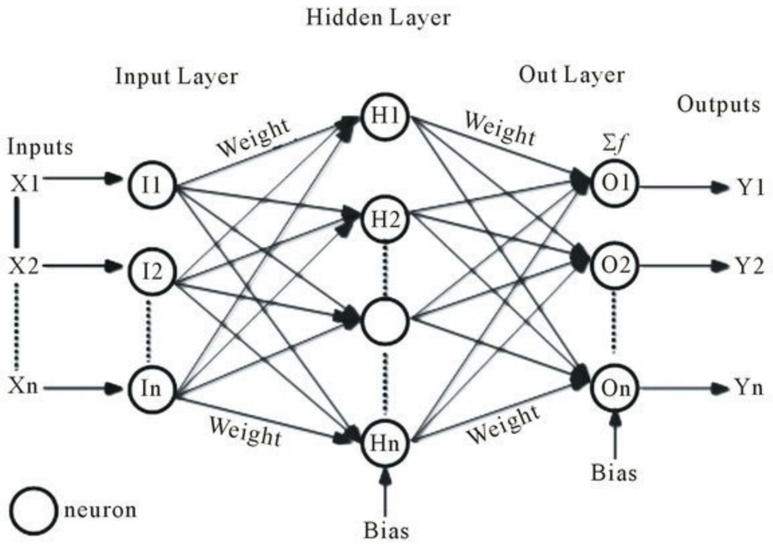
Identity Function



Motivation & History, Introduction

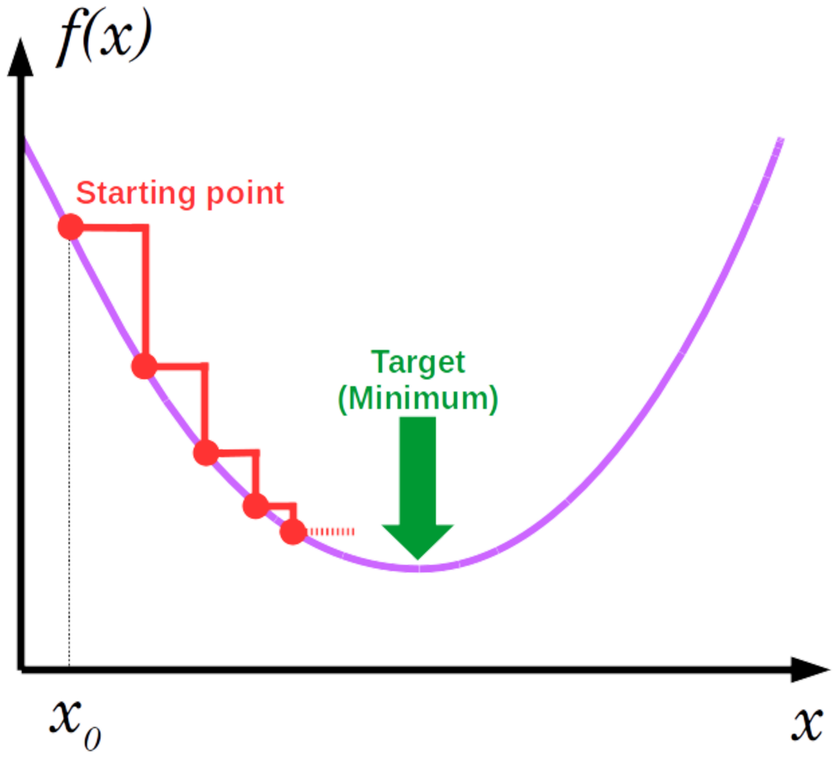
Network Architecture: Layering

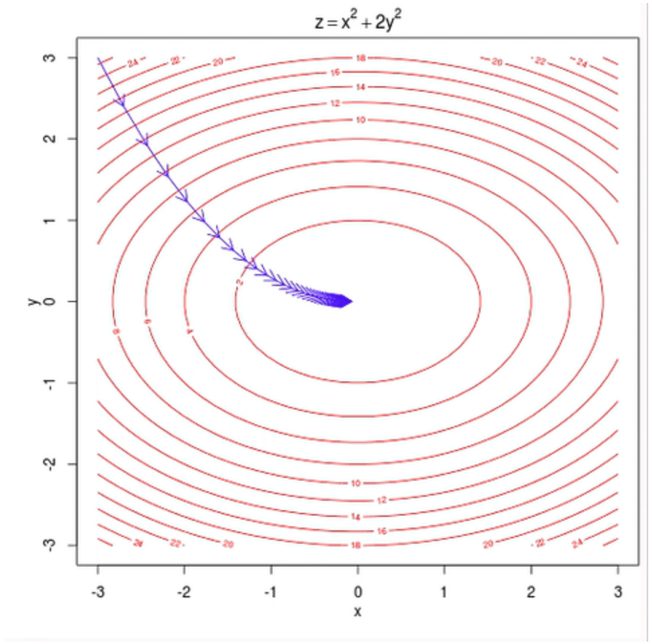




Motivation & History, Introduction

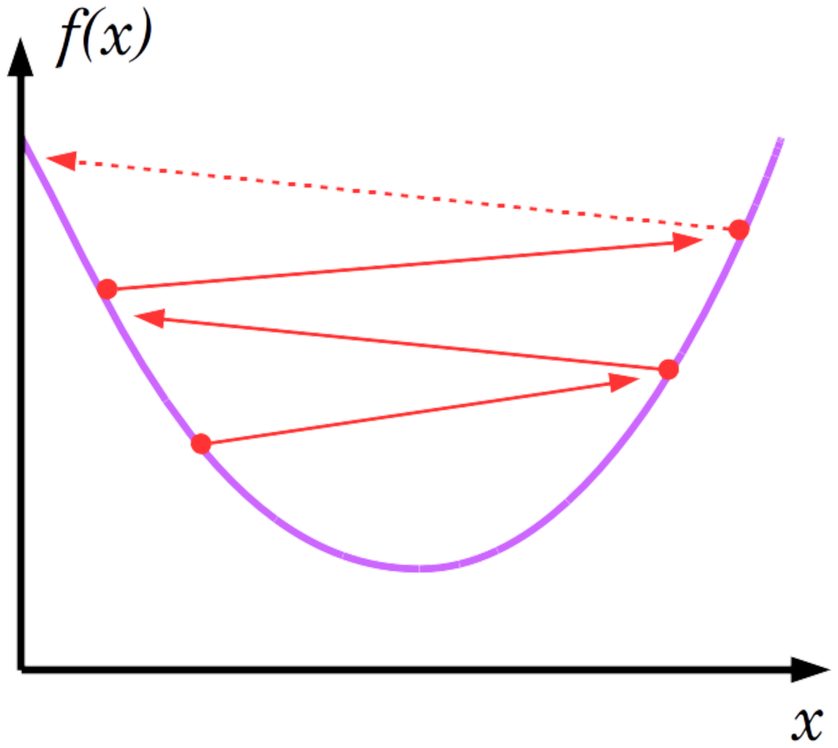
Network Training: Gradient Descent

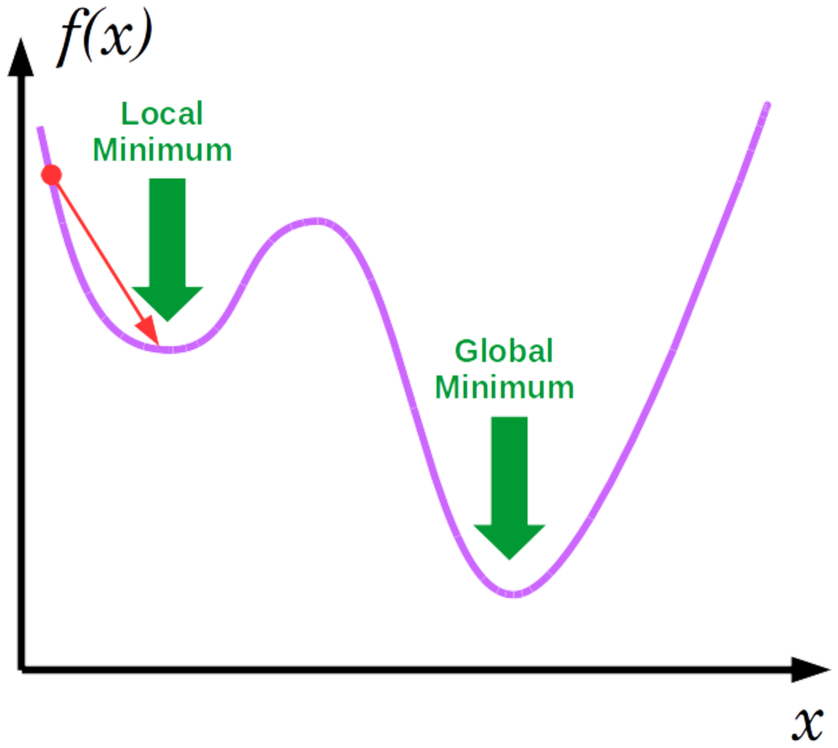




Motivation & History, Introduction

Network Training: Challenges





Motivation & History, Introduction

Neural Networks in Python

(Simple) neural networks (that we are discussing today) are usually trained in Python using the `scikit-learn` (or `sklearn`) package suite.
This is also capable of doing linear models, support vector machines, random forests, and many more.

Motivation & History, Introduction

Neural Networks in R

Targeted Packages

- `neuralnet` around for a long time
- `nnet` a bit more modern
- and many more

In ML "Frameworks"

- `caret` can make use of both the packages mentioned on the left
- other solutions like `h2o` also provide implementation for neural network training

Classification Problems

Classification Problems

How to classify with a Neural Network?

If we only have a binary classification problem, we need one neuron in the output layer with final values of 0 and 1 representing the two classes.
Based on domain and problem specific considerations one can choose a good threshold to separate the two or simply us mathematical rounding (i.e. set the threshold to 0.5)

Classification Problems

German Credit Scores (Lab 1)

In the first example we will have a look at some German Credit data: «These data have two classes for the credit worthiness: 'no-default' or 'default'. There are predictors related to attributes, such as: checking account status, duration, credit history, purpose of the loan, amount of the loan, savings accounts or bonds, employment duration, Installment rate in percentage of disposable income, personal information, other debtors/guarantors, residence duration, property, age, other installment plans, housing, number of existing credits, job information, Number of people being liable to provide maintenance for, telephone, and foreign worker status.»

Classification Problems

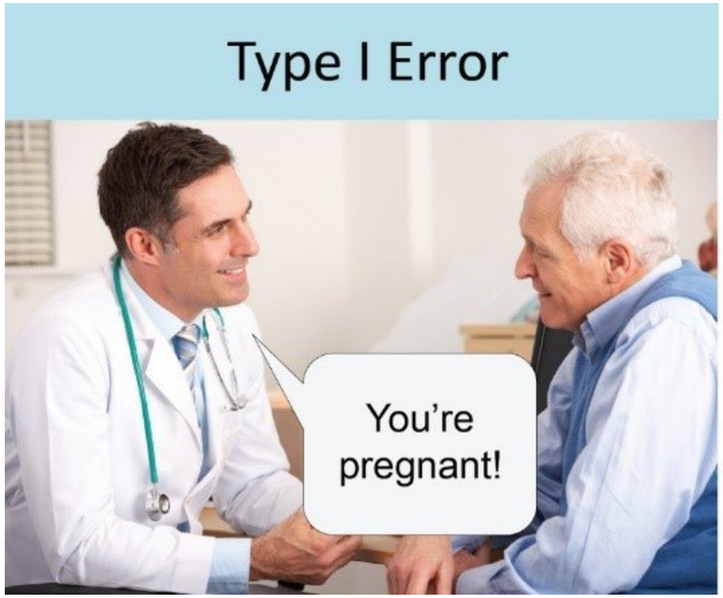
Side Note: Confusion Matrices

- A tool often used in classification problems
- Gives an overview of the successfull classifications
- But also shows both error types
- Many standard measures are being derived from these numbers

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	**True Positive**	**False Positive** (Type I error)	Positive predictive value = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Test Outcome Positive}}$
	Test Outcome Negative	**False Negative** (Type II error)	**True Negative**	Negative predictive value = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Test Outcome Negative}}$
		Sensitivity = $\frac{\Sigma \text{ True Positive}}{\Sigma \text{ Condition Positive}}$	**Specificity** = $\frac{\Sigma \text{ True Negative}}{\Sigma \text{ Condition Negative}}$	

Classification Problems

Side Note: Error Types



Source: To Err is Human: What are Type I and II Errors?

Classification Problems

German Credit Scores (Exercise 1)

- Work in groups of two so you can talk about what works and what does not
- Using the RMarkdown file from the German Credit Score lab make sure you can run the code yourself
- Try to vary the number of hidden neurons and find an "optimal" number
  - How could you compare two versions?
- Of the many variables given to the input layer surely some are more important than others
  - Can you devise a model that is only a bit worse than the original but only uses 10% of the variables?

Classification Problems

How to work with more than two Classes

If there are more than two classes we need a neural network with as many neurons in the output layer as there are different classes.
Usually the classification is done by looking for the output neuron with the maximal value.

Classification Problems

Iris Data (Lab 2)

In the second example (now in R) we will have a detailed look at predicting the classes in the `iris` dataset.
«The [...] data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.»

Regression Problems

Regression Problems

How to predict continuous Variables

In order to predict a continuous dependent variable we can simply have one neuron in the output layer of the network and use it's final value as the prediction.
Of course, in this case the sum of signals to the output neuron is not passed through the activation function (achieved with e.g. `linear.output = TRUE` for the `neuralnet` library in R and by using the `MLPRegressor` class for `sklearn` in Python.)

Regression Problems

Cereal Ratings (Lab 3)

In this example we will try to predict the customer rating of breakfast cereals given some nutritional information.
«The data come from the 1993 ASA Statistical Graphics Exposition, and are taken from the mandatory F&DA food label. The data have been normalized here to a portion of one American cup.»

Classification Problems

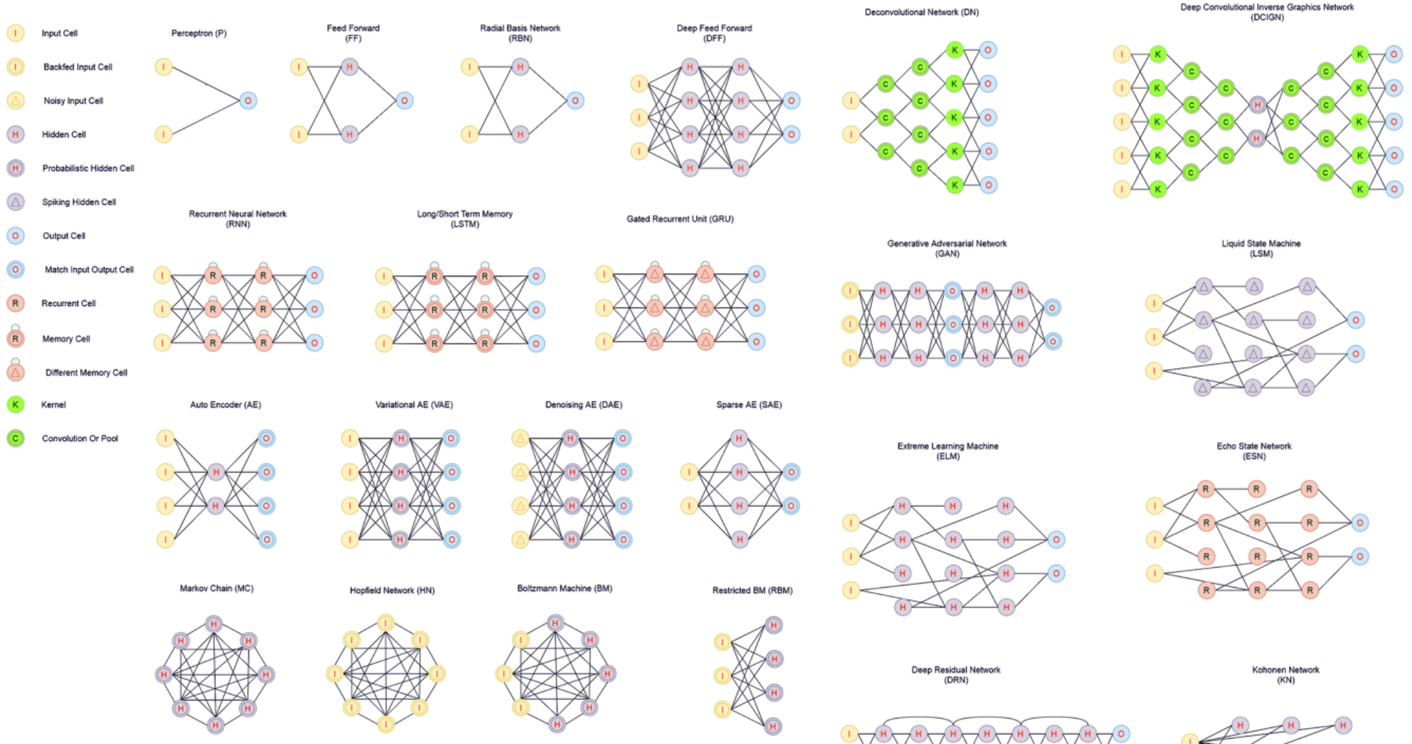
Concrete Compressive Strength (Exercise 2)

- Work in groups of two so you can talk about what works and what does not
- Download the data about concrete compressive strength from the UCI Machine Learning Repository
  - there you will also find some description of the dataset
  - you can convert the Excel-File to CSV or read directly via `read_xlsx`
- Have a short look at the data and maybe create a few descriptive plots
- Train a neural network to predict the strength
- Test the trained network and compare to the true values

Deep Learning & Generative Adversial Networks (GANs)

Deep Learning & Generative Adversial Networks (GANs)

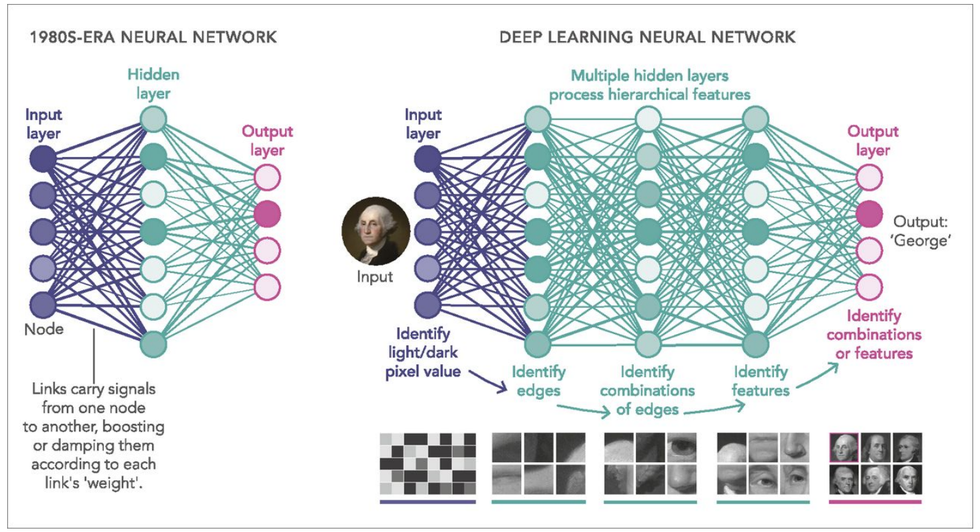
Many new Network Architectures



Deep Learning & Generative Adversial Networks (GANs)

Why is it called Deep Learning?

- Classical Network with many hidden layers (deep)
- Also some special layer types (convolution etc.)
- Does some feature engineering on its own
- Needs parallel processing power (e.g. GPU)
- Used for video, audio, text recognition etc.



Deep Learning & Generative Adversial Networks (GANs)

Who knows one of these three persons?

Source: Analyzing and Improving the Image Quality of StyleGAN. Tero Karras et al. 2019

Deep Learning & Generative Adversial Networks (GANs)

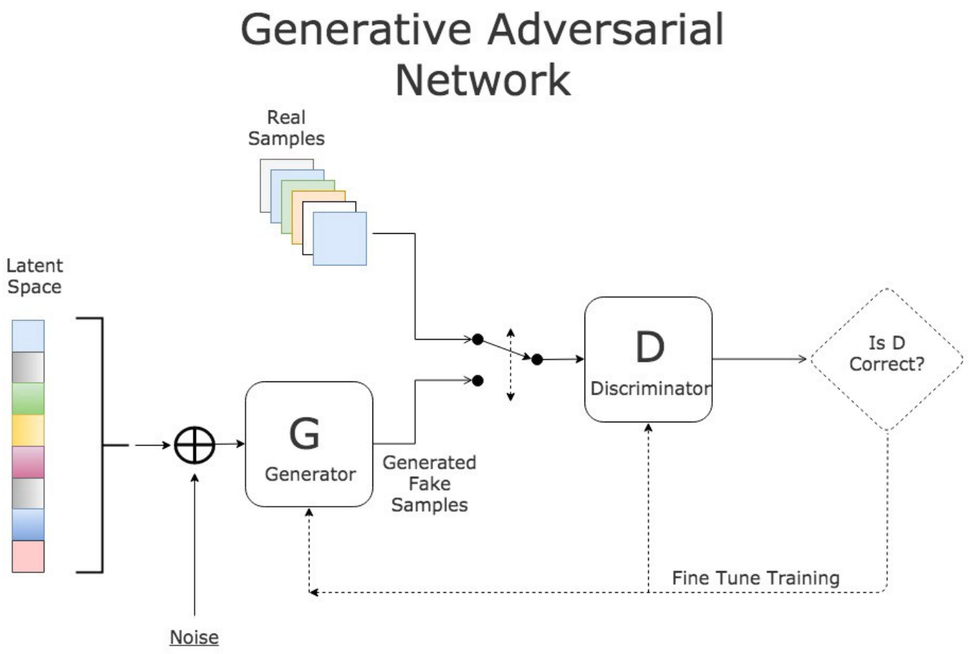
What are GANs (1)

The main goal of a Generative Adversarial Network (GAN) is to generate realistic data from scratch. It is mostly used for images but has been applied to other domains like music.
To train the model you only provide example data of realistic images (e.g. of faces but can also use cars, cats, living rooms etc.).
The idea was first developed in a paper from 2014 by Ian Goodfellow and is today still an active field of research.

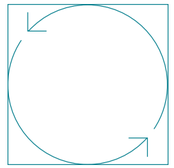
Deep Learning & Generative Adversial Networks (GANs)

What are GANs (2)

- Two ANN are involved in the process
- One tries to create very realistic "fake" images (the generator)
- The other tries to differentiate between "fake" and real images (the discriminator)
- Training those two in sync leads to impressive results



Summary



Summary

- Not a new concept but has recently regained popularity (due to increased computing power)
- ANN have the ability to learn and model non-linear and complex relationships
- Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables
- Due the large numbers of degrees of freedom they can be very sensitive to overfitting
- Very much research an effort in recent years especially in newer concepts in Deep Learning
  - which is an extremly powerful tool if used for a matching problem

datahouse

October 23rd, 2025

Thank you for your attention!

Daniel Meister
+41 44 289 92 30
daniel.meister@datahouse.ch

Datahouse AG
Alte Börse
Bleicherweg 5
CH-8001 Zürich

www.datahouse.ch

Sources and Further Readings

Sources and Further Readings

Artificial Neural Networks

https://en.wikipedia.org/wiki/Artificial_neural_network
https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

GAN
https://paperswithcode.com/method/gan#