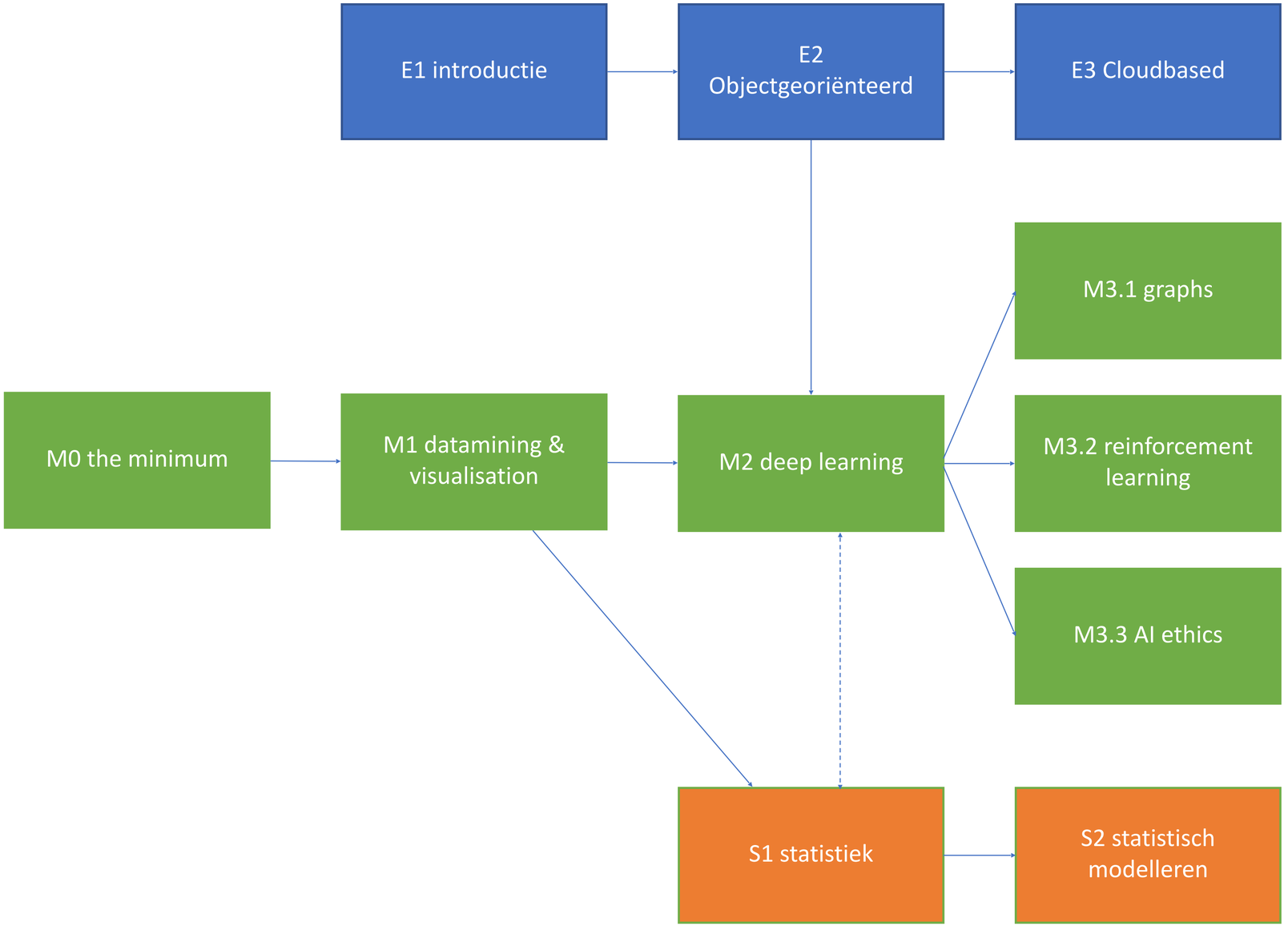
Deep Learning 1

Raoul Grouls, 25 april 2022

Hello world

- Raoul Grouls, data alchemist, https://the-pttrn.nl/
- Building AI solutions from strategy to deployment
- Specialised in Deep Learning & Responsible AI strategy

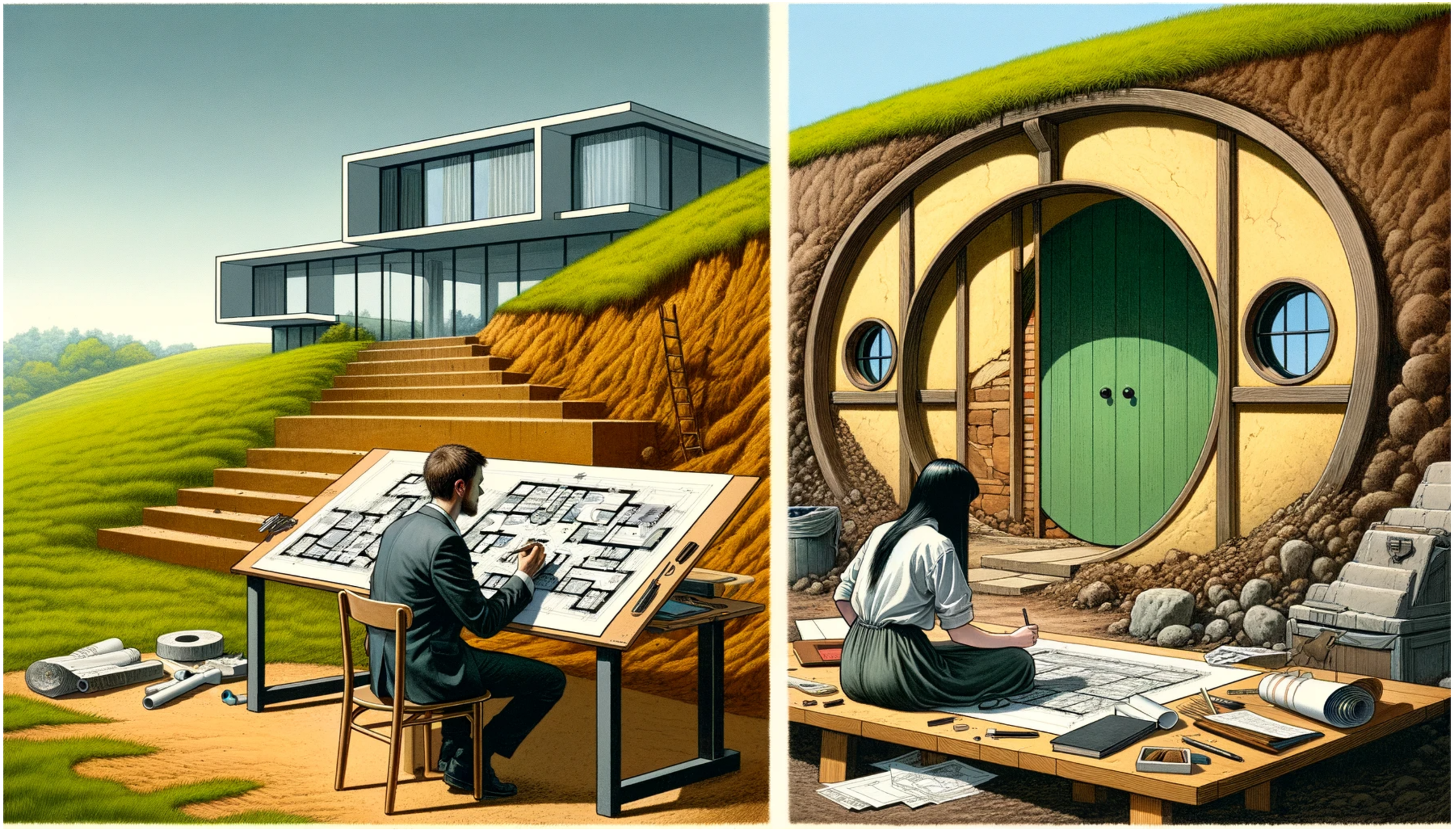
E: Machine learning / Engineering
M: Machine learning / Modelleren
S: Statistiek



What is intelligence?

Exercise:

- 3 minute break out
- Find a definition of intelligence



Clarifying some terminology

This is an hierarchical list, where every item is a subset of the preceding item.

1. **Automation**: Autonomous agent executing fixed rules
2. **Artificial Intelligence**: Autonomous agent optimizing an objective (including Expert Systems, Fuzzy Logic, Symbolic AI, Constraint Satisfaction Problems, Cellular Automata, Genetic Algorithms and Swarm Intelligence)
3. **Machine Learning**: Learning from data. This includes both gradient-based and non-gradient-based methods. (linear regression, random forests, Support Vector Machines)
4. **Deep Learning**: Complex learning from data, gradient-based and with backpropagation (CNN, RNN, Transformers)

What is deep learning?

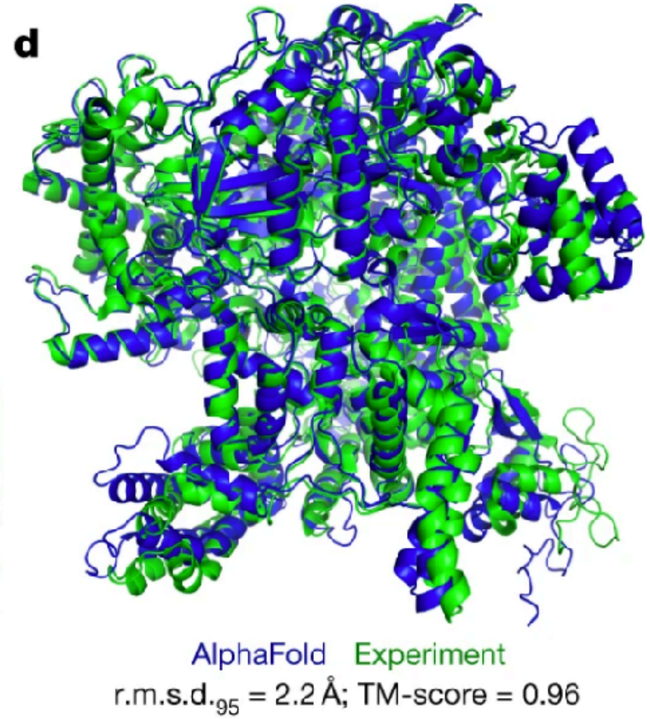
A machine learning technique that uses **multiple layers of non-linear transformations**

complex learning techniques using gradient-based optimization and backpropagation to process large amounts of data effectively

A technique to solve problems by providing data, and objective and often some sort of label, letting the computer find the solution using neural networks.

A very short history of AI

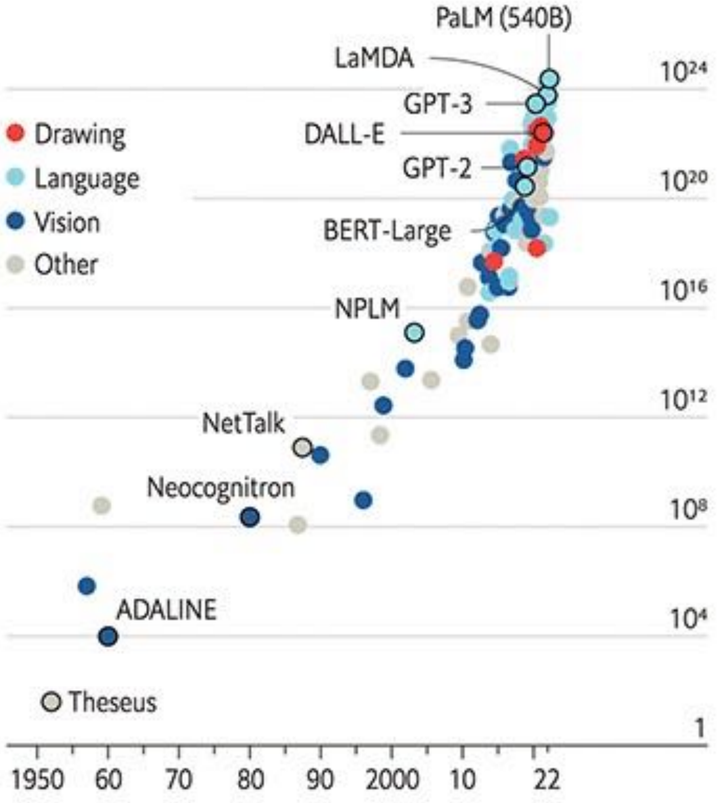
- 1950 Mathematics of Neural Networks invented
- 1990-2010 Support Vector Machines, Random Forest
- 2010 Deep learning
- 2015 Graph Convolutional Networks
- 2017 Attention mechanisms
- 2021 Protein folding
- 2023 chatGPT



The blessings of scale

AI training runs, estimated computing resources used

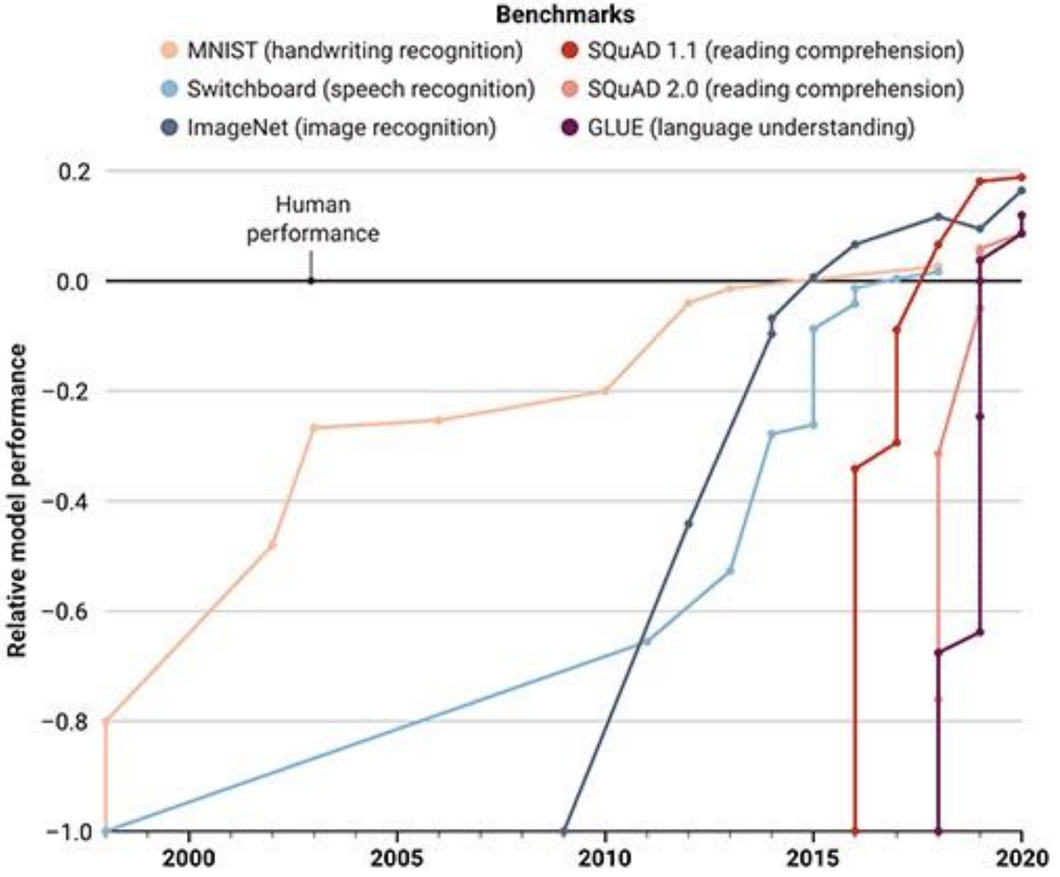
Floating-point operations, selected systems, by type, log scale



Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

Quick learners

The speed at which artificial intelligence models master benchmarks and surpass human baselines is accelerating. But they often fall short in the real world.



(GRAPHIC) K. FRANKLIN/*SCIENCE*; (DATA) D. KIELA *ET AL.*, DYNABENCH: RETHINKING BENCHMARKING IN NLP, DOI:10.48550/ARXIV.2104.14337

Complexity 1990 - SVM

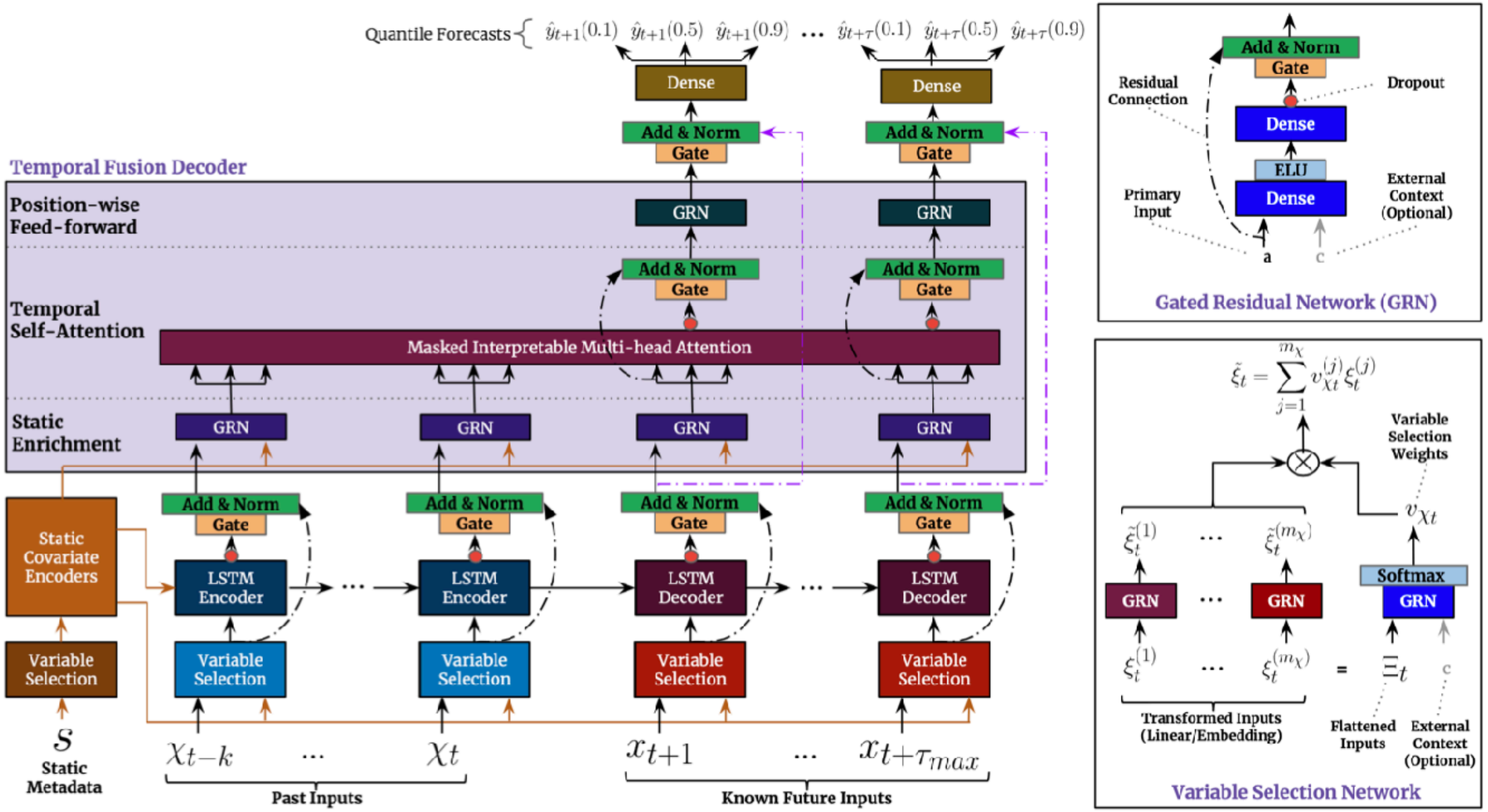
$$max_a \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j y_i y_j K(x_i, x_j)$$

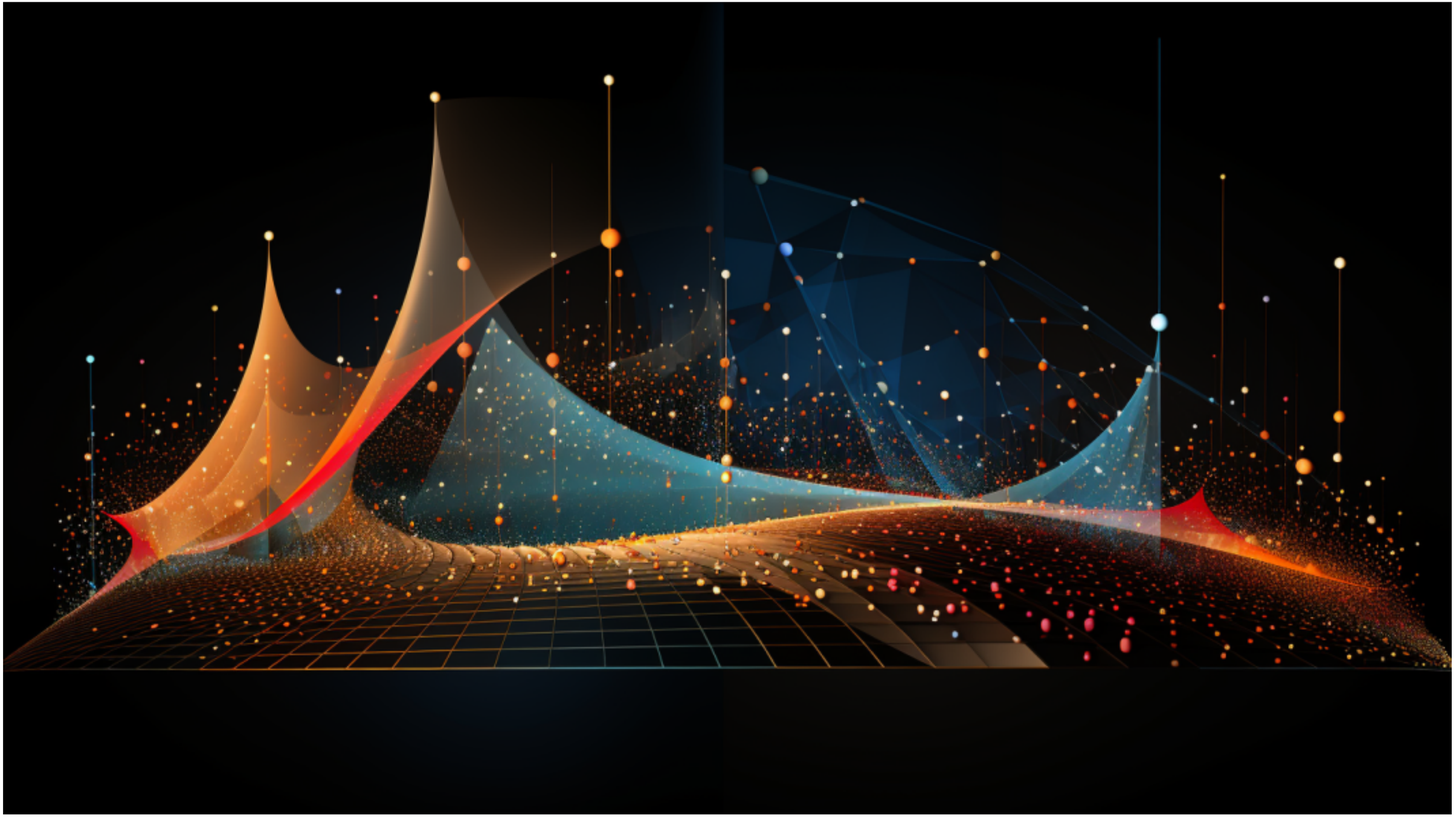
$$K(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2)$$

$$J(w, b) = \frac{1}{2} w^T w + C \sum_{i=1}^{m} max(0, 1 - t^i(w^T x^i + b))$$

m	number of observations
a	counter for errors
x	observations
y	labels
γ, C	hyperparameters
w, b	weights we need to learn

Complexity 2021 - TFT





Linear models

Even simpler models are called “retarded”

Linear models are one of the simplest models available. Their advantages are:

- The simplicity helps us to avoid overfitting
- They are fast and we don’t have to worry about getting stuck in a local minimum.
- They are ideal as a baselinemodel

Linear models

The basic mathematical shape of a linear model is:

$$Y = WX + b$$

- $Y = \{y_1, \ldots, y_m\}$ are labels, so we have m labels.
- Every label Y_i has corresponding features $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$, so we have n features.
- We can store the observations in an (m, n) matrix X
- We can store n weights for every feature in a matrix W
- b is an additional bias weight

The matrix notation is concise. We could write this out in full as:

$$y_i = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b$$

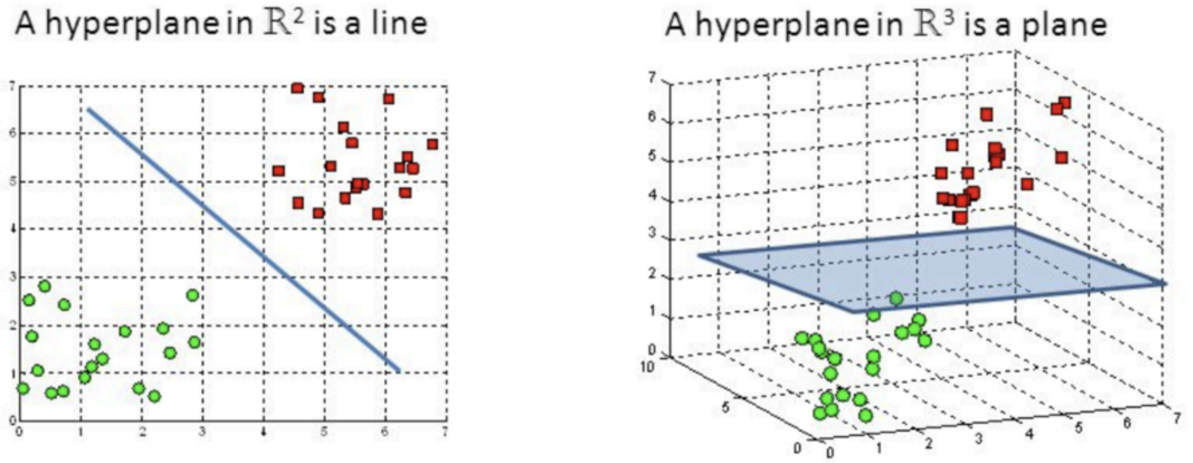
Hyperplanes

It's a bird… It's a plane…

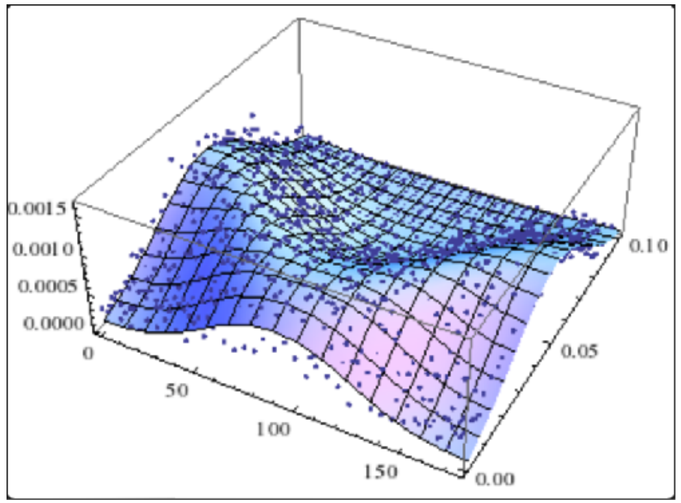
- For classification, our outcome are discrete classes (e.g. "yes" or "no")
- For regression, our outcome is a real number (e.g. 1.4 or 26.834)

We can use a linear model for both cases. We call these models "hyperplanes": they have one dimension less than the ambient space.

With classification, we want the data to be separated by the hyperplane. With regression, we want to points to get as close as possible to the hyperplane



Hyperplane for classification



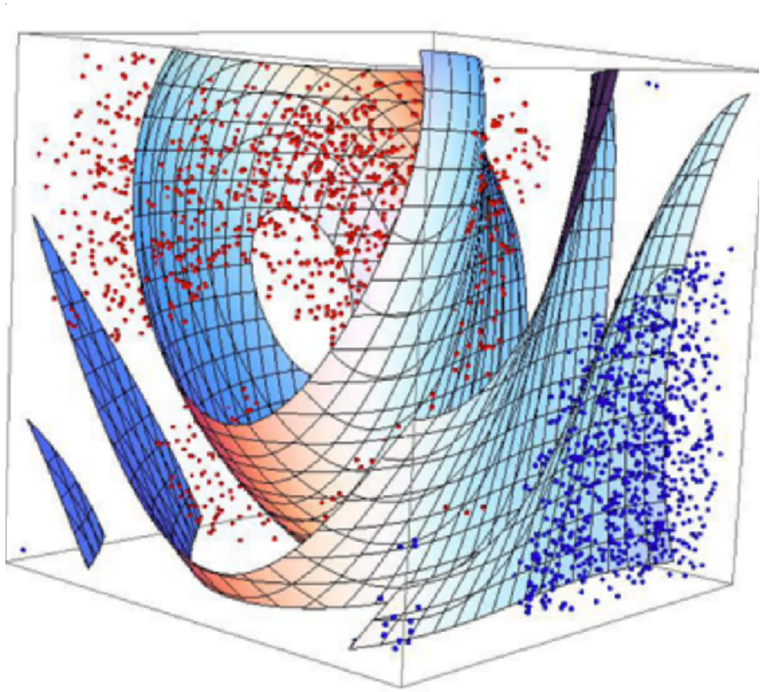
Hyperplane for regression

Non-linear models

A lot of data is non-linear. This means we would need a curved hyperplane.

One trick to do this is the kernel-trick, which is commonly used with Support Vector Machines, which we touched upon in the short history.

Deep learning has found another trick, we will look at the labs to see how this works.



Training Neural networks

Data	$X = \{x_1, \ldots, x_n\}, y = \{y_1, \ldots, y_n\}$
Trainable Weights	W, b
Predict	$\hat{y} = \sigma(f(X)) \qquad f(X) = WX + b$
Loss	$Loss(y, \hat{y})$
Optimize	$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$

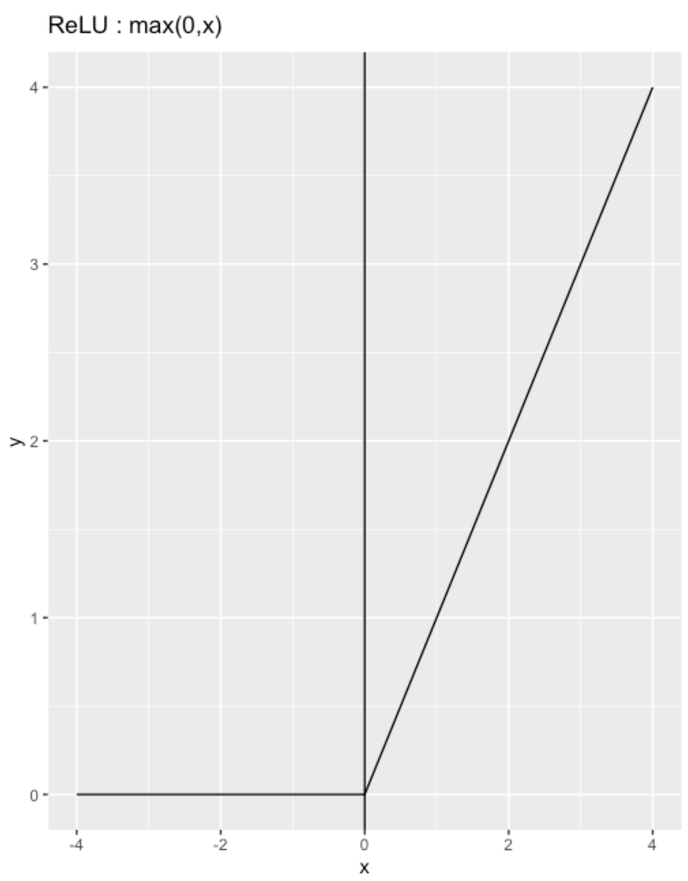
Training Neural networks

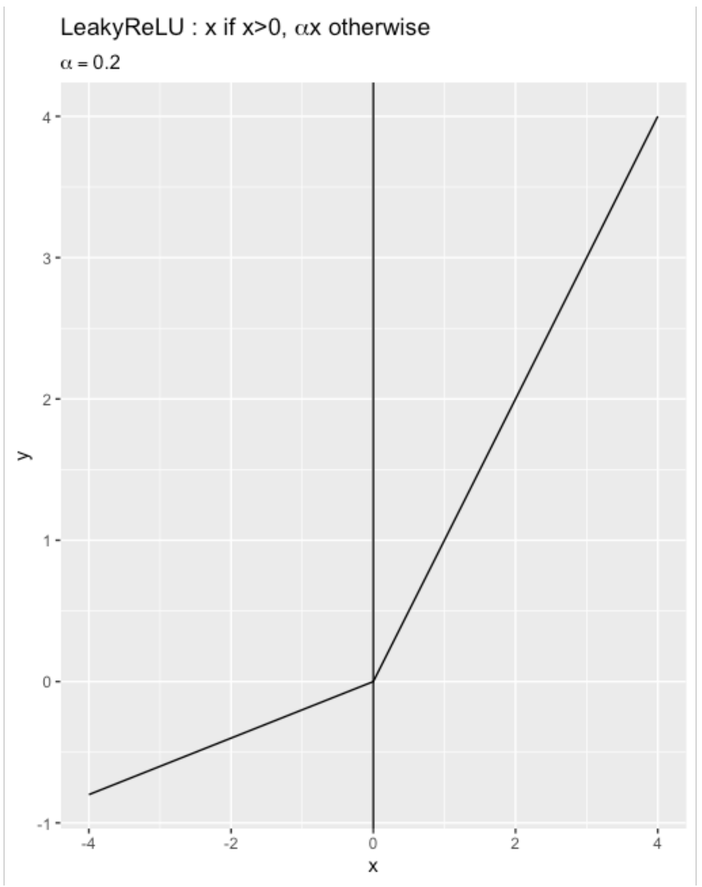
$$X \rightarrow \sigma(f(X)) \rightarrow Loss(y, \hat{y})$$

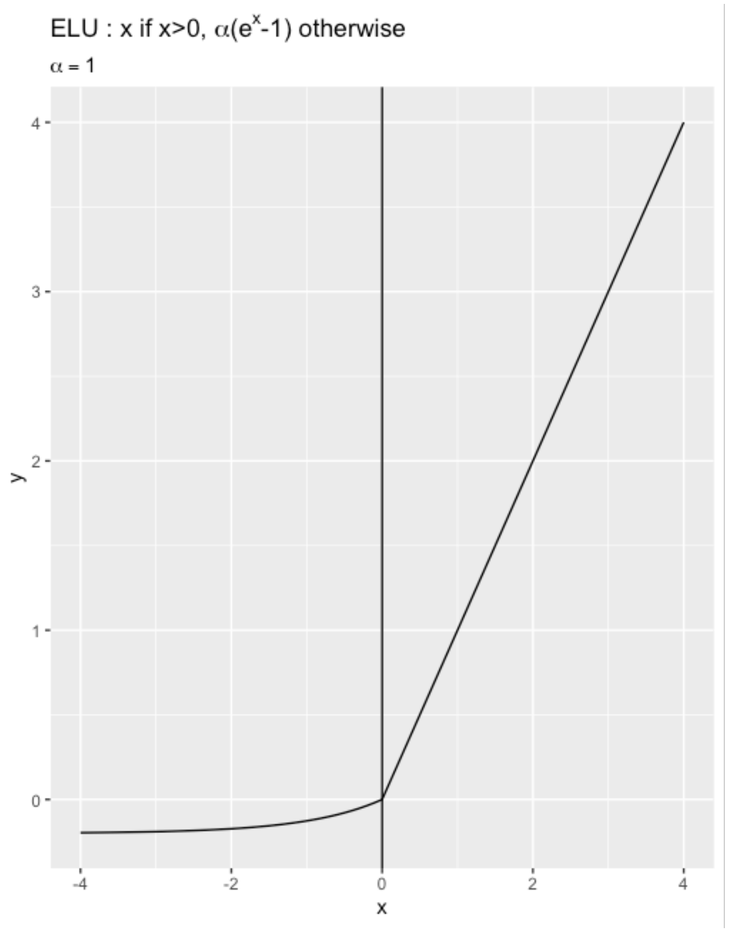
$$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$$

Activations

non-linearities







Code style standards

- See references/codestyle

Topic	Testing a Concept	Proof of Concept	Product	Deployment
Prefer pathlib.Path over os.path	💡	🎖️	🎖️	🎖️
Prefer loguru over print	💡	🎖️	🎖️	🎖️
Use pydantic for all settings	💡	🎖️	🎖️	🎖️
pyproject.toml for dependencies	💡	🎖️	🎖️	🎖️
Use cookiecutters	💡	🎖️	🎖️	🎖️
Git	💡	🎖️	🎖️	🎖️
Use formatters and linting	💡	🎖️	🎖️	🎖️
Use typehinting	🐌	💡	🎖️	🎖️
Makefiles	🐌	💡	🎖️	🎖️
Encapsulation	🐌	💡	🎖️	🎖️
Open-Closed Principle	🐌	💡	🎖️	🎖️
single responsibility	🐌	💡	🎖️	🎖️
Abstract classes (ABC, Protocol)	🐌	🐌	💡	🎖️
Write tests (pytest)	🐌	🐌	💡	🎖️