Timeline

60min [09:00 - 10:00] **Intro & Fundamentals**
15min [10:00 - 10:15] Break
60min [10:15 - 11:15] **Supervised learning**
15min [11:15 - 11:30] Break
30min [11:30 - 12:00] **Unsupervised learning**
90min [12:00 - 13:30] Lunch
45min [13:30 - 14:15] **Pytorch & Keras overview**
15min [14:15 - 14:30] **Visual clutter classifier**
15min [14:30 - 14:45] **Fitts law regressor**
15min [14:45 - 15:00] Break
30min [15:00 - 15:30] **Clustering of UI wireframes**

SAARBRÜCKEN

6TH SUMMER SCHOOL ON COMPUTATIONAL INTERACTION

INFERENCE, OPTIMIZATION AND MODELING FOR THE ENGINEERING OF INTERACTIVE SYSTEMS | 13 - 18 JUNE 2022

Deep Learning for Human–Computer Interaction

Session 1: Intro & Fundamentals

Luis Leiva & **Bereket Yilma**
University of Luxembourg

UNIVERSITÉ DU LUXEMBOURG

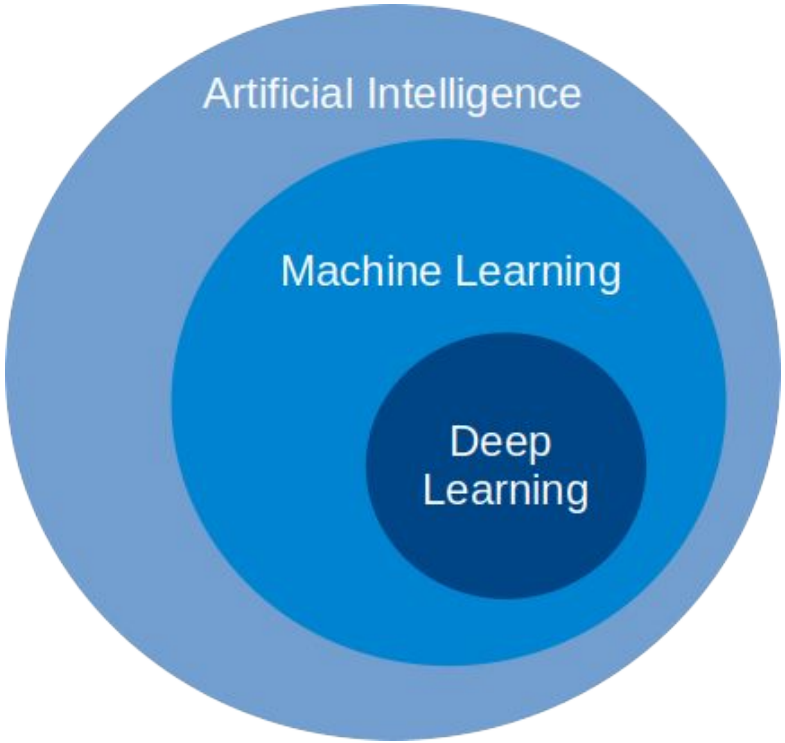
Learning outcomes

After this lecture you will be able to:

- Understand ML and DL foundations
- Recognize key building blocks in DL models
- Formulate HCI problems from a ML/DL perspective

Introduction

Where we are



Artificial Intelligence

Try to create machines that can simulate human behavior

Machine Learning

Automatically learn from data without explicit programming

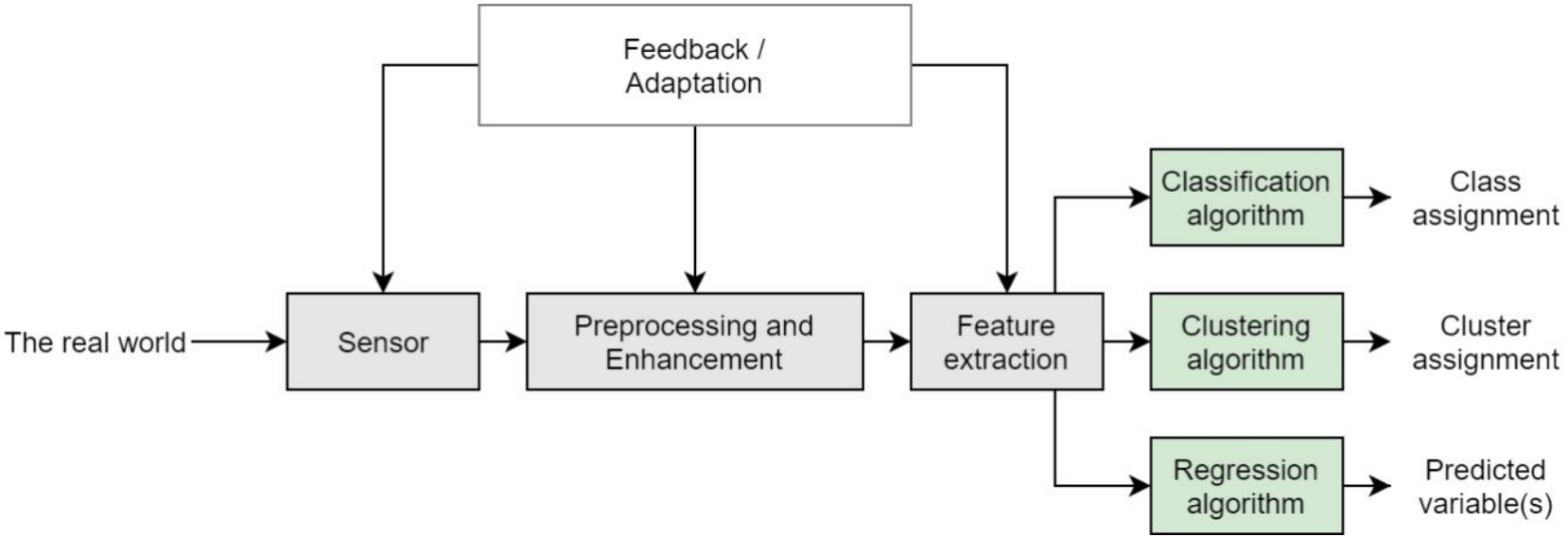
Deep Learning

Machine learning with neural networks

Machine Learning

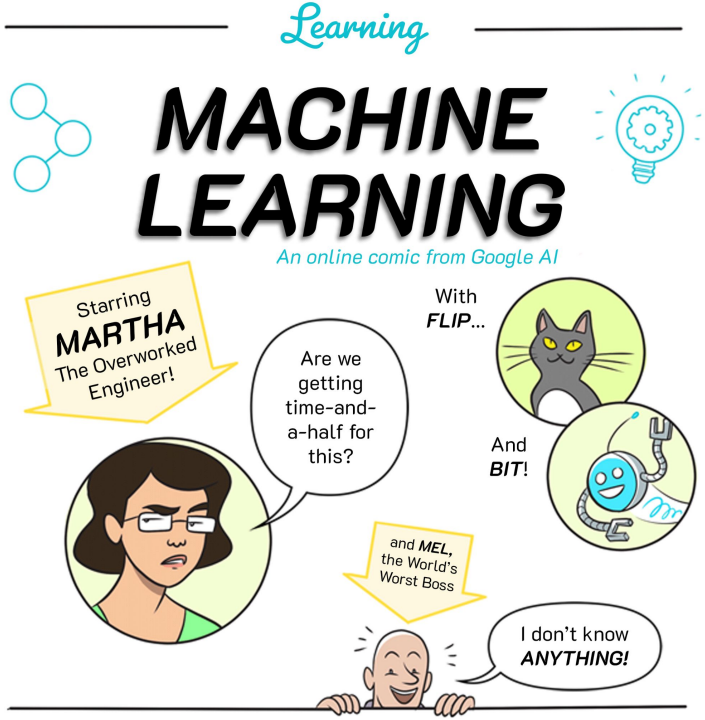
Pattern recognition and inference

Often guided by examples

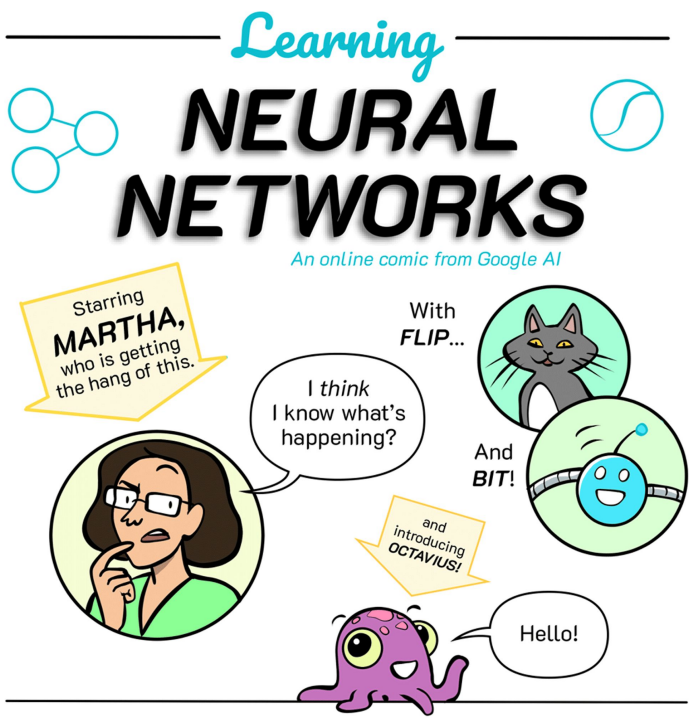


https://huspi.com/blog-open/pattern-recognition-in-machine-learning/

ML & DL preliminaries



https://cloud.google.com/products/ai/ml-comic-1/



https://cloud.google.com/products/ai/ml-comic-2/

Why is it called DL?

Input layer

One or more hidden layers

Output layer

https://xkcd.com/1838/

Why DL *now*?

More **hardware**

TensorFlow

PYTORCH

theano

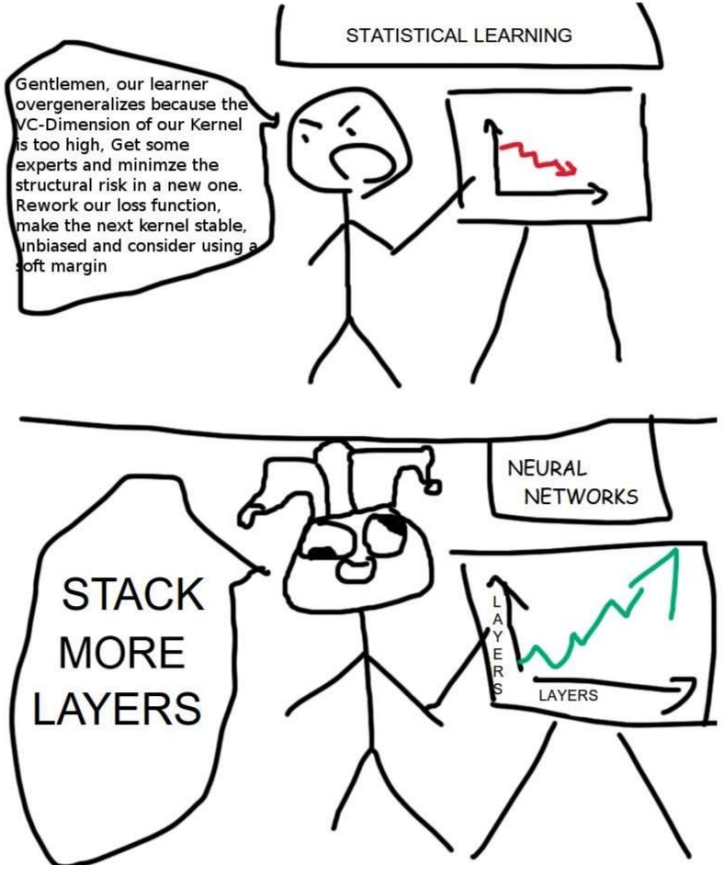
mxnet

Microsoft CNTK

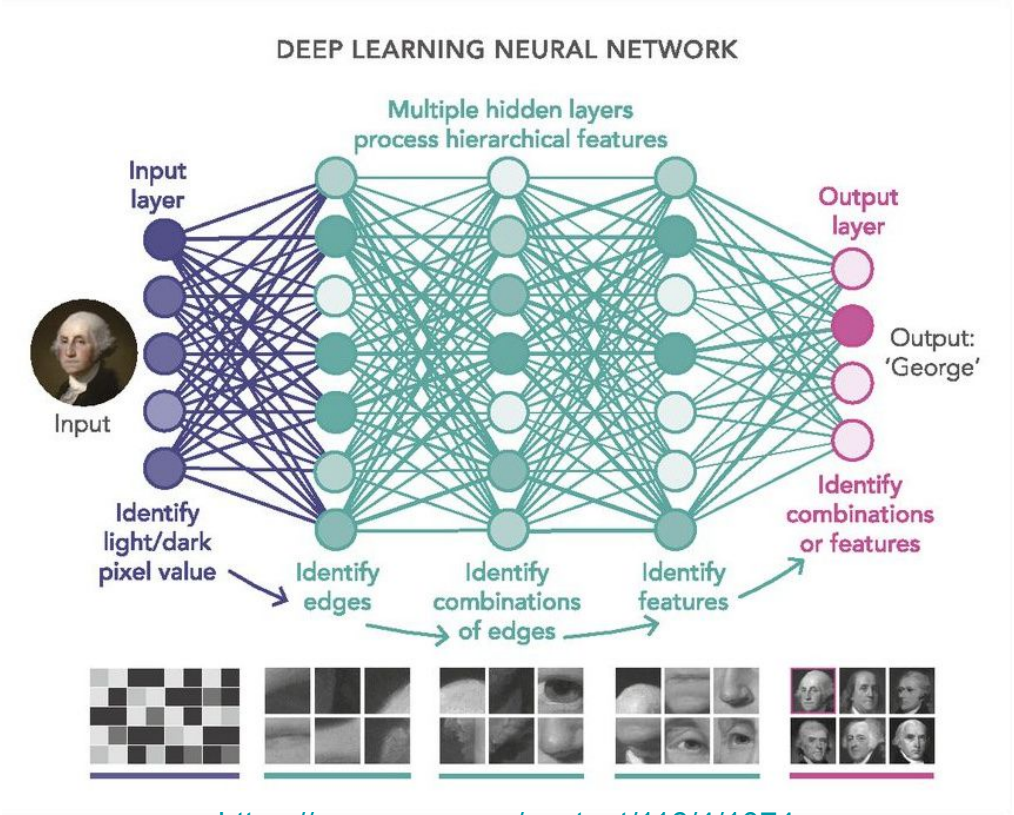
More **software**

More **data**

Why DL *now*?

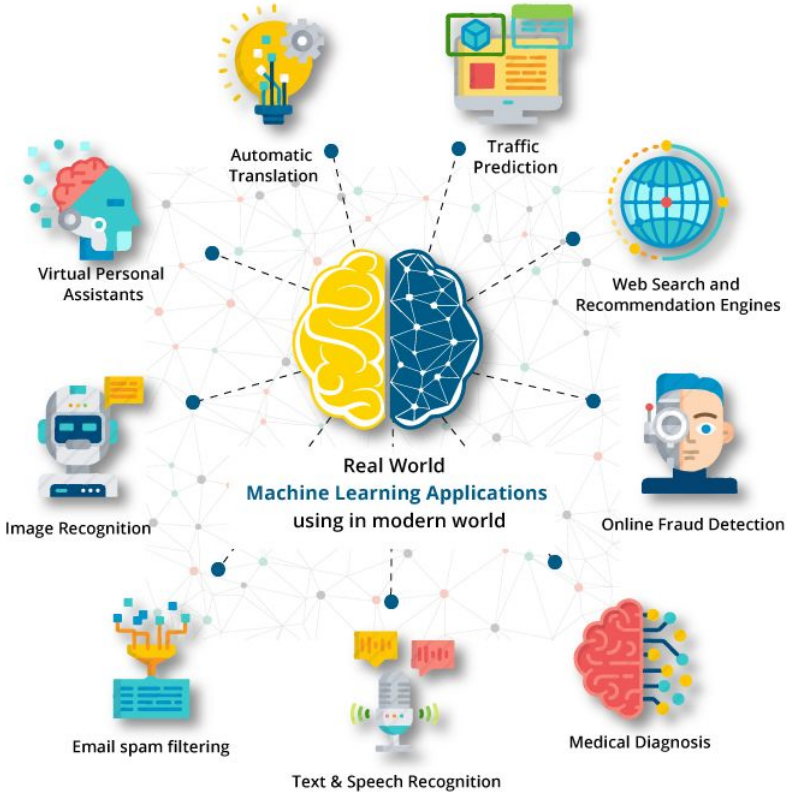

STATISTICAL LEARNING
Gentlemen, our learner overgeneralizes because the VC-Dimension of our Kernel is too high, Get some experts and minimze the structural risk in a new one. Rework our loss function, make the next kernel stable, unbiased and consider using a soft margin
NEURAL NETWORKS
STACK MORE LAYERS
LAYERS
LAYERS

Example



https://www.pnas.org/content/116/4/1074

Applications



Learning paradigms

Supervised Learning

Examples:
Classification
Regression

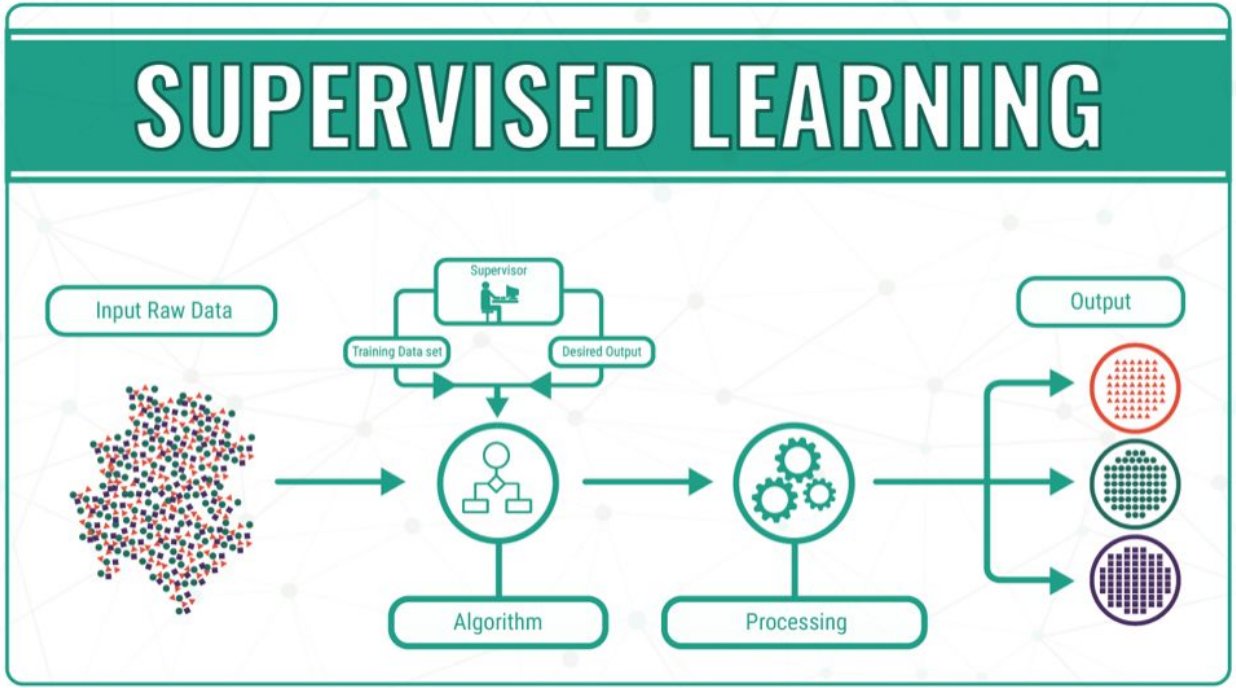
Unsupervised Learning

Examples:
Autoencoders
Clustering

Reinforcement Learning

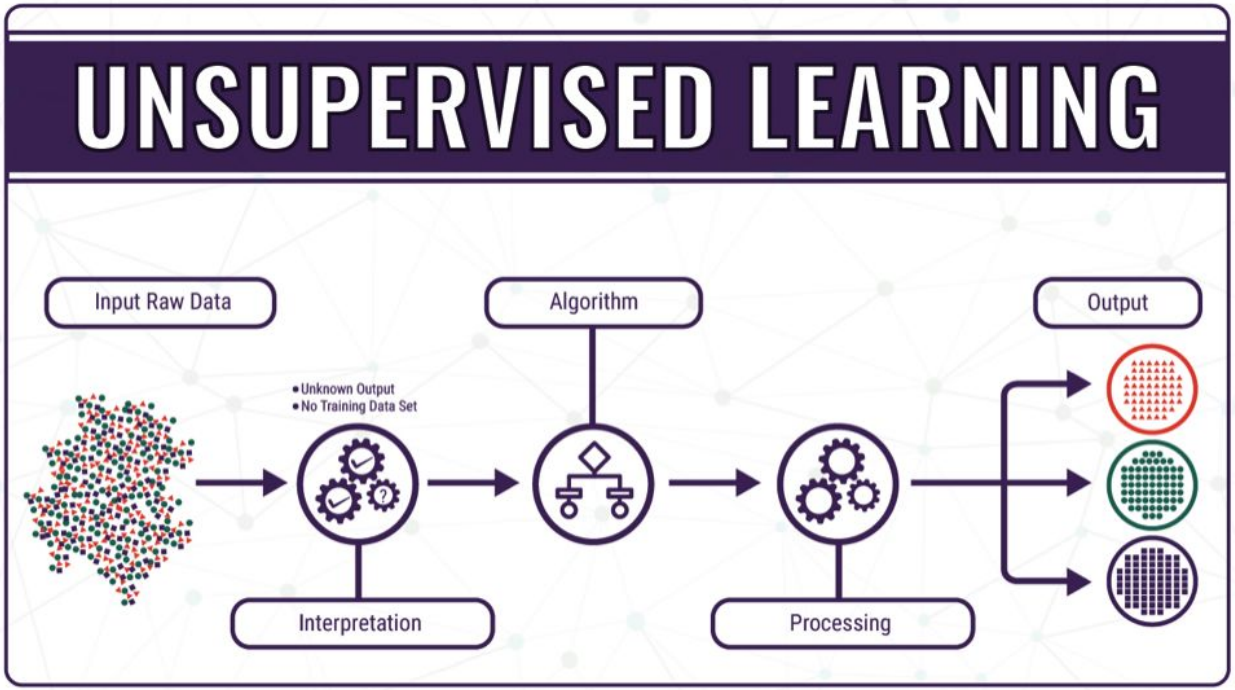
Examples:
Navigation
Planning

Learning paradigms



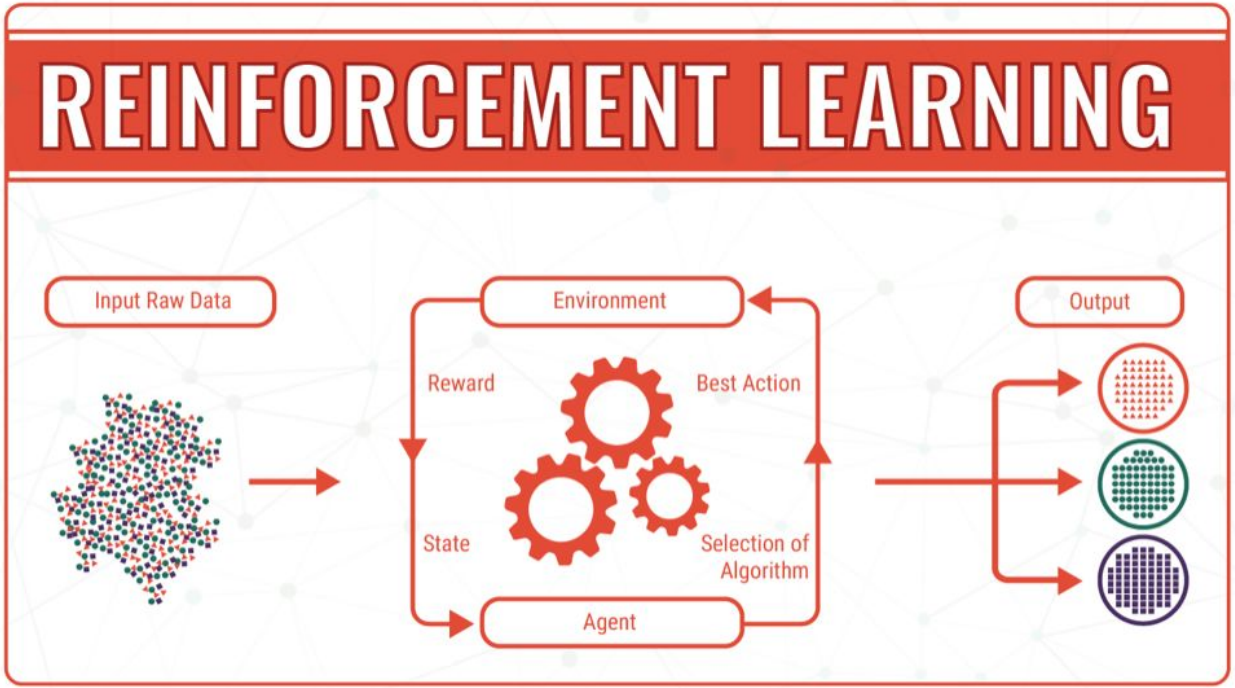
https://www.linkedin.com/pulse/machine-learning-explained-understanding-supervised-ronald-van-loon

Learning paradigms



https://www.linkedin.com/pulse/machine-learning-explained-understanding-supervised-ronald-van-loon

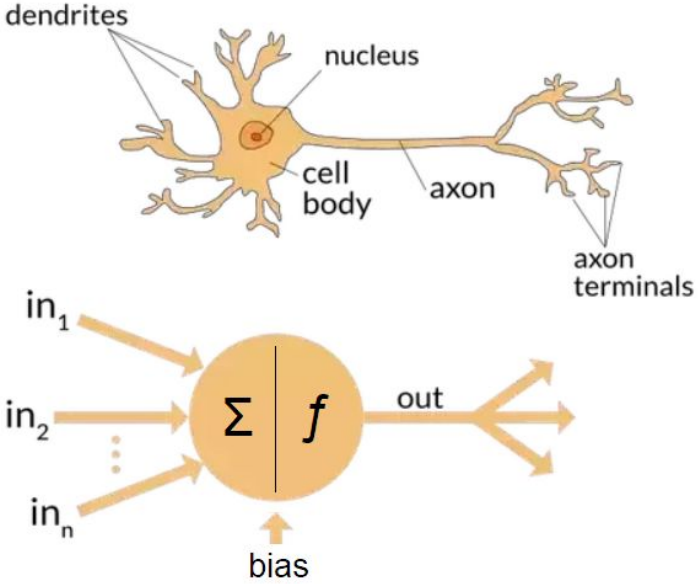
Learning paradigms

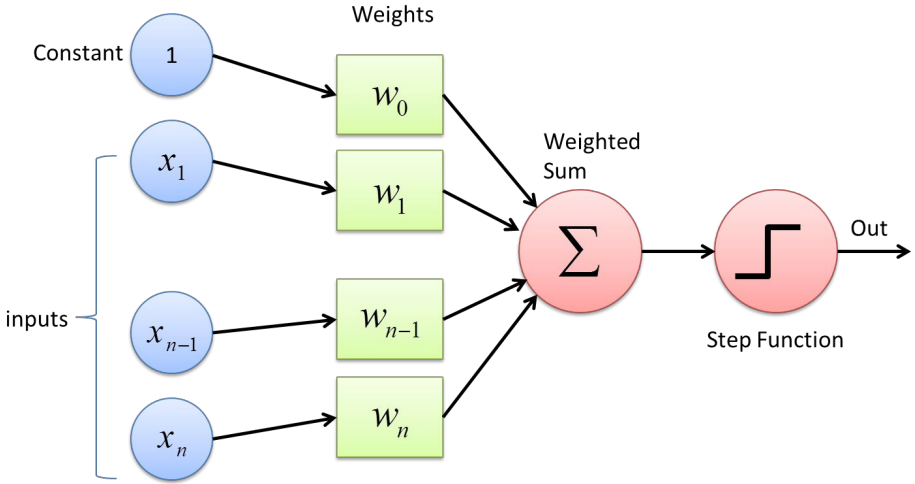


https://www.linkedin.com/pulse/machine-learning-explained-understanding-supervised-ronald-van-loon

Foundations

Perceptron algorithm





https://towardsdatascience.com/a8b46db828b7

https://towardsdatascience.com/626217814f53

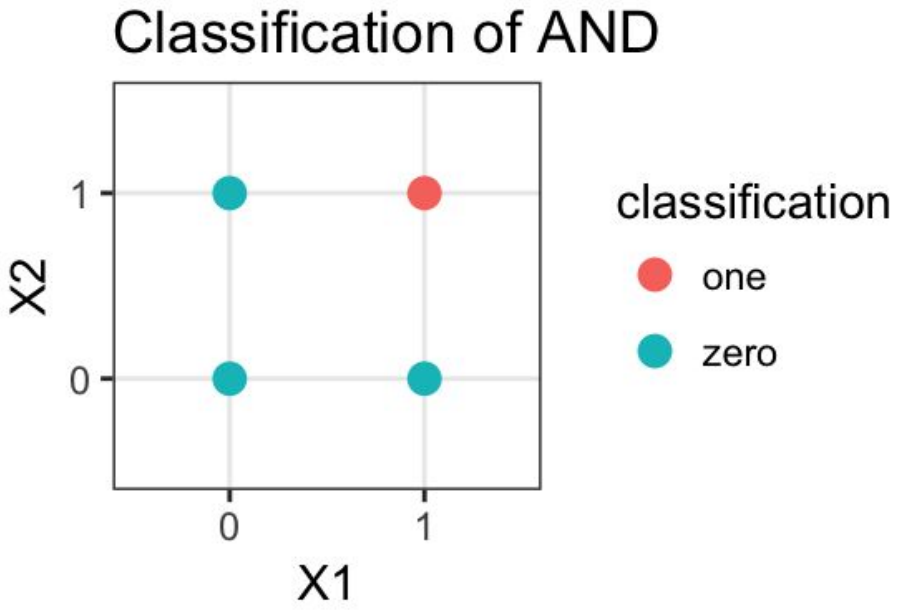
Perceptron algorithm

1. Initialize weights **w**
2. For *N* iterations (or until convergence) do:
   2.1. Process each instance f(*z*) = f($\mathbf{w}^T$ **x**)
   2.2. Update weights: w_i += [*y* - f(*z*)] x_i
3. Return weights

$$\hat{y} = f(z) = \begin{cases} 1 & w_0 + \sum_{i=1}^{n} x_i w_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{y} = f(z) = \begin{cases} 1 & \sum_{i=0}^{n} x_i w_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Perceptron algorithm



x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

Perceptron algorithm

Let $y = f(b, x_1, x_2)$ and $\mathbf{w} = (0,0,0)$.

Train $\mathbf{x} = (1,1,1)$; $y = 1$

$z = \mathbf{x} \cdot \mathbf{w}^T = 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 0 = 0 \rightarrow f(z) = 0$

$\mathbf{w} = (0,0,0) + (1 - 0) \cdot (1,1,1) = (1,1,1)$

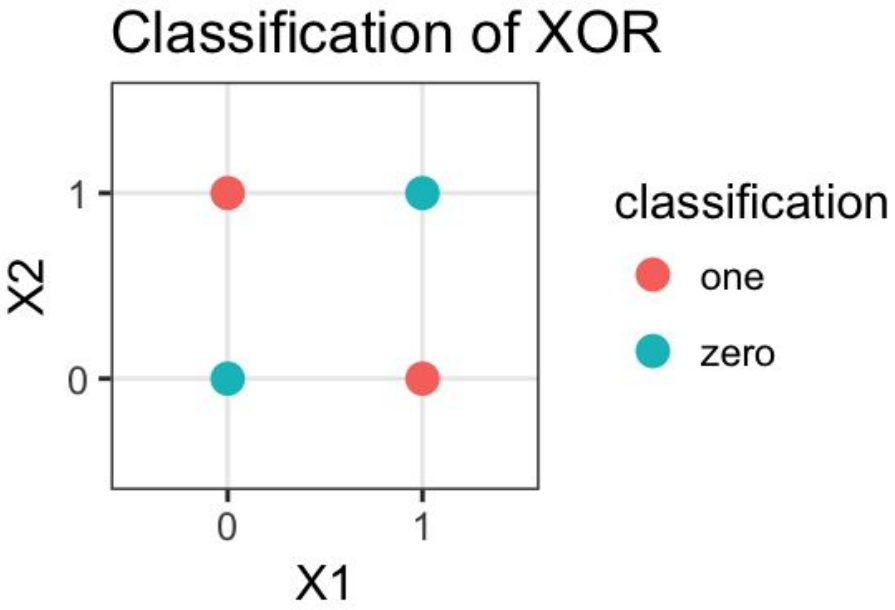
Train $\mathbf{x} = (1,1,0)$; $y = 0$

$z = \mathbf{x} \cdot \mathbf{w}^T = 1 \cdot 1 + 1 \cdot 1 + 0 \cdot 1 > 0 \rightarrow f(z) = 1$

$\mathbf{w} = (1,1,1) + (0 - 1) \cdot (1,1,0) = (0,0,1)$

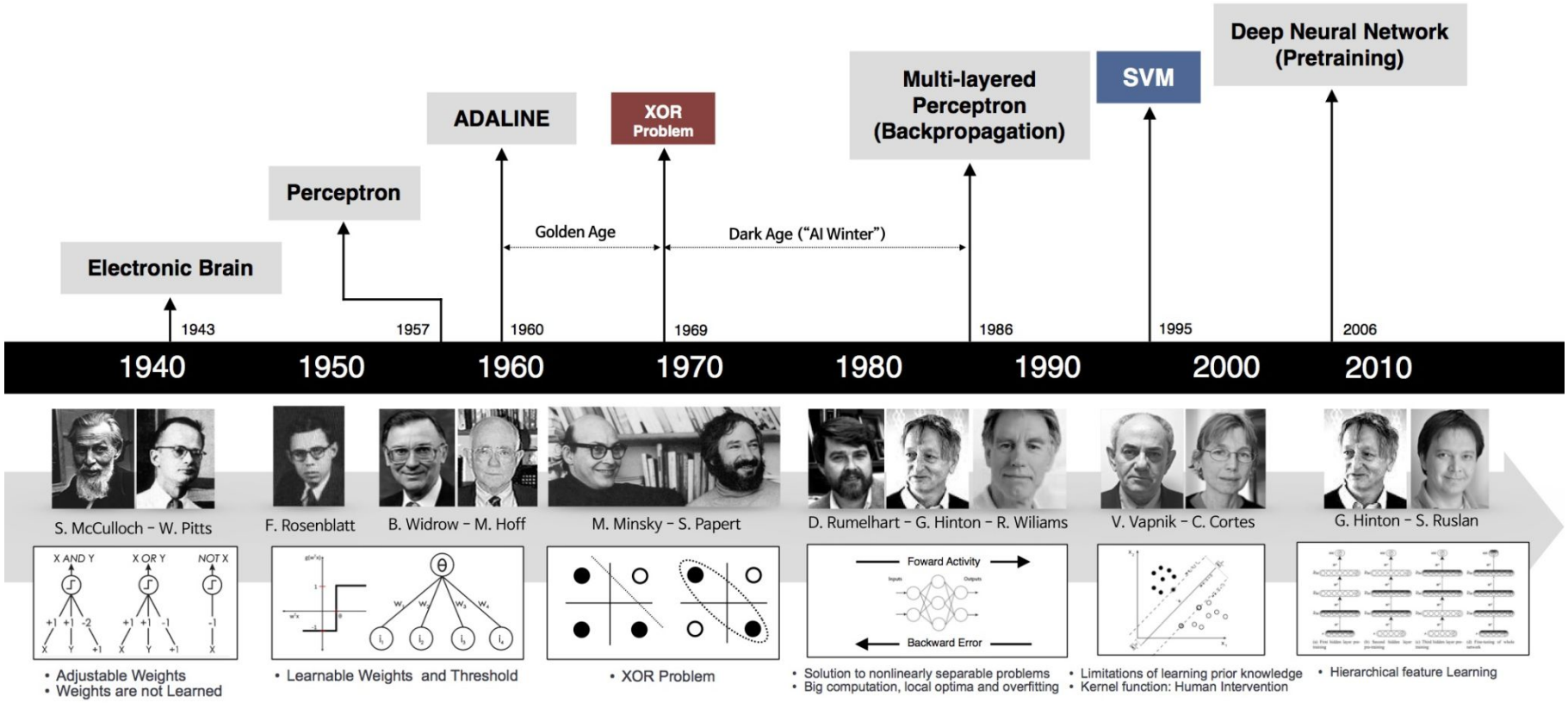
x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

Perceptron algorithm

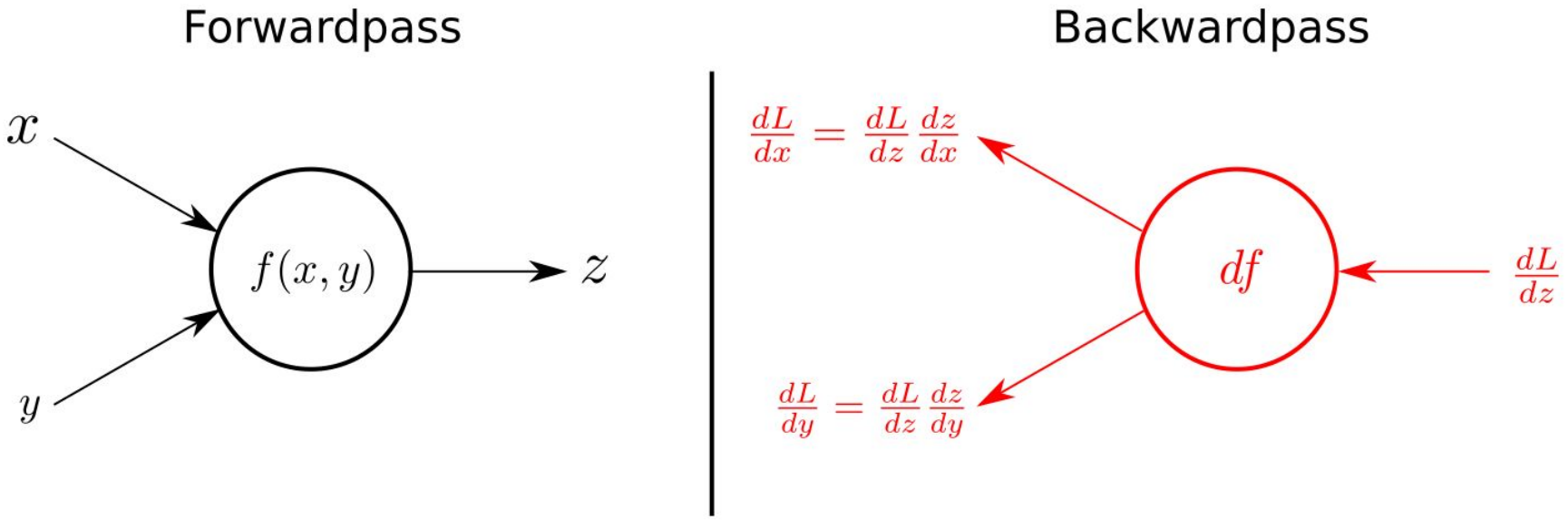


x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

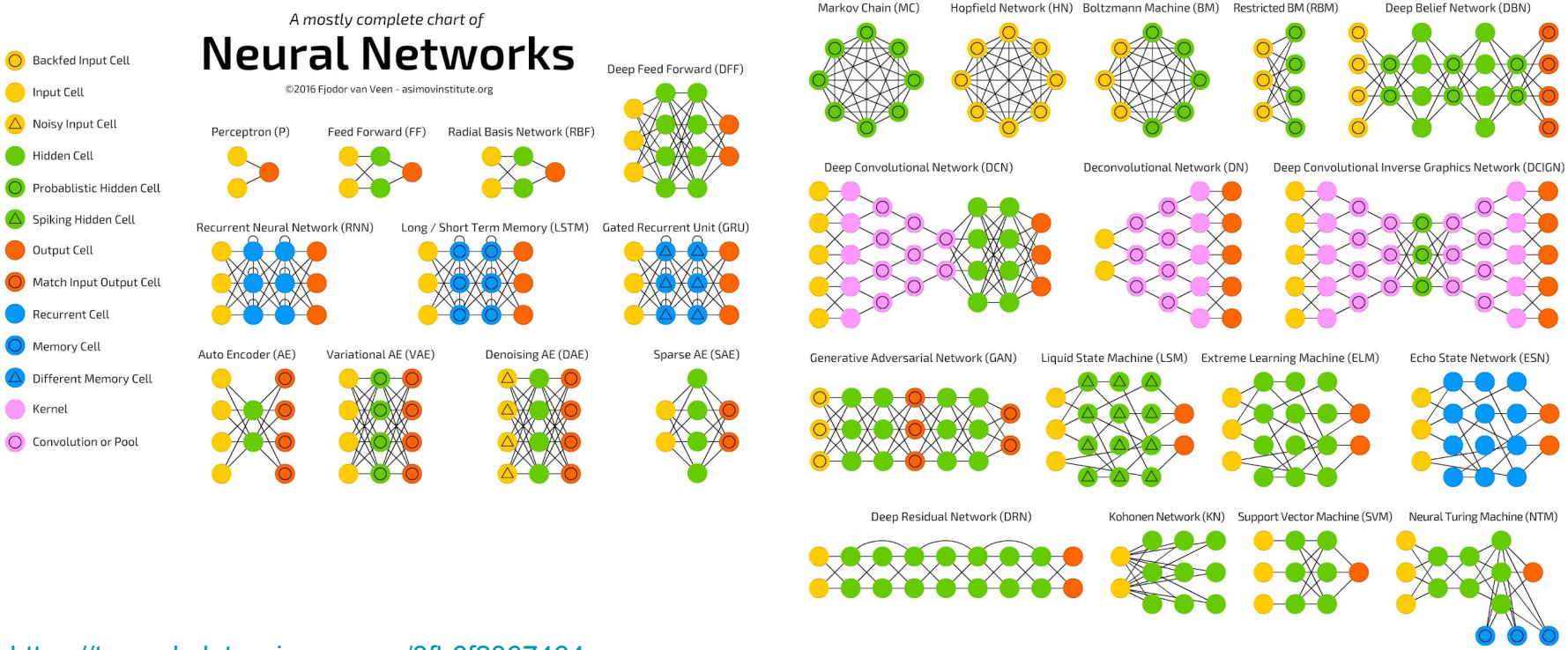
Timeline


Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
ADALINE
XOR Problem
Perceptron
Electronic Brain
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

Backprop

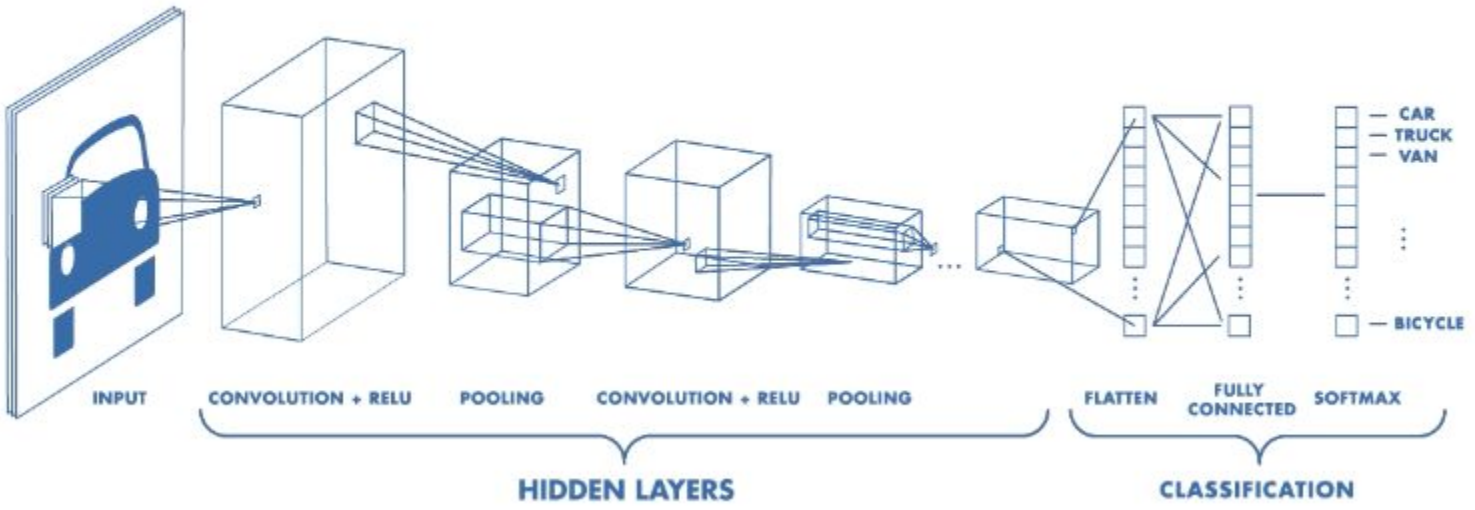

Forwardpass
Backwardpass
x
y
$f(x, y)$
z
$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$
$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$
df
$\frac{dL}{dz}$

Some architectures



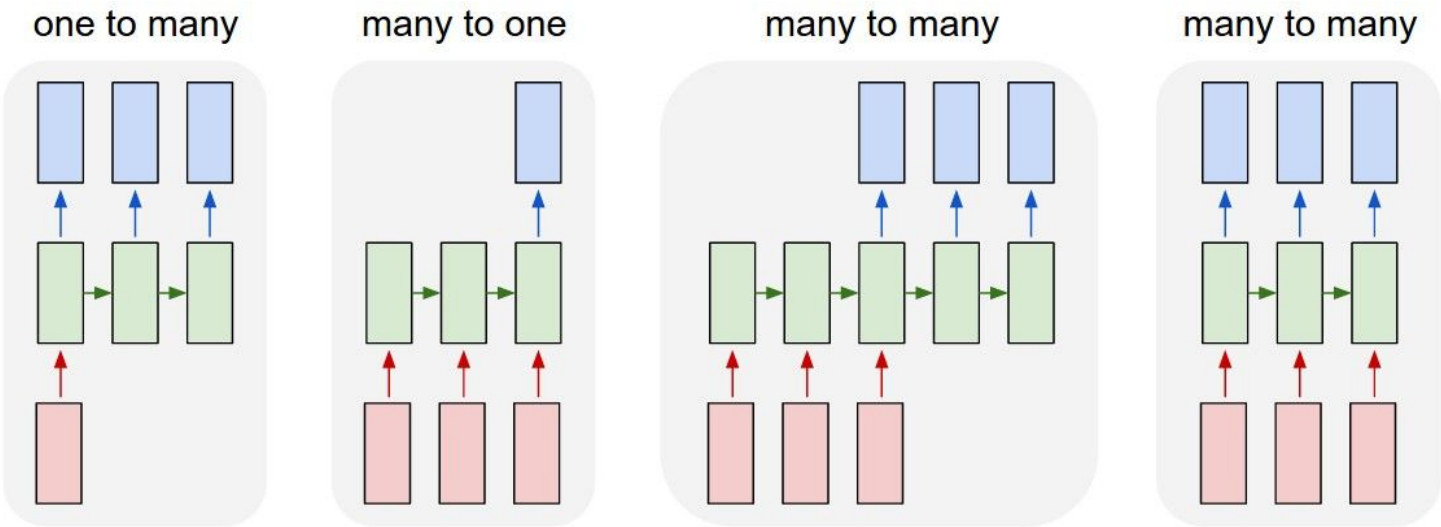
https://towardsdatascience.com/3fb6f2367464

Convolutional Neural Nets



https://towardsdatascience.com/3bd2b1164a53

Recurrent Neural Nets



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Building blocks

Layers

Activation functions

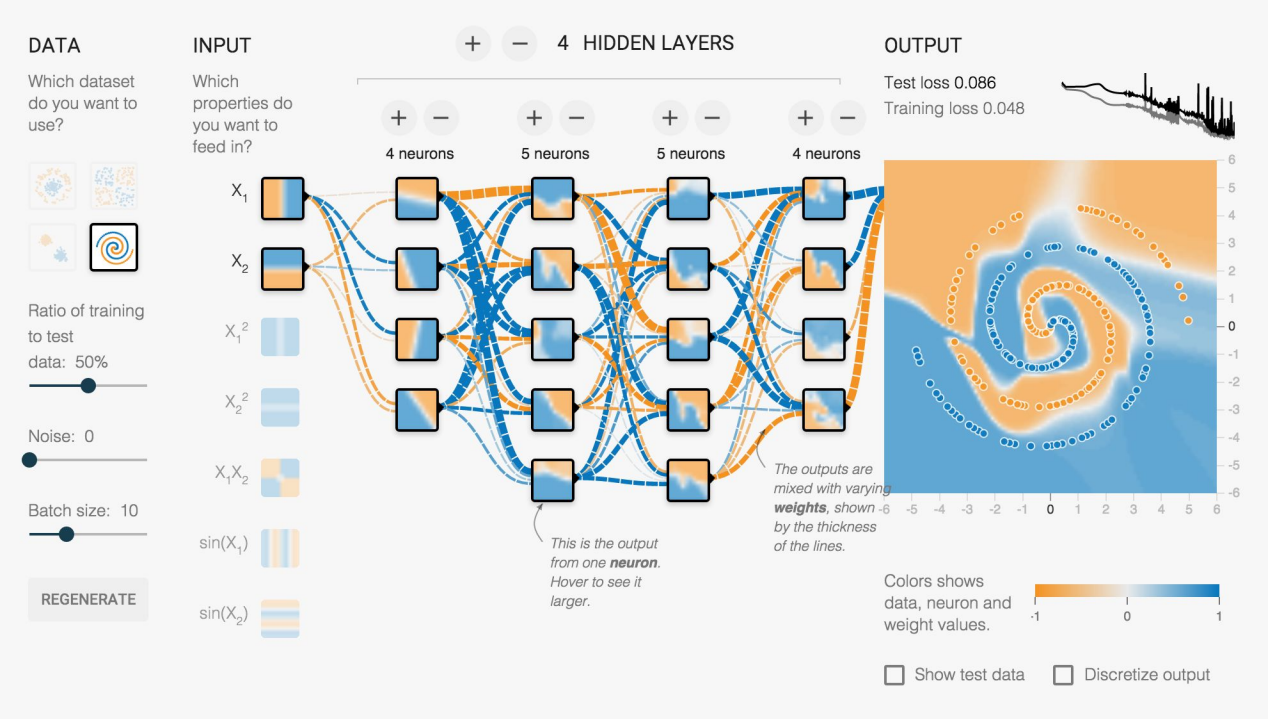
Loss functions

Optimizers

Regularizers

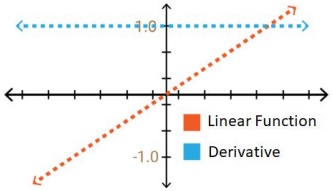
Evaluation metrics

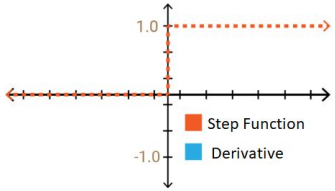
Layers

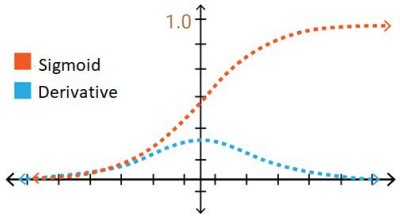


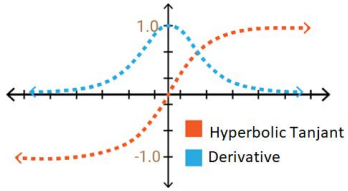
https://blogs.scientificamerican.com/sa-visual/unveiling-the-hidden-layers-of-deep-learning/

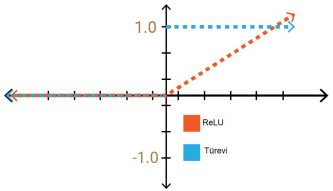
Activation functions

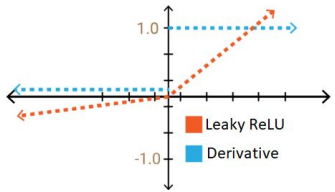


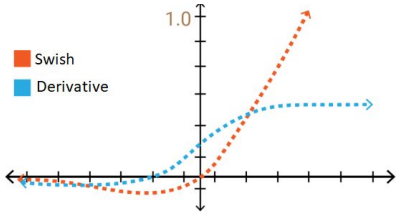










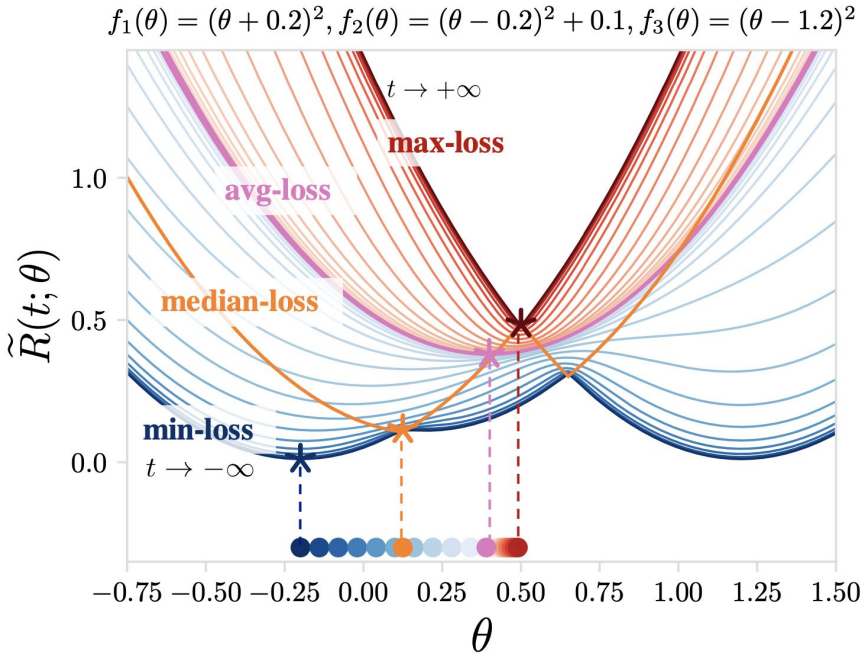


https://towardsdatascience.com/706ac4284c8a and https://en.wikipedia.org/wiki/Activation_function

Loss functions

Goal: Minimize empirical loss

$$J(\mathbf{W}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(\hat{y}_i, y_i)$$



https://pythonawesome.com/tilted-empirical-risk-minimization-in-python/

Loss functions for **classification** tasks

Binary Crossentropy loss for *C* = 2 classes:

$$\mathcal{L}(\widehat{y_i}, y_i) = -\frac{1}{N}\sum_i^N y_i \log \widehat{y_i} + (1 - y_i) \log (1 - \widehat{y_i})$$

Categorical Crossentropy loss for *C* > 2 classes:

$$\mathcal{L}(\widehat{y_i}, y_i) = -\frac{1}{N}\sum_i^N \sum_c^C \mathbb{1}_{y_i \in C_c} \log \widehat{y_i}$$

Note: $\widehat{y_i} = p(y_i)$

Loss functions for **regression** tasks

Mean Squared Error loss:

$$\mathcal{L}(\hat{y}_i, y_i) = (y_i - \hat{y}_i)^2$$

Mean Absolute Error loss:

$$\mathcal{L}(\hat{y}_i, y_i) = |y_i - \hat{y}_i|$$

Notes:

RMSE = √MSE

MAE ≤ RMSE

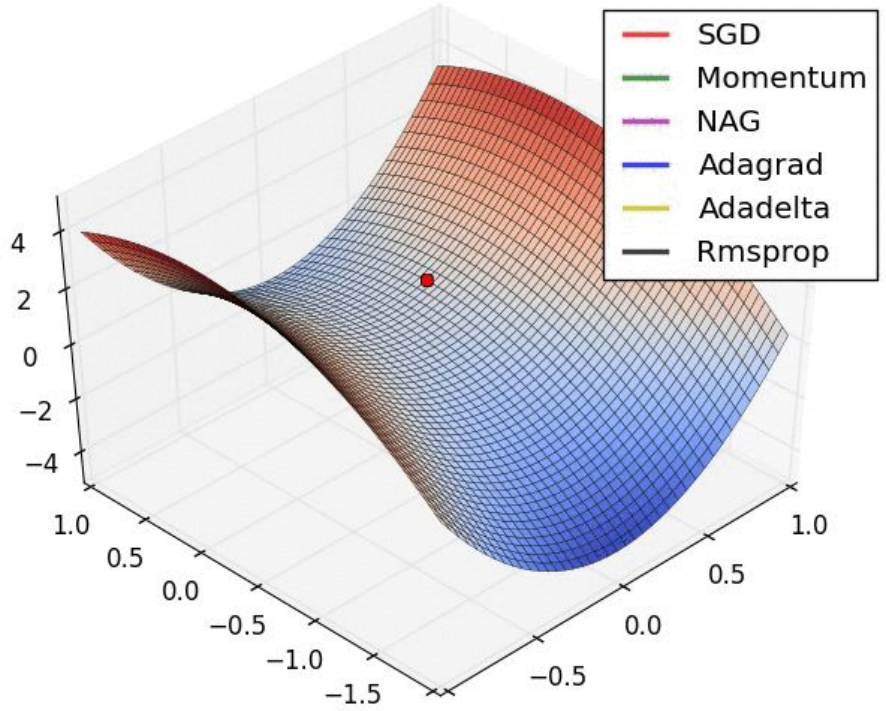
RMSE ≤ MAE · √n

Optimizers

Mostly based on Gradient Descent:

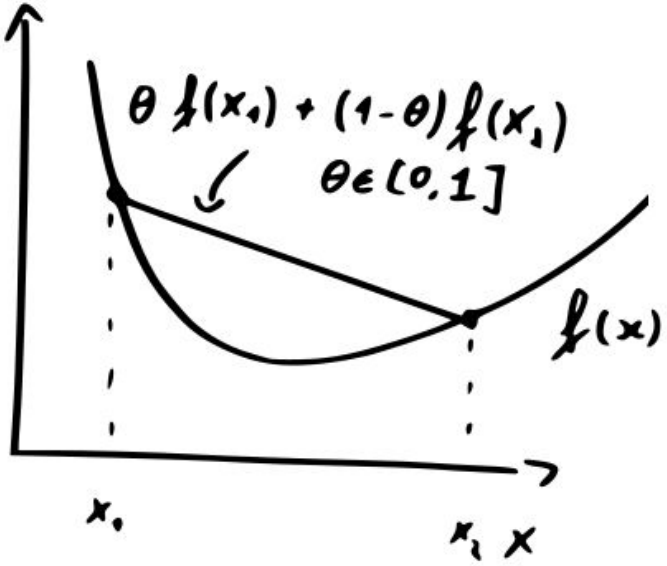
$\theta = \theta - \eta \cdot \nabla J(\mathbf{W}; \theta)$

- SGD
- RMSProp
- Adam
- AdaGrad
- AdaDelta

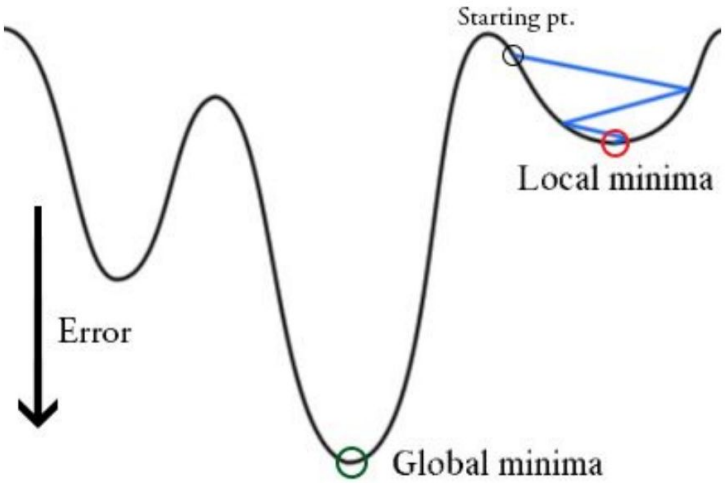


https://ruder.io/optimizing-gradient-descent/

Non-convex optimization



https://michielstock.github.io/posts/2018/2018-03-07-ConvexSummary/



https://www.cs.ubc.ca/labs/lci/mlrg/slides/non_convex_optimization.pdf

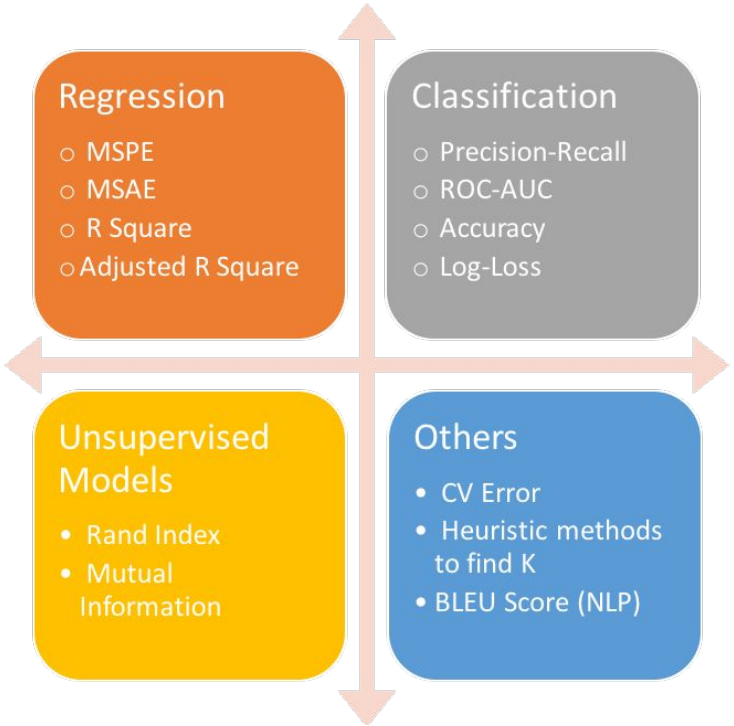
Regularizers

Goal: Improve model generalization

- Dropout*
- BatchNorm
- Early stopping*
- L1 & L2 penalty
- Adaptive learning rate*
- Data augmentation

* Very very useful, really

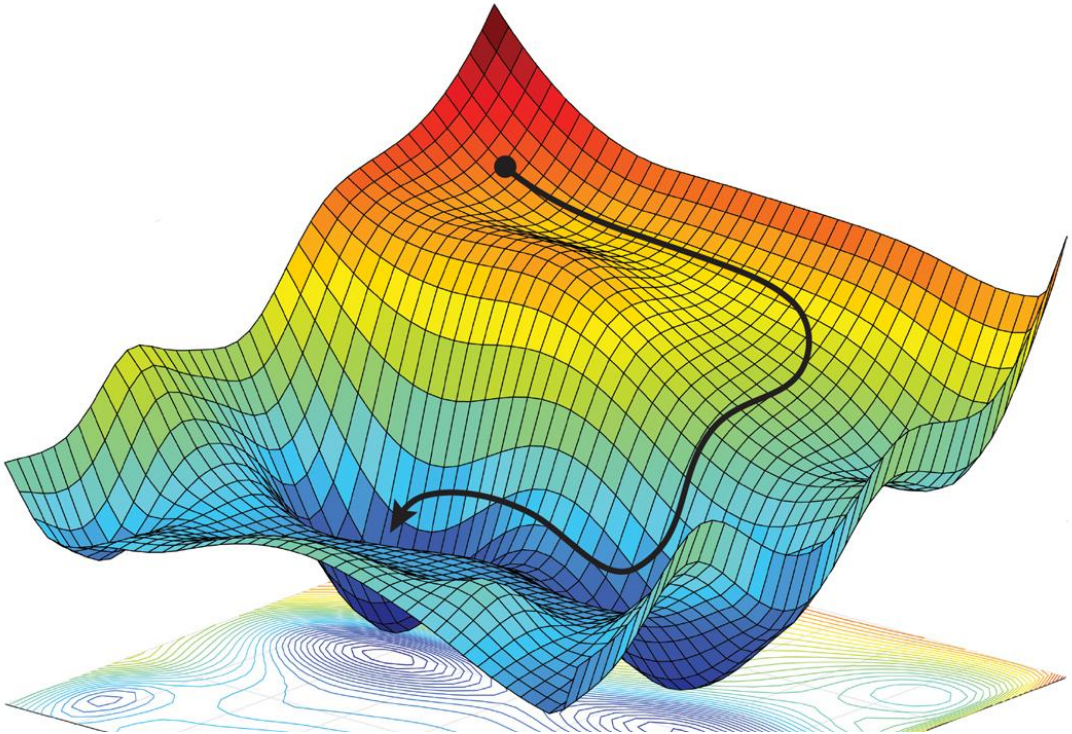
Evaluation metrics



https://medium.com/usf-msds/a99d7d7414e4

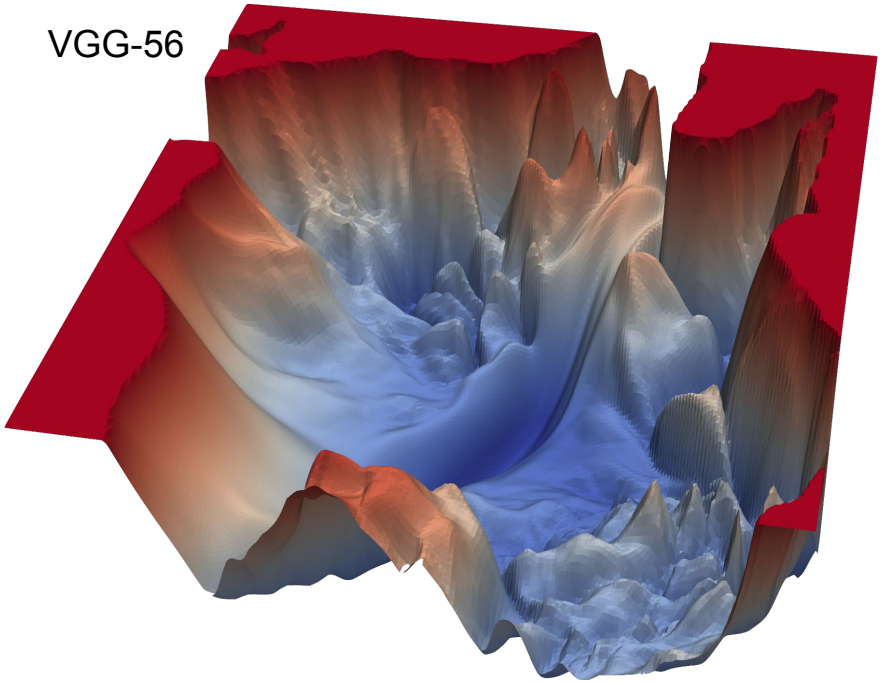
Key notes

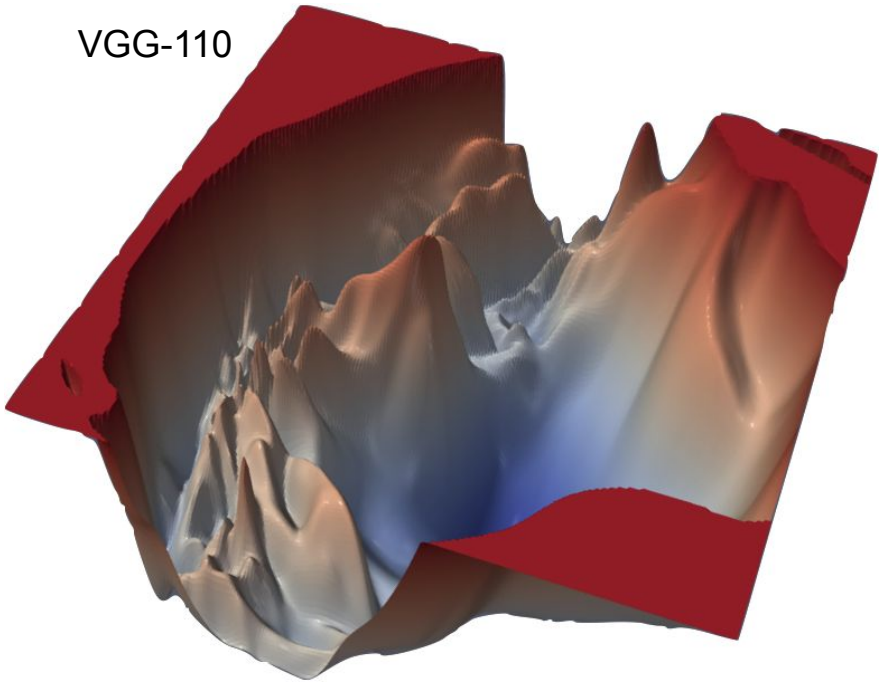
DL in practice



https://www.science.org/content/article/ai-researchers-allege-machine-learning-alchemy

Loss landscapes





https://www.cs.umd.edu/~tomg/projects/landscapes/

Generalization

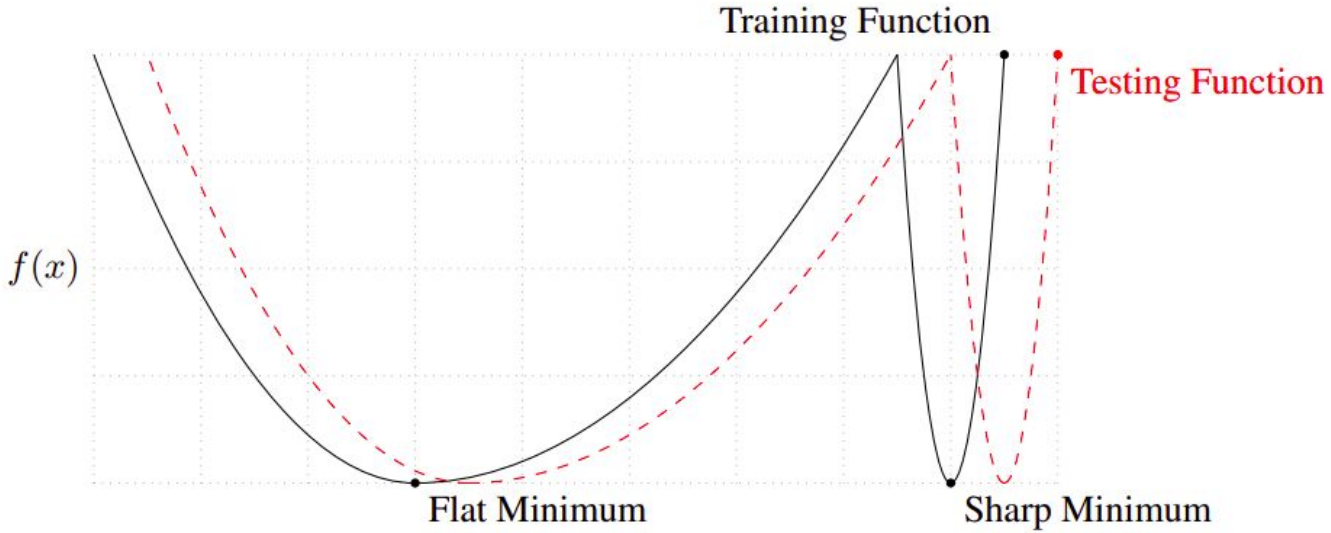
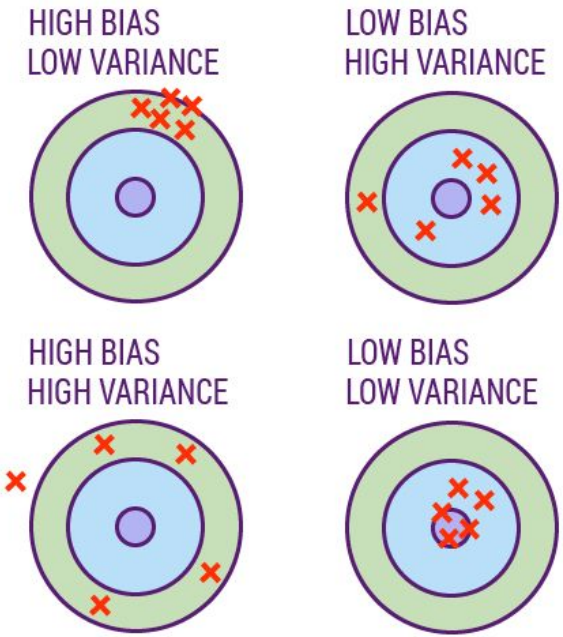
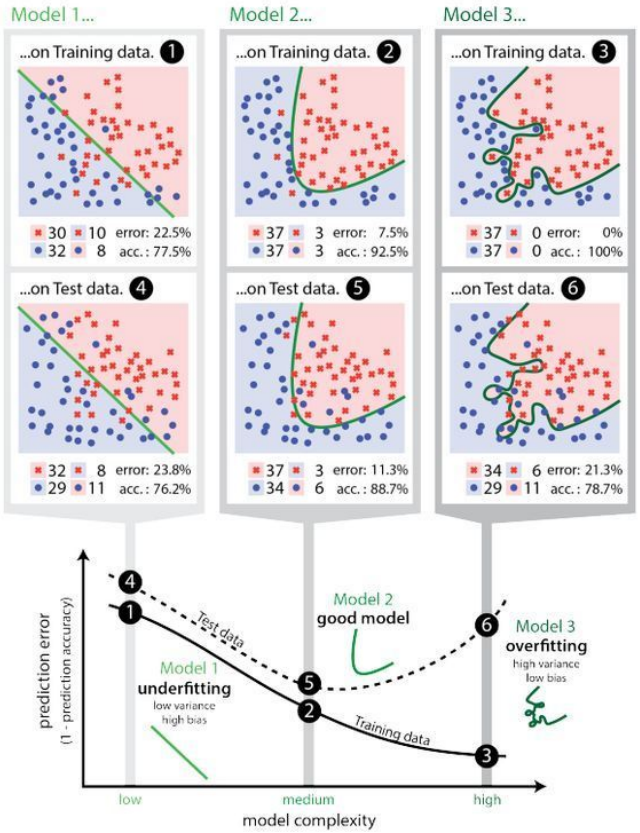


Figure 1: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

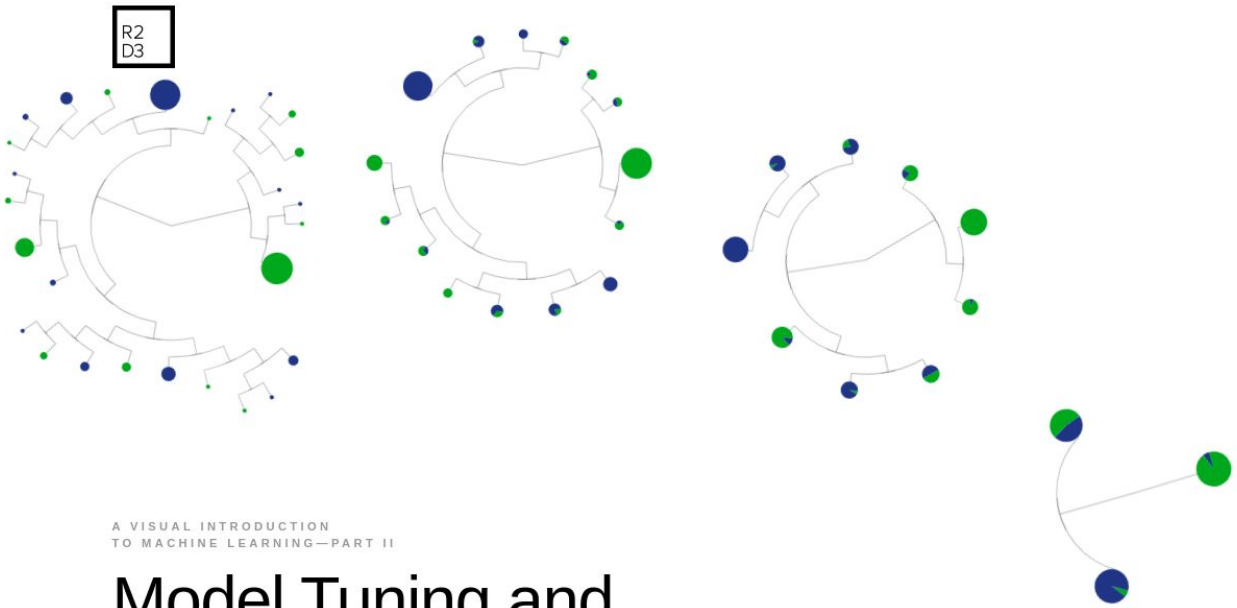
https://arxiv.org/pdf/1609.04836.pdf

Bias-Variance tradeoff



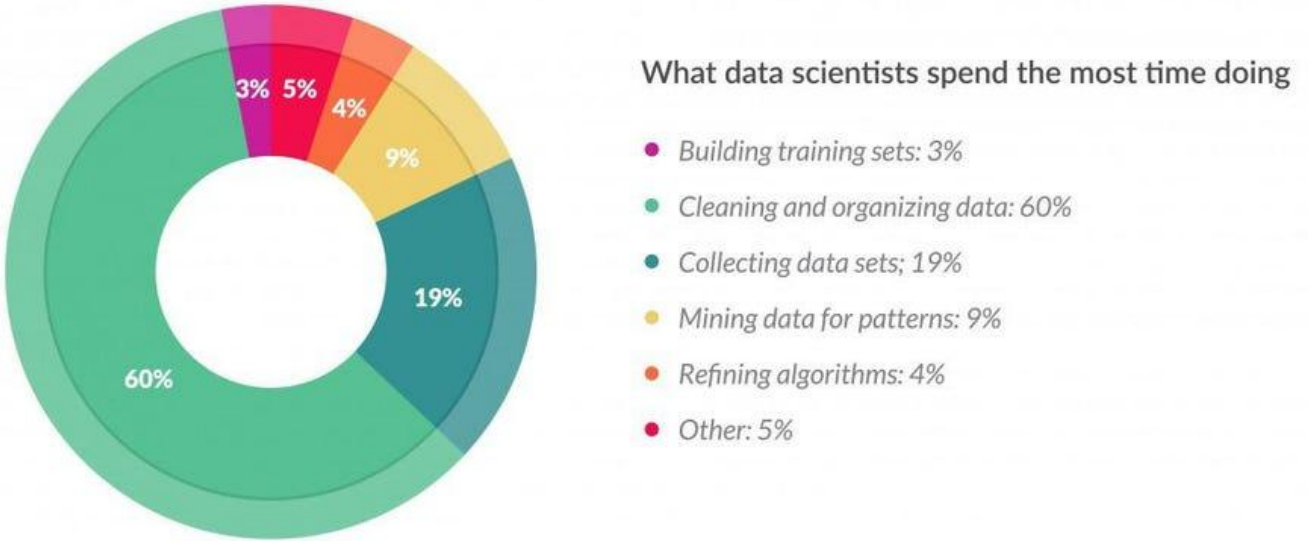


Bias-Variance tradeoff **interactive demo**



http://www.r2d3.us/visual-intro-to-machine-learning-part-2/

Data science



https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/

Limitations of DL

Lack of reasoning

Lack of generalization → DL excels at *automating* tasks

Lack of prior knowledge → solvable with transfer learning

Lack of interpretability (black-box models) → XAI working on this

Lots of training data often needed → Recent research on *Efficient DL*

Models prone to adversarial noise → Hot topic!

DL is a “too-empirical” field → but so do HCI

Some tricks and tips

http://cs231n.github.io/neural-networks-3/

http://karpathy.github.io/2019/04/25/recipe/

https://towardsdatascience.com/3ca24c31ddc8

https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks

Bibliography

1. F. Chollet. **Deep Learning with Python**, 2021
2. M. Nielsen. **Neural Networks and Deep Learning**, 2015
3. I. Goodfellow, Y. Bengio, A. Courville. **Deep Learning**, 2016
4. A. Gibson, J. Patterson. **Deep Learning: A Practitioners Approach**, 2017
5. R. O. Duda, P. E. Hart, D. G. Stork. **Pattern Classification**, 2000