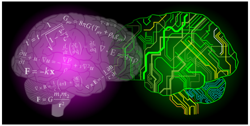
ML and Physics

Lec I: Introduction to ML



Ghadir Jafari

University of Cape Town, Cape Town, South Africa

Farhangian University, Tehran, Iran

The Objectives

1. Motivate to learn ML
2. Physical intuition for ML concepts
3. Introduce some research lines
4. Physics and ML connections

Outline

1. An Introduction to ML for Physicist
2. ML for Physics
3. Physics for ML
4. Physics and ML

Where is ML?

- Search engines, Spam filter in email, Recommender systems, Social networks
- Computer Vision, Speech recognition,
- Medical diagnosis, Medicine
- Industry, Economics,
- Theorem proving, Climate science,
- Self Driving Cars
- ...

Technology and ML

- Advances in computational hardware, the widespread use of GPUs
- Moore's law
- "Big Data" revolution
- Developments in theoretical aspects and programming
- Artificial neural networks

Physics And ML

- Information: Physics and Machine Learning
- Gathering and analyzing Data
- Finding patterns, Design models
- Predict the behaviour of (complex) systems
- Learning: passing information from input to outputs

$$\boxed{\text{Input}} \xrightarrow[\text{Computation,time}]{\text{Laws,Models}} \boxed{\text{Output}},$$

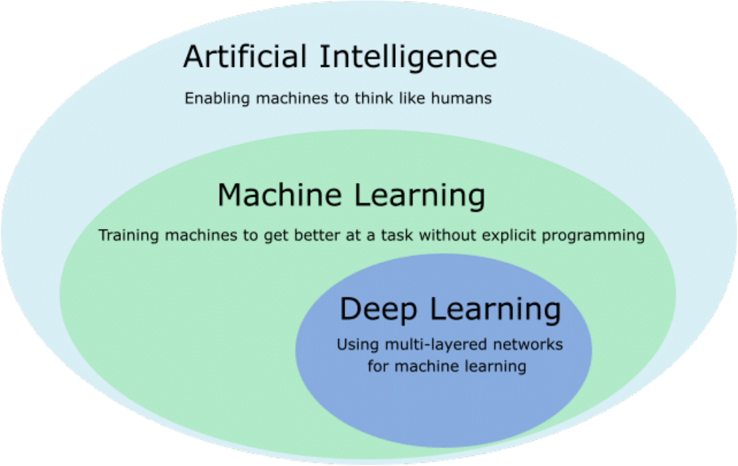
Physics Vs Machine Learning

- Physicists using their own knowledge, intelligence and intuition to inform their models
- Machine Learning: models are agnostic and the machine provides the intelligence by extracting it from data.

Machine Learning

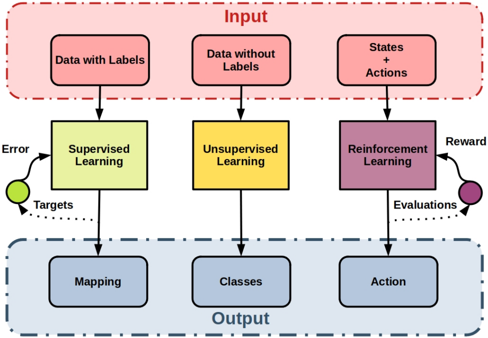
- Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.
- Task easy for humans to do, but difficult for humans to describe in terms of elementary operations.
- Problems for which existing solutions require a lot of fine-tuning or long lists of rules
- Complex problems for which using a traditional approach yields no good solution
- Getting insights about complex problems and large amounts of data

AI, ML and DL


Artificial Intelligence
Enabling machines to think like humans
Machine Learning
Training machines to get better at a task without explicit programming
Deep Learning
Using multi-layered networks
for machine learning

Approaches to ML

- Supervised learning: Data inputs and outputs (labels)
- Unsupervised learning: No labels, Clustering, find structure in its input
- Reinforcement learning: dynamic input space, maximize a reward function



Main Elements In ML

1. Data Preparation
2. Model Selection,
3. Training, Optimization
4. Validation, Test

From Data to Machines

- A Collection of Pairs $(\mathrm{x_i}, \mathrm{y_i})$

$$\mathrm{Data} = \{(\mathrm{x_1}, \mathrm{y_1}), (\mathrm{x_2}, \mathrm{y_2}), \ldots, (\mathrm{x_N}, \mathrm{y_N})\}$$

- x and y could be anything
- The goal: find (make) a function (machine) f

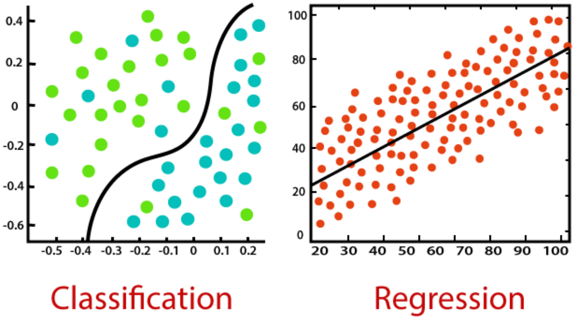
$$\mathrm{x_i} \longrightarrow \mathrm{f}(\mathrm{x_i}) = \mathrm{y_i}$$

- Such that for new data $(\mathrm{x_{new}}, \mathrm{y_{new}})$

$$\mathrm{y_{new}} = \mathrm{f}(\mathrm{x_{new}})$$

Classification Vs Regression

- Classification: the y take discrete values (y: labels)
- Regression: y continuous



- Classification → Nonlinear regression

Modeling: Function Representation (Approximation)

- Polynomial Representation

$$f_a(x) \approx \sum_{n=1}^{N} a_n x^n$$

- Fourier Representation:

$$f_c(x) \approx \sum_{n=-N}^{N} c_n e^{inx}$$

- e.g. $y^i \sim \sum_j^N w^{ij} x_j + b_j$
- Any Representation $\Rightarrow f_\theta(x)$
- A set of parameters which parameterize the space of functions

$$\boldsymbol{\theta} = \{\theta_1, \theta_1, \ldots, \theta_N\}$$

Training (Optimization)

- Finding suitable $\{\theta_{\mathrm{i}}^{\star}\}$ such that

$$\mathrm{y_i} \approx \mathrm{f}_{\theta^{\star}}(\mathrm{x_i})$$

- Define a cost $\mathcal{C}_{\mathrm{i}}(\mathrm{f}_{\theta}, \mathrm{y_i}) \Rightarrow \mathcal{C}_{\mathrm{i}} \approx 0$ when $\mathrm{y_i} \approx \mathrm{f}(\mathrm{x_i})$

$$\text{i.e.} \quad \mathcal{C}_{\mathrm{i}} = (\mathrm{y_i} - \mathrm{f}(\mathrm{x_i}))^2$$

- The Cost Function

$$\mathcal{C}(\theta) = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}}^{\mathrm{N}} \mathcal{C}_{\mathrm{i}}$$

- Want: $\theta^{\star} \rightarrow \mathcal{C}(\theta^{\star}) \approx 0$
- $\Rightarrow$ Minimizing $\mathcal{C}(\theta)$ in function space

Gradient Descent Method

- Change in $\mathcal{C}$ by $\mathrm{d}\theta$:

$$\mathrm{d}\mathcal{C}(\theta) = \mathrm{d}\theta \cdot \nabla_\theta \mathcal{C},$$

- If

$$\mathrm{d}\theta = -\eta \nabla_\theta \mathcal{C}$$

- Then:

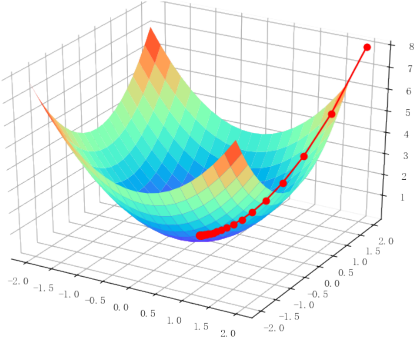
$$\mathrm{d}\mathcal{C}(\theta) = -\eta |\nabla_\theta \mathcal{C}|^2 \leq 0$$

- Steps toward the local minimum:

$$\begin{aligned} \theta &\to \theta - \eta \nabla_\theta \mathcal{C} \\ \mathcal{C} &\to \mathcal{C} - \eta |\nabla_\theta \mathcal{C}|^2 \end{aligned}$$

Gradient Descent Method

- η: Learning rate, n Number of steps (epochs)



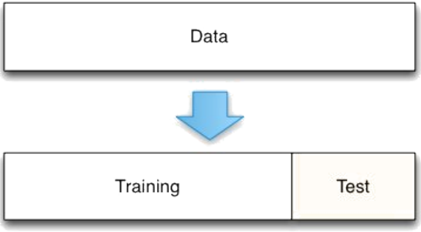
- Particle moving in a potential

Testing

- New Data $\{x_{new}, y_{new}\}$

$$y_{new} = f(x_{new})?$$

- Training and Test Split:



Example: Linear Regression

- data: $\{\mathrm{x_i}, \mathrm{y_i}\}$
- objective: predict $\mathrm{y} = \mathrm{f}(\mathrm{x})$
- model: $\mathrm{f}(\mathrm{x};\vec{\theta}) = \mathrm{mx} + \mathrm{b}, \quad \vec{\theta} = (\mathrm{m}, \mathrm{b})$
- loss function:

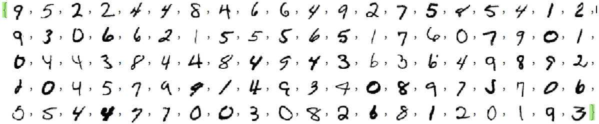
$$\mathcal{C}(\theta) = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{y_i} - \mathrm{f}\left(\mathrm{x_i}\right)\right)^2$$

- model training: optimal parameters

$$\hat{\vec{\theta}} = \arg\min \mathcal{C}(\vec{\theta})$$

Example: Learning The Digits

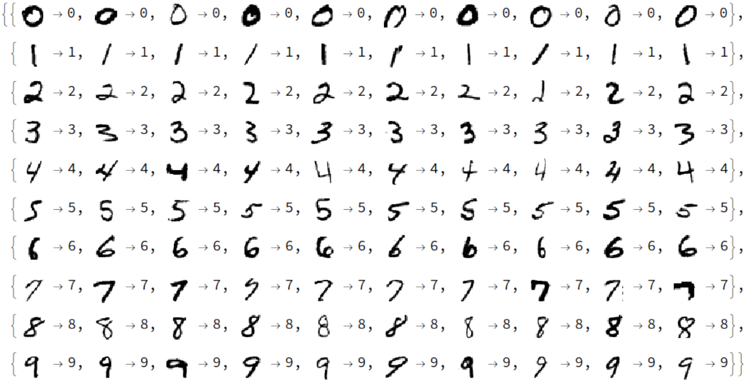
- Problem: A program to recognize the images of handwritten digits
- Input:



- Output: Digit letters $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- Impossible algorithmic programming
- Machine Learning: computers learn (doing tasks) without being explicitly programmed

Example: Learning The Digits

- The Data:



Example: Learning The Digits

- Each image a matrix:

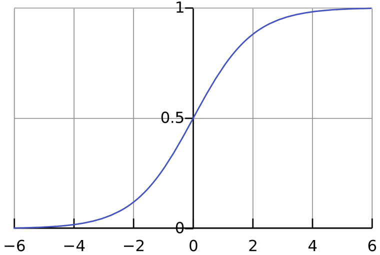
$$
\begin{pmatrix}
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&1&1&0&0&0&0&0&1&1&1&1&1\\
1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}
$$

- Outputs are labels (discrete values)

$$20 \times 20 \text{ Matrices } ([0,1]^{400}) \to \{1, 2, \ldots, 10\}$$

Example: Learning The Digits

- How to model?
- Discrete output: classification problem
- A function to $[0, 1]^{10}$
- A linear model $\mathrm{f}(\mathrm{x_n}) = \mathrm{w^{in}x_n + b^i}$
- Using the logistic sigmond function : $\sigma(\mathrm{x}) = \frac{1}{1+\mathrm{e}^{-\mathrm{x}}}$



- The model $\mathrm{f}(\mathrm{x_n}) = \sigma(\mathrm{w^{in}x_n + b^i})$

Example: Learning The Digits

- Optimization: Find a minimum for

$$\mathcal{C}(\mathrm{w}, \mathrm{b}) = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\mathrm{y_i} - \mathrm{f}\left(\mathrm{x_i}\right)\right)^2$$

- No much precision!
- Is the model simple?
- How to generalize?
- $\mathrm{x_n} \rightarrow \mathrm{w^{in}x_n} + \mathrm{b^i} \rightarrow \sigma(\mathrm{w^{in}x_n} + \mathrm{b^i})$
- The number of parameters : $10 * 400 + 10$

Neural Networks

$$z_i^{(1)} \equiv b_i^{(1)} + \sum_{j=1}^{n_0} W_{ij}^{(1)} x_j, \quad \text{for} \quad i = 1, \ldots, n_1$$

$$z_i^{(1)} \to \sigma(z_i^{(1)})$$

$$z_i^{(2)} \equiv b_i^{(2)} + \sum_{j=1}^{n_1} W_{ij}^{(2)} \sigma(z_j^{(1)}), \quad \text{for} \quad i = 1, \ldots, n_2$$

$$\vdots$$

$$z_i^{(\ell)}(x) \equiv b_i^{(\ell)} + \sum_{j=1}^{n_{\ell-1}} W_{ij}^{(\ell)} \sigma\left(z_j^{(\ell-1)}\right), \quad \text{for} \quad i = 1, \ldots, n_\ell$$

$$\vdots$$

$$\text{for} \quad \ell = 1, \ldots, L$$

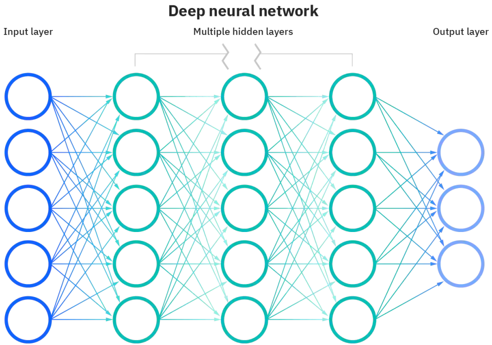
$$z_i^{(L)} \to \sigma(z_i^{(L)})$$

$$\Rightarrow y = f_{\{w,b\}}(x) = \sigma(z_j^{(L)})$$

Neural Networks

- A neural network is a composite nonlinear function :

$$y = f_L(w^L f_{L-1}(w^{L-1} ... f_1(w^1 x) \dots))$$



- Basic Ingredients: Neurons
- Parameters $\{w^{\ell}_{i,j}, b^{\ell}_i\}$

Learning The Digits

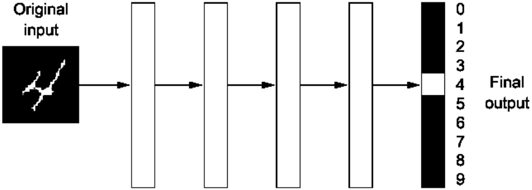
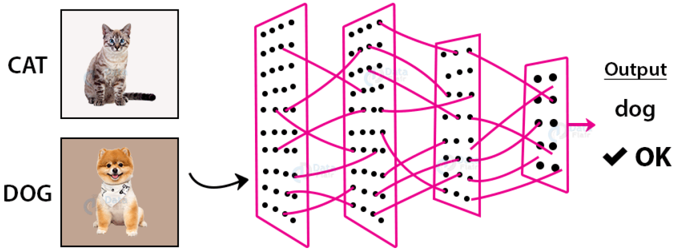
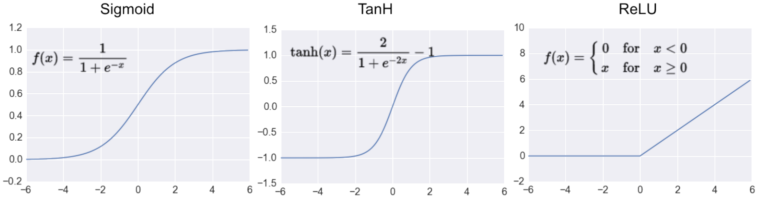


Image Recognition



Neural Networks Architecture

- The Hyper-parameters: number of layers L, The number of neurons in each layer n_ℓ, Initialization of Parameters, Learning Rate η, The Number of Epochs, ...
- The activation function $\sigma(x)$ e.g.:



NN as Universal Approximation

- Neural Networks can represent any functions
- For every y,d,D and ϵ

$$\mathrm{y} : \mathbb{R}^{\mathrm{d}} \rightarrow \mathbb{R}^{\mathrm{D}}$$

- There exist a network $\mathrm{f_{w,b}}$ such that

$$\|\mathrm{y(x)} - \mathrm{f_{w,b}(x)}\| < \epsilon$$

- $\Rightarrow$ Neural network: A flexible representation for high-dim functions
- Importance: Every process is a function computation

The backpropagation Algorithm

- Training: the calculation of $\nabla_{\theta}\mathcal{C}$ at each step

$$\mathcal{C}(\vec{\theta}) = \frac{1}{\mathrm{N}} \sum_{\mathrm{i}=1}^{\mathrm{N}} (\mathrm{y_i} - \mathrm{f}(\mathrm{x_i}))^2$$

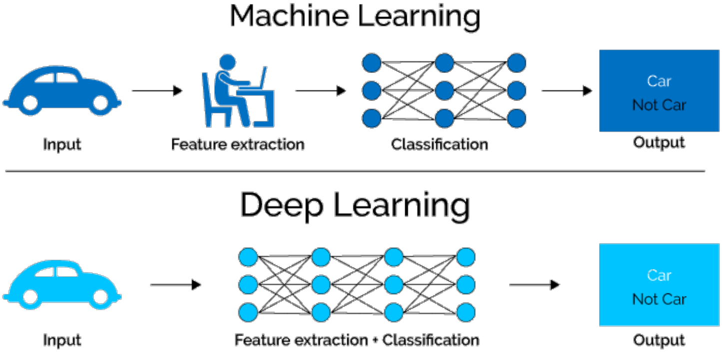
$\vec{\theta} = \{\mathrm{w}^{\ell}_{\mathrm{i,j}}, \mathrm{b}^{\ell}_{\mathrm{i}}\}$

- How to compute for NN?
- Suppose $\mathrm{w}^{\ell}_{\mathrm{i,j}} \to \mathrm{w}^{\ell}_{\mathrm{i,j}} + \delta \mathrm{w}^{\ell}_{\mathrm{i,j}}$
- Chain Rule:

$$\nabla_{\mathrm{w}^{\ell}}\mathcal{C} = [\left(\mathrm{f}^{\ell}\right)' \circ \left(\mathrm{w}^{\ell+1}\right)^{\mathrm{T}} \cdots \circ \left(\mathrm{w}^{\mathrm{L}-1}\right)^{\mathrm{T}} \cdot \left(\mathrm{f}^{\mathrm{L}-1}\right)' \circ \left(\mathrm{w}^{\mathrm{L}}\right)^{\mathrm{T}} \cdot \left(\mathrm{f}^{\mathrm{L}}\right)' \circ \nabla_{\mathrm{z}^{\ell}}\mathrm{C}] \left(\mathrm{z}^{\ell-1}\right)^{\mathrm{T}}$$

Deep Learning

- Large number of layers in NN
- $\Rightarrow$ More efficient



The Black Box?

- No Theoretical explanations of deep learning
- Why the output $f_{\theta^\star}(x)$ is given on an input x
- Hyper-Parameters: trial-and-error
- The activation functions, the widths of the layers and the depth
- which learning problems are computationally tractable?
- $\Rightarrow$ Black box

A Theory for Deep Learning?

- Complexity $\Rightarrow$ Statistical explanation

$$\mathrm{P(x|f_\theta(x))} \quad \mathrm{Or/And} \quad \mathrm{P(f_\theta(x)|x)}$$

- Statistical Theory beneath experimental thermodynamic rules
- Like statistical mechanics explains how the macroscopic laws of thermodynamics describing the steam engines.
- $\Rightarrow$ A theory for machines

Unsupervised Learning

- No outputs (labels)

$$\mathcal{D} = \{\mathrm{x_i}\}$$

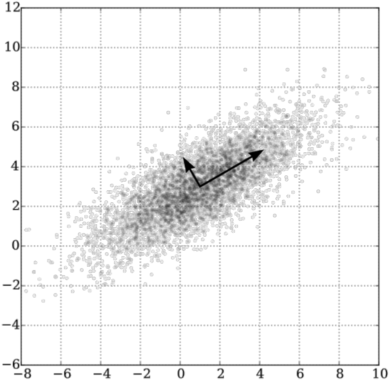
- Depend on the specific details of the task
- We assume that there exists some probability distribution $\mathrm{P(x)}$
- The model $\mathrm{Q}_\theta(\mathrm{x})$
- Make them similar
- Reduce the relative entropy

$$\mathrm{D_{KL}}(\mathrm{P} \parallel \mathrm{Q}) = \sum_{\mathrm{x} \in \mathcal{D}} \mathrm{P(x)} \log \left(\frac{\mathrm{P(x)}}{\mathrm{Q}_\theta(\mathrm{x})} \right).$$

- Both modeling and optimization $\Rightarrow$ Deep Learning help

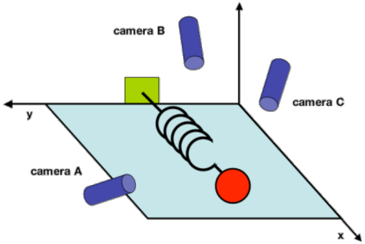
Principal component analysis

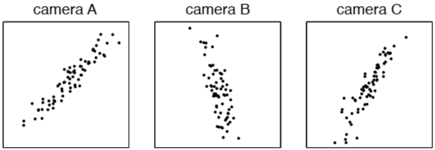
- Compress the data to loss information as less as possible
- Dimension Reduction



Simple Example[1]

- Data from an oscillating spring





[1] arXiv:1404.1100

Which Problems could be the subject of ML?

- ML: Direct relation of input and outputs,
- Data, Modeling, Optimization
- $\Rightarrow$ In Principle any problem

Some Textbooks

- Pattern recognition and machine learning , Bishop, C M (2006),
- Deep Learning, Ian Goodfellow, Yoshua Bengio, Aaron Courville [1]
- Neural Networks and Deep Learning, Michael Nielsen [2]

[1] www.deeplearningbook.org
[2] neuralnetworksanddeeplearning.com

Programming

- Most Common languages: Python and R
- Libraries: Tensorflow, Scikit-learn, Keras, PyTorch
- Prerequisites: NumPy, pandas, and Matplotlib
- Hands-On Machine Learning with Scikit-Learn and TensorFlow, Aurélien Geron[1]
- Wolfram Mathematica

[1] github.com/ageron/handson-ml2

Summery

- ML deal with modeling and predicting similar to physics
- Main elements include modeling and optimization
- Neural networks provide a flexible modeling
- With the cost of being black box