Machine Learning Introduction

Matteo Matteucci

matteucci@elet.polimi.it

Department of Electronics and Information

Politecnico di Milano

Lectures Outline

Lectures on Knowledge Engineering: 60% of classes, 50% of grade!

- Uncertaint Reasoning
- Fuzzy Logic and Fuzzy Sets
- Reinforcement Learning

Lectures on Machine Learning: 40% of classes, 50% of grade!

- Introduction to Natural Computation an Machine Learning
- Artificial Neural Networks
  - Feed-forward Neural Networks
  - Radial Basis Functions
  - Recurrent Architectures
  - Neuro-Fuzzy Systems
- Bayesian Networks
  - Inference in Bayesian Networks
  - Dynamical Bayesian Networks (Hidden Markov Models)

Course Info

- Course Material from Machine Learning Perspective
  - Reti Neurali e Metodi Statistici, S. Ingrassia, C. Davino, 2002
  - Neural Networks and Pattern Recognition, C. Bishop, 1995
  - Probabilistic Networks and Expert Systems, R.G. Cowell, A.P. Dawid, S.L. Lauritzen and D.J. Spiegelhalter, 1999.
  - . . .
- Evaluation and Grading:

  “Standard” written exam on the course material (up to 32/32)

Dartmouth 1955 – Conception of AI

A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

J. McCarthy, Dartmouth College – M.L. Minsky, Harvard University
N. Rochester, I.B.M. Corporation – C.E. Shannon, Bell Telephone Laboratories

August 31, 1955

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html

Dartmouth 1956 – The AI Program

1. Automatic Computers
2. How Can a Computer be Programmed to Use a Language
3. Neuron Nets
4. Theory of the Size of a Calculation
5. Self-Improvement
6. Abstractions
7. Randomness and Creativity

We'll look at least to 3 of these points:

- Self-Improvement $\longrightarrow$ Machine Learning
- Neuron Nets $\longrightarrow$ Artificial Neural Networks
- Randomness $\longrightarrow$ Genetic Algorithms (Reinforcement Learning)

Motivation for Natural Computation (vs. GOFAI)

Artificial Intelligence:
A subfield within computer science concerned with developing technology to enable computers to solve problems (or assist humans in solving problems) that, according to common people thought, could be solved only by humans.

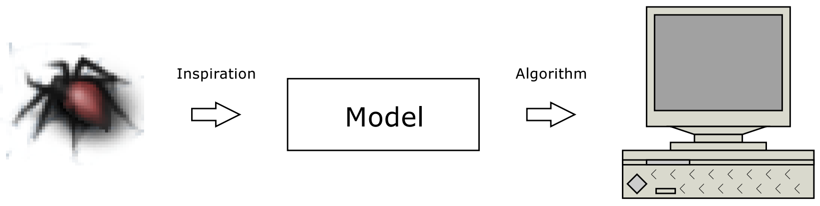
This resulted mostly on Symbolic Processing Algorithms and Languages with Expert Systems being the most remarkable applications . . . however:

- Human brain is not a symbolic system neither any other living being!
- Nature has solutions (HW & SW) to problem we are interested in, can we steal ideas from nature?
- Nature has adapted to solve "hard" problems can we make our computer able to do the same?

Here we go! $\longrightarrow$ Natural Computation

Natural Computation: Models from Nature

Natural Computation:
A subfield within computer science concerned with developing computational models and technologies inspired by nature to solve problems that in some extent nature has already solved in the last millennia and man is still struggling on.



Natural models of computation evidence:

- Learning & Adaptivity $\longrightarrow$ Robust to modeling errors
- Parallelism & Fault Tolerance $\longrightarrow$ Robust to failures
- Embedding in the real world!

Machine Learning

– Basics & Examples –

Self-Improvement and Learning

A computer program is said to **learn** from experience **E**
with respect to some class of **task T** and a **performance measure P**,
if its performance at tasks in **T**, as measured by **P**,
improves because of experience **E**.

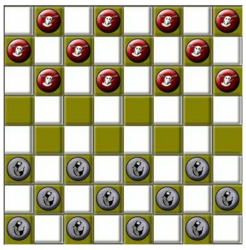
Tom Mitchell – Machine Learning

Machine Learning: the study or development of models and algorithms that make systems automatically improve their performance during execution. We call these models: **Adaptive Models.**

Matteo Matteucci

Example: Learning to Playing Checkers

- Task **T**
  - Play checkers
- Experience **E**
  - Games played with other players
  - Games played against itself
- Performance **P**
  - Percentage of games won

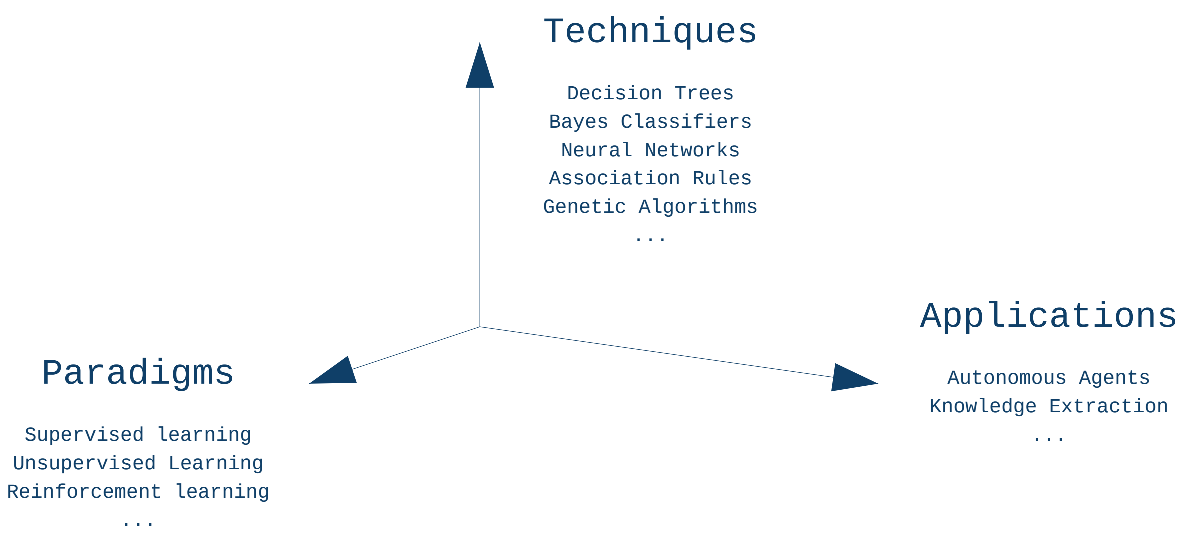


Any other example?

Applications: “What’s your flava?”

- Self customizing programs
  - Newsreader that learns user interests
  - Email anti-spam filters
  - . . .
- Data mining
  - medical records $\longrightarrow$ medical knowledge
  - using historical data to improve decisions
  - . . .
- Software applications we can’t program by hand
  - autonomous driving
  - speech recognition
  - . . .

Apps – Paradigms – Techniques



Learning Paradigms

Imagine an organism or machine that experiences a series of inputs:

$$\mathbf{E} = x_1, x_2, x_3, x_4, \ldots$$

- Supervised learning: given the **desired outputs** $y_1, y_2, \ldots,$ learn to produce the correct output given new input
- Unsupervised learning: **exploit regularities in $\mathbf{E}$** to **build a representation** that can be used for reasoning or prediction
- Reinforcement learning: **producing actions** $a_1, a_2, \ldots$ which affect the environment, and **receiving rewards** $r_1, r_2, \ldots$ learn to act in a way that **maximizes rewards** in the long term

In this course we'll focus on **Supervised Learning!**

Terminology

Classification: The desired outputs y_i are discrete class labels. The goal is to classify new inputs correctly.

Regression: The desired outputs y_i are continuous valued. The goal is to predict the output accurately for new inputs.

Generalization: The capability of a model to correctly predict new samples never seen during the training.

Inductive Hypothesis: A solution that approximate the target function over a sufficiently large set of training examples will also approximate the target function over unobserved examples

Supervised Learning

- **The Task T:** extract from a finite set of examples a model of the observed phenomenon to be used in the future for prediction or decision making about it
- **The Experience E:** a set of examples for the desired “behavior” pre-processed by an expert [the supervisor] as pairs input / output
- **The Performance P:** is the measure of the distance between the desired output for new examples and the output provided by the model

Supervised Learning Examples

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction has to be based on demographic, diet, and clinical measurements.
- Predict the price of a stock in six months from now, on the basis of company performance measures and economic data.
- Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption of that person’s blood.
- Identify the risk factors for prostate cancer, based on clinical and demographic variables.
- Identify the numbers in handwritten ZIP code, from a digitalized image.

Example: ZIP Codes Images

There are 7291 training observations and 2007 test observations.
Each observation is a 16 x 16 gray-scale image

References for the Lecture

To prepare this lecture I had a look at:

- An Introduction to Natural Computation, D.H. Ballard, 1997.
- The Computational Beauty of Nature, G.W. Flake, 1998. (you can find more at: http://mitpress.mit.edu/books/FLAOH/cbnhtml/home.html)
- Emula e non Simula, M. Somalvico, in Aspettando Robot: Il futuro prossimo dell’Intelligenza Artificiale, 1987.
- Machine Learning, T. Mitchell, 1997

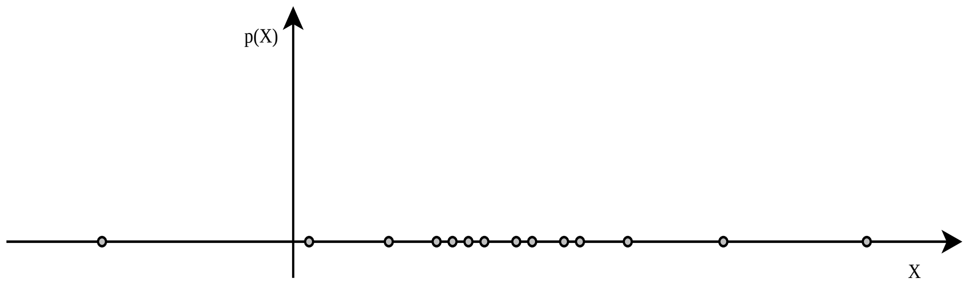
Supervised Learning

– Maximum Likelihood Approach –

An Introduction to Maximum Likelihood Estimation

Suppose we observe some i.i.d. samples coming from a Gaussian distribution with known σ^2:

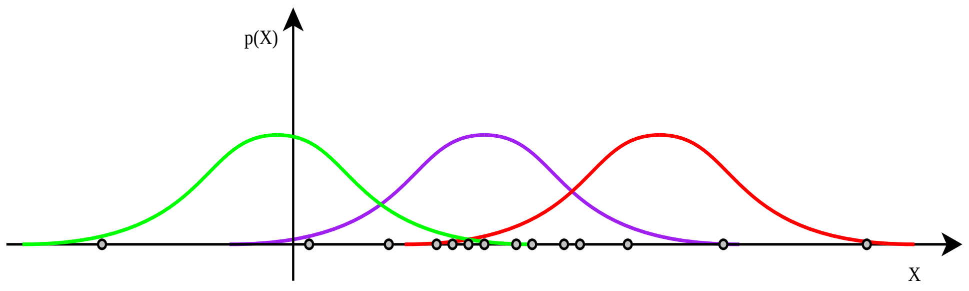
$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$



An Introduction to Maximum Likelihood Estimation

Suppose we observe some i.i.d. samples coming from a Gaussian distribution with known σ^2:

$$x_1, x_2, \ldots, x_K \sim N(\mu, \sigma^2) \qquad p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{\sigma^2}}$$



Which one do you prefer? Why?

Maximum Likelihood $\rightarrow$ Choose parameters that maximize data probability

The Maximum Likelyhood Estimation Recipe

Suppose $\theta = (\theta_1, \theta_2, \ldots, \theta_n)^T$ is a vector of parameters and we want to find the MLE for θ assuming known form for $p(Data|\theta)$:

1. Write the likelihood $L = P(Data|\theta)$ for the data
2. (Take the logarithm of likelihood $\mathcal{L} = \log P(Data|\theta)$)
3. Work out $\partial L / \partial \theta$ or $\partial \mathcal{L} / \partial \theta$ using high-school calculus
4. Solve the set of simultaneous equations $\partial \mathcal{L} / \partial \theta_i = 0$
5. Check that θ^{mle} is a maximum

To maximize/minimize the (log)likelihood you can use:

- Analytical techniques (i.e., solve the disequations)
- Optimizaion techniques (e.g., Lagrange mutiplier)
- Numerical tecniques (e.g., gradient descend)

Maximum Likelihood for Univariate Gaussians (I)

Suppose we have N indipendent samples $x_1, x_2, \ldots, x_N$ coming from a Gaussian distribution with known σ^2. Which is the MLE for μ?

1. Write the likelihood $L = P(Data|\theta)$ for the data

$$\begin{aligned} L(\mu) &= p(x_1, x_2, \ldots, x_N|\mu, \sigma^2) = \prod_{n=1}^{N} p(x_n|\mu, \sigma^2) \\ &= \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_n-\mu)^2}{2\sigma^2}} \end{aligned}$$

Maximum Likelihood for Univariate Gaussians (I)

Suppose we have N indipendent samples $x_1, x_2, \ldots, x_N$ coming from a Gaussian distribution with known σ^2. Which is the MLE for μ?

1. Write the likelihood $L = P(Data|\theta)$ for the data
2. (Take the logarithm of likelihood $\mathcal{L} = \log P(Data|\theta)$)

$$\begin{aligned} \mathcal{L} &= log \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \\ &= \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x_n-\mu)^2}{2\sigma^2}) \\ &= N(\log \frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (x_n - \mu)^2 \end{aligned}$$

Maximum Likelihood for Univariate Gaussians (I)

Suppose we have N indipendent samples $x_1, x_2, \ldots, x_N$ coming from a Gaussian distribution with known σ^2. Which is the MLE for μ?

1. Write the likelihood $L = P(Data|\theta)$ for the data
2. (Take the logarithm of likelihood $\mathcal{L} = \log P(Data|\theta)$)
3. Work out $\partial\mathcal{L}/\partial\theta$ using high-school calculus

$$\begin{aligned}
\frac{\partial\mathcal{L}}{\partial\mu} &= \frac{\partial}{\partial\mu}N(\log\frac{1}{\sqrt{2\pi}\sigma}) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 \\
&= -\frac{1}{2\sigma^2}\frac{\partial}{\partial\mu}\sum_{n=1}^{N}(x_n-\mu)^2 = \\
&= \frac{1}{2\sigma^2}\sum_{n=1}^{N}2(x_n-\mu)
\end{aligned}$$

Maximum Likelihood for Univariate Gaussians (I)

Suppose we have N indipendent samples $x_1, x_2, \ldots, x_N$ coming from a Gaussian distribution with known σ^2. Which is the MLE for μ?

1. Write the likelihood $L = P(Data|\theta)$ for the data
2. (Take the logarithm of likelihood $\mathcal{L} = \log P(Data|\theta)$)
3. Work out $\partial\mathcal{L}/\partial\theta$ using high-school calculus
4. Solve the unconstrained equations $\partial\mathcal{L}/\partial\theta_i = 0$

$$\begin{aligned}
\frac{1}{2\sigma^2}\sum_{n=1}^{N}2(x_n-\mu) &= 0 \\
\sum_{n=1}^{N}(x_n-\mu) &= 0 \\
\sum_{n=1}^{N}x_n &= \sum_{n=1}^{N}\mu
\end{aligned}$$

Maximum Likelihood for Univariate Gaussians (II)

The best Maximum Likelihood Estimate of the mean of a gaussian distribution is the mean of the sample:

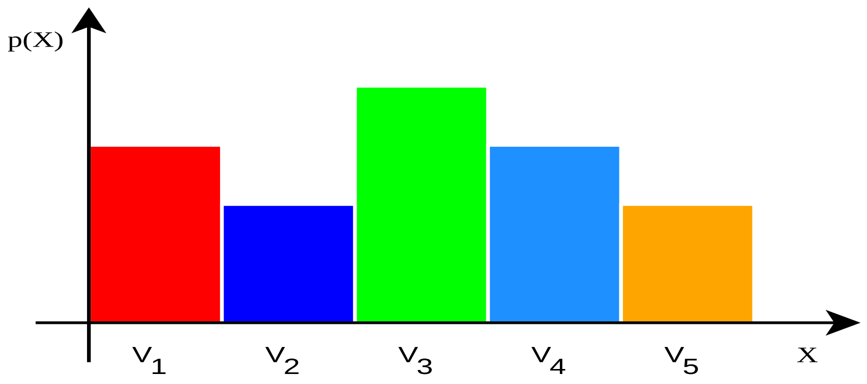
$$\mu^{mle} = \frac{1}{N}\sum_{r=1}^{N} x_n$$

"This kind of pedantic, algebra-filled and ultimately unsurprising fact is exactly the reason people throw down their Statistics book and pick up their Agent Based Evolutionary Data Mining Using The Neuro-Fuzzy Transform book."

– Andrew W. Moore

Maximum Likelihood for Multinomial Distribution (I)

Suppose we have N indipendent samples $x_1, x_2, \ldots, x_n$ from a discrete random variable X that can assume a finite set of K values $V_1, V_2, \ldots, V_K$ each with probability θ_k. Which is the MLE for θ_V?



Maximum Likelihood for Multinomial Distribution (I)

Suppose we have N indipendent samples $x_1, x_2, \ldots, x_n$ from a discrete random variable X that can assume a finite set of K values $V_1, V_2, \ldots, V_K$ each with probability θ_k. Which is the MLE for θ_V?

1. Write the likelihood $L = P(Data|\theta)$ for the data
$$L(\theta) = p(x_1, x_2, \ldots, x_N|\theta_1, \ldots, \theta_K) = \prod_{n=1}^{N} \theta_{x_n}$$
2. Remind the constraints for the problem:
    - $\theta_k \geq 0 \;\; \forall k = 0, 1, \ldots, K$
    - $\sum_{k=1}^{K} \theta_k = 1$
3. Maximize using Lagrange Multipliers
    - $\theta^{mle} = \arg\max_\theta \left[L(\theta) + \lambda \left(\sum_{k=1}^{K} \theta_k - 1\right)\right] = \arg\max_\theta J(\theta)$
    - Solve the system of equations
        - $\frac{\partial J(\theta)}{\partial \theta_k} = \frac{N_k}{\theta_k} L(\theta) - \lambda = 0$
        - $\frac{\partial J(\theta)}{\partial \lambda} = \sum_{k=1}^{K} \theta_k - 1 = 0$

Maximum Likelihood for Multinomial Distribution (II)

The best Maximum Likelihood Estimate for the θ_k parameters of a multinomial distribution is the ratio of the number of records with value $X = V_k$ and the total number of records:

$$\theta_k^{mle} = \frac{N_k}{N} = \frac{\text{Num. of Records with } X = V_k}{\text{Total Num. of Records}}$$

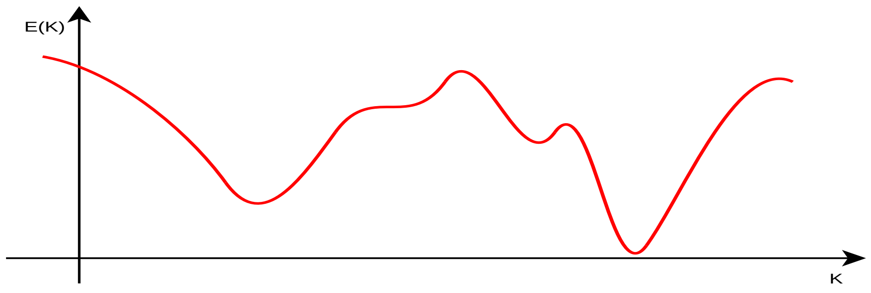
“This kind of pedantic, algebra-filled and ultimately unsurprising fact is exactly the reason people throw down their Statistics book and pick up their Agent Based Evolutionary Data Mining Using The Neuro-Fuzzy Transform book.”

– Andrew W. Moore

We’ll see more interesting MLE problems using Neural Networks ;)

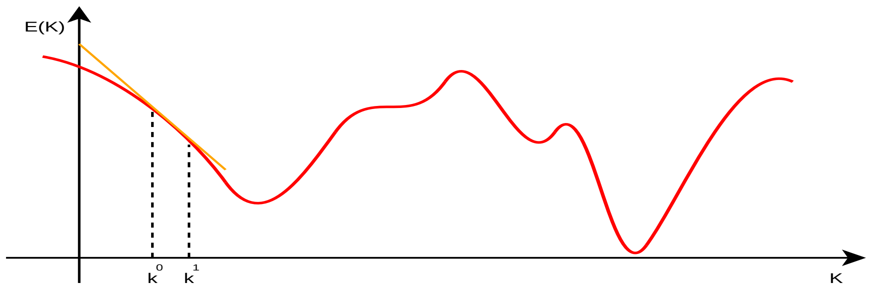
Numerical Methods for Minimization

Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend



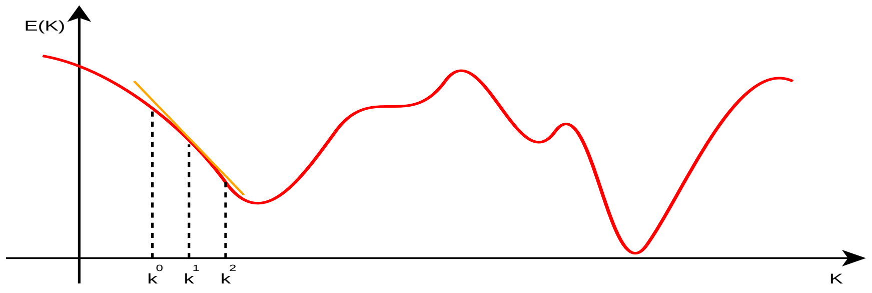
Numerical Methods for Minimization

Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend



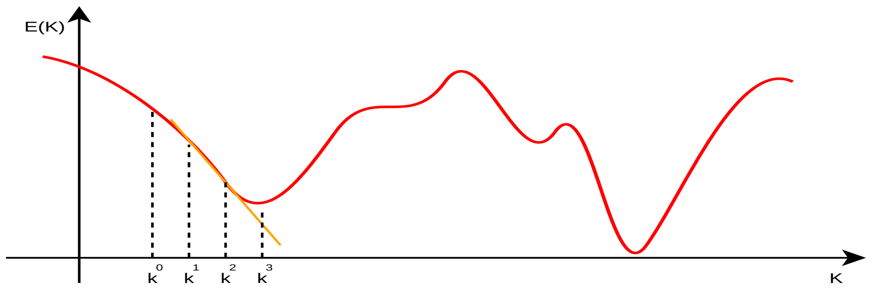
Numerical Methods for Minimization

Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend



Numerical Methods for Minimization

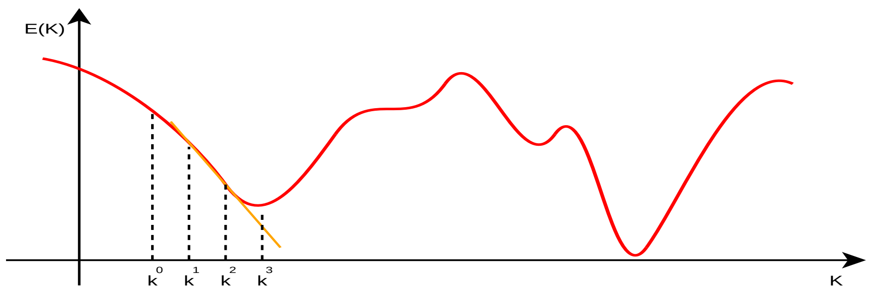
Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend



1. Pick up a possible solution k^0 (at random)
2. Compute the function derivative $\frac{\partial E(k)}{\partial k}|_{k^n}$
3. Update your solution $k^n = k^{n-1} - \gamma \frac{\partial E(k)}{\partial k}|_{k^{n-1}}$
4. Repeat 2 and 3 untill convergence

Numerical Methods for Minimization

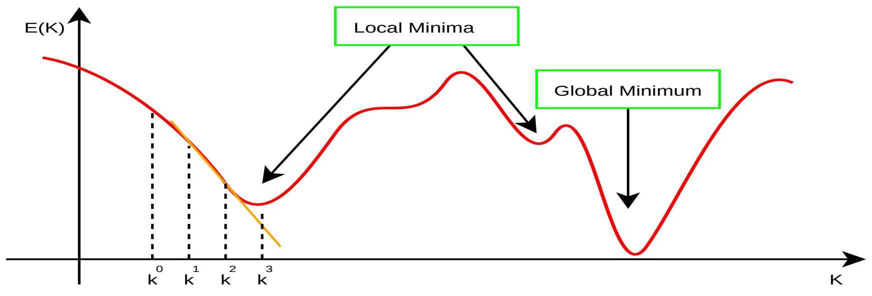
Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend



Can you guess any possible flaw?

Numerical Methods for Minimization

Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend

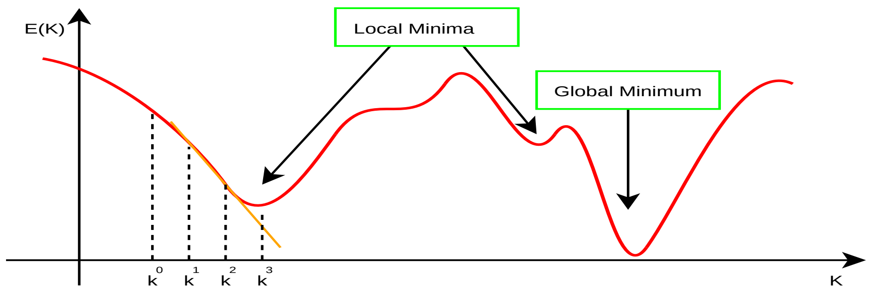


Can you guess any possible flaw?

- Lots of local minima
- Slow convergence
- No convergence at all

Numerical Methods for Minimization

Sometimes it not that easy to minimize/maximize analytically a function so we need to apply a numerical approach $\longrightarrow$ gradient descend

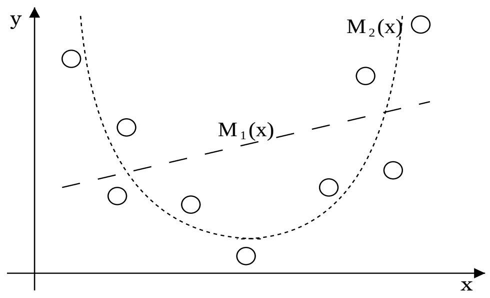


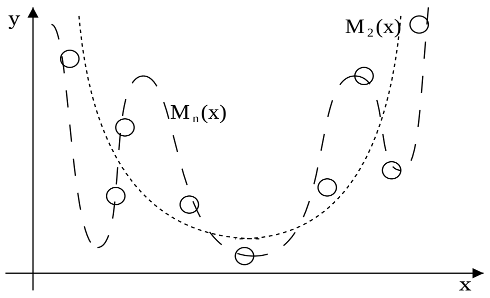
Can you guess any possible flaw?

- Lots of local minima $\longrightarrow$ Many random initializations
- Slow convergence $\longrightarrow$ Use second order derivative
- No convergence at all $\longrightarrow$ Reduce γ during minimization

Picking Up The Right Model . . .

Suppose we have a dataset taken from a 2^{nd} order polinome (but we don't know) with some kind of noise in it ...





- A 1^{st} order polimome has a poor fitting of the data $\longrightarrow$ Underfitting
- Higher order have a perfect fit, but no generalization $\longrightarrow$ Overfitting

"Entia non sunt multiplicanda praeter necessitatem" [Occam]

"Things should be made as simple as possible, but not simpler" [A. Einstein]