Convex and Distributed Optimization

Franck Iutzeler **Jérôme Malick** **Thomas Ropars** **Dmitry Grishchenko**




from **LJK**, the applied maths and computer science laboratory

and **LIG**, the Grenoble Informatics Laboratory

- To reach us: `firstname.name@univ-grenoble-alpes.fr`
- To find these slides: `http://ljk.imag.fr/membres/Jerome.Malick/CDO.pdf`
- Practical informations on this course: `http://www.iutzeler.org/CDO/`
  (including schedule, tutorials, articles, info on grades...)

- Positioning of the course

About this course

Optimization & Learning

- We have entered the **Big Data** area...
- Huge amounts of data are collected, routinely and continuously
  - Consumer and people data (phone calls and text, social media, email, surveillance cameras, web activity...)
  - Scientific data (biological, genomic, astronomical,...)
- Challenges in the whole chain of data processing
  from data collection to computation, analysis, interpretation

- Optimization is at the core of numerical methods...

Illustration: images reconstruction in radio-astronomy

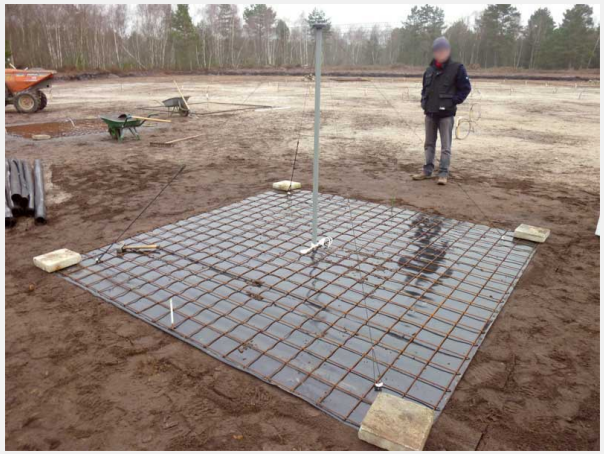
(example maybe unusual for you)

technology of the past

Illustration: images reconstruction in radio-astronomy

(example maybe unusual for you)

technology of the past



technology of the future !!

software-telescope

- large, flexible, and cheap networks
- huge data flow, huge numerical treatment
- with in particular: large-scale optimization problems

Goals of data analysis

- Extract meaning from data: understand statistical properties, learn important features and fundamental structures in the data
- Use this knowledge to make decisions or predictions about other data

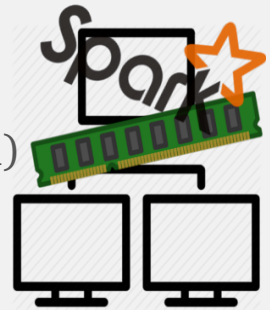
Highly multidisciplinary area
with foundations in statistics and computer science (artificial intelligence, machine learning, databases, parallel systems...)

and **Optimization** here ?

- Optimization provides a toolkit of modeling and algorithmic techniques
- Ongoing challenges because of increasing **scale and complexity** of data analysis applications

Need for **scalable optimization algorithms**...

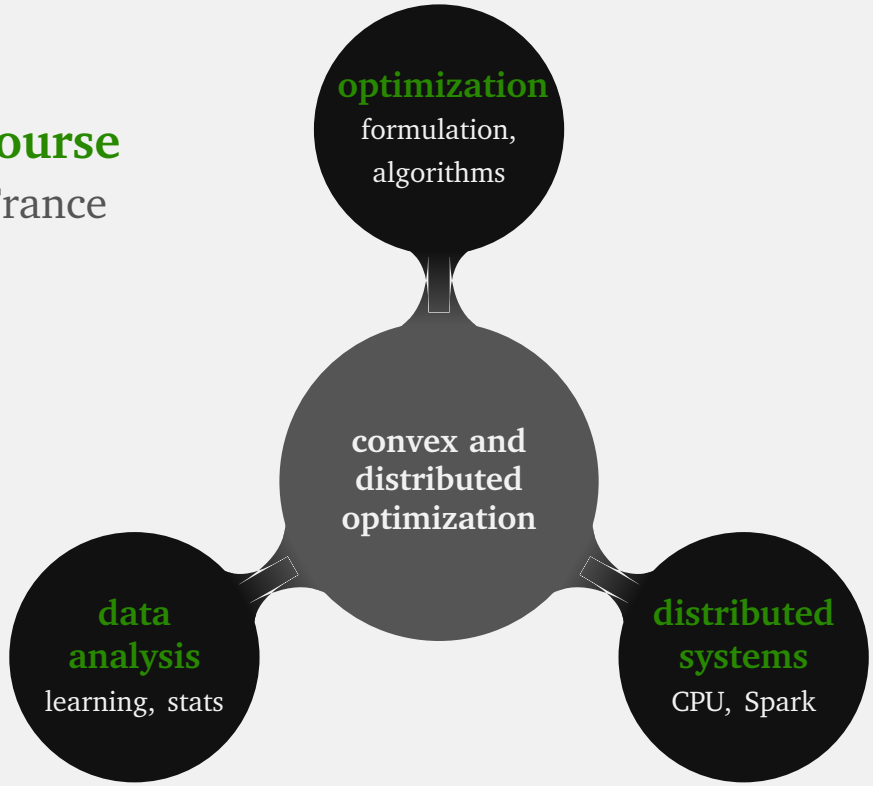
Leveraging on the **new distributed systems**



- Hardware improvements
  - Explosion of available computing resources (data centers, cloud)
  - Improvement of multicore infrastructure and networks
- Software improvements
  - developed by a wide scientific community and powerful industrial partners (Google, Facebook, Twitter)

Positioning of this course

original and unique in France



Positioning of the course

- About this course

Optimization & Learning

topic positioning

- not a course on distributed systems
  but we manipulate the hottest technologies of this domain
- not a standard optimization course
  rather a data analysis-related optimization course
- not a course on stats or machine learning
  but we discuss standard learning problems

contents

- not a maths course – but requires some maths agility
- not an algorithmic course – but requires some programming skills

prerequisites

- basic programming skills in Python (check-out online tutorials if necessary)
- basic knowledge in matrix calculus (matrix operations, norms) and differential calculus (definition and manipulation of gradients...)
- basic ideas on optimization (e.g. convex functions, gradient algorithm...)
  Refresh your mind about the refresher course!

Extended **subtitle** could be:

algorithmic aspects of optimization
for data analysis applications

Main **objectives** of the course:

- present optimization algorithms that scale up to high dimensions: stochastic, incremental, coordinate, random, and distributed algorithms
- implement them efficiently on data problems with high-level tools currently used in big data companies
- provide a complementary viewpoint on data analysis from an optimization perspective

5 lectures in amphitheater: (most of them: old-school way, on blackboard...)

- Lecture 1: advanced first-order optimization methods
- Lecture 2: stochastic optimization methods
- Lecture 3: distributed optimization framework
- Lecture 4: distributed computing with Spark
- Lecture 5: communication-efficient distributed learning

3 tutorials on machines:

- Tutorial 1: (variance-reduced) stochastic gradient descent
- Tutorial 2: parsing and manipulating data
- Tutorial 3: distributed sparse logistic regression

objectives of the tutorials:

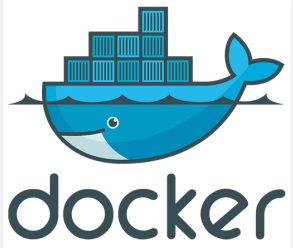
- understand the basics of optimization algorithms in large-scale settings
- review standard learning models and interpret numerical results
- programming: play with the hottest big data technologies

we work on Jupyter notebooks with Python,
Spark for computation, and Docker for installation

Spark (v2.0.1, october 2016)

- open-source distributed computing framework
- high-level paradigm (higher than MPI, OpenMP...) that automatically adapts to underlying hardware infrastructure
- is becoming one of the main big data technology (with thousand of developers) adopted by Twitter, Facebook, Google, Amazon...

Docker



- open-source project that automates
the deployment of applications inside software "containers"
- container $\simeq$ small virtual machines = provides an environment with a full OS and all softwares and libraries needed
- our docker contains a linux system + python, pyspark, jupyter...
- nothing else to install and everyone has the same soft environment

Report on tutorials (by group of < 3)

- report on the accomplished work on tutorials
  (with tables, plots, comments... but no code !)
- with highlights on chosen aspects

  Examples: learning (interpretation of results, other models...), maths (proof of related results, theoretical analysis of special cases,...) or numerical extensions
- in a very open format – before christmas break

Presentation of a research article (by group of ≤ 3)
(only constraint on groups: intersection of your two groups is reduced to yourself)

- list of various articles (theoretical, algorithmical, computations, or applications-oriented)
- oral presentation of $\sim$ 8 mins
- again, in a very open format – beginning of January

Find our own way to valorize your work !

Final grade : 1/2 report + 1/2 article

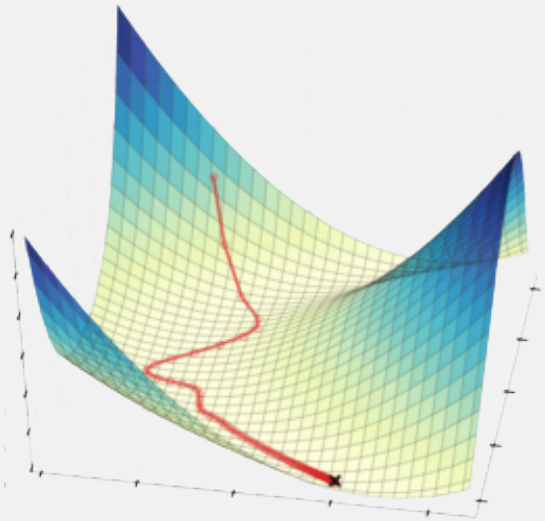
Positioning of the course

About this course

- Optimization & Learning

Mathematical Optimization

- branch of applied maths
  (theory, algorithms, software, modeling)
- being revolutionized by its interactions with data analysis
  (computational statistics and machine learning)
- specific problems (size, structure,...)
  $\rightarrow$ active domain of research/dev.



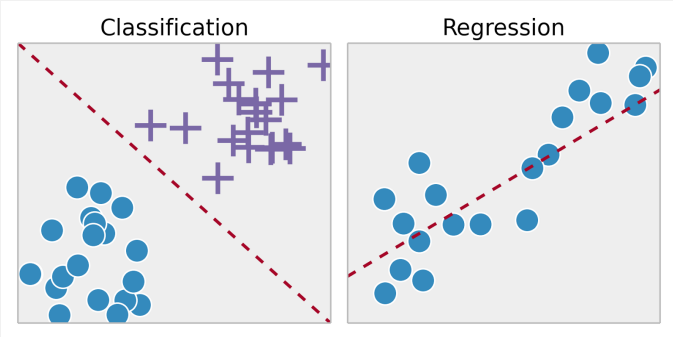
Example: numerical engine of (supervised) learning

- optimization algorithms work on (large) datasets
- to compute parameters of learning models
- used afterward to predict, classify, or make a decision

- **Data:** n observations (a_i, y_i) $i = 1, \dots, n$
  feature $a_i \in \mathbb{R}^d$ label $y_i \in \mathbb{R}$ or $\in \{0, 1\}$
- **Prediction** function $h(\cdot, x)$ parametrized by x (usually w in learning, β in stats)
- mapping from a new data $a \in \mathbb{R}^d$ to a prediction $y = h(a, x)$
- two popular examples of prediction functions

1) Linear prediction: $h(a, x) = \langle a, x \rangle$

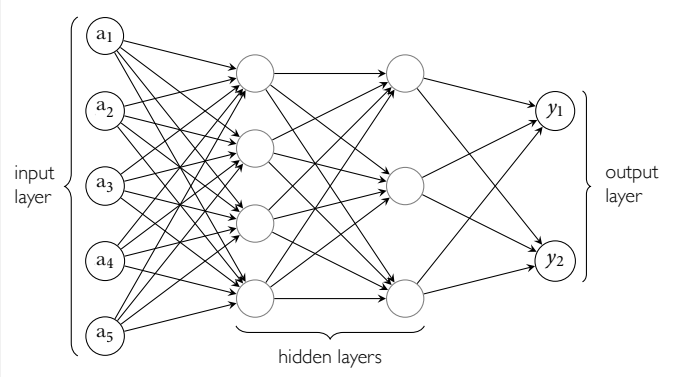
$(h(a, x) = \langle \phi(a), x \rangle$ for kernel methods)



2) Highly non-linear prediction by artificial **neural networks**

$$h(a, x) = \langle x_m, \sigma(\langle x_{m-1}, \cdots \sigma(\langle x_1, a \rangle)) \rangle \rangle$$

$\sigma(z) = 1/(1 + \exp(-z))$ or $\tanh(z)$
$\sigma(z) = \max\{0, z\}$ ReLU



Goal: compute the parameter $x^\star \in \mathbb{R}^d$ which explains **at best** the data.

- Prediction should be "close" to labels on known data (=training data set)

$$h(a_i, x) \simeq y_i \qquad \text{for } i = 1, \dots, n$$

- should generalize to unseen data (testing set)

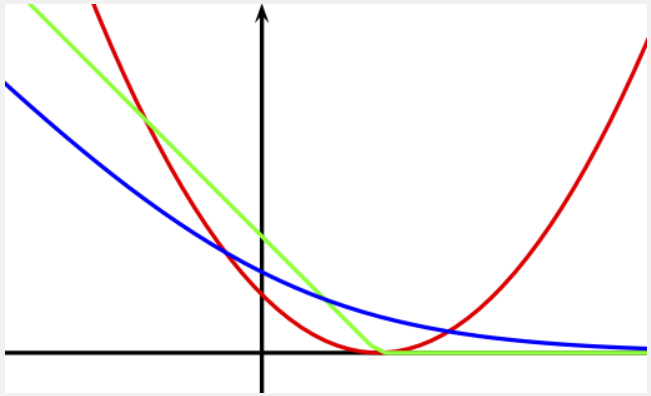
Measure of the difference with a loss function $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$

$\ell(y, z) = \frac{1}{2}(y - z)^2$ (least-square)

$\ell(y, z) = |y - z|$ (robust regression)

$\ell(y, z) = \log(1 + \exp(-yz))$ (logistic regr.)

$\ell(y, z) = \max\{0, 1 - yz\}$ (SVM classif)



Learning is optimizing $\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, h(a_i, x)\big)$

- Assume (x_i, y_i) are (iid) samples from a random variable $(a, y) \in \mathbb{R}^{d+1}$
- Underlying problem: minimization of the expectation of the error

$$\min_{x \in \mathbb{R}^d} \ \mathbb{E}\left[\ell\big(y, h(a, x)\big)\right]$$

- Learning problem = empirical estimation of the expected error

$$\min_{x \in \mathbb{R}^d} \ \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, h(a_i, x)\big)$$

- In theory: convergence when $n \to 0$ (weak law of large numbers)
- In practice: only finite data... independence assumption questionable...
- To remember: empirical risk minimization is **not** the must
  because the statistical error is neglected !
- Practical consequences for optimization algorithms

> try to find (quickly) solutions with medium (not high) precision
>
> be cautious and add a regularization term

Regularized empirical risk minimization

$$\min_{x \in \mathbb{R}^d} \; f(x) \;=\; \frac{1}{n} \sum_{i=1}^{n} \ell\big(y_i, h(a_i, x)\big) \;+\; \lambda\, \Omega(x)$$

Two extreme cases of regularization parameter: λ close to 0 and λ large

Interests of the regularisation term Ω

- avoid over-fiiting on known data to better generalize to new data
- promote a special structure on the optimal learning parameter
  E.g. sparsity with $\Omega(x) = \|x\|_1$
- help solving the optimization problem
  E.g. strong-convexity of $\Omega(x) = {\|x\|_2}^2$

Example: linear model $h(a, x) = \langle a, x \rangle$, least-square loss and ℓ_2 regularization

$$\min_{x \in \mathbb{R}^d} \; \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} (y_i - \langle a_i, x \rangle)^2 + \frac{\lambda}{2} {\|x\|_2}^2 \;=\; \frac{1}{2n} {\|Ax - y\|_2}^2 + \frac{\lambda}{2} {\|x\|_2}^2$$

Matrix $A \in \mathbb{R}^{n \times d}$ with lines ${a_i}^\top$

Often the objective function is **differentiable** (h, ℓ, and Ω differentiable) with easy-to-compute gradient (explicit expression or efficient algorithms)

Examples:

- back-propagation for neural networks (automatic differentiation)
- ridge regression $\nabla f(x) = \left(\frac{1}{n}A^\top A + \lambda I\right) x - \frac{1}{n}A^\top y$

Often the objective function is **differentiable** (h, ℓ, and Ω differentiable) with easy-to-compute gradient (explicit expression or efficient algorithms)

Examples:

- back-propagation for neural networks (automatic differentiation)
- ridge regression $$\nabla f(x) = \left(\frac{1}{n}A^\top A + \lambda I\right)x - \frac{1}{n}A^\top y$$

But not always... we may prefer **nonsmooth** losses or regularizers

Examples:

- Classification by SVM (support vector machines)
$$\min_{x\in\mathbb{R}^d} \ \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i {a_i}^\top x\} \ + \ \frac{\lambda}{2}\|x\|_2^2$$
- Lasso regression with $\|\cdot\|_1$ for sparse solutions (compressed sensing)
$$\min_{x\in\mathbb{R}^d} \ \frac{1}{2n}\|Ax - y\|_2^2 \ + \ \lambda\|x\|_1$$

Consequence: we will (briefly) discuss nonsmooth (first-order) algorithms

In this course, we focus on optimization problems of the form

$$\min_{x \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x)$$

where:

- we minimize over x, the parameter of the learning model
- f_i measure the quality of the prediction on a data block (data-fidelity term)
- r allows better learning (regularization term)
- functions f_i and r are smooth or nonsmooth
  (with simple first-order information, as gradient or prox...)

How to solve ?

→ just apply usual (first-order) optimization algorithms ??

In this course, we focus on optimization problems of the form

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x) + r(x)$$

where:

- we minimize over x, the parameter of the learning model
- f_i measure the quality of the prediction on a data block (data-fidelity term)
- r allows better learning (regularization term)
- functions f_i and r are smooth or nonsmooth
  (with simple first-order information, as gradient or prox...)

How to solve ?

→ just apply usual (first-order) optimization algorithms ??

Difficulty is not in the form but in the size !

- Volume of data (n large, *big data*),
- complexity of models (d large/h complex, *big models*)

Example with smooth f_i (and $r = 0$)

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

Apply gradient descent

$$x_{k+1} \;=\; x_k - \gamma_k \nabla f(x_k) \;=\; x_k - \frac{\gamma_k}{n} \sum_{i=1}^{n} \nabla f_i(x_k)$$

Costly operation in large dimension !

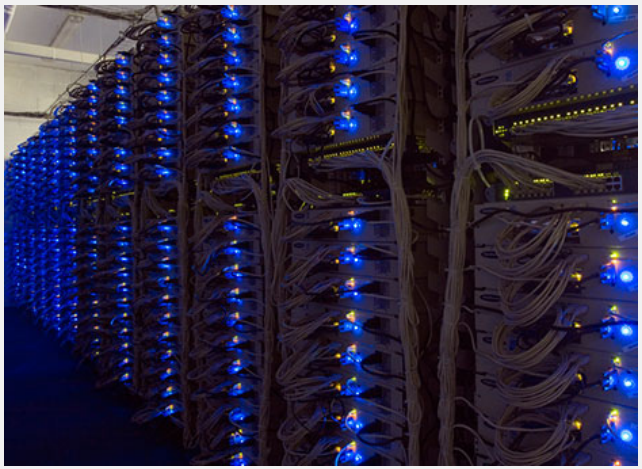
- Need a pass on all the data to compute all $\nabla f_i(x_k)$
- Compute the sum of n vectors of $\mathbb{R}^d$

How to solve **with scalable algorithms for d or n large ?**

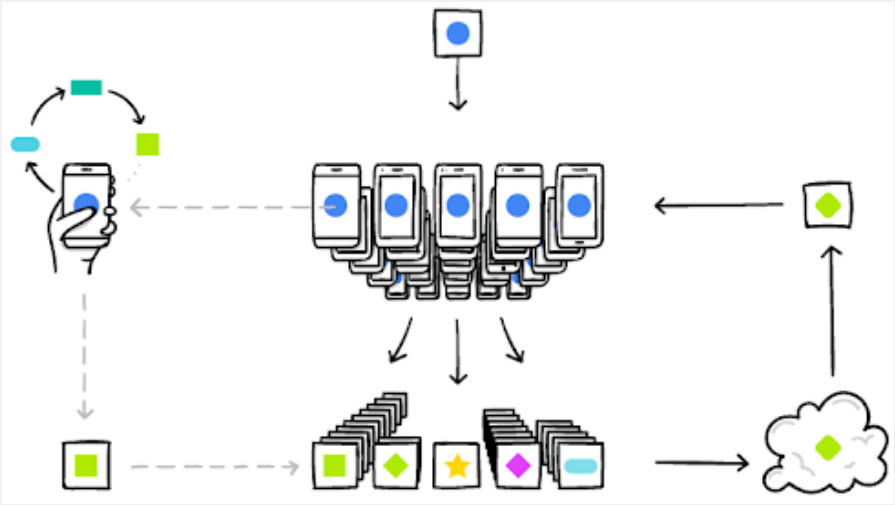
key idea: do not treat all the data together

1. incremental algorithms (one iteration = one block of data)
2. distributed algorithms (when data is stored on different machines)

Distributed optimization/learning...
...on a variety of computing systems ! e.g.:



computing clusters,
cloud services,...



federated learning
(all training data remains on device)

Exciting research on **flexible** optimization algorithms

(**data-robust**, **delay-tolerant**, **privacy-preserving**...)

We will discuss some underlying ideas associated to this research... more next!
Get ready by looking back at refresher courses (lectures + tutorials !)