Optimization Techniques for Big Data Analysis

Chapter 1. Introduction

Master of Science in Signal Theory and Communications
Dpto. de Señales, Sistemas y Radiocomunicaciones
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

2024

Motivation

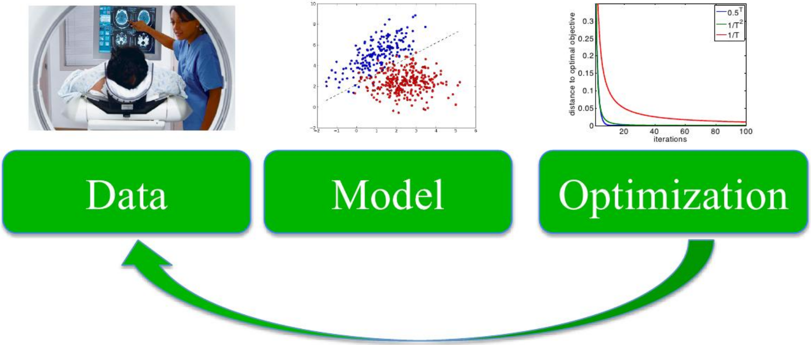
Optimization is a supporting technology in many numerical computation-related research fields, such as machine learning, signal processing, industrial design, and operation research.

machine learning = representation + optimization + evaluation

Motivation

Optimization is a supporting technology in many numerical computation-related research fields, such as machine learning, signal processing, industrial design, and operation research.

machine learning = representation + optimization + evaluation



Some basic concepts

Regarding optimization in machine learning, it is worth to mention:

- Mathematical Modelling:
  Defining and modelling the problem

Some basic concepts

Regarding optimization in machine learning, it is worth to mention:

- Mathematical Modelling:
  Defining and modelling the problem
- Computational optimization:
  Algorithms to solve these optimization problems optimally or near optimially

Some basic concepts

Regarding optimization in machine learning, it is worth to mention:

- Mathematical Modelling:
  Defining and modelling the problem
- Computational optimization:
  Algorithms to solve these optimization problems optimally or near optimially
- Continuous optimization:
  It often appears as a relaxation of risk/error minimisation problems. The *learning* problem in many parametrized models involves Continuous Optimization.

Some basic concepts

Regarding optimization in machine learning, it is worth to mention:

- Mathematical Modelling:
  Defining and modelling the problem
- Computational optimization:
  Algorithms to solve these optimization problems optimally or near optimially
- Continuous optimization:
  It often appears as a relaxation of risk/error minimisation problems. The *learning* problem in many parametrized models involves Continuous Optimization.
- Discrete optimization:
  It occurs in inference problems in structured spaces, such as Feature selection, Data subset selection, Data summarization, Architecture search etc.

Continuous optimization in ML

- Supervised Learning: Logistic Regression, Least Square, Support Vector Machines, Deep Models.
- Unsupervised Learning: k-Means Clustering, Principal Component Analysis.
- Contextual bandits and Reinforcement learning: Soft-Max estimators and Policy Exponential Models.
- Recommender systems: Matrix Completion, Non-Negative Matrix Factorization, Collaborative Filtering.

Continuous optimization in ML

- Supervised Learning: Logistic Regression, Least Square, Support Vector Machines, Deep Models.
- Unsupervised Learning: k-Means Clustering, Principal Component Analysis.
- Contextual bandits and Reinforcement learning: Soft-Max estimators and Policy Exponential Models.
- Recommender systems: Matrix Completion, Non-Negative Matrix Factorization, Collaborative Filtering.

Countless ML libraries available implement all kinds of optimization algorithms (Tensorflow, PyTorch, Scipy, Sklearn, Vowpal Wabbit, ...)

Large scale optimization

Different setups and contexts can lead to different algorithm requirements:

- Data collection: Whether data is collected in one node or in several.

Large scale optimization

Different setups and contexts can lead to different algorithm requirements:

- Data collection: Whether data is collected in one node or in several.
- Accelerated optimization: Increase the convergence rate without making much stronger assumptions.

Large scale optimization

Different setups and contexts can lead to different algorithm requirements:

- Data collection: Whether data is collected in one node or in several.
- Accelerated optimization: Increase the convergence rate without making much stronger assumptions.
- Parallel optimization: Run the algorithm in different cores or threats of a node.

Large scale optimization

Different setups and contexts can lead to different algorithm requirements:

- Data collection: Whether data is collected in one node or in several.
- Accelerated optimization: Increase the convergence rate without making much stronger assumptions.
- Parallel optimization: Run the algorithm in different cores or threats of a node.
- Distributed optimization: Run the algorithm in different nodes using different portions of the data. The solution could be exact or approximated.

Large scale optimization

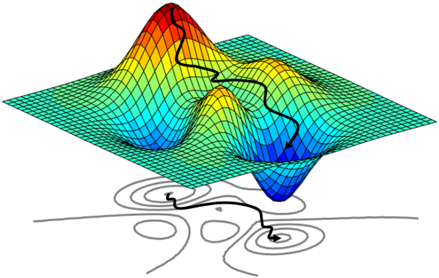
Different setups and contexts can lead to different algorithm requirements:

- Data collection: Whether data is collected in one node or in several.
- Accelerated optimization: Increase the convergence rate without making much stronger assumptions.
- Parallel optimization: Run the algorithm in different cores or threats of a node.
- Distributed optimization: Run the algorithm in different nodes using different portions of the data. The solution could be exact or approximated.
- Federated learning: Run the algorithm in different nodes without sharing any data among nodes.

Nomenclature

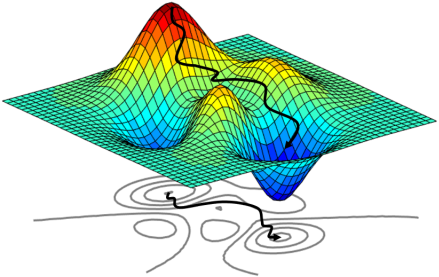
- For the most part through this course, we will use n as the number of training instances and d as the number of dimensions (features).
- Function: $f(\cdot)$
- Scalar: x; single-input function $f(x)$
- d-dimensional vector: $\mathbf{x} \in \mathbb{R}^{\mathbf{d}}$; $\mathbf{x}_i = [x_{1i}, x_{2i}, \cdots, x_{di}]$
- multi-input function $f(\mathbf{x})$; vector-valued function $\mathbf{f}(\mathbf{x})$
- Vector Space: $\mathcal{X}$
- Vector Norm: $||\mathbf{x}||_L$
- Inner product: Given two vectors $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$, define the inner product $\langle \mathbf{w}, \mathbf{x} \rangle = \sum_{j=1}^{d} \mathbf{w}_j \mathbf{x}_j = \mathbf{w}^T \mathbf{x}$
- Random variable X; Matrix: $\mathbf{X}$

Mathematical optimization



"The selection of the best element, with regard to some criterion, from some set of available alternatives" [1].

Mathematical optimization



"The selection of the best element, with regard to some criterion, from some set of available alternatives" [1].

So typically we are given a problem like the following:

$$\hat{x} = \arg\max_{x} f(x) \;\; \text{s.t. } g(x) < a \tag{1}$$

- $f(\cdot)$ function subject to optimization.
- $x \in \mathcal{X}$ variables/parameters that need to be adjusted.
- $\mathcal{X}$ is the search space. $\hat{x}$ is the optimum.
- $g(\cdot)$ restrictions. $f|_{\mathcal{G}}$, where $\mathcal{G} \subseteq \mathcal{X}$; $\mathcal{G}$ is the feasible set.

ML set-up

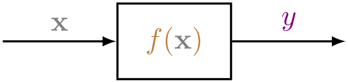
A typical ML task can be seen like:

$$\xrightarrow{\mathbf{x}} \boxed{f(\mathbf{x})} \xrightarrow{y}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a function $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$,

ML set-up

A typical ML task can be seen like:



where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a function $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$, and often we assume that:

$$y = f_{\theta}(\mathbf{x}) + \varepsilon; \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

ML set-up

A typical ML task can be seen like:

$$\xrightarrow{\mathbf{x}} \boxed{f(\mathbf{x})} \xrightarrow{y}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a function $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$, and often we assume that:

$$y = f_{\theta}(\mathbf{x}) + \varepsilon; \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

Using a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^{n}\}$ and some criterion $J(\theta)$ we approximate $f(\cdot)$. This allow us to make predictions of y^* given a new $\mathbf{x}^*$.

ML set-up

A typical ML task can be seen like:

$$\xrightarrow{\mathbf{x}} \boxed{f(\mathbf{x})} \xrightarrow{y}$$

where $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$; so we want to find a function $f(\cdot)$ that performs the mapping $f : \mathcal{X} \to \mathbb{R}$, and often we assume that:

$$y = f_\theta(\mathbf{x}) + \varepsilon; \;\; \varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{2}$$

Using a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$ and some criterion $J(\theta)$ we approximate $f(\cdot)$. This allow us to make predictions of y^* given a new $\mathbf{x}^*$.

Training $f_\theta(\cdot)$ corresponds to the optimization of $J(\theta)$!

Supervised Learning: Modelling

- Data: Given training examples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector and y_i is the label.
- Model: Denote the Model by $f_\theta(\mathbf{x})$ with θ being the parameters of the model. e.g. $f_\theta(\mathbf{x}) = \theta^T \mathbf{x}$
- Loss Functions: The loss function ℓ tries to measure the distance between $f_\theta(\mathbf{x}_i)$ and y_i.

Supervised Learning: Modelling

- Data: Given training examples $\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector and y_i is the label.
- Model: Denote the Model by $f_\theta(\mathbf{x})$ with θ being the parameters of the model. e.g. $f_\theta(\mathbf{x}) = \theta^T \mathbf{x}$
- Loss Functions: The loss function ℓ tries to measure the distance between $f_\theta(\mathbf{x}_i)$ and y_i.

The generic problem denoted as the *expected loss function* can be expressed as:

$$\arg\min_\theta \mathcal{L}(\theta) = \mathbb{E}[\ell_\theta(\mathbf{x}, y)] + \lambda r(\theta) \tag{3}$$

where ℓ is the *instantaneous loss*, $r(\cdot)$ is the *regularizer* and λ a regularization factor.

Empirical loss functions

In most of this course' cases, we are dealing with a supervised learning problem, so given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$ the loss function takes the form:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell_\theta(\mathbf{x}_i, y_i) + \lambda r(\theta) \tag{4}$$

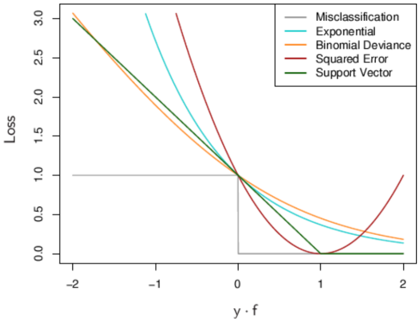
Empirical loss functions

In most of this course' cases, we are dealing with a supervised learning problem, so given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\}$ the loss function takes the form:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell_\theta(\mathbf{x}_i, y_i) + \lambda r(\theta) \tag{4}$$

Examples of ℓ:

- Logistic loss: $\log(1 + \exp(-y_i f_\theta(\mathbf{x}_i)))$
- Hinge Loss: $\max\{0, 1 - y_i f_\theta(\mathbf{x}_i)\}$
- Absolute Error: $|f_\theta(\mathbf{x}_i) - y_i|$
- Least Squares: $(f_\theta(\mathbf{x}_i) - y_i)^2$



Instantaneous loss function

A common simplification approximates the original function by its instantaneous value:

$$\mathcal{L}(\theta) = \ell_\theta(\mathbf{x}_i, y_i) + \lambda r(\theta) \tag{5}$$

- The minimization is performed by using an instantaneous version of the original gradient.

Instantaneous loss function

A common simplification approximates the original function by its instantaneous value:

$$\mathcal{L}(\theta) = \ell_\theta(\mathbf{x}_i, y_i) + \lambda r(\theta) \tag{5}$$

- The minimization is performed by using an instantaneous version of the original gradient.
- When minimizing it, we will have to consider the gradient noise that will affect convergence properties.

Instantaneous loss function

A common simplification approximates the original function by its instantaneous value:

$$\mathcal{L}(\theta) = \ell_\theta(\mathbf{x}_i, y_i) + \lambda r(\theta) \tag{5}$$

- The minimization is performed by using an instantaneous version of the original gradient.
- When minimizing it, we will have to consider the gradient noise that will affect convergence properties.
- This simplification makes the algorithm very attractive in big data applications, both because of hardware requirements and for distributed settings.

Main characteristics of our problems

- Too many data points, so it is needed efficient iterative solutions.

Main characteristics of our problems

- Too many data points, so it is needed efficient iterative solutions.
  - Stochastic optimization

Main characteristics of our problems

- Too many data points, so it is needed efficient iterative solutions.
  - Stochastic optimization
  - Algebraic implementations of the algorithms are crucial.

Main characteristics of our problems

- Too many data points, so it is needed efficient iterative solutions.
    - Stochastic optimization
    - Algebraic implementations of the algorithms are crucial.
- High dimensional data: Because of the dimensionality of the parameter spaces, we can't afford second-order optimization methods. First-order or simplified second-order methods are requested.

Main characteristics of our problems

- Too many data points, so it is needed efficient iterative solutions.
  - Stochastic optimization
  - Algebraic implementations of the algorithms are crucial.
- High dimensional data: Because of the dimensionality of the parameter spaces, we can't afford second-order optimization methods. First-order or simplified second-order methods are requested.
- Distributed processing is highly desirable.

Acknowledgments

I would like to acknowledge several sources I have used to create slides

- Rishabh Iyer's course at University of Texas, DA
  https://github.com/rishabhk108/OptimizationML
- Martin Jaggi & Nicolas Flammarion's course at EPFL
  https://github.com/epfml/OptML_course

Questions?

[1] Melvyn W Jeter. *Mathematical programming: an introduction to optimization.* Routledge, 2018.

Thank You

Julián D. Arias-Londoño
julian.arias@upm.es