Second-order Inertial Algorithms for Smooth and Non-smooth Large-scale Optimization

Camille Castera

University of Tübingen

Faculty of Mathematics

Part 1: Joint work with J. Bolte, C. Févotte, E. Pauwels

Part 2: Joint work with H. Attouch, J. Fadili, and P. Ochs

Dynamical Systems and Semi-algebraic Geometry: Optimization and Deep Learning

Dalat, July 2023

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Introduction: Machine Learning & Optimization for Training Neural Networks

Machine Learning (ML)

Concept

Predict some **output** variable $y \in \mathbb{R}^D$ from an **input** variable $x \in \mathbb{R}^M$.

ML Models

Function f, **parameterized** by $\theta \in \mathbb{R}^P$. We want $f(x, \theta) = y$.

Neural networks (NN): a class of ML models

Compositional structure in *layers* $(f_\ell)_{\ell \in \{1,\ldots,L\}}$:

$$f = f_L \circ f_{L-1} \circ \ldots \circ f_1.$$

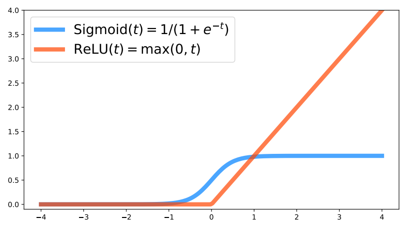
Typical layer: $f_1(x, \theta_1) = g_1(W_1 x + b_1)$, where,

– W_1 is a matrix, b_1 a vector,

– g_1 is an **activation function** (non-linear).

– **Parameter** $\theta \in \mathbb{R}^P$ of f: coefficients of the matrices and vectors of the layers.

Common activation functions



– **Deep learning:** ML with neural networks.

Training neural networks: an optimization problem

Central question: How to select the parameter θ?

Loss function

– **Training dataset:** a collection of N examples

$$(x_n, y_n)_{n \in \{1,\dots,N\}}.$$

– **Loss function:** sum of the errors made by a neural network on the training set, e.g.,

$$\mathcal{J}(\theta) \overset{\text{e.g.}}{=} \frac{1}{N}\sum_{n=1}^{N} \|f(x_n,\theta) - y_n\|_2^2.$$

Training is an optimization problem

We seek $\theta \in \mathbb{R}^P$ which minimizes $\mathcal{J}$:

$$\min_{\theta\in\mathbb{R}^P} \mathcal{J}(\theta) \overset{\text{def}}{=} \min_{\theta\in\mathbb{R}^P} \frac{1}{N}\sum_{n=1}^{N} \mathcal{J}_n(\theta).$$

Main topic of the talk:

Designing new algorithms to train neural networks, i.e., to minimize $\mathcal{J}$.

Temporary simplification

We first assume that $\mathcal{J}$ is twice differentiable.

Denote $\nabla\mathcal{J}$ and $\nabla^2\mathcal{J}$ its gradient and Hessian matrix.

Basic minimization algorithm

To minimize a function we can follow the direction of *steepest descent*.

Gradient descent (GD)

Given θ_0, $\gamma > 0$, for all $k \in \mathbb{N}$,

$$\theta_{k+1} = \theta_k - \gamma \nabla \mathcal{J}(\theta_k).$$

GD only requires being able to evaluate $\nabla \mathcal{J}$.

A difficult practical optimization problem

Selecting $\theta \in \mathbb{R}^P$ that minimizes $\mathcal{J}(\theta) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{J}_n(\theta)$ was unachievable for decades.

Drawback: High computational cost

Neural networks:

– Usually have millions of parameters (P large).

– Are trained on large datasets (N large, $\mathcal{J}$ is the sum of many terms).

Assets

– Modern computers, in particular **GPUs**.

– The **backpropagation** (Rumelhart and Hinton, 1986): efficient way of evaluating the $\nabla \mathcal{J}_n$'s.

Consequences

– **Possible but expensive** to compute $\mathcal{J}$ and $\nabla \mathcal{J}$. Expensive to store $\nabla \mathcal{J}$.

– **Unreasonable** to compute $\nabla^2 \mathcal{J}$.

Mini-batch algorithms

GD remains expensive, another strategy is preferable.

Mini-batch (MB) sub-sampling

Let $\mathrm{B} \subset \{1, \ldots, N\}$ a sub-sample of indices, define

$$\mathcal{J}_{\mathrm{B}} = \frac{1}{\mathrm{Card}(\mathrm{B})} \sum_{n \in \mathrm{B}} \mathcal{J}_n \text{ and } \nabla \mathcal{J}_{\mathrm{B}} = \frac{1}{\mathrm{Card}(\mathrm{B})} \sum_{n \in \mathrm{B}} \nabla \mathcal{J}_n.$$

Approximations of $\mathcal{J}$ and $\nabla \mathcal{J}$ with a *sub-sample* of the training set.

Stochastic gradient descent (SGD)

Given θ_0, step-sizes $\gamma_k > 0$ and **random** mini-batches $\mathrm{B}_k \subset \{1, \ldots, N\}$, for $k \in \mathbb{N}$,

$$\theta_{k+1} = \theta_k - \gamma_k \nabla \mathcal{J}_{\mathrm{B}_k}(\theta_k).$$

Consequences of mini-batches

– Empirically **faster** in most large-scale problems (Bottou and Bousquet, 2008).

– Brings **new hardships** (imprecise update directions, need for using vanishing step-sizes, etc.).

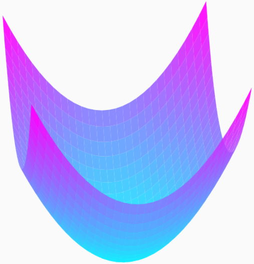
A possibly difficult optimization problem

Non-convexity

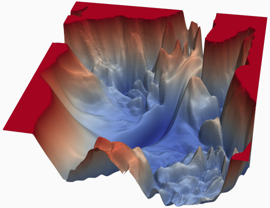
The compositional structure of NNs makes $\mathcal{J}$ **non-convex**. Critical points need not be minima.

Non-smoothness

Loss function $\mathcal{J}$ may be **non-differentiable** (e.g., due to ReLU). Gradient is not always well defined.



Usually easier to minimize this function...



than this one!

In deep learning, $\mathcal{J}$ is more likely to be similar to right figure.[1]

[1] Right figure credited to `https://www.cs.umd.edu/~tomg/projects/landscapes/`

Motivation: New algorithms

– SGD is widely used in DL. Two main variations: **Momentum** and **adaptivity**.

– ADAM (Kingma and Ba, 2015) combines adaptivity and momentum (10^6 citations).

Goal: Making use of second-order information. Why?

– Potentially **faster training**.

– **Tuning** step-sizes.– And more (escaping saddles, etc.).

Many limitations: High computational cost, non-smoothness (lack of existence of 1st and 2nd-order derivatives), mini-batch sub-sampling, etc.

Coming next:

– *A **practical** second-order algorithm, built despite the limitations.*

– ***Convergence guarantees** in this restrained theoretical framework.*

Overview

Part 1: INNA, An Inertial Newton Algorithm for Deep Learning

ODEs and optimization algorithms

Assume **temporarily** that $\mathcal{J}$ is twice differentiable. Optimization algorithms with small step-sizes can be modeled by **Ordinary Differential Equations** (ODEs).

Discrete gradient descent

$$\theta_{k+1} = \theta_k - \gamma \nabla \mathcal{J}(\theta_k)$$

$$\iff \frac{\theta_{k+1} - \theta_k}{\gamma} + \nabla \mathcal{J}(\theta_k) = 0$$

$\xrightarrow{\gamma \to 0}$

$\xleftarrow{\text{discretize}}$

Gradient system

$$\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \nabla \mathcal{J}(\theta(t)) = 0, \forall t > 0$$

The same can be done for Newton's method

Newton's method

$$\theta_{k+1} = \theta_k - \gamma \left(\nabla^2 \mathcal{J}(\theta_k)\right)^{-1} \nabla \mathcal{J}(\theta_k)$$

$\longrightarrow$

$\longleftarrow$

Continuous Newton's method

$$\nabla^2 \mathcal{J}(\theta(t)) \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \nabla \mathcal{J}(\theta(t)) = 0$$

Strategy

We start from an ODE to build an algorithm.

ODEs and optimization algorithms

Assume **temporarily** that $\mathcal{J}$ is twice differentiable. Optimization algorithms with small step-sizes can be modeled by **Ordinary Differential Equations** (ODEs).

Discrete gradient descent

$$\theta_{k+1} = \theta_k - \gamma \nabla \mathcal{J}(\theta_k)$$
$$\iff \frac{\theta_{k+1} - \theta_k}{\gamma} + \nabla \mathcal{J}(\theta_k) = 0$$

$\xrightarrow{\gamma \to 0}$ / $\xleftarrow{\text{discretize}}$

Gradient system

$$\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \nabla \mathcal{J}(\theta(t)) = 0, \forall t > 0$$

The same can be done for Newton's method

Newton's method

$$\theta_{k+1} = \theta_k - \gamma \left(\nabla^2 \mathcal{J}(\theta_k)\right)^{-1} \nabla \mathcal{J}(\theta_k)$$

$\longrightarrow$ / $\longleftarrow$

Continuous Newton's method

$$\nabla^2 \mathcal{J}(\theta(t)) \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \nabla \mathcal{J}(\theta(t)) = 0$$

Strategy

We start from an ODE to build an algorithm. **Why not mix both methods?**

DIN (Alvarez, Attouch, Bolte, and Redont, 2002)

Take two hyper-parameters $\alpha \geq 0$ and $\beta > 0$. Consider for all $t > 0$,

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t) + \alpha\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \beta\nabla^2\mathcal{J}(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \nabla\mathcal{J}(\theta(t)) = 0. \qquad \text{(DIN)}$$

DIN mixes **gradient descent**, **Newton's method**, and an acceleration term connected to **momentum** methods (Polyak, 1964).

The solution of DIN are attracted by critical points

The limits (as $t \to \infty$) of the solutions of DIN (when they exist) are critical points of $\mathcal{J}$ (Alvarez et al., 2002).

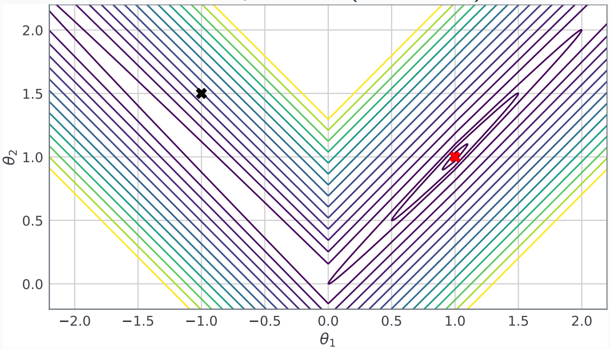
Related work (non-exhaustive)

Extensions and properties of DIN further studied by many (Attouch, Peypouquet, and Redont, 2014, 2016; Boț, Csetnek, and László, 2021). Link with Nesterov's method (Alecsa, László, and Pința, 2021; Shi, Du, Jordan, and Su, 2021), etc.

Numerical illustration

$$\underbrace{\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t)}_{\text{Accel.}} + \underbrace{\alpha\,\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Friction}} + \underbrace{\beta\,\nabla^2\mathcal{J}(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Transverse damping}} + \underbrace{\nabla\mathcal{J}(\theta(t))}_{\text{Gravity}} = 0 \qquad \text{(DIN)}$$

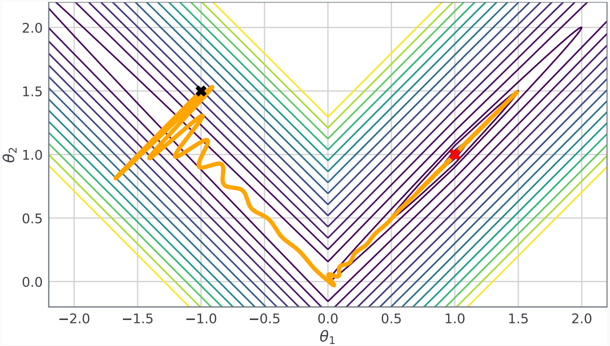
Landscape of $\mathcal{J}$ (level lines)



Minimization of $\mathcal{J}(\theta_1, \theta_2) = 100\,(\theta_2 - |\theta_1|)^2 + |1 - \theta_1|$

(inspired by H.H. Rosenbrock and Y. Nesterov)

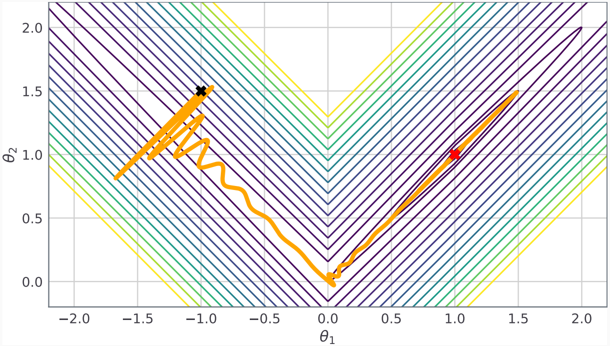
$$\underbrace{\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t)}_{\text{Accel.}}+\underbrace{\alpha\,\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Friction}}+\underbrace{\beta\,\nabla^2\mathcal{J}(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Transverse damping}}+\underbrace{\nabla\mathcal{J}(\theta(t))}_{\text{Gravity}}=\ 0 \qquad \text{(DIN)}$$



Minimization of $\mathcal{J}(\theta_1,\theta_2) = 100\,(\theta_2 - |\theta_1|)^2 + |1-\theta_1|$
(inspired by H.H. Rosenbrock and Y. Nesterov)

Numerical illustration

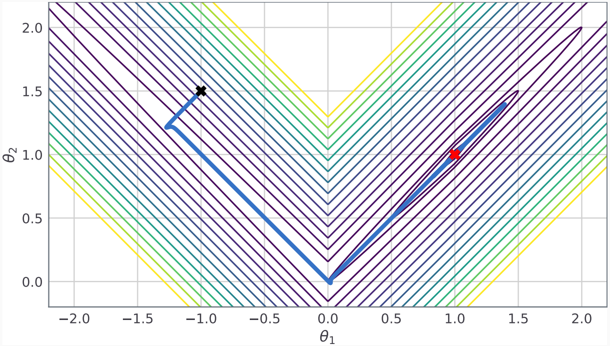
$$\underbrace{\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t)}_{\text{Accel.}} + \underbrace{\alpha\,\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Friction}} + \underbrace{\beta\,\nabla^2\mathcal{J}(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Transverse damping}} + \underbrace{\nabla\mathcal{J}(\theta(t))}_{\text{Gravity}} = 0 \qquad \text{(DIN)}$$



Minimization of $\mathcal{J}(\theta_1, \theta_2) = 100\,(\theta_2 - |\theta_1|)^2 + |1 - \theta_1|$
(inspired by H.H. Rosenbrock and Y. Nesterov)

$$\underbrace{\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t)}_{\text{Accel.}} + \underbrace{\alpha\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Friction}} + \underbrace{\beta\,\nabla^2\mathcal{J}(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Transverse damping}} + \underbrace{\nabla\mathcal{J}(\theta(t))}_{\text{Gravity}} = 0 \qquad \text{(DIN)}$$

With larger β:

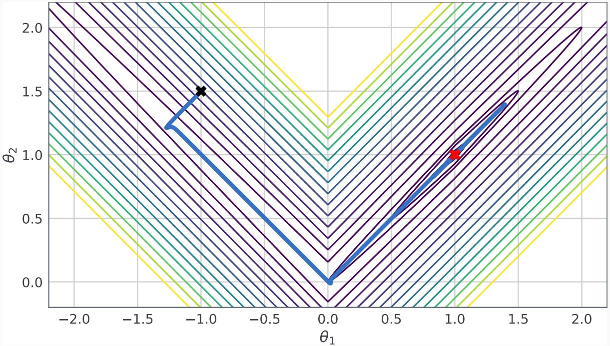


Minimization of $\mathcal{J}(\theta_1, \theta_2) = 100\,(\theta_2 - |\theta_1|)^2 + |1 - \theta_1|$
(inspired by H.H. Rosenbrock and Y. Nesterov)

Numerical illustration

$$\underbrace{\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t)}_{\text{Accel.}} + \underbrace{\alpha \frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Friction}} + \underbrace{\beta \nabla^2 \mathcal{J}(\theta(t)) \frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Transverse damping}} + \underbrace{\nabla \mathcal{J}(\theta(t))}_{\text{Gravity}} = 0 \qquad \text{(DIN)}$$

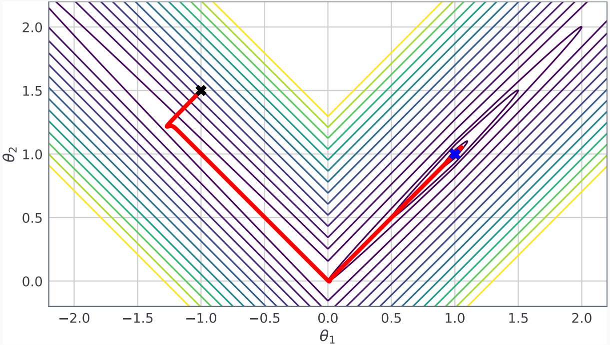
With larger β:



Minimization of $\mathcal{J}(\theta_1, \theta_2) = 100\,(\theta_2 - |\theta_1|)^2 + |1 - \theta_1|$
(inspired by H.H. Rosenbrock and Y. Nesterov)

$$\underbrace{\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t)}_{\text{Accel.}} + \underbrace{\alpha\,\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Friction}} + \underbrace{\beta\,\nabla^2\mathcal{J}(\theta(t))\frac{\mathrm{d}\theta}{\mathrm{d}t}(t)}_{\text{Transverse damping}} + \underbrace{\nabla\mathcal{J}(\theta(t))}_{\text{Gravity}} = 0 \qquad \text{(DIN)}$$

With larger α:



Minimization of $\mathcal{J}(\theta_1, \theta_2) = 100\,(\theta_2 - |\theta_1|)^2 + |1 - \theta_1|$
(inspired by H.H. Rosenbrock and Y. Nesterov)

How to deal with the Hessian term $\nabla^2 \mathcal{J}$?

Equivalent form of DIN (Alvarez et al., 2002)

For $\mathcal{J}$ twice differentiable, this **second-order** ODE...

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}(t) + \alpha \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \beta \nabla^2 \mathcal{J}(\theta(t)) \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + \nabla \mathcal{J}(\theta(t)) = 0, \tag{DIN}$$

can be rewritten into an equivalent first-order system with **no explicit Hessian term!**

$$\begin{cases} \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) + \beta \nabla \mathcal{J}(\theta(t)) & = 0 \\ \frac{\mathrm{d}\psi}{\mathrm{d}t}(t) + (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) & = 0 \end{cases} \tag{g-DIN}$$

Computational cost is now affordable

For **differentiable** functions we could discretize g-DIN, and incorporate mini-batches.

Remark: g-DIN was also used by others at a similar time (Chen and Luo, 2019; Attouch, Chbani, Fadili, and Riahi, 2020).

Main challenge: how to adapt this to **non-differentiable** loss functions?

Handling non-smoothness

From now on, $\mathcal{J}$ is **not assumed to be differentiable** anymore.

– However, $\mathcal{J}$ is locally Lipschitz continuous hence **differentiable almost everywhere** (by Rademacher's theorem).

– Let $\theta \in \mathbb{R}^P$, for convex function, the sub-differential of $\mathcal{J}$ at θ is,

$$\left\{ v \in \mathbb{R}^P \middle| \forall \psi \in \mathbb{R}^P, \mathcal{J}(\psi) - \mathcal{J}(\theta) \geq \langle v, \psi - \theta \rangle \right\}.$$

– Not suited for some non-convex functions (e.g., $t \in \mathbb{R} \mapsto -|t|$).

Sub-differential suited for deep learning: Clarke sub-differential (Clarke, 1990)

Denote R the set of points where $\mathcal{J}$ is differentiable. For $\theta \in \mathbb{R}^P$,

$$\partial \mathcal{J}(\theta) \stackrel{\text{def}}{=} \operatorname{conv} \left\{ v \in \mathbb{R}^P \mid \exists (\theta_k)_{k \in \mathbb{N}} \in \mathrm{R}^{\mathbb{N}}, \text{ s.t. } \theta_k \xrightarrow[k \to \infty]{} \theta \text{ and } \nabla \mathcal{J}(\theta_k) \xrightarrow[k \to \infty]{} v \right\}.$$

This is formally the **convex hull of the limits of all neighboring gradients.**

Next step: mini-batch sub-sampling for sub-differentials.

Non-smooth mini-batch sub-sampling

Sum of sub-differentials of two functions g_1, g_2 $\quad (\mathrm{dom}(g_1) = \mathrm{dom}(g_2) = \mathbb{R})$

Convex case: $\partial(g_1 + g_2) = \partial g_1 + \partial g_2$.

Non-convex case: $\partial(g_1 + g_2) \subset \partial g_1 + \partial g_2$.

Example

$g : t \in \mathbb{R} \mapsto |t| - |t| = 0$. Clearly $\partial g(0) = \{0\}$, whereas,

$$\partial(|0|) + \partial(-|0|) = [-1, 1] + [-1, 1] = [-2, 2] \supset \{0\}.$$

Consequence: $\partial \mathcal{J}$ not suited for mini-batch sub-sampling!

– **Ideally** we would approximate $\partial \mathcal{J} = \partial \left(\frac{1}{N} \sum_{n=1}^{N} \mathcal{J}_n \right)$.

– **Unfortunately** we can only approximate $D\mathcal{J} \stackrel{\text{def}}{=} \frac{1}{N} \sum_{n=1}^{N} \partial \mathcal{J}_n$.

– This models what practitioners do (often unconsciously).

So we use $D\mathcal{J}$ **in place** of $\partial \mathcal{J}$ to keep theory close to practice.

Mini-batch sub-sampling: for $\mathrm{B} \subset \{1, \dots, N\}$, $\quad D\mathcal{J}_{\mathrm{B}} \stackrel{\text{def}}{=} \frac{1}{\mathrm{Card}(\mathrm{B})} \sum_{n \in \mathrm{B}} \partial \mathcal{J}_n$.

On computing $D\mathcal{J}$ and $\partial\mathcal{J}_n$

Simplification

Here I am assuming that computing elements of each Clarke subdifferential $\partial\mathcal{J}_n$ is "easy" so that $D\mathcal{J}_{\mathrm{B}} = \frac{1}{\mathrm{Card(B)}} \sum_{n\in\mathrm{B}} \partial\mathcal{J}_n$ is also easy to evaluate.

It is not!

But not today

Reality

In practice $\partial\mathcal{J}_n$ is replaced by the output of **automatic differentiation**: the PyTorch/Tensorflow implementation of backpropagation.

Can be taken into account in convergence analyses (Bolte and Pauwels, 2020a,b).

Incorporate $D\mathcal{J}$ into g-DIN: new differential inclusion

$$\begin{cases} \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) & +(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) + \beta\nabla\mathcal{J}(\theta(t)) = 0 \\ \frac{\mathrm{d}\psi}{\mathrm{d}t}(t) & +(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) = 0 \end{cases}, \quad \text{for all } t \in (0, +\infty).$$

The INNA algorithm

Incorporate $D\mathcal{J}$ into g-DIN: new differential inclusion

$$\begin{cases} \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) & +(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) + \beta D\mathcal{J}(\theta(t)) \ni 0 \\ \frac{\mathrm{d}\psi}{\mathrm{d}t}(t) & +(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) = 0 \end{cases}, \quad \text{for a.e.} \quad t \in (0, +\infty).$$

The INNA algorithm

Incorporate $D\mathcal{J}$ into g-DIN: new differential inclusion

$$\begin{cases} \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) & +(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) + \beta D\mathcal{J}(\theta(t)) \ni 0 \\ \frac{\mathrm{d}\psi}{\mathrm{d}t}(t) & +(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) = 0 \end{cases}, \quad \text{for a.e. } \ t \in (0, +\infty).$$

Finally: explicit Euler discretization and non-smooth mini-batch sub-sampling.

INNA: an Inertial Newton Algorithm

Choose hyper-parameters $\alpha \geq 0$, $\beta > 0$, a sequence of step-sizes $(\gamma_k)_{k\in\mathbb{N}}$, nonempty mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$ i.i.d uniformly at random (e.g., with fixed cardinality), and an initialization $(\theta_0, \psi_0) \in \mathbb{R}^P \times \mathbb{R}^P$.

Iterate for $k = 0, \ldots,$
$$\begin{cases} v_k & \in D\mathcal{J}_{\mathsf{B}_k}(\theta_k) = \frac{1}{\mathrm{Card}(\mathsf{B}_k)} \sum_{n\in\mathsf{B}_k} \partial\mathcal{J}_n(\theta_k) \\ \theta_{k+1} & = \theta_k + \gamma_k\left(-(\alpha - \frac{1}{\beta})\theta_k - \frac{1}{\beta}\psi_k - \beta v_k\right) \\ \psi_{k+1} & = \psi_k + \gamma_k\left(-(\alpha - \frac{1}{\beta})\theta_k - \frac{1}{\beta}\psi_k\right) \end{cases}$$

Convergence of INNA for tame functions

The loss function $\mathcal{J}$ is tame

Functions are tame when their graph can be described with a finite number of polynomials, logarithms, exponentials, etc. (extension of semi-algebraic functions).

Theorem (C., Bolte, Févotte, Pauwels, 2019)

If each $\mathcal{J}_n$ is locally Lipschitz continuous, tame, if $\gamma_k = o(1/\log k)$ and $\sum_k \gamma_k = +\infty$. Assume that the iterates are bounded almost surely. Then, a sequence $(\theta_k, \psi_k)_{k\in\mathbb{N}}$ generated by INNA is such that almost surely,

- **any accumulation point $(\bar{\theta}, \bar{\psi})$ is such that $0 \in D\mathcal{J}(\bar{\theta})$,**
- **the sequence $(\mathcal{J}(\theta_k))_{k\in\mathbb{N}}$ of values of the loss function converges.**

Why does it work?

Formally, when the step-size $\gamma_k \xrightarrow[k\to\infty]{} 0$, “INNA behaves like g-DIN” (Benaïm, 1999; Benaïm, Hofbauer, and Sorin, 2005).

Proof sketch 1/2: connecting discrete and continuous dynamics

Let $(\theta_k, \psi_k)_{k\in\mathbb{N}}$ generated by INNA, with step-sizes $(\gamma_k)_{k\in\mathbb{N}}$ and mini-batches $(\mathsf{B}_k)_{k\in\mathbb{N}}$.

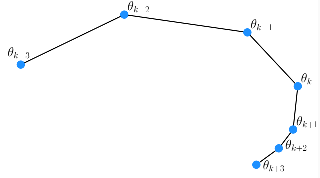
INNA

$$\begin{cases} v_k & \in D\mathcal{J}_{\mathsf{B}_k}(\theta_k) \\ \frac{\theta_{k+1}-\theta_k}{\gamma_k} & = -(\alpha - \frac{1}{\beta})\theta_k - \frac{1}{\beta}\psi_k - \beta v_k \\ \frac{\psi_{k+1}-\theta_k}{\gamma_k} & = -(\alpha - \frac{1}{\beta})\theta_k - \frac{1}{\beta}\psi_k \end{cases}$$

g-DIN

$$\begin{cases} \frac{\mathrm{d}\theta}{\mathrm{d}t}(t) + (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) + \beta D\mathcal{J}(\theta(t)) \ni 0 \\ \frac{\mathrm{d}\psi}{\mathrm{d}t}(t) + (\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t) = 0 \end{cases}$$

We compare the solutions of g-DIN with INNA by **interpolating** the iterations.



Main elements to connect the dynamics

– By assumption $(\theta_k, \psi_k)_{k\in\mathbb{N}}$ is **bounded**, $\gamma_k = o(1/\log k)$ and $\sum_{k\in\mathbb{N}} \gamma_k = \infty$.

– The mini-batches are chosen so that $\forall k \in \mathbb{N}$, $\mathbb{E}[v_k \mid \mathsf{B}_{k-1}, \ldots, \mathsf{B}_0] \in D\mathcal{J}(\theta_k)$.

– Interpolated INNA behaves asymptotically like solutions of g-DIN (it is a *bounded perturbed solution* of g-DIN, Benaïm et al. (2005, Thm. 1.3&1.4)).

Proof sketch 2/2: Lyapunov analysis and limit

Lyapunov function

Let (θ, ψ) solution of g-DIN. Define for $t > 0$,

$$E(\theta(t), \psi(t)) = (1 + \alpha\beta)\mathcal{J}(\theta(t)) + \frac{1}{2}\|(\alpha - \frac{1}{\beta})\theta(t) + \frac{1}{\beta}\psi(t)\|^2.$$

By differentiating $E(\theta, \psi)$ (chain rule for tame functions, see Davis et al. (2020)), we show that for a.e. $t > 0$, $E(\theta(t), \psi(t)) \leq E(\theta(0), \psi(0))$ with **strict inequality** if

$$(\theta(0), \psi(0)) \notin \mathsf{S} = \left\{(\tilde{\theta}, \tilde{\psi}) \in \mathbb{R}^P \times \mathbb{R}^P \mid 0 \in D\mathcal{J}(\tilde{\theta}), \tilde{\psi} = (1 - \alpha\beta)\tilde{\theta}\right\}.$$

Sard's lemma (based on Bolte, Daniilidis, Lewis, and Shiota 2007)

The loss function $\mathcal{J}$ is tame so has a finite number of D-critical values. Since $E \equiv (1 + \alpha\beta)\mathcal{J}$ on S, $E(\mathsf{S})$ **is finite**.

Finally combine (Benaïm et al., 2005, Theorem 3.6&3.27)

– $E(\theta, \psi)$ is a Lyapunov function $\implies$ the limit set L of bounded perturbed solutions is in S (so any accumulation point is D-critical).

– $E(\mathsf{S})$ is finite $\implies$ $E(\mathsf{L})$ is a singleton, and $E \equiv (1 + \alpha\beta)\mathcal{J}$ on S.

More guarantees

Rates of convergence

Linear or Sub-linear rates of convergence for the solutions of the differential inclusion (g-DIN). Based on the **Kurdyka-Łojasiewicz** (KL) inequality (Bolte, Daniilidis, Lewis, and Shiota, 2007). We provide a recipe to extend this to a large class of dynamical systems.

Escape of strict saddles

Almost sure convergence to minimizers of $\mathcal{J}$, under stronger assumptions (smooth, deterministic setting) (Castera, 2021).
Not true for usual Newton's methods (Dauphin et al., 2014), but also holds for a few of them (Truong et al., 2020).

Important disclaimer

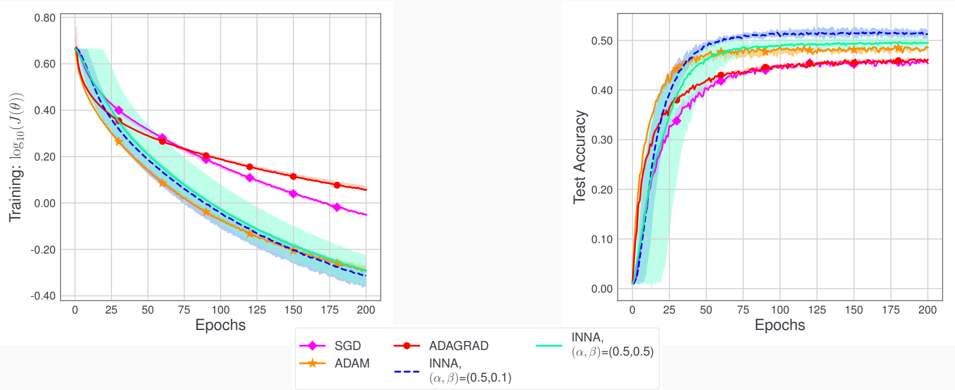
Many tricks and tuning hidden behind DL problems (NN, dataset, batchnorm, weight decay, hyper-parameter tuning, etc.).

It is hardly possible to accurately benchmark optimizers in DL (Sivaprasad et al., 2019).

Results for image classification on CIFAR-100

Classification of images in 100 categories with NiN (moderately large network, Lin et al. 2014).

Training loss (lower is better) | **Test accuracy** (higher is better)



More experiments in the paper

– Comparisons on more problems.

– Hyper-parameters sensitivity.

– Faster convergence with slower step-size schedule.

Conclusions on INNA

Take-home messages

- A second-order algorithm for training neural networks.
- Convergence guarantees in a stochastic non-smooth non-convex framework.
- Promising experiments and good generalization on DL problems.

What's next?

- In the experiments, the gap of performance is satisfying but not tremendous.
- What should we improve? Is DIN the right ODE? Is our discretization good?

Let's go back to the smooth setting to find out.

Part 2: Continuous Newton-like Methods featuring Inertia and Variable Mass

Let us take a step back

More favorable framework for optimization than deep learning

– Usual optimization notations: $\theta \to x$, $\mathcal{J} \to f$.

– Assume f to be strongly convex and smooth (C^2).

– Consider (for now) only the ODE framework.

Recall DIN?

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) + \alpha \quad \frac{\mathrm{d}x}{\mathrm{d}t}(t) + \beta \nabla^2 f(x(t)) \frac{\mathrm{d}x}{\mathrm{d}t}(t) + \nabla f(x(t)) = 0, \qquad \text{(DIN} \qquad)$$

Typically $\alpha(t) = 3/t$ (Nesterov, 1983; Su et al., 2014).

Observation

DIN-like ODEs feature a Newtonian term, yet no guarantee of faster convergence compared to first-order methods.

This raises an important question

Are DIN-like ODEs really Newton-like methods?

More favorable framework for optimization than deep learning

- Usual optimization notations: $\theta \to x$, $\mathcal{J} \to f$.
- Assume f to be strongly convex and smooth (C^2).
- Consider (for now) only the ODE framework.

Recall DIN? Here is a popular extension

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) + \alpha(t)\frac{\mathrm{d}x}{\mathrm{d}t}(t) + \beta\nabla^2 f(x(t))\frac{\mathrm{d}x}{\mathrm{d}t}(t) + \nabla f(x(t)) = 0, \qquad \text{(DIN-AVD)}$$

Typically $\alpha(t) = 3/t$ (Nesterov, 1983; Su et al., 2014).

Observation

DIN-like ODEs feature a Newtonian term, yet no guarantee of faster convergence compared to first-order methods.

This raises an important question

Are DIN-like ODEs really Newton-like methods?

A new system VM-DIN-AVD

Introduce an additional parameter $\varepsilon(t)$ in front of the acceleration

$$\varepsilon(t)\frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) + \alpha(t)\frac{\mathrm{d}x}{\mathrm{d}t}(t) + \beta\nabla^2 f(x(t))\frac{\mathrm{d}x}{\mathrm{d}t}(t) + \nabla f(x(t)) = 0, \tag{VM-DIN-AVD}$$

where ε and α are two non-negative functions of $[0, +\infty[$.

Intuition

$$\beta\nabla^2 f(x_N(t))\frac{\mathrm{d}x_N}{\mathrm{d}t}(t) + \nabla f(x_N(t)) = 0. \tag{CN}$$

$$\alpha(t)\frac{\mathrm{d}x_{LM}}{\mathrm{d}t}(t) + \beta\nabla^2 f(x_{LM}(t))\frac{\mathrm{d}x_{LM}}{\mathrm{d}t}(t) + \nabla f(x_{LM}(t)) = 0. \tag{LM}$$

When $\varepsilon(t)$ and $\alpha(t)$ vanish asymptotically, the solution x of (VM-DIN-AVD) *should* get close either to x_N or x_{LM}.

Prior work: a partial answer (Alvarez et al., 2002)

Setting and Notation

We fix initial conditions $x(0) = x_0$, $\frac{\mathrm{d}x}{\mathrm{d}t}(0) = \dot{x}_0$.

Prior work: fixed ε and $\alpha = 0$ (Alvarez et al., 2002)

$$\varepsilon \frac{\mathrm{d}^2 x}{\mathrm{d}t^2}(t) + \cancel{\alpha \frac{\mathrm{d}x}{\mathrm{d}t}(t)} + \cancel{\beta} \nabla^2 f(x(t)) \frac{\mathrm{d}x}{\mathrm{d}t}(t) + \nabla f(x(t)) = 0. \qquad (\varepsilon\text{-DIN})$$

There exists $C > 0$ such that **for all** $0 \leq \varepsilon \leq 1$ and for all $t \geq 0$,

$$\|x(t) - x_N(t)\| \leq C\sqrt{\varepsilon}.$$

Can we say more? What about $\alpha \neq 0$, $\varepsilon(t)$, $\alpha(t)$, etc.?

Convergence to CN under moderate viscous damping

Theorem (C., Attouch, Fadili, Ochs, 2023)

In the case $\varepsilon(t) \geq \alpha(t)$:

Under mild assumptions, there exist $C_0, C_1, C_2 \geq 0$ s.t. $\forall(\varepsilon, \alpha)$ for which the assumptions hold the solution x of (VM-DIN-AVD) is s.t. $\forall t \geq 0$,

$$\|x(t) - x_N(t)\| \leq C_0 e^{-\frac{t}{\beta}} \varepsilon_0 \|\dot{x}_0\| + C_1 \sqrt{\varepsilon(t)} + C_2 \int_{s=0}^{t} e^{\frac{1}{\beta}(s-t)} \sqrt{\varepsilon(s)} \, \mathrm{d}s.$$

Corollary: simpler bound

$$\|x(t) - x_N(t)\| \leq C_0 e^{-\frac{t}{\beta}} \varepsilon_0 \|\dot{x}_0\| + C_3 \sqrt{\varepsilon(t)}.$$

NB: The case $\varepsilon(t) < \alpha(t)$ is also covered but is more involved.

Illustration: a control perspective

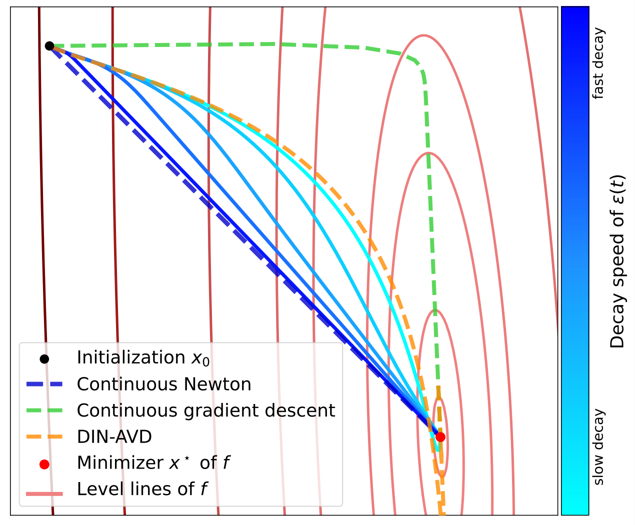
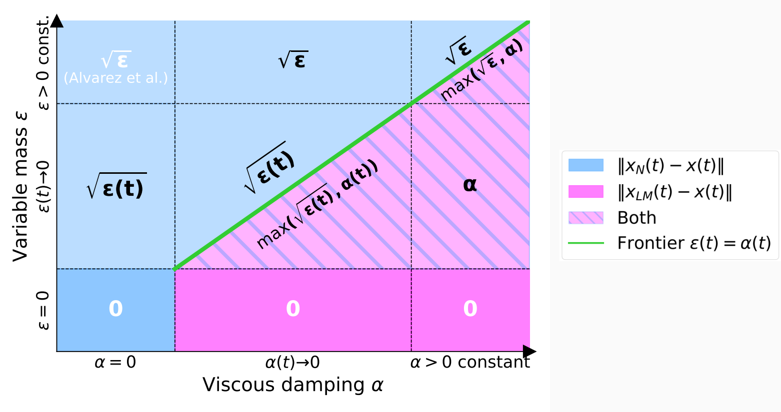


Illustration of the result on a 2D quadratic function in the case $\varepsilon(t) \geq \alpha(t)$

$$\varepsilon(t)\frac{\mathrm{d}^2x}{\mathrm{d}t^2}(t) + \alpha(t)\frac{\mathrm{d}x}{\mathrm{d}t}(t) + \beta\nabla^2 f(x(t))\frac{\mathrm{d}x}{\mathrm{d}t}(t) + \nabla f(x(t)) = 0 \qquad \text{(VM-DIN-AVD)}$$



- VM-DIN-AVD is really a second-order method, that we can control.
- This paves the way to the design of new algorithms.
- **But:** finding the right trade-off between cheap and efficient discretization is (extremely) challenging. I am currently working on it.

Associated publications

Journal papers

- **Part 1:** An Inertial Newton Algorithm for Deep Learning. C. Castera, J. Bolte, C. Févotte, and E. Pauwels (2021). In *Journal of Machine Learning Research (JMLR)*.
- Inertial Newton Algorithms Avoiding Strict Saddle Points. C. Castera (2021). In *arXiv 2111.04596*.
- **Part 2:** Continuous Newton-like Methods featuring Inertia and Variable Mass. C. Castera, H. Attouch, J. Fadili, P. Ochs (2023). *arXiv 2301.08726*.

Code repository

INNA is available as a ready-to-use optimizer for Pytorch and Tensorflow.

https://github.com/camcastera/INNA-for-DeepLearning

References

References

Alecsa, C. D., S. C. László, and T. Pința (2021). An extension of the second order dynamical system that models Nesterov's convex gradient method. *Applied Mathematics & Optimization 84*(2), 1687–1716.

Alvarez, F., H. Attouch, J. Bolte, and P. Redont (2002). A second-order gradient-like dissipative dynamical system with Hessian-driven damping: Application to optimization and mechanics. *Journal de Mathématiques Pures et Appliquées 81*(8), 747–779.

Attouch, H., Z. Chbani, J. Fadili, and H. Riahi (2020). First-order optimization algorithms via inertial systems with Hessian driven damping. *Mathematical Programming*.

Attouch, H., J. Peypouquet, and P. Redont (2014). A dynamical approach to an inertial forward-backward algorithm for convex minimization. *SIAM Journal on Optimization 24*(1), 232–256.

Attouch, H., J. Peypouquet, and P. Redont (2016). Fast convex optimization via inertial dynamics with Hessian driven damping. *Journal of Differential Equations 261*(10), 5734–5783.

Benaïm, M. (1999). Dynamics of stochastic approximation algorithms. In *Séminaire de Probabilités XXXIII*, pp. 1–68. Springer.

Benaïm, M., J. Hofbauer, and S. Sorin (2005). Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization 44*(1), 328–348.

Bolte, J., A. Daniilidis, A. Lewis, and M. Shiota (2007). Clarke subgradients of stratifiable functions. *SIAM Journal on Optimization 18*(2), 556–572.

Bolte, J. and E. Pauwels (2020a). Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, 1–33.

Bolte, J. and E. Pauwels (2020b). A mathematical model for automatic differentiation in machine learning. In *Advances in Neural Information Processing Systems (NIPS)*.

Boț, R. I., E. R. Csetnek, and S. C. László (2021). Tikhonov regularization of a second order dynamical system with Hessian driven damping. *Mathematical Programming 189*(1), 151–186.

Bottou, L. and O. Bousquet (2008). The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 161–168.

Castera, C. (2021). Inertial Newton algorithms avoiding strict saddle points. *preprint arXiv:2111.04596*.

Chen, L. and H. Luo (2019). First order optimization methods based on Hessian-driven Nesterov accelerated gradient flow. *arXiv preprint:1912.09276*.

Clarke, F. H. (1990). *Optimization and nonsmooth analysis*. SIAM.

Dauphin, Y. N., R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems (NIPS)*, Volume 27.

Davis, D., D. Drusvyatskiy, S. Kakade, and J. D. Lee (2020). Stochastic subgradient method converges on tame functions. *Foundations of Computational Mathematics 20*(1), 119–154.

Kingma, D. P. and J. Ba (2015). Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Lin, M., Q. Chen, and S. Yan (2014). Network in Network. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Nesterov, Y. (1983). A method for unconstrained convex minimization problem with the rate of convergence $o(1/k^2)$. In *Doklady an USSR*, Volume 269, pp. 543–547.

Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics 4*(5), 1–17.

Rumelhart, D. E. and G. E. Hinton (1986). Learning representations by back-propagating errors. *Nature 323*(9), 533–536.

Shi, B., S. S. Du, M. I. Jordan, and W. J. Su (2021). Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*.

Sivaprasad, P. T., F. Mai, T. Vogels, M. Jaggi, and F. Fleuret (2019). Optimizer benchmarking needs to account for hyperparameter tuning. *arXiv preprint:1910.11758*.

Su, W., S. Boyd, and E. Candes (2014). A differential equation for modeling Nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2510–2518.

Truong, T. T., T. D. To, T. H. Nguyen, T. H. Nguyen, H. P. Nguyen, and M. Helmy (2020). A fast and simple modification of Newton's method helping to avoid saddle points. *arXiv preprint arXiv:2006.01512*.

Second-order Inertial Algorithms for Smooth and Non-smooth Large-scale Optimization

Camille Castera
University of Tübingen
Faculty of Mathematics

Part 1: Joint work with J. Bolte, C. Févotte, E. Pauwels
Part 2: Joint work with H. Attouch, J. Fadili, and P. Ochs

Dynamical Systems and Semi-algebraic Geometry: Optimization and Deep Learning
Dalat, July 2023

EBERHARD KARLS UNIVERSITÄT TÜBINGEN