A mathematical model for automatic differentiation in machine learning

Jerome Bolte, Edouard Pauwels

A mathematical model for automatic differentiation in machine learning

Jérôme Bolte*
Toulouse School of Economics
Univ. Toulouse
Toulouse, France

Edouard Pauwels
IRIT, CNRS
Univ. Toulouse
Toulouse, France

Abstract

Automatic differentiation, as implemented today, does not have a simple mathematical model adapted to the needs of modern machine learning. In this work we articulate the relationships between differentiation of programs as implemented in practice and differentiation of nonsmooth functions. To this end we provide a simple class of functions, a nonsmooth calculus, and show how they apply to stochastic approximation methods. We also evidence the issue of artificial critical points created by algorithmic differentiation and show how usual methods avoid these points with probability one.

1 Introduction

Optimization algorithms based on backpropagation oracles, and more generally on automatic or algorithmic differentiation (AD) [41, 39], are one of the most widely used training tools for modern learning architectures [14, 32, 15, 18, 20, 3, 16]. They often rely on popular numerical implementations as TensorFlow or PyTorch [1, 36]. However, for nonsmooth, nonconvex losses, AD does not have a stable theory [23, 25, 26, 2, 30, 28, 29, 12], matching the actual practice. We wish to present a simple mathematical framework addressing this issue. Let us progressively explain our approach.

1.1 What is backpropagation?

Algorithmic differentiation acts on programs not on functions: To convey this fact we carry out a small experiment in TensorFlow [1] with the function $\mathrm{relu}\colon t \mapsto \max\{0, t\}$, see Appendix A.2 for implementation details. Algorithmic differentiation is displayed in Figure 1, in particular, we have $\mathrm{relu}'(0) = 0$. Consider the two functions

$$\mathrm{relu}_2\colon t \mapsto \mathrm{relu}(-t) + t, \qquad \mathrm{relu}_3\colon t \mapsto \frac{1}{2}(\mathrm{relu}(t) + \mathrm{relu}_2(t)).$$

As mathematical functions on $\mathbb{R}$ these are *equal to* relu. However TensorFlow returns $\mathrm{relu}_2'(0) = 1$ and $\mathrm{relu}_3'(0) = 1/2$ (Figure 1). Indeed, AD does not act on functions, but on their representations, i.e., on programs. Different programs implementing the same function may provide different results, beyond numerical precision; we refer to this as the spurious behaviour of AD for nonsmooth functions[2]. Let us explore this phenomenon further. The function $\mathrm{zero}\colon t \mapsto \mathrm{relu}_2(t) - \mathrm{relu}(t)$, outputs constantly 0 but AD gives $\mathrm{zero}'(0) = 1$. More generally, one can modify the value of the derivative of a given function at prescribed arguments (Figure 1). This may generate artificial critical points; for instance $x \to x - \mathrm{zero}$ is the identity but its derivative at 0 according to AD is 0.

*Authors in alphabetical order.

[2]The validity domain of AD is restricted in theory to smooth functions [23], yet it is common practice to use it for nonsmooth functions.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

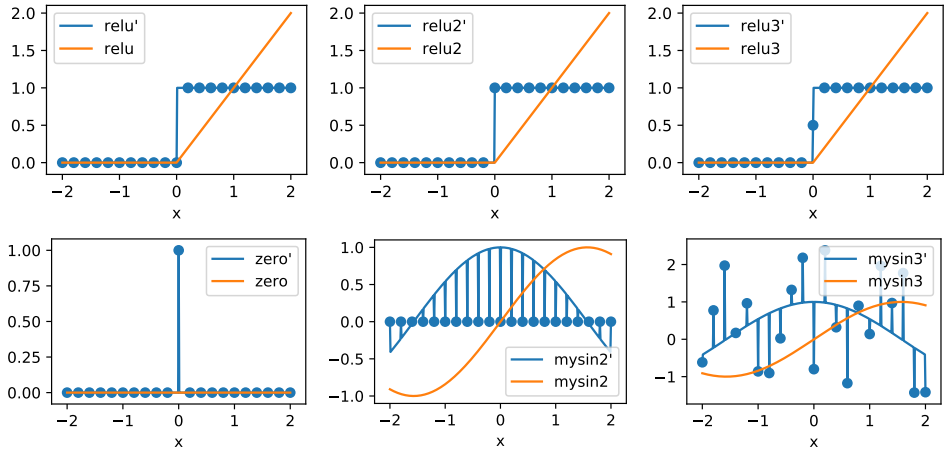


Figure 1: Top: AD applied to relu and two different implementations of the same function. Bottom: Algorithmic differentiation of a constant function, creation of artificial critical point or arbitrary derivatives at prescribed arguments for the sine function.

This discussion was limited to univariate functions, but these pathologies grow in size and in complexity when occurring in higher dimensions. Besides, as the "compositional depth" of functions increases the phenomenon gets more complex, making the geometry of artificial point difficult to grasp.

Canonical surjection between programs functions: Numerical programs combine basic mathematical functions within an algorithm and return an output. This can be understood in two ways:

- Computer science: it is a sequence of instructions with numerical inputs-outputs,
- Mathematics: the program is a function[3] of its arguments.

It is tempting to identify both, but functions can be represented by different programs. This defines a surjection $\mathcal{F}$ mapping a program to a function (in the class of functions "accessible through coding").

Algorithmic differentiation: As presented above, AD is an operation on programs, $\mathcal{A}$ which takes as argument a program and returns a program with the same input variables. This operation can be "pushed" to the space of functions using the canonical surjection $\mathcal{F}$. Remarkably, if we restrict ourselves to programs $\mathcal{P}$ which only smoothly combine smooth functions, then we have the following fundamental relation, depicted in Figure 2:

$$\mathcal{F}(\mathcal{A}(\mathcal{P})) = \nabla\mathcal{F}(\mathcal{P}). \tag{1}$$

In other words, algorithmic differentiation of a program which smoothly combines smooth functions, is equivalent, through the canonical surjection, to derivation.

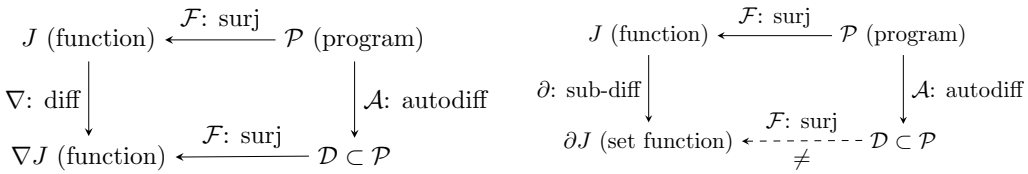


Figure 2: Left: Algorithmic differentiation applied to programs combining smooth functions in a smooth way, the diagram commutes. Right: Algorithmic differentiation in nonsmooth settings, connection with known notion of generalized derivative is much less clear.

[3] In the usual mathematical sense.

However practitioners use AD and backpropagation beyond smooth programs with nonsmooth elementary functions or program branching for instance. Can we find a proper operational interpretation of this widespread practice?

Algorithmic differentiation cannot be represented through a variational operator At first, it is tempting to simply use AD to induce a differential operator on functions generalizing classical differentiation. This operator, say ∂^A, should:

(a) encompass the outputs of algorithmic differentation for all functions

(b) be such that 0 is an element of $\partial^A(\mathrm{relu})$ at 0.

Unfortunately such an operator does not exist:

Theorem 1 (Algorithmic differentiation does not induce an operator on functions) *There is no nontrivial operator on functions satisfying* (a) *and* (b).

1.2 Contribution and related work

We address this impossibility result and provide a class of functions together with an operational nonsmooth differential calculus which is able to cope with spurious behaviours.

Elementary selections and selection derivatives: We introduce a new class of nonsmooth nonconvex functions, encompassing most objective functions met in machine learning, having appealing stability properties. This allows us to define simple differential objects called selection derivatives. Selection derivatives turn out to have an operational calculus adapted to the analysis of many learning methods, as backpropagation or stochastic first order methods. They thus provide an operational model to capture nonsmooth AD as implemented in current numerical software.

Algorithmic differentiation, algorithms This framework allows to formalize properly the relationships between, functions, algorithmic differentiation and capture the corresponding notion of critical points as met in practice. These characterize the set of attractors (limit points) for stochastic approximation algorithms based on nonsmooth backpropagation [37, 4, 31, 5, 13]. It is important to stress that these attractors, which models sharply the whole scope of AD-induced stationnarity, are different from the traditional notions as Clarke criticality [17, 38, 20]. This is described in Theorems 3 and 4.

Avoidance of traps: As sketched above and in the introduction AD produces artificial critical points, i.e. stationary points which are not Clarke critical. These points have a parasitic nature which could be detrimental to training purposes, were they met in practice. We show that randomly initialized mini-batch stochastic gradient method do not lead to artificial critical points (Theorem 4). This result applies to modern machine learning software libraries based on AD [1, 36], seen as performing operation over the reals, without any modification. Although AD may have unpredictable behavior in nonsmooth contexts, both theoretically and numerically, this result justifies theoretically that the practical impact is somewhat negligible in the context of common machine learning usage.

Related work: Spurious behaviour of AD in nonsmooth context has been investigated in [23, 25, 26, 27, 2, 30, 12]. In particular, [27, 30] considers qualification conditions allowing to construct AD algorithms which compute proper Clarke subgradients [17, 38, 20]. However qualification is extremely hard to check and almost impossible to enforce in practice. Let us also mention [2] which uses the notion of lexicographic derivatives, but, at this day, algorithmic computations are limited to forward mode for the moment which is of little use in machine learning.

[23, 25, 27, 26, 28, 29] use settings closer to ours. Piecewise smooth functions, selection derivatives and their variational properties are extensively described in [40]. Our approach differs because we adopt more stringent definitions and rigidity assumptions, which allows in turn for much stronger properties. For instance, we fully treat backward algorithmic differentiation which is the most useful tool in machine learning.

Altogether, our contribution is an accessible and elementary framework for the conservative fields recently introduced in [12], without explicitly requiring the introduction of semialgebraic geometry and o-minimal structures [21, 19].

Stochastic approximation algorithms [37, 4, 31, 5, 13] are widely used in machine learning contexts [39, 14, 35, 32, 15, 18, 16]. For example [20] describes asymptotics of stochastic subgradient algorithms in nonsmooth, nonconvex settings. In contrast, we do not assume access to subgradients and instead explicitly model the behaviour of AD in optimization contexts. Our convergence results are based on [12], complemented by a new result on "the avoidance of critical traps" in the line of [7] in the context of long run convergence.

Notations The ambient space is Euclidean $\mathbb{R}^p$. For each k, e_k is the k-th vector of the canonical basis. We use $D\colon \mathbb{R}^m \rightrightarrows \mathbb{R}^q$ for set valued functions, *i.e* functions from $\mathbb{R}^m$ to the subsets of $\mathbb{R}^q$. The convex hull of $A \subset \mathbb{R}^p$ is denoted by $\mathrm{conv}(A)$. All proofs are postponed to the Appendix.

2 Basic piecewise differentiable functions and selection gradient

We introduce a simple but vast class of functions that model rigorously the machine learning models and losses for applications such as deep learning.

Definition 1 (Elementary (log-exp) functions) *Elementary (log-exp) functions* are functions on $\mathbb{R}^p$ described by a finite compositional expression involving basic operations, $+, -, \times, /$ as well as affine mappings, exponential and logarithms, inside their domain of definition. We denote by $\mathcal{E}$ the set of elementary functions in any dimension p.

Examples include polynomials, logistic loss, boosting loss, Gaussian likelihood. Observe that the corresponding functions are C^∞ smooth on their open domains. Note also that if log and exp are not present we obtain the field of rational functions. See Remark 1 in Appendix A.3.

Definition 2 (Elementary index) *$s\colon \mathbb{R}^p \mapsto \{1, \dots, m\}$ is an elementary (log-exp) index if the set $\{x \in \mathbb{R}^p,\ s(x) = i\}$ is the solution set of a finite number of inequalities and equalities involving elementary functions on $\mathbb{R}^p$. The set of such functions is denoted by $\mathcal{I}$ (for any input dimensions p).*

Examples: The Heaviside function, the index of the largest or k-th largest element in a vector, the sign pattern of a vector in $\mathbb{R}^p$ which is indexed by integers from 1 to 2^p.

Definition 3 (Elementary selection) Let $f\colon \mathbb{R}^p \mapsto \mathbb{R}$ be continuous. We say that f has an *elementary (log-exp) selection* $(s, f_1, \dots, f_m)$ if $s\colon \mathbb{R}^p \mapsto (1, \dots, m)$ is an elementary index in $\mathcal{I}$ and for $i = 1 \dots, m$, $f_i\colon \mathbb{R}^p \mapsto \mathbb{R}$ are elementary functions in $\mathcal{E}$, such that for all $x \in \mathbb{R}^p$,

$$f(x) = f_{s(x)}(x). \tag{2}$$

The $m+1$-uplet $(s, f_1, \dots, f_m)$ is a *representation* of f, and f admits an *elementary (log-exp) selection*. The class of such functions is denoted by $\mathcal{S}_{\log \exp}$ or simply here $\mathcal{S}$. This extends to functions from $\mathbb{R}^p$ to $\mathbb{R}^m$ by applying a coordinatewise definition with a common elementary index.

Observe that the representation is *never* unique, both in s and in the sequence $f_1, \dots, f_m$. The ReLU, hinge loss, maximal entry, k-th largest entry functions are elementary selections. Note also that continuity is part of the definition.

Proposition 1 (Stability of $\mathcal{S}$ by $\circ, +, \times$) *The class $\mathcal{S}$ of elementary selections is stable by composition, sum and product.*

The class $\mathcal{S}$ is close to the one of piecewise C^k functions, see e.g [40], but it is also much more disciplined since indices and functions are required to satisfy strong "log-exp" rigidity assumptions.

2.1 Selection derivative

Functions in $\mathcal{S}$ can be associated with a flexible notion of generalized derivative based on the selection structure of the underlying function.

Definition 4 (Selection gradient) (i) Let $f\colon \mathbb{R}^p \mapsto \mathbb{R}$, in $\mathcal{S}$ with selection $(s, f_1, \dots, f_m)$. We set the *selection derivative of f with respect to s* to be

$$\widehat{\nabla}^s f\colon x \mapsto \nabla f_{s(x)}(x). \tag{3}$$

This extends to multivariate outputs by applying the definition coordinatewise, which leads to a notion of a *selection Jacobian* denoted by $\widehat{J}^s$.
(ii) Given a function $f \in \mathcal{S}$, a *selection derivative* is a derivative of the form (3) for a given representation. In that case a selection derivative of f is merely denoted by $\widehat{\nabla} f$.

Example: Set for all $x \in \mathbb{R}$, $f_1(x) = 0$, $f_2(x) = x$ and $s(x) = 1$ for $x \leqslant 0$ and $s(x) = 2$ for $x > 0$. This this defines the relu function and its selection derivative at 0 is 0. See more in Appendix A.2.

Remark: (a) $\widehat{\nabla} f$ is different from any known notion of subgradient. Set for all $x \in \mathbb{R}$, $f_1(x) = 0$, $f_2(x) = x$ and $s(x) = 1$ for $x \neq 0$ and $s(0) = 2$. This defines a elementary selection for the null function however, $\widehat{\nabla}^s f(0) = 1$. This is the zero function of the introduction.
(b) This formalizes what one would obtained by differentiating a code with all decision branches frozen and hence represents the numerical output of AD (see 4). Note that one only needs one branch and do not need to explore all possible outcomes, avoiding combinatorial explosion.

The properties of selection derivatives might seem too liberal at first sight and too disconnected from the original function, but this is not the case as shown below.

Proposition 2 (Integration along segments) *Let $f\colon \mathbb{R}^p \mapsto \mathbb{R}$ be in $\mathcal{S}$, with elementary selection $(s, f_1, \dots, f_m)$. Then f is locally Lipschitz and for all y, x in $\mathbb{R}^p$.:*

$$f(y) - f(x) = \int_0^1 \Big\langle y - x, \widehat{\nabla}^s f(x + t(y - x)) \Big\rangle \, dt$$

Proposition 3 (Gradient almost everywhere) *Let $f\colon \mathbb{R}^p \mapsto \mathbb{R}$ be in $\mathcal{S}$, with elementary selection $(s, f_1, \dots, f_m)$. There exists sets $U_1, \dots, U_N$ with nonempty interior such that $\bigcup_{i=1}^N \mathrm{cl}(U_i) = \mathbb{R}^p$ and for each $i = 1$, and for all x in the interior of U_i, $\widehat{\nabla}^s f(x) = \nabla f(x)$. Furthermore, the U_i are solution sets of equations and inequalities involving functions in $\mathcal{E}$.*

Remark: Although less transparent, Proposition 2 is not a consequence of Proposition 3. Both results crucially rely on the rigidity of elementary functions in $\mathcal{E}$ (Definition 3), not only on their piecewise smoothness. This a central novelty of our approach.

2.2 A calculus for selection derivatives

One has an unusual differential calculus: although it does not involve the linearity of some (sub)differential operator, the selection derivative of a sum gives a sum of selection derivatives provided that the selection is refined.

Proposition 4 (Chain rule) *Let $F\colon \mathbb{R}^{p_1} \mapsto \mathbb{R}^{p_2}$ such that each of its coordinate f_i, $i = 1 \dots p_2$, is in $\mathcal{S}$ and $g\colon \mathbb{R}^{p_2} \mapsto \mathbb{R}$, $g \in \mathcal{S}$. Consider a selection Jacobian for F, $\widehat{J}_F\colon \mathbb{R}^{p_1} \mapsto \mathbb{R}^{p_2 \times p_1}$*

$$x \mapsto \begin{pmatrix} \widehat{\nabla} f_1(x)^T \\ \vdots \\ \widehat{\nabla} f_q(x)^T \end{pmatrix} \tag{4}$$

Then $g \circ F \in \mathcal{S}$ and the function $x \mapsto \widehat{J}_F(x)^T \widehat{\nabla} g(F(x))$ is a selection derivative for $g \circ F$.

Proposition 4 extends readily to the case when the outer function g is multivariate. For example, we have a sum rule $\widehat{\nabla}(f + g) = \widehat{\nabla} f + \widehat{\nabla} g$ for full-domain functions f, g in $\mathcal{S}$. Indeed, if F_1 and F_2 are elementary selections then $F_1 \circ F_2 \in \mathcal{S}$ and

$$\widehat{J}_{F_1 \circ F_2} = (\widehat{J}_{F_1} \circ F_2) \times \widehat{J}_{F_2}. \tag{5}$$

3 Programs and elementary selections

Numerical programs encode numerical functions by combining elementary functions using a predecessor relation which models program execution. In what follows, m can be seen as an estimate of the memory footprint of a program[4], while p and q the number of inputs and outputs respectively.

Given positive integers $m \geqslant p+q$, a *predecessor relation* is a set valued map $\mathtt{pr}\colon \{1,\dots,m\} \rightrightarrows \{1,\dots,m\}$ such that

- For $i \in \{1,\dots,m\}$ and $j \in \mathtt{pr}(i)$, $j < i$.
- For $i \in \{p+1,\dots,m\}$, $\mathtt{pr}(i)$ is nonempty.

A predecessor relation induces a partial order on the set of integers from 1 to m and hence can be represented by a directed acyclic graph [34, Theorem 9.4.9]. Given (p,q,m) and a predecessor relation $\mathtt{pr}$, a elementary function sequence $\mathcal{G} = (g_i)_{i=p+1}^m$ is a set of functions such that $g_i\colon \mathbb{R}^{|\mathtt{pr}(i)|} \mapsto \mathbb{R}$, and $g_i \in \mathcal{S}$, for all $i = p+1,\dots,m$. A program P is then given by the data $P = (p,q,m,\mathtt{pr},\mathcal{G})$, while its *evaluation* is described in Algorithm 1. We denote by $\mathcal{P}$ the set of programs, and $\mathcal{P}_{p,q}$ when input-output dimensions have to be made explicit.

Algorithm 1: Program evaluation

Program data: $p, q \geqslant 1$, $m \geqslant p+q$, $\mathtt{pr}$ a predecessor relation, $\mathcal{G} = (g_i)_{i=p+1}^m$ an adapted function sequence.

Input: $x = (x_1, \dots x_p)$

1: **for** $k = p+1, p+2, \dots m$ **do**
2: $\quad x_k = g_k(x_{\mathtt{pr}(k)})$ where $x_{\mathtt{pr}(k)} = (x_i)_{i\in\mathtt{pr}(k)}$.
3: **end for**

Return: $y := (x_j)_{j=m-q+1}^m$.

By definition a program encodes a function, but the representation is not unique. We express this fact below through the canonical surjection $\mathcal{F}$ of the introduction. The following proposition illustrates the fact that practitioners *implicitly* implement selection functions when writing programs.

Proposition 5 (Programs represents elementary selections) *Through its input-output correspondence each program P of the form* (3) *induces a function which is an elementary selection. In other words $\mathcal{F}(P) \in \mathcal{S}$.*

4 Algorithmic differentiation and a variational model

Algorithmic differentiation is based on the idea of propagating infinitesimal variations in a program P through the chain rule, either forward or backward.

Algorithm 2: Algorithmic differentiation computes selection gradients

Program data: $p \geqslant 1$, $m \geqslant p+1$, $\mathtt{pr}$ a predecessor relation, $\mathcal{G} = (g_i)_{i=p+1}^m$ an adapted function sequence.

Input: variables $(x_1, \dots x_m)$ computed by Algorithm 1, $d_i = (d_i[j])_{j=1}^{|\mathtt{pr}(i)|} = \widehat{\nabla} g_i(x_{\mathtt{pr}(i)})$, $i = p+1 \dots m$

1: **Forward mode:**
2: Initialize: $\frac{\partial x_k}{\partial x} = e_k$, $k = 1,\dots,p$.
3: **for** $k = p+1, \dots m$ **do**
4:
$$\frac{\partial x_k}{\partial x} = \sum_{j\in\mathtt{pr}(k)} \frac{\partial x_j}{\partial x} d_k[j]$$
where $x = (x_1,\dots,x_p)$.
5: **end for**

Return: $\frac{\partial x_m}{\partial x}$.

1: **Backward mode:**
2: Initialize: $v = e_m$
3: **for** $t = m, \dots p+1$ **do**
4: $\quad$ **for** $j \in \mathtt{pr}(t)$ **do**
5: $\quad\quad$ Update coordinate j of v:
$$v[j] := v[j] + v[t] d_t[j]$$
6: $\quad$ **end for**
7: **end for**

Return: $(v[1], v[2], \dots, v[p])$.

[4] We consider programs which do not overwrite values in memory

Consider Algorithm 1, and assume for simplicity that $q = 1$. The program can be seen as the implementation of $m - p$ successive transformations on $\mathbb{R}^m$, of the form

$$\begin{aligned} G_k \colon \mathbb{R}^m &\mapsto \mathbb{R}^m \\ x &\mapsto x + e_k(g_k(x_{\mathtt{pr}(k)}) - x_k), \end{aligned}$$

for $k = p+1, \dots, m$ which belong to $\mathcal{S}$. Algorithm 2 combines gradients dynamically along two modes: forward or backward. Let us describes these two forms.

Fix $x \in \mathbb{R}^m$. After applying Algorithm 1, for each k, let $d_k \in \mathbb{R}^m$ be the selection gradient $\widehat{\nabla} g_k(x_{\mathtt{pr}(k)})$, appending 0 to non dependant coordinates. A selection Jacobian of G_k (at x) is given by

$$\widehat{J}_{G_k} = I - e_k e_k^T + e_k d_k^T$$

Denote by $J_p \in \mathbb{R}^{m \times p}$, the matrix whose entries are 0, except for diagonal entries which are 1. In Algorithm 2, the forward mode computes

$$e_m^T \widehat{J}_{G_m} \dots \widehat{J}_{G_{p+1}} J_p = e_m^T \left(I - e_m e_m^T + e_m d_m^T\right) \dots \left(I - e_{p+1} e_{p+1}^T + e_{p+1} d_{p+1}^T\right) J_p$$

which is a selection Jacobian thanks to the chain rule in (5). On the other hand the backward mode computes

$$J_p^T \left(I + d_{p+1} e_{p+1}^T\right) \dots \left(I + d_m e_m^T\right) e_m.$$

This quantity turns out to be the same as the one computed by the forward mode thanks to:

Lemma 1 *Let $p, m \in \mathbb{N}$, $0 < p < m$. Assume that for $i = p+1, \dots, m$ we have $d_i \in \mathbb{R}^m$. Then we have*

$$P_p \left(I - e_{p+1} e_{p+1}^T + d_{p+1} e_{p+1}^T\right) \dots \left(I - e_m e_m^T + d_m e_m^T\right) = P_p \left(I + d_{p+1} e_{p+1}^T\right) \dots \left(I + d_m e_m^T\right) \tag{6}$$

where $I \in \mathbb{R}^{m \times m}$ is the identity matrix and $P_p \in \mathbb{R}^{m \times m}$ denotes the projection on the first p coordinates.

Denote by $\mathcal{A} : \mathcal{P}_{p,1} \to \mathcal{P}_{p,p}$ the algorithmic-differentiation operator. This establishes the following fundamental fact which is at the root of this work. This result asserts that practitioners *implicitly* implement selection derivatives when writing numerical programs and calling forward or backward AD on these programs.

Theorem 2 (Algorithmic differentiation outputs a selection gradient) *Algorithmic differentiation of a given program, i.e., $\mathcal{A}(P)$, outputs a selection derivative of the underlying numerical function. In other words there exists a representation of the numerical function $\mathcal{F}(P)$ with elementary index s such that:*

$$\mathcal{F}(\mathcal{A}(P)) = \widehat{\nabla}^s \mathcal{F}(P).$$

5 Algorithmic differentiation at work

5.1 Selection derivatives, conservative fields and Clarke subgradient

The asymptotic study of first-order optimization methods implies limiting processes and necessitates thus the introduction of graph closed operators. Given a representation for f, we may construct such a convex-valued mapping pointwise as follows[5].

Definition 5 (Representation minimal operator) Let $f \in \mathcal{S}$ with elementary selection $(s, f_1, \dots, f_m)$. For any $x \in \mathbb{R}^p$, set $I(x) = \{i \in \{1, \dots, m\}\,,\ f(x) = f_i(x)\}$. The index closure of $\widehat{\nabla}^s f$ is given by the set valued map

$$\begin{aligned} D_f^s \colon \mathbb{R}^p &\rightrightarrows \mathbb{R}^p \\ x &\rightrightarrows \operatorname{conv}\left(\{\nabla f_i(x),\ i \in I(x)\}\right). \end{aligned}$$

where the double arrows express that the map has values in subsets of $\mathbb{R}^p$, much like subgradients, and conv denotes the convex hull.

[5]Minimality relates to the *representation of the function*, not the function itself. This is the minimal convex-valued operator, constructed pointwise and guaranteed to be graph-closed.

The role of D_f^s is to capture all possible outputs of AD including all possible program branches. Of course, due to combinatorial explosion, this quantity is intractable in practice. Its introduction here is only instrumental, we do not use it in algorithms, we just need to access one of its element, for example using a selection derivatives, obtained from AD. A point x satisfying $0 \in D_f^s(x)$ is called a *selection critical point*. We will often drop the index s and write $D_f = D_f^s$.

The two following results highlight crucial properties of D_f in terms of optimization, they again rely on the rigidity constraint of elementary functions.

Theorem 3 *Let $f \in \mathcal{S}$ with elementary selection $(s, f_1, \dots, f_m)$ and D_f be as in Definition 5. Then D_f is conservative for f, that is for all absolutely continuous curves $\gamma\colon [0,1] \mapsto \mathbb{R}^p$, for almost all $t \in [0,1]$, $f \circ \gamma$ is differentiable and*

$$\frac{d}{dt} f(\gamma(t)) = \langle v, \dot{\gamma}(t) \rangle, \qquad \forall v \in D_f(\gamma(t)).$$

The previous result generalizes Proposition 2 by allowing to integrate arbitrary selections along absolutely continuous curves. This connects our work to the general setting of [12], note that D_f has a closed graph thanks to Proposition 6 in Appendix A.3.

In [40], the author considers the *essential index set*, for each $x \in \mathbb{R}^p$,

$$S_E(x) = \{i \in \{1, \dots, m\}\,,\, x \in \mathrm{cl}(\mathrm{int}(\{y,\, f(y) = f_i(y)\}))\} \subset S(x).$$

Considering Definition 5 with $S_E(x)$ instead of $I(x)$ leads to the Clarke subgradient, which can also be defined as

$$\partial^c f(x) = \mathrm{conv}\{d \in \mathbb{R}^p : \exists x_k \in \Delta_f, x_k \to x, \nabla f(x_k) \to d\}$$

where Δ_f is the dense set of differentiability points of f. While $I(x)$ can be computed pointwise (check finitely many equalities), it might be very hard to check membership in $S_E(x)$ without restrictive qualification conditions on programs [30].

Illustration with ReLU and sorting: (a) Set for all $x \in \mathbb{R}$, $f_1(x) = 0$, $f_2(x) = x$, $s(x) = 1$ for $x \leqslant 0$ and $s(x) = 2$ for $x > 0$. This is relu. In this case $D_f = \partial \mathrm{relu}$, the convex subgradient.
(b) Let $F\colon \mathbb{R}^p \mapsto \mathbb{R}^p$ to be the sorting function which associates to x a vector Px where P is any permutation such that Px belongs to the set of vectors which values are sorted in descending order coordinatewise. F obviously has an elementary selection and the construction which we have proposed leads to

$$D_F\colon x \mapsto \mathrm{conv}\,\{P \in \Delta, \quad Px = F(x)\}\,,$$

where Δ denotes the set of permutation matrices of size $p \times p$. Then D is a conservative mapping for F and it actually corresponds to the Clarke Jacobian.

5.2 Convergence of gradient type algorithm and criticality of limit points

Optimization processes in learning are supposed to provide at least a critical point x of the loss, i.e. a point satisfying $0 \in \partial^c f(x)$. When using AD one enlarges the definition of criticality into $0 \in D_f(x)$ and *artificial critical points* appear, they satisfy $0 \notin \partial^c f(x)$ and $0 \in D_f(x)$. Artificial critical points could possibly trap the optimization process in strongly non-optimal situations, we thus have to determine if they have an impact on learning phases.

We consider the problem

$$\min_{x \in \mathbb{R}^p} J(x) = \frac{1}{n} \sum_{i=1}^{n} f_i(x) \tag{7}$$

where $f_i\colon \mathbb{R}^p \mapsto \mathbb{R}$, $f_i \in \mathcal{S}$, $i = 1, \dots, n$. We consider the following algorithm, given $x_0 \in \mathbb{R}^p$, a sequence of positive step sizes $(\gamma_k)_{k \in \mathbb{N}}$ and a sequence of *iid* indices $(I_k)_{k \in \mathbb{N}}$ taken uniformly in the nonempty subsets of $\{0, \dots, n\}$,

$$x_{k+1} = x_k - \gamma_k \widehat{\nabla} f_{I_k}(x_k) \text{ where } f_I = \frac{1}{|I|} \sum_{i \in I} f_i,\ I \subset \{1, \dots, n\}. \tag{8}$$

Note that as discussed in Section 4 selection derivatives can be computed by AD if f_i are given by the data of numerical programs as in (3), and could be far from usual notions of subgradients. Hence this algorithm models explicitly the training of a nonsmooth deep network using existing backpropagation implementations. Note that $J \in \mathcal{S}$ and that $1/n \sum_{i=1}^{n} \widehat{\nabla} f_i$ is a selection gradient for J as stated in Proposition 4, denote by $\widehat{\nabla} J$ this quantity and D_J the corresponding set valued field (Definition 5). The following result illustrates that selection critical points are the only attractors for the recursion and that generically such attractors are actually Clarke critical. The first result stands on the theory developed in [5]. The second parallels developments in [7] in the context of long run convergence. The spurious behaviour illustrated in Figure 1 does not affect asymptotics, for typical initialization.

Theorem 4 (Convergence and insignificance of artefacts) *Let for all k, $\gamma_k = c\alpha_k$ where $c \in (0, 1]$ and $\alpha_k = o(1/\log k)$ and $K \subset \mathbb{R}^p$ be open. Assume that for all $c \in (0, 1]$ and all $x_0 \in K$ the sequence in* (8) *is bounded almost surely.*

- *For all $x_0 \in K$, almost surely, $J(x_k)$ converges as k tends to infinity and all accumulation points, $\bar{x}$, of $(x_k)_{k\in\mathbb{N}}$ are selection critical points: $0 \in D_J(\bar{x})$.*
- *For almost all $c \in (0, 1]$, almost all $x_0 \in K$, and almost surely, any accumulation point, $\bar{x}$, of $(x_k)_{k\in\mathbb{N}}$ is Clarke critical: $0 \in \partial^c J(\bar{x})$.*

6 Conclusion

The current work departs from existing approaches to nonsmooth algorithmic differentiation in a fundamental way. We propose to study the backward mode of AD, as implemented in machine learning, without any modification. Our theoretical results model thus AD "as is", and our focus is precisely on its unpredictable behavior in a nonsmooth context, addressing an issue which is ubiquitous in machine learning. Our main contribution was to prove that, in a stochastic optimization context, this spurious behavior is essentially harmless from a theoretical point of view, providing justifications for the use of AD outside of its original domain of validity in machine learning.

We achieve our goal by modeling sharply common machine learning functions and their differentiation using selection derivatives, a known concept, which models the way AD differentiates nonsmooth programs. We restrict it to certain classes of elementary functions, opening the possibility to use powerful geometric techniques.

Further questions include convergence rates and complexity issues, hardly tackled at this day, let us mention the attempt of [43]. Our theory is limited to continuous functions and an interesting venue is to extend it to discontinuous functions, in view of treating ranking operations [10] ubiquitous in recommendation systems, or more generally differentiating through an argmax [6].

Broader impact

One of the goals of the paper is to raise awareness about an important issue of in the training of ML methods: the spuriousness of AD. To address adequately this issue, we think it is necessary to include algorithmic differentiation explicitly in the study of optimization algorithms, a point of view which is largely ignored by today's machine learning community.

Acknowledgments and Disclosure of Funding

The authors acknowledge the support of ANR-3IA Artificial and Natural Intelligence Toulouse Institute, Air Force Office of Scientific Research, Air Force Material Command, USAF, under grant numbers FA9550-19-1-7026, FA9550-18-1-0226, and ANR MasDol. J. Bolte acknowledges the support of ANR Chess, grant ANR-17-EURE-0010 and ANR OMS. The authors would like to thank anonymous referees for careful reading of this work and useful suggestions. The authors would like to thank N. Asher and S. Gerchinovitz for useful discussions. We also warmly thank J. Malick who triggered this research.

References

[1] Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M., Kudlur M., Levenberg J., Monga R., Moore S., Murray D., Steiner B., Tucker P., Vasudevan V., Warden P., Wicke M., Yu Y. and Zheng X. (2016). Tensorflow: A system for large-scale machine learning. In Symposium on Operating Systems Design and Implementation.

[2] Barton, P. I., Khan, K. A., Stechlinski, P., Watson, H. A. (2018). Computationally relevant generalized derivatives: theory, evaluation and applications. Optimization Methods and Software, 33(4-6), 1030-1072.

[3] Baydin A., Pearlmutter B., Radul A. and Siskind J. (2018). Automatic differentiation in machine learning: a survey. Journal of machine learning research, 18(153).

[4] Benaïm, M. (1999). Dynamics of stochastic approximation algorithms. In Séminaire de probabilités XXXIII (pp. 1-68). Springer, Berlin, Heidelberg.

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

[6] Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J. P., Bach, F. (2020). Learning with differentiable perturbed optimizers. arXiv preprint arXiv:2002.08676.

[7] Bianchi, P., Hachem, W., and Schechtman, S. (2020). Convergence of constant step stochastic gradient descent for non-smooth non-convex functions. arXiv preprint arXiv:2005.08513.

[8] Bischof, C., Carle, A., Corliss, G., Griewank, A., Hovland, P. (1992). ADIFOR-generating derivative codes from Fortran programs. Scientific Programming, 1(1), 11-29.

[9] Bischof, C., Khademi, P., Mauer, A., Carle, A. (1996). ADIFOR 2.0: Automatic differentiation of Fortran 77 programs. IEEE Computational Science and Engineering, 3(3), 18-32.

[10] Blondel, M., Teboul, O., Berthet, Q., Djolonga, J. (2020). Fast Differentiable Sorting and Ranking. arXiv preprint arXiv:2002.08871.

[11] Bolte, J., Daniilidis, A., Lewis, A., Shiota, M. (2007). Clarke subgradients of stratifiable functions. SIAM Journal on Optimization, 18(2), 556-572.

[12] Bolte, J. and Pauwels, E. (2020). Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. Mathematical Programming.

[13] Borkar, V. (2009). Stochastic approximation: a dynamical systems viewpoint (Vol. 48). Springer.

[14] Bottou L. and Bousquet O. (2008). The tradeoffs of large scale learning. In Advances in neural information processing systems (pp. 161-168).

[15] Bottou L., Curtis F. E. and Nocedal J. (2018). Optimization methods for large-scale machine learning. Siam Review, 60(2), 223-311.

[16] Castera C., Bolte J., Févotte C., Pauwels E. (2019). An Inertial Newton Algorithm for Deep Learning. arXiv preprint arXiv:1905.12278.

[17] Clarke F. H. (1983). Optimization and nonsmooth analysis. Siam.

[18] Chizat, L., and Bach, F. (2018). On the global convergence of gradient descent for over-parameterized models using optimal transport. In Advances in neural information processing systems, 3036-3046.

[19] Coste M., *An introduction to o-minimal geometry*. RAAG notes, Institut de Recherche Mathématique de Rennes, 81 pages, November 1999.

[20] Davis, D., Drusvyatskiy, D., Kakade, S., Lee, J. D. (2018). Stochastic subgradient method converges on tame functions. Foundations of Computational Mathematics.

[21] van den Dries L. and Miller C. (1996). Geometric categories and o-minimal structures. Duke Math. J, 84(2), 497-540.

[22] Griewank, A., Juedes, D., Utke, J. (1996). Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++. ACM Transactions on Mathematical Software (TOMS), 22(2), 131-167.

[23] Griewank, A., Walther, A. (2008). Evaluating derivatives: principles and techniques of algorithmic differentiation (Vol. 105). SIAM.

[24] Griewank, A. (2012). Who invented the reverse mode of differentiation. Documenta Mathematica, Extra Volume ISMP, 389-400.

[25] Griewank A. (2013). On stable piecewise linearization and generalized algorithmic differentiation. Optimization Methods and Software, 28(6), 1139-1178.

[26] Griewank A., Walther A., Fiege S. and Bosse T. (2016). On Lipschitz optimization based on gray-box piecewise linearization. Mathematical Programming, 158(1-2), 383-415.

[27] Griewank, A., Walther, A. (2016). First-and second-order optimality conditions for piecewise smooth objective functions. Optimization Methods and Software, 31(5), 904-930.

[28] Griewank, A., Rojas, A. (2019, September). Treating artificial neural net training as a nonsmooth global optimization problem. In International Conference on Machine Learning, Optimization, and Data Science (pp. 759-770). Springer, Cham.

[29] Griewank, A., Walther, A. (2020). Beyond the Oracle: Opportunities of Piecewise Differentiation. In Numerical Nonsmooth Optimization (pp. 331-361). Springer, Cham.

[30] Kakade, S. M. and Lee, J. D. (2018). Provably correct automatic sub-differentiation for qualified programs. In Advances in Neural Information Processing Systems (pp. 7125-7135).

[31] Kushner H. and Yin, G. G. (2003). Stochastic approximation and recursive algorithms and applications (Vol. 35). Springer Science & Business Media.

[32] LeCun Y., Bengio Y., Hinton, G. (2015). Deep learning. Nature, 521(7553).

[33] Lee, W., Yu, H., Rival, X., Yang, H. (2020). On Correctness of Automatic Differentiation for Non-Differentiable Functions. arXiv preprint arXiv:2006.06903.

[34] Lehman, E., Leighton, T., and Meyer, A. R. (2010). Mathematics for computer science. Technical report, 2006. Lecture notes.

[35] Moulines E. and Bach, F. (2011). Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In Advances in Neural Information Processing Systems (pp. 451-459).

[36] Paszke A., Gross S., Chintala S., Chanan G., Yang E., DeVito Z., Lin Z., Desmaison A., Antiga L. and Lerer A. (2017). Automatic differentiation in pytorch. In NIPS workshops.

[37] Robbins H. and Monro, S. (1951). A stochastic approximation method. The annals of mathematical statistics, 400-407.

[38] Rockafellar, R. T., Wets, R. J. B. (1998). Variational analysis. Springer.

[39] Rumelhart E., Hinton E., Williams J. (1986). Learning representations by back-propagating errors. Nature 323:533-536.

[40] Scholtes, S. (2012). Introduction to piecewise differentiable equations. Springer Science & Business Media.

[41] Speelpenning, B. (1980). Compiling fast partial derivatives of functions given by algorithms (No. COO-2383-0063; UILU-ENG-80-1702; UIUCDCS-R-80-1002). Illinois Univ., Urbana (USA). Dept. of Computer Science.

[42] Wilkie, A. J. (1999). A theorem of the complement and some new o-minimal structures. Selecta Mathematica, 5(4), 397-421.

[43] Zhang, J., Lin, H., Sra, S., Jadbabaie, A. (2020). On Complexity of Finding Stationary Points of Nonsmooth Nonconvex Functions. arXiv preprint arXiv:2002.04130.

This is the appendix for "A mathematical model for automatic differentiation in machine learning".

A A more comprehensive discussion and auxiliary results

A.1 Related work and contribution

The use of backward mode of algorithmic differentiation (AD) for neural network training expanded in the 80's, the most cited reference being [39]. However the theory applies to much more optimization problems, see for example [24]. Indeed, numerical libraries implementing the backward mode of AD were already available in the 90's for `FORTRAN` code [8, 9] or `C/C++` code [22], 30 years before the emergence of `python` libraries. These early implementation could differentiate virtually any code, but their domain of validity, i.e., the setting for which one could predict what the output would be, was restricted to differentiable functions evaluated on their (open) domain of differentiability.

This was well known to the AD community, see for example [23], and exploring further the domain of validity of AD, beyond mere differentiability, was already a vivid problem.

Let us mention [23] who used notions such as finite selection, "isolated criticalities", stable domain or regular arcs, and argued that "functions given by evaluation procedures are almost everywhere real analytic or stably undefined" where "undefined" meant that a nonsmooth elementary function is used in the evaluation process. For piecewise smooth functions which nonsmoothness can be described using the absolute value function (abs-normal form), [25] developped a piecewis linearisation formalism and local approximation related to AD, [26] proposed an AD based bundle type method. These developments are based on the notion of piecewise smooth functions [40] which we use in this work. More recently, [28] applied these techniques to single layer neural network training and [29] proposed to avoid the usage of subgradient "oracles" in nonsmooth analysis as they are not available in practice. In a similar vein, let us mention [2] study lexicographic derivatives, a notion of directional derivatives which satisfy a chain rule making them compatible by forward mode AD, and [43] who use directional derivatives in the context of local sampling stochastic approximation algorithms for machine learning.

Constraint qualification is known in nonsmooth analysis to ensure favorable behavior of chain rules of differential calculus for nonsmooth objects (see [38]). These already appeared in the context of piecewise smooth functions of Scholtes with the notion of "essential selections". Such an approach was used in [30] to propose an AD algorithm for subgradient computation under constrant qualification. Similarly [27] study first and second order optimality, in relation to AD using constraint qualification.

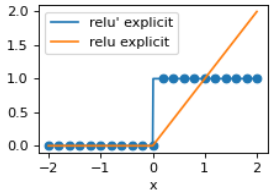
The current work departs from all these approaches in a fundamental way. We propose to study backward mode of AD, as implemented for nonsmooth functions by standard software (e.g. TensorFlow, PyTorch), without any modification, addition of operations or hypotheses. Our theoretical results model AD as implemented in current machine learning libraries. Contrary to previous works, our focus is precisely on the unpredictable behavior of AD in nonsmooth context. Our main contribution is to show that in a stochastic optimization context, this spurious behavior is essentially harmless from a theoretical point of view, providing justifications for the use of AD outside of its original domain of validity in machine learning.

At the time this paper was accepted, we learnt about a paper proposing an analysis close to ours [33]. The authors show that AD applied to programs involving piecewise analytic continuous functions, under analytic partitions, compute gradients almost everywhere. This is the counterpart of Proposition 3, replacing log-exp elementary function in Definitions 1 and 2, by analytic functions.

A.2 Implementation of relu

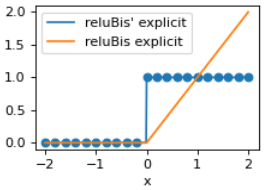
The implementation of the $relu$ function used in Figure 1 is given by the function `tf.nn.relu` in Tensorflow software library [1]. This implementation corresponds to the selection function described in Section 2 and the same result may be obtained by an explicit implementation of this branching selection as illustrated in the following figure

```
def myRelu(x):
    if x<=0:
        return 0
    else:
        return x

def myReluVec(x):
    return tf.map_fn(myRelu,x)
```



One can imagine an equivalent implementation of $relu$ with a slightly different branching involving a strict inequality, that would correspond to an equivalent implementation of the same function, but the computed derivative at 0 is different due to the implementation

```
def myReluBis(x):
    if x<0:
        return 0
    else:
        return x

def myReluBisVec(x):
    return tf.map_fn(myReluBis,x)
```



A.3 Auxiliary results and remarks

Remark 1 (Elementary piecewise differentiable functions)
(a) The building blocks in the construction of $\mathcal{S}$ in Definition 3 could be modified and adapted to other needs. Besides, the results we present in this article would remain true if we added real analytic functions restricted to compact sets.
(b) Note also that in Definition 3, functions are actually real analytic on their (open) domain of definition. Yet their extension might not be analytic, as for instance the function $f : x \neq 0 \to \exp(-1/x^2)$ extended by $f(0) = 0$.
(c) The construction of elementary piecewise functions in Definition 3, does not coincide in general with some natural minimal o-minimal, but are contained in a larger such structure. For instance, when the basic bricks are polynomial functions, we obtain the field of rational functions which differs from the set of semi-algebraic functions.

Proposition 6 (D_f has a closed graph) *As $k \to \infty$, assume that $x_k \to \bar{x} \in \mathbb{R}^p$ and $v_k \in D_f(x_k)$, $v_k \to \bar{v}$. Then $\bar{v} \in D(\bar{x})$.*

B Proofs

Proof of Theorem 1: Recall the operator is denoted by ∂^A. Fix a function f, by point (a), the operator $\partial^A f$ should contain

$$\begin{cases} \mathbb{R}^p \rightrightarrows & \mathbb{R}^p \\ x \to & \{\mathcal{A}(P)(x) : \mathcal{F}(P) = f, P \in \mathcal{P}\} \end{cases}$$

Let us show that the graph of the above is $\mathbb{R}^p \times \mathbb{R}^p$. Assume $p = 1$ for simplicity. For real numbers r, s, consider the functions $f_{r,s} = f + r\,\mathrm{zero}(\cdot - s)$ which coincide with f but whose form induces programs $P_{r,s}$ of f. These satisfy $\mathcal{F}(P_{r,s}) = f$ and $\mathcal{A}(P_{r,s})(s) \ni \mathcal{A}(f)(s) + r$. Since r is arbitrary, $\partial^A f(s) = \mathbb{R}^p$ and since s is arbitrary, we actually have

$$\mathrm{graph}\, \partial^A f = \mathbb{R}^p \times \mathbb{R}^p.$$

Since f is arbitrary, we have shown that ∂^A is trivial. □

Proof of Proposition 2: The proposition is a consequence of Theorem 3 and (11) but it admits a more elementary proof which we detail here. Fix $x, y \in \mathbb{R}^p$. Let us admit the following claim –whose independent proof is given in Section C.

Claim 1 *There exists a finite set of numbers $0 = a_0 < a_1 < \ldots < a_N = 1$, such that for all $i \in 0, \ldots N-1$, the function $t \mapsto s(x + t(y - x))$ is constant.*

Fix $i \in 0 \dots, N-1$, and $j \in 1 \dots m$ such that $f = f_j$ on $(x + a_i(y-x), x + a_{i+1}(y-x))$. Since $f_j \in \mathcal{E}_p$, it is C^1 and we have by the fundamental theorem of integral calculus

$$\begin{aligned} f(x + a_{i+1}(y-x)) - f(x + a_i(y-x)) &= \int_{a_i}^{a_{i+1}} \langle \nabla f_j(x + t(y-x)), y - x \rangle \, dt \\ &= \int_{a_i}^{a_{i+1}} \left\langle \hat{\nabla} f(x + t(y-x)), y - x \right\rangle dt. \end{aligned}$$

The conclusion follows because

$$\begin{aligned} f(y) - f(x) &= \sum_{i=0}^{N-1} f(x + a_{i+1}(y-x)) - f(x + a_i(y-x)) \\ &= \sum_{i=0}^{N-1} \int_{a_i}^{a_{i+1}} \left\langle \hat{\nabla} f(x + t(y-x)), y - x \right\rangle dt \\ &= \int_0^1 \left\langle \hat{\nabla} f(x + t(y-x)), y - x \right\rangle dt. \end{aligned}$$

□

Proof of Proposition 3: Constructs the sets U_i by considering sets $V_j = \{x \in \mathbb{R}^p,\ s(x) = j\}$, $j = 1 \dots m$, the proof of the following claim is postponed to Section C.

Claim 2 *The boundary of each V_j has zero measure and* $\mathrm{cl}\left(\cup_{i=j}^m \mathrm{int}(V_j)\right) = \mathbb{R}^p$.

Hence, we may define $U_1, \dots, U_N$ by keeping only those sets with nonempty interior and take their closure. On each set U_i, f is identical to f_k for some k and the result follows. □

Lemma 2 *Let $t \in \mathcal{I}$ be an elementary index on $\mathbb{R}^{p_2}$ and $F \colon \mathbb{R}^{p_1} \mapsto \mathbb{R}^{p_2}$ with each coordinate in $\mathcal{E}$, then $t \circ F$ is an elementary index on $\mathbb{R}^{p_1}$.*

Proof : Fix an arbitrary integer i in the image of t, by Definition 2, there exists elementary functions $h_1, \dots, h_J$, $J \in \mathbb{N}$ on $\mathbb{R}^{p_2}$ such that $t(y) = i$ if and only if $y \in K_i := \{z \in \mathbb{R}^{p_2},\ h_j(z) \diamond_j 0,\ j = 1, \dots J\}$ where $\diamond_j$ is an equality or inequality sign depending on j. Then $t(F(x)) = i$ if and only if $F(x) \in K_i$ which is equivalent to say that $x \in \tilde{K}_i := \{x \in \mathbb{R}^{p_1}, h_j(F(x)) \diamond_j 0,\ j = 1, \dots J\}$. By Definition 1, $h_j \circ F$ is an elementary function for $j = 1, \dots, J$ and i was an arbitrary integer, this shows that we have an elementary index. □

Proof of Proposition 1: Let $F \colon \mathbb{R}^{p_1} \mapsto \mathbb{R}^{p_2}$ such that each of its coordinate f_i, $i = 1 \dots p_2$, is in $\mathcal{S}$ and $g \colon \mathbb{R}^{p_2} \mapsto \mathbb{R}$, $g \in \mathcal{S}$. We establish that $g \circ F$ is an elementary selection, the other cases are similar. We may consider all possible intersections of constant index domains across all coordinates of F in $\{1, \dots, p_2\}$. We obtain $(s, F_1, \dots, F_m)$, an elementary selection for F (each $F_i \colon \mathbb{R}^{p_1} \mapsto \mathbb{R}^{p_2}$ has coordinates in $\mathcal{E}$) . Consider $g \in \mathcal{S}$ with elementary selection $(t, g_1, \dots, g_l)$. The composition $g \circ F$ may be written as

$$g(F(x)) = g_{t(F(x))}(F(x)) = g_{t(F_{s(x)}(x))}(F_{s(x)}(x)).$$

For each $i = 1 \dots, m$ and $j = 1, \dots, l$, consider the set

$$U_{ij} = \{x \in \mathbb{R}^p,\ s(x) = i, t(F_i(x)) = j\}.$$

Fix (i, j) in $\{1, \dots, m\} \times \{1, \dots, l\}$, by Lemma 2, $t \circ F_i$ is an elementary index on $\mathbb{R}^{p_1}$. Hence U_{ij} is the solution set of finitely many equalities and inequalities involving functions in $\mathcal{E}$. We associate to the bi-index (i, j) the corresponding set U_{ij} and the function $g_j(F_i(x)) \in \mathcal{E}$. Note that we assumed that the composition is well defined. Identifying each pair (i, j) with a number in $\{1, \dots, nm\}$, we obtain an elementary selection for $g \circ F$ and hence $g \circ F \in \mathcal{S}$. □

Proof of Proposition 4: The derivation formula follows from the proof argument of Proposition 1, for each pair (i, j), the function $g_j \circ F_i$ is the composition of two C^1 functions and its gradient is given by $J_{F_i} \times \nabla g_j \circ F_i$ on U_{ij}. By construction of U_{ij} and definition of the selection derivative, this corresponds to (5) on U_{ij} and the result follows. □

Proof of Lemma 1: We actually prove a slightly stronger result, namely for each $i \in \{p+1, \dots, m-1\}$

$$P_i \left(I - e_{i+1}e_{i+1}^T + d_{i+1}e_{i+1}^T\right) \dots \left(I - e_m e_m^T + d_m e_m^T\right) = P_i \left(I + d_{i+1}e_{i+1}^T\right) \dots \left(I + d_m e_m^T\right) \tag{9}$$

We argue by exhaustion from $i = m-1$ downward to $i = p$, which is the result of interest. If $i = m-1$, we indeed have

$$P_{m-1} \left(I - e_m e_m^T + d_m e_m^T\right) = P_{m-1} \left(I + d_m e_m^T\right)$$

since $P_{m-1}e_m e_m^T = 0$. Now assume that (9) holds true for an index i within $\{p+1, \dots, m-1\}$, then we have

$$\begin{aligned}
& P_{i-1} \left(I - e_i e_i^T + d_i e_i^T\right) \dots \left(I - e_m e_m^T + d_m e_m^T\right) \\
= \; & P_{i-1} \left(I - e_i e_i^T + d_i e_i^T\right) \left(I - e_{i+1} e_{i+1}^T + d_{i+1} e_{i+1}^T\right) \dots \left(I - e_m e_m^T + d_m e_m^T\right) \\
= \; & P_{i-1} \left(I - e_i e_i^T + d_i e_i^T\right) P_i \left(I - e_{i+1} e_{i+1}^T + d_{i+1} e_{i+1}^T\right) \dots \left(I - e_m e_m^T + d_m e_m^T\right) \\
= \; & P_{i-1} \left(I + d_i e_i^T\right) P_i \left(I + d_{i+1} e_{i+1}^T\right) \dots \left(I + d_m e_m^T\right) \\
= \; & P_{i-1} \left(I + d_i e_i^T\right) \left(I + d_{i+1} e_{i+1}^T\right) \dots \left(I + d_m e_m^T\right),
\end{aligned}$$

where step 1 is expanding the product, step 2 is because $P_{i-1}P_i = P_{i-1}$ and $e_i^T P_i = e_i^T$, step 3 combines the fact that $P_{i-1}e_i = 0$ and (9) which we assumed to be true, the last step uses again the fact that $P_{i-1}P_i = P_{i-1}$ and $e_i^T P_i = e_i^T$. Hence the result holds by exhaustion. □

Proof of Proposition 6: Consider the sequence $s_k = S(x_k)$, by taking a subsequence we may assume that s_k is constant, say equal to $\{1, \dots, r\}$. Hence for all k, $v_k \in \operatorname{conv}\left(\{\nabla f_i(x_k),\ i = 1, \dots r\}\right)$ and $f(x_k) = f_i(x_k)$, $i = 1, \dots, r$. Passing to the limit, we have $f(\bar{x}) = f_i(\bar{x})$, $i = 1, \dots, r$ and hence $\{1, \dots, r\} \in S(x)$. Furthermore, $\bar{v} \in \operatorname{conv}\left(\{\nabla f_i(\bar{x}),\ i = 1, \dots r\}\right) \subset D_f(\bar{x})$. □

C o-minimal structures, definability and conservative fields

C.1 $(\mathbb{R}, \exp)$-definability

We recall here the results of geometry that we use in the present work. Some references on this topic are [19, 21].

An *o-minimal structure* on $(\mathbb{R}, +, \cdot)$ is a collection of sets $\mathcal{O} = (\mathcal{O}_p)_{p \in \mathbb{N}}$ where each $\mathcal{O}_p$ is itself a family of subsets of $\mathbb{R}^p$, such that for each $p \in \mathbb{N}$:

(i) $\mathcal{O}_p$ is stable by complementation, finite union, finite intersection and contains $\mathbb{R}^p$.

(ii) if A belongs to $\mathcal{O}_p$, then both $A \times \mathbb{R}$ and $\mathbb{R} \times A$ belong to $\mathcal{O}_{p+1}$;

(iii) if $\pi : \mathbb{R}^{p+1} \to \mathbb{R}^p$ is the canonical projection onto $\mathbb{R}^p$ then, for any $A \in \mathcal{O}_{p+1}$, the set $\pi(A)$ belongs to $\mathcal{O}_p$;

(iv) $\mathcal{O}_p$ contains the family of real algebraic subsets of $\mathbb{R}^p$, that is, every set of the form

$$\{x \in \mathbb{R}^p \mid g(x) = 0\}$$

where $g : \mathbb{R}^p \to \mathbb{R}$ is a polynomial function;

(v) the elements of $\mathcal{O}_1$ are exactly the finite unions of intervals.

A subset of $\mathbb{R}^p$ which belongs to an o-minimal structure $\mathcal{O}$ is said to be *definable in* $\mathcal{O}$. A function is *definable in* $\mathcal{O}$ whenever its graph is definable in $\mathcal{O}$). A set valued mapping (or a function) is said to be definable in $\mathcal{O}$ whenever its graph is definable in $\mathcal{O}$. The terminology *tame* refers to definability in an o-minimal structure without specifying which structure.

The simplest o-minimal structure is given by the class of real semialgebraic objects. Recall that a set $A \subset \mathbb{R}^p$ is called *semialgebraic* if it is a finite union of sets of the form

$$\bigcap_{i=1}^{k} \{x \in \mathbb{R}^p \mid g_i(x) < 0,\ h_i(x) = 0\}$$

where the functions $g_i, h_i : \mathbb{R}^p \to \mathbb{R}$ are real polynomial functions and $k \geqslant 1$. The key tool to show that these sets form an o-minimal structure is Tarski-Seidenberg principle which ensures that (iii) holds true.

According to [42], there is an o-minimal structure which contains all semialgebraic sets and the graph of the exponential function, we fix this o-minimal structure and call it $\mathcal{O}$. As a consequence, all functions which can be described by a finite compositional expression involving polynomials, quotients, exponential and logarithms are definable in $\mathcal{O}$. In particular any function $f \in \mathcal{S}$ is definable in $\mathcal{O}$, which opens the use of powerful geometric tools [19, 21] for functions in $\mathcal{S}$. From now on, we call an object *definable* if it is definable in $\mathcal{O}$.

As detailed in [19] the following holds true

Proposition 7 (Quantifier elimination) *Any first order formula (quantification on variables only) involving definable functions and definable sets describes a definable set.*

This allows to prove Claim 1

Proof of Claim 1: The function $t \mapsto s(x + t(y - x))$ is definable and has values in $\{1, \dots, m\}$. For each $j \in \{1, \dots, m\}$, the set $S_j = \{t \in [0, 1],\ s(x + t(y - x)) = j\}$ is definable, and by (v), it is a finite union of intervals. For each j consider only the endpoints of those intervals with nonempty interior, this provides the desired partition. □

C.2 Properties of definable sets

The tangent space at a point x of a manifold M is denoted by $T_x M$. Given a submanifold[6] M of a finite dimensional Riemannian manifold, it is endowed by the Riemanninan structure inherited from the ambient space. Given $f \colon \mathbb{R}^p \to \mathbb{R}$ and $M \subset \mathbb{R}^p$ a differentiable submanifold on which f is differentiable, we denote by $\operatorname{grad}_M f$ its Riemannian gradient or even, when no confusion is possible, $\operatorname{grad} f$.

A C^r stratification of a (sub)manifold M (of $\mathbb{R}^p$) is a partition $\mathcal{S} = (M_1, \dots, M_m)$ of M into C^r manifolds having the property that $\operatorname{cl} M_i \cap M_j \neq \varnothing$ implies that M_j is entirely contained in the boundary of M_i whenever $i \neq j$. Assume that a function $f : M \to \mathbb{R}$ is given and that M is stratified into manifolds on which f is differentiable. For x in M, we denote by M_x the strata containing x and we simply write $\operatorname{grad} f(x)$ for the gradient of f with respect to M_x.

Stratifications can have many properties, we refer to [21] and references therein for an account on this question and in particular for more on the idea of a Whitney stratification that we will use repeatedly. We pertain here to one basic definition: a C^r-stratification $\mathcal{S} = (M_i)_{i \in I}$ of a manifold M has the *Whitney-*(a) *property,* if for each $x \in \operatorname{cl} M_i \cap M_j$ (with $i \neq j$) and for each sequence $(x_k)_{k \in \mathbb{N}} \subset M_i$ we have:

$$\left.\begin{array}{l} \lim\limits_{k \to \infty} x_k \ = x \\ \\ \lim\limits_{k \to \infty} T_{x_k} M_i \ = \mathcal{T} \end{array}\right\} \implies T_x M_j \ \subset \ \mathcal{T}$$

where the second limit is to be understood in the Grassmanian, i.e., "directional", sense. In the sequel we shall use the term *Whitney stratification* to refer to a C^1-stratification with the Whitney-(a) property. The following can be found for example in [21].

Theorem 5 (Whitney stratification) *Let $A_1, \dots, A_k$ be definable subsets of $\mathbb{R}^p$, then there exists a definable Whitney stratification $(M_i)_{i \in I}$ compatible with $A_1, \dots, A_k$,* i.e. *such that for each $i \in I$, there is $t \in \{1, \dots k\}$, such that $M_i \subset A_t$.*

This allows for example to prove Claim 2

Proof of Claim 2: The sets $V_1, \dots, V_m$ form a definable partition of $\mathbb{R}^p$. Consider a Whitney stratification of $\mathbb{R}^p$, $(M_i)_{i \in I}$ compatible with the closure of $V_1, \dots, V_m$. The boundary of each V_i is a finite union of strata of dimension strictly smaller than p and hence has measure zero. The remaining strata (open of maximal dimension) have to be dense in $\mathbb{R}^p$ since we started with a partition. □

[6] We only consider embedded submanifolds

C.3 Variational stratification and projection formulas

Definition 6 (Variational stratification) Let $f\colon \mathbb{R}^p \to \mathbb{R}$, be locally Lipschitz continuous, let $D\colon \mathbb{R}^p \rightrightarrows \mathbb{R}^p$ be a set valued map and let $r \geqslant 1$. We say that the couple (f, D) has a C^r *variational stratification* if there exists a C^r Whitney stratification $\mathcal{S} = (M_i)_{i\in I}$ of $\mathbb{R}^p$, such that f is C^r on each stratum and for all $x \in \mathbb{R}^p$,

$$\mathrm{Proj}_{T_{M_x}(x)} D(x) = \{\mathrm{grad}\, f(x)\}\,, \tag{10}$$

where $\mathrm{grad}\, f(x)$ is the gradient of f restricted to the active strata M_x containing x.

The equations (10) are called *projection formulas* and are motivated by Corollary 9 in [11] which states that Clarke subgradients of definable functions have projection formulas.

Let us recall the definition of conservative set-valued mappings from [12] and one of its characterization.

Definition 7 (Conservative set-valued mappings) Let f be a Lipschitz continuous function. A set valued vector field D is called *conservative* if for any absolutely continuous path $\gamma\colon [0,1] \mapsto \mathbb{R}$, we have

$$f(\gamma(1)) - f(\gamma(0)) = \int_0^1 \min_{v\in D(\gamma(t))} \langle v, \dot\gamma(t)\rangle\, dt = \int_0^1 \max_{v\in D(\gamma(t))} \langle v, \dot\gamma(t)\rangle\, dt. \tag{11}$$

Equivalently D is conservative for f, if for all absolutely continuous curves $\gamma\colon [0,1] \mapsto \mathbb{R}^p$, for almost all $t \in [0,1]$, $f \circ \gamma$ is differentiable and

$$\frac{d}{dt} f(\gamma(t)) = \langle v, \dot\gamma(t)\rangle\,, \qquad \forall v \in D(\gamma(t)).$$

The following combines other results from [12], where one implication is essentially due to [20] based on [11].

Theorem 6 (Characterization of conservativity) *Let $D\colon \mathbb{R}^p \rightrightarrows \mathbb{R}^p$ be a definable, nonempty compact valued, graph closed set valued field and $f\colon \mathbb{R}^p \mapsto \mathbb{R}$ be a definable locally Lipschitz function. Then the following are equivalent*

- *D is conservative for f.*
- *For any $r \geqslant 1$, (f, D) admit a C^r variational stratification.*

This result allows to prove the following

Proof of Theorem 3: We prove that there is a C^1 projection formula (see Theorem 6). For each $I \subset \{1,\dots,m\}$, set $V_I = \{x \in \mathbb{R}^p, S(x) = I\}$. On each set V_I, $f(x) = f_i(x)$ for all $i \in I$. These sets are definable, hence, there is a definable Whitney stratification of $\mathbb{R}^p$ which is compatible with them (Theorem 5). For any C^1 manifold M in the stratification there is an index set $I \subset \{1,\dots,m\}$ such that for all $i \in I$ and all $x \in M$, $f(x) = f_i(x)$ and $S(x) = I$. Since each f_i, $i \in I$ is C^1 and they agree on M, they represent the same function when restricted to M. Hence they have the same differential on M and since they are all globally C^1 this agrees with the projection of their gradient on the tangent space of M. Hence the projection of $D_f(x)$ to the tangent space to M at x is single valued and corresponds to the derivative of f restricted to M. This is sufficient to conclude as this is precisely the variational stratification required by Theorem 6. □

D Convergence to selection critical points

Proof of Theorem 4, first part: We use here the results on conservative fields developed in [12]. To prove the theorem it suffices to establish that:

- D_J is a conservative field for J
- the number of D_J critical values are finite.

The first point is Theorem 6 while the second one is the consequence of the latter and the definability of the couple f, D_f, see Proposition 8 (ii). To conclude it suffices to apply the convergence results in [12, Theorem 9]. □

Proof of Theorem 4, second part: This result is a consequence of the more general Theorem 7 established in Section E. Let F be the finite set given in Theorem 7, the set

$$\{c \in (0,1],\, \exists k \in \mathbb{N},\, c\gamma_k \in F\},$$

is countable, and hence has zero measure. So for almost all $c \in (0,1]$, $\{c\gamma_k\}_{k\in\mathbb{N}}$ does not intersect F. Using Theorem 7, there is a zero measure set N such that any initialization outside N provides almost surely a subgradient sequence. By hypothesis, for almost every $x_0 \in K\backslash N$, the sequence is bounded almost surely and the result follows from Theorem 7. □

E Artificial critical points

Being given a Lipschitz continuous function on $\mathbb{R}^p$ and a conservative field D, one has two types of D-critical points:

- Clarke critical points: $\partial^c f(x) \ni 0$, we denote the set of these points by $\mathrm{crit}^c f$
- Artificial critical points $\partial^c f(x) \not\ni 0$ and $D(x) \ni 0$, we denote this set by $\mathrm{crit}^a f$

Critical values are defined accordingly as images of critical points.

Proposition 8 (Artificial critical points) *Assume $f : \mathbb{R}^p \to \mathbb{R}$ and $D : \mathbb{R}^p \rightrightarrows \mathbb{R}^p$ are definable in a common o-minimal structure. The connected components C_i of $\mathrm{crit}^a f$, which are in finite number, satisfy*

(i) $\dim C_i < p$

(ii) *$f(C_i)$ is a singleton, and as a consequence the D critical values of f are in finite number,*

(iii) *$\mathrm{crit}^a f$ does not contain local minimum (nor local maximum)*

Proof : By definability of $\mathrm{crit}^a f$, the number of connected components is finite.

If C_i had full dimension it would contain a non trivial ball on which f should be constant by the integral property. This would in turn imply that the points in the ball would also be local minimum and thus Clarke critical, which is impossible.

To see that the critical values are in finite number it suffices to evoke the fact that Clarke critical values are finite [11] and use that artificial critical values are in finite number.

By definability the connected components are arcwise-connected with piecewise C^1 paths. Using the integral property this shows f is constant on C_i.

(iii) is obvious since local minimum or maximum are Clarke critical. □

As explained in the introduction, artificial critical points are "computing artefacts", whence their names. For algorithmic differentiation the "gradient" provided by a program is zero while the point might even be a smooth non critical point. We consider the setting of the mini-batch algorithm of the last section.

Theorem 7 *Assume that each $f_1, \ldots, f_n$ belongs to $\mathcal{S}$. There exists a finite subset of steps $F \subset (0, +\infty)$ and a zero measure meager subset N of $\mathbb{R}^p$, such that for any positive sequence $\gamma_k = o(1/\log k)$ avoiding values in F, and any almost surely bounded sequence with initial condition in $\mathbb{R}^p\backslash N$, we have*

- *$J(x^k)$ converges towards a Clarke critical value almost surely,*
- *the cluster points of x^k are Clarke critical point almost surely.*

Proof : The proof is twofold. We first prove that the set of initial conditions leading to an artificial critical point or more generally to a non differentiability point within a finite time is "small". We then use this fact to conclude.

Claim 3 *Let $g : \mathbb{R}^p \to \mathbb{R}$ be a definable differentiable function. Set, for $\lambda > 0$,*

$$\Phi_\lambda = \lambda Id - \nabla g,$$

where Id denotes the identity. There exists a finite set F in $(0, +\infty)$ such that,

$$\forall \lambda \in (0, +\infty) \backslash F,\ \forall Z \subset \mathbb{R}^p \textit{ definable }, \dim Z < p \Rightarrow \dim \Phi_\lambda^{-1}(Z) < p. \tag{12}$$

Proof of the claim. Denote by L the set of points where g is twice differentiable so that L is dense and definable. Denote by $\lambda_1, \ldots, \lambda_p : L \to \mathbb{R}$ a representation of the eigenvalues of $\nabla^2 g$. Refine L to be contained in the common domain of differentiability for each λ_i, L remains open and dense. By the definable Sard's theorem the critical values of each function λ_i is finite, so that the set of all these values which we denote by F is itself finite.

Take a positive real $\lambda \notin F$ and consider the set

$$K_\lambda := \{x \in L : \Phi'_\lambda(x) = \lambda Id - \nabla^2 g(x) \text{ is not invertible}\}.$$

By diagonalization, we see that the determinant of $\Phi'_\lambda(x)$ is $\prod_{i=1}^{d}(\lambda - \lambda_i(x))$ for any x, thence

$$K_\lambda \subset \bigcup_{i=1}^{m} \{x \in L,\ \lambda_i(x) = \lambda\}.$$

Since λ is a regular value for each λ_i the previous set is a finite union of manifolds of dimension $p-1$, see e.g., [19]. This implies that the set $\mathbb{R}^p \backslash K_\lambda = \{x \in L : \Phi'_\lambda(x)$ is invertible $\}$ is dense. Using the above, we deduce that there exists finitely many open connected subsets $U_1, \ldots, U_r \subset L$ of $\mathbb{R}^p \backslash K_\lambda$ such that $U_1 \cup \ldots \cup U_r$ is dense in L and thus in $\mathbb{R}^p$. Take now $Z \subset \mathbb{R}^p$ definable with $\dim Z < p$. Assume towards a contradiction that there exists a nonempty open ball B in $\Phi_\lambda^{-1}(Z)$. In that case B must have a nonempty intersection with some U_{i_0}. The set $\Phi_\lambda(B \cap U_{i_0})$ is open because Φ_λ is a diffeomorphism on U_i on its image. Since on the other hand we have $\Phi_\lambda(B \cap U_{i_0}) \subset Z$, we have a contradiction and the claim is proved. □

For each $I \subset \{1, \ldots, n\}$, we denote by $f_{I,1}, \ldots, f_{I,m_I}$ the bricks attached to f_I where $m_I \geqslant 1$. Denote by Sing the set of points on which at least one f_I is non differentiable and C the set of points for which $\widehat{\nabla} f_I \neq \nabla f_I$ for at least one I. By Proposition 3 and definability, Sing and C are finite unions of manifolds of dimension at most $p-1$.

Set $\Phi^k_{I,j} = Id - \gamma_k \nabla f_{I,j}$, with $I \subset \{1, \ldots, m\}$, $j \in \{1, \ldots, m_I\}$ and Id denotes the identity. Applying Claim 3, we can find a finite set F for which $\gamma_k \notin F$ implies that each $\Phi^k_{I,j}$ has the property (12). Indeed, for each $I \subset \{1, \ldots, m\}$, $j \in \{1, \ldots, m_I\}$, there is $F_{I,j} \subset \mathbb{R}$ finite such that $f_{I,j}$ has property (12). Since the subsets I are in finite number and each m_I is finite, the set $F = \bigcup_{I \subset \{1,\ldots,m\}} \bigcup_{j \in \{1,\ldots,m_I\}} F_{I,j}$, is also finite. For each $k \in \mathbb{N}$, $I \subset \{1, \ldots, m\}$, $j \in \{1, \ldots, m_I\}$. Remark that if $\gamma_k \notin F$ then $\Phi^k_{I,j}$ has property (12).

For $k \leqslant k_0$ fixed, let us consider the finite set of definable mappings defined by

$$\Psi_{k_0} := \left\{ \prod_{j=1}^{k} \Phi^j_{I_j, i_j} : k \leqslant k_0, I_j \subset \{1, \ldots, n\}, i_j \in \{1, \ldots, m_{I_j}\} \right\}.$$

We now assume that $\gamma_k \notin F, \forall k \geqslant 0$, so that each mapping in Ψ_{k_0} has the property (12) and

$$N_{k_0} := \{x \in \mathbb{R}^p : \exists k \leqslant k_0, \exists \Phi \in \Psi_k(x) \in C \cup \mathrm{Sing}\}$$

These are initial conditions in U leading to an artificial critical or a non-differentiability point within U before time k_0.

We can also write

$$N_{k_0} \subset \bigcup_{\Phi \in \Psi_{k_0}} \Phi^{-1}(C \cup \mathrm{Sing}).$$

From stratification arguments we know that Sing has a dimension lower than $p-1$. On the other hand, C has dimension strictly lower than p by Proposition 3. Claim 3 applies and yields

$\dim \Phi^{-1}(C \cup \mathrm{Sing}) < p$ for all $\Phi \in \Phi_{k_0}$. As a consequence N_{k_0} is closed with nonempty interior and so does $N := \cup_{k\in\mathbb{N}} N_k$ by Baire's theorem. Similarly N has zero measure as a countable union of zero measure sets.

This proves that any sequence with initial condition out of N must remain in the zone of differentiability of J as well as all f_I. In particular if I is taken uniformly at random among possible subsets, for all $x \notin N$, we have $\mathbb{E}_I[\widehat{\nabla} f_I(x)] = \widehat{\nabla} J(x) = \nabla J(x) = \partial^c J(x)$, so that these specific sequences can also be seen as stochastic subgradient sequences for J. To be more specific, the sequence x_k can be seen as one of the sequence generated by the algorithm

$$y_{k+1} \in y_k - \gamma_k \partial^c J(y_k) + \epsilon_k$$

where ϵ_k is a random noise with zero mean. Using general results [20, 5], we know that y_k sequences, when bounded almost surely, have limit points which are Clarke critical. □