



**UNIVERSIDAD DE INVESTIGACION DE TECNOLOGIA
EXPERIMENTAL YACHAY TECH**

Escuela de Ciencias Matemáticas y Computacionales

**TITULO: Theory and Implementation of the Savvy Ball Method
with application to machine learning**

Trabajo de integración curricular presentado como requisito para la
obtención del título de Matemático

Autor:

Manosalvas Holguin Peter Sly

Tutor:

Ph.D. Peluffo Diego

Urququí, julio de 2020

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE MATEMÁTICA
ACTA DE DEFENSA No. UITEY-ITE-2020-00024-AD

A los 15 días del mes de abril de 2020, a las 17:30 horas, de manera virtual mediante videoconferencia, y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa	Dr. MENDEZ PEREZ, MIGUEL ANGEL , Ph.D.
Miembro No Tutor	Dr. ACOSTA ORELLANA, ANTONIO RAMON , Ph.D.
Tutor	Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.

El(la) señor(ita) estudiante **MANOSALVAS HOLGUIN, PETER SLY**, con cédula de identidad No. **0953446614**, de la **ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES**, de la Carrera de **MATEMÁTICA**, aprobada por el Consejo de Educación Superior (CES), mediante Resolución **RPC-SO-15-No.174-2015**, realiza a través de videoconferencia, la sustentación de su trabajo de titulación denominado: **THEORY AND IMPLEMENTATION OF THE SAVVY BALL METHOD WITH APPLICATION TO MACHINE LEARNING**, previa a la obtención del título de **MATEMÁTICO/A**.

El citado trabajo de titulación, fue debidamente aprobado por el(los) docente(s):

Tutor	Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.
--------------	---

Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.

Previamente cumplidos los requisitos legales y reglamentarios, el trabajo de titulación fue sustentado por el(la) estudiante y examinado por los miembros del Tribunal Calificador. Escuchada la sustentación del trabajo de titulación a través de videoconferencia, que integró la exposición de el(la) estudiante sobre el contenido de la misma y las preguntas formuladas por los miembros del Tribunal, se califica la sustentación del trabajo de titulación con las siguientes calificaciones:

Tipo	Docente	Calificación
Presidente Tribunal De Defensa	Dr. MENDEZ PEREZ, MIGUEL ANGEL , Ph.D.	8,0
Miembro Tribunal De Defensa	Dr. ACOSTA ORELLANA, ANTONIO RAMON , Ph.D.	9,5
Tutor	Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.	7,0

Lo que da un promedio de: **8.2 (Ocho punto Dos)**, sobre 10 (diez), equivalente a: **APROBADO**

Para constancia de lo actuado, firman los miembros del Tribunal Calificador, el/la estudiante y el/la secretario ad-hoc.

MANOSALVAS HOLGUIN, PETER SLY
Estudiante

MIGUEL ANGEL MENDEZ PEREZ
Firmado digitalmente por MIGUEL ANGEL MENDEZ PEREZ
 Fecha: 2020.05.29 08:21:24 -05'00'

Dr. MENDEZ PEREZ, MIGUEL ANGEL , Ph.D.
Presidente Tribunal de Defensa

DIEGO HERNAN PELUFFO ORDONEZ
Firmado digitalmente por DIEGO HERNAN PELUFFO ORDONEZ
 Fecha: 2020.05.28 22:01:50 -05'00'

Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.
Tutor

ANTONIO RAMON ACOSTA ORELLANA
Firmado digitalmente por ANTONIO RAMON ACOSTA ORELLANA
 Fecha: 2020.06.12 07:34:55 -05'00'

Dr. ACOSTA ORELLANA, ANTONIO RAMON , Ph.D.
Miembro No Tutor

DAYSY
MARGARITA
MEDINA BRITO

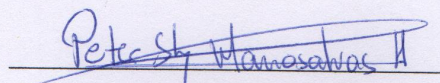
Firmado digitalmente
por DAYSY MARGARITA
MEDINA BRITO
Fecha: 2020.06.11
21:00:47 -05'00'

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Autoría

Yo, **Peter Sly Manosalvas Holguín**, con cédula de identidad **0953446614**, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así como, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autora (a) del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, marzo 2020.



Peter Sly Manosalvas Holguin

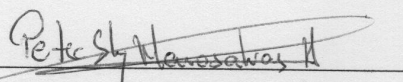
C.I. 0953446614

Autorización de publicación

Yo, **Peter Sly Manosalvas Holguín**, con cédula de identidad **0953446614**, cedo a la Universidad de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Urucuí, marzo 2020.



Peter Sly Manosalvas Holguin

C.I. 0953446614

Acknowledgements

I am sincerely thankful to Andreas Griewank, Ph.D., for providing me with the opportunity to work by his side and guiding me through the many discussions we had.

I am also thankful to my very last advisor, Diego Peluffo, Ph.D., for supporting me in the administrative procedures of the degree project.

I would like to express my special thanks of gratitude to my friends, family and classmates who helped me to face all the hardships during the last five years.

We did it!!!

Abstract

Recently, computer scientists have considered the use of second order differential equations [1] in order to provide a dynamic search trajectory to train neural networks [2], [3]. They are based on the *heavy ball* method of B.T. Polyak [4]. Previous research focused on using Polyak's method in order to speed up the convergence rate to a local minimizer compared to simple steepest descent. We focus here on the global optimization aspect and weaken the requirement on the objective to Lipschitz continuity instead of twice continuous differentiability [2]. We analyze theoretically the non-smooth but convex case where the ODE generalizes to an Ordinary Differential Inclusion (ODI). We show numerical results for implementation of what we call Savvy Ball method which was referred to as TOAST in [5], using the parallel programming environment OpenMP.

Key Words: ANNs, ODI, Heavy Ball, MNIST

Resumen

Actualmente, científicos computacionales han considerado el estudio de ecuaciones diferenciales de segundo orden, para brindar una trayectoria de búsqueda dinámica al entrenamiento de redes neuronales artificiales. Estas están basadas en el método de B.T. Polyak. Investigaciones pasadas se han enfocado en el uso de este método para acelerar la tasa de convergencia a un mínimo local comparado al método steepest descent. Esta tesis se enfoca en el aspecto de optimización global y asume a la función objetivo como Lipschitz continua en vez de dos veces Lipschitz diferenciable. Analizamos teóricamente los casos no continuos y convexos donde la EDO se generaliza como una Inclusión Diferencial Ordinaria. Mostramos resultados numéricos de la implementación de nuestro método Savvy Ball mencionado antes como TOAST a través de la región paralela dada por la librería OpenMP.

Palabras Claves: Redes Neuronales, ODI, Heavy Ball, MNIST

Contents

1	Introduction and Motivation	8
2	Derivation of the Savvy Ball Trajectory	11
3	Fundamental Results of Nonsmooth Analysis	15
4	Analyzing the convex case	16
5	Properties in the homogeneous case	21
6	Closed form solution in prox-linear case	23
7	The Generalized Abs-Normal and Abs-Linear Forms	27
8	Piecewise Numerical Integration	31
9	Experimental Results	32
9.1	Classification Problem: The MNIST Dataset for Digit Recognizer	32
9.2	Numerical Results	36
10	Summary and Conclusion	37
10.1	Conclusions	38
A	Savvy Ball method for the MNIST character recognition problem: Code	42

1 Introduction and Motivation

Reproducing the human learning process is a study of almost thousands of years ago. The first step to reinforce this insight appeared in 1943 with the model of a simple neural network with electrical circuits [6]. So far the most significant transition happened with the Perceptron Algorithm in 1958 by Rosenblatt, a neurobiologist from Cornell. He proposed the oldest neural network capable of being trained with datasets [6]. The basics of the Perceptron Algorithm has been extended to the so called Artificial Neural Networks (ANN).

Prediction models such as ANNs and Machine Learning (ML) algorithms solve a learning problem through steps such as: training, validating and testing the input datasets. Namely, the objective function depends on a set of parameters and a known dataset called samples. These functions are nonsmooth and nonnegative. Considering a non-smooth optimization problem, ANNs can be studied as a globally minimization problem for piecewise smooth objective functions [7]. The most common approach is to neglect the non-smoothness applying a stochastic gradient and to hope that it works with a global optimality.

In the context of ML, especially supervised learning, ANN looks for a prediction function (i.e. predictor) to fit the training sample and then to predict data based on the parameters adjustment. Here, computer scientists named the average fitness error Empirical Risk (ER), which is the value to minimize as a non-smooth optimization problem. For example, we may consider the single-layer case with the most used activation function for ANN, the Rectifier with constant output weights $p \in \{-1, 1\}^d$. For this case, the prediction function is defined as follows

$$f(\tilde{W}; x) \equiv p^\top \max(0, Wx + b) \text{ with } \tilde{W} \equiv (W, b) \in \mathbb{R}^{d(n+1)}$$

where the feature vector $x \in \mathbb{R}^n$ and the corresponding label $y \in \mathbb{R}^m$ comes from the training set. The predictor has parameters such as the weights $\tilde{W} \in \mathbb{R}^{d \times (n+1)}$ including the inhomogeneous shift $b \in \mathbb{R}^d$. Hence, the optimization problem is stated as follows,

$$\min_{\tilde{W}} \varphi(x) = \frac{1}{qm} \sum_{i=1}^m \|f(\tilde{W}; x_i) - y_i\|_q^2 \quad \text{for } q \in \{1, 2\}$$

given a training sample set of m pairs $(x_i, y_i)_{i=1}^m$. Here, φ is the empirical risk and the objective function of our learning task. [8]. Later on in describing optimization algorithms we will denote

the optimization variables \tilde{W} as x as customary when no confusion is possible.

In a neural network, the activation function transform the summed weighted input from the node, i.e. the activation of the node for that input. In general, the most popular activation function is the rectified linear activation function

$$Re(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \quad (1)$$

It has become the default activation function because of its performance and training proficiency, even though it is not differentiable at the origin, which turns the learning problem into a nonsmooth optimization task.

Moreover, as observed before by Griewank and Rojas in [5], classical optimizers such as Stochastic Gradient Method (SGM) and Steepest Descent (SD) also known as back-propagation may be stumped at saddle points or local minima. Moreover, they assumed that the standard gradient exists and is available at all iterates. As they already stated in [8] we should consider the following

Proposition 1.1. *For $q = 1$ let's assume that the locally Lipschitz continuous function $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ has a non-empty bounded level set $\{x \in \mathbb{R}^n : \varphi(x) \leq \varphi(x_0)\}$ for some $x_0 \in \mathbb{R}^n$. Then, φ is not differentiable at all geometrically isolated local minimizers and at least one global minimizers.*

Here, we consider the generalized gradient $\partial\varphi(x)$ defined by Clarke [9] whose existence follows from the assumption that the objective function $\varphi(x)$ is everywhere locally Lipschitz continuous. Then the steepest descent differential inclusion

$$-\dot{x}(t) \in \partial\varphi(x(t)) \quad \text{from } x(0) = x_0 \in \mathbb{R}^n \quad (2)$$

has at least one absolutely continuous solution trajectory $x(t)$ according to the theory developed by Filippov (see [10] and [11]). According to the theorem of Picard-Lindelöf the solution $x(t)$ is unique for the smooth case i.e. when $\partial\varphi(x)$ is equivalent to the singleton $\{\nabla\varphi(x)\}$ and the gradient $\nabla\varphi(x)$ is Lipschitz continuous. Moreover, uniqueness of $x(t)$ by (2) can also be assured if φ is not everywhere differentiable but globally convex [12][8]. Moreover, it can then also be proven that the limiting point $x_* = \lim_{t \rightarrow \infty} x(t)$ exists and is a global minimizer, unless φ does not attain its global infimum, possibly because it is unbounded below. However, the minimizing trajectory $x(t)$ may have kinks and is in general difficult to follow unless φ is piecewise linear as shown in [13].

Although that is not really needed in the convex case, one may try to increase the chances of approaching a global or at least low local minimum by following the second order differential inclusion

$$-\ddot{x}(t) \in \partial\varphi(x(t)) \quad \text{from } x(0) = x_0 \quad \text{with } \dot{x}(0) \in \partial\varphi(x_0). \quad (3)$$

In the smooth case, this equation models the motion of a body in a potential field given by $\varphi(x)$ which motivated its naming by Polyak [14]. Hence methods that are based on this equation and its discretizations are called *heavy ball* methods. In terms of global optimization this method has the advantage that the momentum of the balls motion makes it to achieve an optimal convergence rate. This is the idea behind the method of Snyman and Fatti [15] who developed a complex optimization scheme involving random restarts and local minimization runs. In [16] they did a detailed numerical comparison with differential evolution and other currently popular derivative free methods. Global optimization methods are notoriously difficult to compare, especially if nonsmoothness is allowed, see for example the survey [17]. Thus we will refrain from any claims of empirical efficiency and instead emphasize theoretical properties. However, we note that practically all objectives in machine learning are piecewise smooth in a way that allows the evaluation of gradients or rather their piecewise linearization [18].

2 Derivation of the Savvy Ball Trajectory

As already argued in [2] the drawback of the heavy ball motion is that it picks up speed and momentum when going down hill and slows down when climbing up hill. It may thus race by an attractive local minimum near the bottom of a valley but settle down in a dent on top of a mountain. This is exactly the opposite behavior of what one wants in global minimization. We therefore prefer the *savvy ball equation*

$$\frac{d}{dt} \left[\frac{\dot{x}(t)}{(\varphi(x(t)) - c)^e} \right] \in \frac{-e \partial\varphi(x(t))}{(\varphi(x(t)) - c)^{e+1}} = \partial \left[\frac{1}{(\varphi(x(t)) - c)^e} \right]. \quad (4)$$

It can be rewritten as a first order system of one vector differential equation and one vector inclusion

$$\dot{x}(t) = v(t)(\varphi(x(t)) - c)^e, \quad (5)$$

$$\dot{v}(t) \in \frac{-e \partial\varphi(x(t))}{(\varphi(x(t)) - c)^{e+1}}. \quad (6)$$

Now we prove that the conditions of the Filippov theorem [11] are satisfied, namely the values of the multi-function on the right hand side are convex and outer semi-continuous(o.s.c) by definition of the Clarke differential.

Definition 2.1. *A set-valued mapping $F : X \rightrightarrows Y$ is outer semi-continuous (osc) at x_0 if*

$$\limsup_{x \rightarrow x_0} F(x) \subset F(x_0)$$

Proposition 2.2. *Let $\partial\varphi$ be the gradient of a Lipschitzian function. Then, the multi-valued function*

$$F(x, v) = \left[(\varphi(x) - c)v, \frac{-eg}{(\varphi(x) - c)^{e+1}} \right]_{g \in \partial\varphi(x)}$$

has convex images and is outer semi continuous.

Proof. It is well known that the Clarke gradient is outer semi-continuous, i.e., for any sequence

$$x_i \rightarrow x_* \quad (7)$$

with $w_i \in \partial\varphi(x_i)$ such that

$$w_i \rightarrow w_* \tag{8}$$

we have

$$w_* \in \partial\varphi(x_*)$$

Now, let's consider a double sequence

$$(x_i, v_i) \rightarrow (x_*, v_*)$$

and

$$s_i = [v_i(\varphi(x_i) - c)^e, \frac{-ew_i}{(\varphi(x_i) - c)^{e+1}}] \in F(x_i, v_i)$$

Since φ is continuous, from (7) and (8), we have immediately

$$v_i(\varphi(x_i) - c)^e \rightarrow v_*(\varphi(x_*) - c)^e$$

Similarly, it follows that

$$\frac{-ew_i}{(\varphi(x_i) - c)^{e+1}} \rightarrow \frac{-ew_*}{(\varphi(x_*) - c)^{e+1}}$$

We know that $w_* \in \partial\varphi(x_*)$, so that

$$s_* \in F(x_*, v_*)$$

i.e., F is also outer semi-continuous.

Since φ is a convex set, we have for

$$g_i \in \frac{-e\partial\varphi(x)}{(\varphi(x) - c)^{e+1}} \quad \text{and} \quad \sum_i \alpha_i = 1 \quad \text{with} \quad \alpha_i \geq 0$$

$$\begin{aligned} \sum_i \alpha_i (v(\varphi(x) - c)^e, g_i) &= (v(\varphi(x) - c)^e, \sum_i \alpha_i g_i) \\ &= (v(\varphi(x) - c)^e, g) \quad \text{where} \quad g \in \frac{-e\partial\varphi(x)}{(\varphi(x) - c)^{e+1}} \end{aligned}$$

which means that the image of F is convex. □

Hence there exists (at least) one absolutely continuous solution $(x(t), v(t))$ for $t \geq 0$ and obviously the tangent $\dot{x}(t)$ has the same property. Since $v(t)$ is the integral of its almost everywhere existing derivative $\dot{v}(t)$ we get the integrated form

$$v(t) = \frac{\dot{x}(t)}{[\varphi(x(t)) - c]^e} \in \frac{\dot{x}_0}{[\varphi(x_0) - c]^e} - \int_0^t \frac{e \partial\varphi(x(\tau)) d\tau}{[\varphi(x(\tau)) - c]^{e+1}}. \quad (9)$$

Definition 2.3. Given k points $x_1, \dots, x_k \in \mathbb{R}^n$ a conic combination of these k points is a vector of the form $\lambda_1 x_1 + \lambda_2 x_2 + \dots + \dots + \lambda_k x_k$ where $\{\lambda_i\}_i \subset \mathbb{R}_+$. As k tends to infinity we can generalize the sum to an integral with a positive weight function.

Thus we see that the tangent $\dot{x}(t)$ is a conic combination of its initial value and negative generalized gradients along the solution trajectory $x(\tau)$ for $\tau \in [0, t]$. To get a better feel how the trajectory behaves it is good to look at the second order expression for the acceleration $\ddot{x}(t)$, namely

$$-\ddot{x}(t) \in \left[I - \frac{\dot{x}(t) \dot{x}(t)^\top}{\|\dot{x}(t)\|^2} \right] \frac{[e \partial\varphi(x(t))]}{[\varphi(x(t)) - c]^{e+1}} \quad \text{with} \quad \|\dot{x}(0)\| = 1. \quad (10)$$

which can be obtained by differentiation of (9). This formulation was derived directly in [2] for the smooth case where we have a proper gradient $\partial\varphi(x) = \{\nabla\varphi(x)\}$. As we can see the tangent length is normalized such that $\|\dot{x}(t)\| = 1$, which means that t represents an arc length parametrization of the trajectory $x(t)$. Now we can get a better feel for the role of the two constants c called the *target value* and $e > 0$ the *sensitivity parameter*. Since the projection in front of the right hand side merely serves to normalize the tangent length $\|\dot{x}(t)\| = 1$, we have essentially the heavy ball equation (3) for the transformed potential $\log(\varphi(x) - c)^e$. The choice of the two method parameters e and c is critical, as is typical for global optimization schemes, which very often require many more than two more or less intuitive parameter choices. The attribute *savvy ball* suggests that the method knows what it is doing as specified by the two hyper parameters.

The target c indicates at any stage what range of function values we are looking for; it should always be below the values $\varphi(x)$ already obtained at previous points x during the optimization of the current objective. Once we have reached the current target value c it can be lowered to a lower level unless we are already happy with what has been achieved. In machine learning the objective is frequently a nonnegative empirical risk that one wants to push down as close to zero as possible. Then a successive halving strategy makes sense until we reach a level

that might be too ambitious.

As long as the actual function value $\varphi(x(t))$ is much larger than the target c the reciprocal $1/[\varphi(x(t))c]^{e+1}$ and thus the second derivative $\ddot{x}(t)$ is small so that the trajectory moves more or less straight ahead, hopefully ignoring smaller wiggles in the objective function. When $\varphi(x(t))$ comes closer to the target value c the reciprocal grows and the second derivative $\ddot{x}(t)$ corrects the tangent $\dot{x}(t)$ towards the local direction of steepest descent or at least one element of the negative generalized gradient $-e \partial\varphi(x(t))$. In the limit as $\varphi(x(t))$ converges to c the second order trajectory (10) reduces the first order trajectory (2) with a different parametrization.

The sensitivity parameter e is also very important but not quite as variable as c . At least in the convex case e should be selected equal to 1, which we consider as its default value. In general the ideal value of e would be the reciprocal of the growth rate of φ towards infinity so that for some reference point \hat{x}

$$\varphi(x) - \varphi(\hat{x}) \sim \|x - \hat{x}\|^{1/e}.$$

Of course such an exponent is hard to come by, but for smooth and essentially quadratic functions a value $e \in [\frac{1}{2}, 1]$ can be recommended. In our numerical experiments we have used the default value $e = 1$ throughout.

The thesis is organized as follows. In the following Section 3 we show some results from non-smooth analysis which is the background needed to avoid the optimizer's house of horrors showed in [5], in Section 4 we bring further details for the convex case and show some conditions where convergence is guaranteed. We develop some mathematical properties of the Savvy Ball trajectory in the convex but not necessarily smooth case. In particular we show that the inclusion (10) reaches the target level if that is not empty. We also establish rates of convergence in terms of the parameter t , which represents the arclength of the search trajectory. In the subsequent Section 5 we consider the case where the objective is non-convex but homogeneous in the catchments surrounding some local minimizers. There we can show not only convergence to desirable local minimizers below or at the target level but also divergence from undesirable local minimizers above the target level. In Section 6 we show that on affine functions with a proximal term the search trajectory is a circular segment, which allow us to integrate piecewise linear functions in abs-normal form exactly. In Section 7 we introduce the generalized abs-normal form and its abs-linear approximation at a given reference point. In

Section 8 and 9 we consider various numerical integration options and show numerical results on the standard MNIST problem. .

3 Fundamental Results of Nonsmooth Analysis

Since the prediction function and the loss function are Lipschitz continuous we can assume that the empirical risk satisfies the same condition. Then we can apply the following theorem

Theorem 3.1. *Let U be an open subset of \mathbb{R}^n and $\varphi : U \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is locally Lipschitz continuous. Then φ is differentiable almost everywhere (with respect to the Lebesgue measure λ). That is, there is a set $E \subset U$ with $\lambda(U \setminus E) = 0$ and such that for every $x \in E$ there is a linear function $\nabla\varphi(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ with*

$$\lim_{y \rightarrow x} \frac{\|\varphi(y) - \varphi(x) - \nabla\varphi(x)(y - x)\|}{\|y - x\|} = 0$$

At the exceptional points $x \in U \setminus E$ the so called limiting gradient can be defined as

Definition 3.2. *At all points $x \in U$ we set*

$$g \in \partial^L\varphi(x) \iff \left\{ \lim_{n \rightarrow \infty} \nabla\varphi(x_n) \rightarrow g : x_n \rightarrow x \text{ and } x_n \in E \right\}$$

Then, we can define the Clarke's generalized derivative as follows [19]

$$\partial^C\varphi(x) \equiv \text{conv}(\partial^L\varphi(x))$$

So now we can take a look into another known definition for generalized derivative which is defined by the Hadamard derivative or directional derivative.

Definition 3.3. *Let's consider $\varphi : U \rightarrow \mathbb{R}$. The Hadamard derivative in the direction $h \in \mathbb{R}^n$ at $x \in U$ is denoted by*

$$\varphi'(x; h) = \lim_{t \rightarrow 0^+} \frac{[\varphi(x+th) - \varphi(x)]}{t} \in \mathbb{R} \cup \{\pm\infty\}$$

For merely Lipschitzian φ the Hadamard derivative may not exist in some directions h . But it does always exist in the convex case, where we may alternatively define the so-called sub-gradient of φ by:

Definition 3.4. *Let's assume that φ is convex, we define the sub-gradient $\partial\varphi$ through $\varphi'(x_0, \cdot)$,*

$$\partial\varphi \equiv \{u \in \mathbb{R}^n : u^\top h \leq \varphi'(x_0; h), \text{ for } h \in \mathbb{R}^n\}$$

which is identical to the set

$$\{u \in \mathbb{R}^n : \varphi(x) \geq \varphi(x_0) + u^\top(x - x_0), \text{ for } x \in \mathbb{R}^n\}$$

Remark 3.5. *The definition of Clarke's generalized derivative is equivalent to the concept of subgradient when φ is Lipschitz continuous and convex [20].*

Remark 3.6. *Let's show an example where the subgradient can be empty for a Lipschitz function. Let's consider the real-valued function*

$$f(x) = \begin{cases} x^2 \sin(\frac{1}{x}) & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases} \quad (11)$$

Here an important result from convex analysis states that since f is not the upper envelope of the set of all its affine minorants, we have the subgradient is empty. [21]

4 Analyzing the convex case

Practically all convergence proofs for gradient descent methods (see e.g. [22] and [23]) assume that in some vicinity of a stationary point the objective function is convex. To show an asymptotic linear rate, which is sometimes called exponential in the machine learning literature, one typically assumes strict convexity. Certainly the convex case is interesting in its own right for both theoretical and practical reasons. We conjecture that in the convex case the system (5), (6) has only one solution, but our attempts to prove that have so far failed. As it turns out in the convex case we should use $e = 1$ and since $\|x(t)\| = 1$ is enforced automatically we get the

slightly simplified differential inclusion

$$-\ddot{x}(t) \in \left[I - \dot{x}(t) \dot{x}(t)^\top \right] \frac{\partial\varphi(x(t))}{[\varphi(x(t)) - c]} \quad \text{with } -\dot{x}(0) \in \alpha\partial\varphi(x_0) \neq 0 \quad (12)$$

where $0 < \alpha \in \mathbb{R}$ can be selected such that $\|\dot{x}(0)\| = 1$. Notice that, if $\partial\varphi(x_0)$ does include the zero vector 0 the convexity already ensures that x_0 is a global minimizer so that then nothing remains to be done. Otherwise, we can take $-\dot{x}(0)$ to be any element of $\partial\varphi(x_0)$ positively scaled to a unit vector. While such $\dot{x}_0 = \dot{x}(0)$ needs not be a descent direction with respect to the objective function $\varphi(x)$, it is well know that for any differentiable path $x(t)$ from $x(0) = x_0$ with initial tangent \dot{x}_0 and any $\hat{x} \in \mathbb{R}^n$ the Euclidean distance function $r(t) = \|x(t) - \hat{x}\|$ satisfies

$$\begin{aligned} r_0 \dot{r}_0 &= \left. \frac{d}{dt} \frac{1}{2} r(t)^2 \right|_{t=0} = (x_0 - \hat{x})^\top \dot{x}_0 \\ &\in \alpha (\hat{x} - x_0)^\top \partial\varphi(x_0) \leq \alpha (\varphi(\hat{x}) - \varphi(x_0)) . \end{aligned} \quad (13)$$

where the last inequality follows again by convexity. Hence we see that the distance to any point \hat{x} with $\varphi(\hat{x}) < \varphi(x_0)$ is strictly decreasing for all small $t > 0$. That applies especially to any global minimizer \hat{x} if they exist.

Proposition 4.1 (Convergence in the convex case).

Suppose $\varphi : \mathbb{R}^n \mapsto \mathbb{R}$ is globally convex. Then we have for almost all t in the maximal interval $[0, t_) \subset [0, \infty)$ on which the denominator $\varphi(x(t)) - c$ is positive*

(i) *For any reference point \hat{x} the distance $r(t) = \|x(t) - \hat{x}\|$ satisfies*

$$\frac{\dot{r}(t)r(t)}{[\varphi(x(t)) - c]} \leq \frac{\dot{r}_0 r_0}{[\varphi(x_0) - c]} + [\varphi(\hat{x})c] \int_0^t \frac{d\tau}{[\varphi(x(\tau)) - c]^2} \quad (14)$$

(ii) *If the level set $\{x \in \mathbb{R}^n : \varphi(x) \leq c\}$ is non-empty the trajectory converges s.t.*

$$\varphi(x_*) = c \quad \text{for } x_* \in \lim_{t \rightarrow t_*} x(t)$$

provided $\dot{r}(t) < 0$ at $t = 0$ as assumed in (13) or sometime later along the way.

(iii) If (ii) holds with x_* not being a global minimizer then $t_* < \infty$ and

$$\limsup_{t \rightarrow t_*} \{-\dot{x}(t)\} \subset \{v \in \alpha \partial \varphi(x_*) \mid \|v\| = 1, \alpha > 0\}.$$

where \limsup denotes the outer limit of an ordered family of sets.

(iv) If φ is strongly convex at its unique global minimizer $x_* = \hat{x}$ in that

$$\varphi(x) - \varphi(x_*) \geq \frac{\sigma}{2} \|x - x_*\|^2 \quad \text{for some } \sigma > 0$$

and $c = \varphi(x_*)$ we get exponential convergence in that for all $t \in [0, \infty)$

$$\|x(t) - x_*\| \leq \|x_0 - x_*\| \exp(-\tilde{\sigma}t) \quad \text{for some } \tilde{\sigma} > 0.$$

except in the highly unlikely case where $x(t_*) = x_*$ for finite $t_* < \infty$.

Proof. For later reference we let e be general in the beginning. First we note that by definition of $v(t)$ in (9) and its absolute continuity for almost all t

$$\begin{aligned} \frac{d}{dt} \left[\frac{r(t)\dot{r}(t)}{(\varphi(x(t)) - c)^e} \right] &= \frac{d}{dt} \left[(x(t) - \hat{x})^\top v(t) \right] \\ &= \dot{x}(t)^\top v(t) + (x(t) - \hat{x})^\top \dot{v}(t) \\ &\in \frac{1}{(\varphi(x(t)) - c)^e} - e \frac{(x(t) - \hat{x})^\top \partial \varphi(x(t))}{(\varphi(x(t)) - c)^{e+1}} \\ &= \frac{\varphi(x(t)) + e(\dot{x}(t))^\top \partial \varphi(x(t))c}{(\varphi(x(t)) - c)^{e+1}} \end{aligned} \quad (15)$$

$$\leq \frac{(\varphi(\hat{x}) - c)}{[\varphi(x(t)) - c]^2} \quad \text{if } e = 1 \text{ and } \varphi \text{ convex} \quad (16)$$

Integrating with respect to t we obtain (14), which completes the proof of (i).

To prove (ii) let \hat{x} be any point in the level set. Then the second term on the right hand side of (14) is nonpositive and we get the simple bound

$$\dot{r}(t)r(t) \leq -[\varphi(x(t)) - c][-\dot{r}_0 r_0 / (\varphi(x_0) - c)] < 0 \quad \text{for } t \in [0, t_*]. \quad (17)$$

where the last inequality follows from $\dot{r}_0 < 0$ as guaranteed by the choice of \dot{x}_0 . This means that the distance $r(t) = \|x(t) - \dot{x}\|$ is monotonically decreasing for all \dot{x} with $\varphi(\dot{x}) \leq c$, which implies in particular that $x(t)$ stays within a convex compact subset of \mathbb{R}^n . By definition of the supremal t_* we must have $\liminf_{t \rightarrow t_*} \varphi(x(t)) = c$ or $t_* = \infty$ or both. If both conditions hold $x(t)$ must have a cluster point x_* with $\varphi(x_*) = c$ and since for $\dot{x} = x_*$ the distance $r(t)$ is monotonically decreasing x_* must in fact be a proper limit of $x(t)$ as asserted. If the liminf was not zero and thus $t_* = \infty$ the negative right hand side of (14) would be bounded away from zero and the squared distance $r(t)^2$ would go to zero in finite time for any \dot{x} in the target level. This is obviously a contradiction, which completes the proof of (ii).

To prove the third assertion (iii) we note first that due to x_* not being a global minimizer the generalized gradient $\partial\varphi(x_*)$ cannot contain zero and the same is by outer semicontinuity true for all x in some neighborhood of x_* . Moreover, we can assume that the convex hull T_δ of all $\partial\varphi(x)$ with $\|x - x_*\| < \delta$ does not contain zero and we have $\limsup_{\delta \rightarrow 0} T_\delta = \partial\varphi(x_*)$. Then there is a supporting vector $v \in \mathbb{R}^n$ such that for δ sufficiently small and some $\nu > 0$ we have $v^\top g \geq \nu$ for all $g \in T_\delta$. Moreover we may rewrite (9) to

$$-\dot{x}(t) \in -\dot{x}_0 \frac{[\varphi(x(t)) - c]}{[\varphi(x_0) - c]} + [\varphi(x(t)) - c] \int_0^t \frac{\partial\varphi(x(\tau))}{[\varphi(x(\tau)) - c]^2} d\tau. \quad (18)$$

which shows that $-\dot{x}(t)$ is a conic combination of $-\dot{x}(0)$ and vectors in T_δ with the weight of $-\dot{x}(0)$ going to zero as $x(t)$ converges to x_* . Since the vectors in T_δ are bounded the corresponding multipliers α are bounded below by some α_* so that for all $\dot{x}(t)$ with t larger than some $t_0 < t_*$

$$\begin{aligned} g^\top(-\dot{x}(t)) &\geq \alpha_* g^\top v \geq \alpha_* \nu \equiv \nu_* \\ \implies g^\top(-x(t) + x(t_0)) &\geq \nu_*(t - t_0). \end{aligned}$$

No since by (ii) $x(t) \rightarrow x_*$ the last inequality yields the bounds

$$t \leq t_* \leq t_0 + g^\top(-x_* + x(t_0))/\nu_* < \infty,$$

which completes the proof of (iii) as the limiting set inclusion was already shown.

Finally to prove (iv) we note that by (14) and the strong convexity assumption

$$\dot{r}(t)r(t) \leq -\frac{\sigma}{2}r^2(t)[-r_0r_0/(\varphi(x_0) - c)] < 0 \quad \text{for } t \in [0, t_*]. \quad (19)$$

so that division by $r(t) > 0$ and integration of the differential inequality yields the asserted exponential decline. \square

The proposition says essentially that we loose nothing compared to steepest descent provided we know an upper bound c on the infimal value $\inf\{\varphi(x)|x \in \mathbb{R}^n\}$, which might be minus infinity. The trajectory has a finite length t_* if c is a strict upper bound, but unfortunately this does not mean that it is easy to follow. In particular the curvature $\ddot{x}(t)$ may tend to infinity so that classical path following strategies may result in excessively small step and possibly infinitely many drastic changes of direction. So far we have not found an example where this is actually the case, so the question remains to be explored.

While for a strict upper bound c the trajectory will approach the limit point x_* along a convex cone we might get a spiraling convergence if c exactly equals the global minimum. Of course that is rather unlikely to happen in practice, except if we know that a residual or empirical risk can be driven exactly to zero. For the special case $\varphi(x) = r$ and $c = 0$ we get $\dot{r} = \dot{r}_0 < 0$, which defines a spiral of finite length. Actually, we are sofar also lacking an example, where the spiral is infinitely long.

Unfortunately, Proposition 3.1 does not address the thorny question what happens if the target set $\{x \in \mathbb{R}^n|\varphi(x) < c\}$ is empty and thus not reachable. Then one would want the trajectory to run off to infinity, but that does not always need to happen. For example when $\varphi(x) = r^2$ one can easily see that $c < 0$ and $|\dot{r}_0| < 1$ leads to a trajectory that circles around the origin infinitely often. In such situations we can either lower our expectation by gradually increasing the target value c or reduce the sensitivity e to enable the trajectory to climb the wall around a given undesirable minimizer \hat{x} . Of course the latter makes only sense if sufficiently far away from \hat{x} the function attains other local minimizers, some of them hopefully below or at the actual target. Naturally, this cannot happen when φ is convex as we have assumed in this Section 4. Since for a global optimization method getting away from undesirable stationary points or local minimizers is very important, we consider in the next section the nonconvex case with homogeneous center points.

5 Properties in the homogeneous case

In this section we assume that for some reference point $\hat{x} \in \mathbb{R}^n$ and all $x \in \mathbb{R}^n$

$$\varphi(\hat{x} + \rho(x - \hat{x})) - \varphi(\hat{x}) = [\varphi(x) - \varphi(\hat{x})]\rho^d \quad \text{for } \rho > 0 \quad (20)$$

which means that φ is positively homogeneous of degree d with respect to the *center point* \hat{x} . That holds for example if $\varphi(x) - \varphi(\hat{x})$ is equal to its piecewise linearization $\Delta\varphi(\hat{x}; x - \hat{x})$ as defined in Section 7 and the latter is homogeneous. Notice that we do not assume \hat{x} to be a stationary point, let alone a minimizer.

Proposition 5.1. *Let $\hat{x} \in \mathbb{R}^n$ some reference point. For every $x \in \mathbb{R}^n$, from (20), fixing $\rho = 1$, we have*

$$\partial\varphi(x)(x - \hat{x}) = \{d(\varphi(x) - \varphi(\hat{x}))\}. \quad (21)$$

which means that all elements in $\partial\varphi(x)$ have the same inner product with $x - \hat{x}$.

Proof. By first differentiation (20) with respect to ρ , since φ is positively homogeneous of degree d with respect to \hat{x} we have that

$$\partial\varphi(\hat{x} + \rho(x - \hat{x}))(x - \hat{x}) \ni d[\varphi(x) - \varphi(\hat{x})]\rho^{d-1}$$

with equality holding if $\hat{x} + \rho(x - \hat{x}) \in E$ so that the generalized gradient is a singleton. Taking $\rho = 1$,

$$\partial\varphi(x)(x - \hat{x}) \ni d[\varphi(x) - \varphi(\hat{x})] \quad (22)$$

From the other hand, consider any $g \in \partial^L\varphi(x)$, then we have

$$g = \lim_{i \rightarrow \infty} \nabla\varphi(x_i), x_i \in E$$

Then,

$$\nabla\varphi(x_i)(x_i - \hat{x}) = d(\varphi(x_i) - \varphi(\hat{x}))$$

which implies in the limit when $i \rightarrow \infty$

$$g^\top(x - \hat{x}) = d(\varphi(x) - \varphi(\hat{x}))$$

Clearly, this property is then also true for any convex combination of limiting gradients and thus all elements of the generalized gradient $\partial\varphi$. \square

In other words Proposition 5.1 says that the difference between various elements in $\partial\varphi(x)$ is orthogonal to the *radial direction* $x - \hat{x}$. Then it follows from equation (15) that the distance $r(t) = \|x(t) - \hat{x}\|$ satisfies

$$\frac{d}{dt} \left[\frac{r(t)\dot{r}(t)}{(\varphi(x(t)) - c)^e} \right] = \frac{\varphi(x(t)) + e(\hat{x} - x(t))^\top \partial\varphi(x(t)) - c}{(\varphi(x(t)) - c)^{e+1}} \quad (23)$$

$$= \frac{(1 - ed)[\varphi(x(t)) - \varphi(\hat{x})] + (\varphi(\hat{x}) - c)}{[\varphi(x(t)) - c]^{e+1}} \quad (24)$$

Integrating with respect to t we get

$$\frac{\dot{r}(t)r(t)}{[\varphi(x(t)) - c]^e} = \frac{\dot{r}_0 r_0}{[\varphi(x_0) - c]^e} + \int_0^t \frac{(1 - ed)[\varphi(x(\tau)) - \varphi(\hat{x})] + (\varphi(\hat{x}) - c)}{[\varphi(x(\tau)) - c]^{e+1}} d\tau. \quad (25)$$

Now if we choose $e = 1/d$ it is clear that the right hand side is monotonically growing or falling depending on whether the constant $\varphi(\hat{x}) - c$ is positive or negative, respectively. For any trajectory that moves at some time t towards \hat{x} in that $\dot{x}(t)^\top(x(t) - x_0) = 2\dot{r}(t)r(t) < 0$ we thus get the following result.

Proposition 5.2 (Convergence/Divergence in homogeneous case).

Suppose (21) holds with $e \geq 1/d$ and $\dot{r}(t) < 0$ for any $t \geq 0$. Then $x(t)$ must approach the level set $\{\varphi(x) \leq c\}$ if $\varphi(\hat{x})$ is desirable, i.e. at or below the target level c . If $\varphi(\hat{x})$ is undesirable, i.e. above the target level c , and $e \leq 1/d$ then the trajectory either diverges monotonically towards infinity or reaches the target level somewhere else.

Proof. Let us first consider the case $\varphi(\hat{x}) \leq c$, where \hat{x} is a desirable local minimizer. If the level set was not reached $\dot{r}(t)$ would by (25) be negative and bounded away from zero, which leads to a contradiction since $r(t)$ cannot become negative. Thus we are left with the undesirable case $\varphi(\hat{x}) > c$ but with $e \leq 1/d$. If the trajectory stays clear of the feasible set the trajectory $x(t)$ is

well defined for all positive $t \geq 0$. If $r(t)$ was nevertheless bounded above the integral on the RHS of (25) would diverge. Hence $\dot{r}(t)r(t)$ in the numerator of the left hand side would also grow unbounded, which is impossible. Hence $r(t)$ must be unbounded and its derivative $\dot{r}(t)$ must be positive at some t_1 and thus by (25) all subsequent $t \geq t_1$. This completes the proof as we must have monotonic divergence towards infinity. \square

The Proposition ensures that, once the search trajectory $x(t)$ enters a ball of radius r_0 about some \hat{x} in which $\varphi(x)$ is homogeneous with respect to \hat{x} in the sense of (20) and $e = 1/d$ then $x(t)$ does the right thing, i.e. it must reach the level set if $\varphi(\hat{x}) \leq c$ and otherwise, it either finds another point in the ball below the target level or leaves the ball altogether. The last possibility cannot occur if \hat{x} minimizes φ locally. Hence we see that, provided $e = 1/d$, the local homogeneous minimizers repulse the trajectory if they are undesirable and attract it if they are desirable, i.e. above the target level. Irrespective of the local minimality of \hat{x} the undesirable possibility of the trajectory circling within the ball forever without reaching the target has been excluded. Unfortunately, when $e > 1/d$ or φ does not satisfy the rather stringent homogeneity assumption (20) this may still happen. Nevertheless, since all piecewise smooth functions have homogenous local approximations one can be optimistic that the selective behavior of the trajectory will work in many situations for $e = 1$. In the piecewise linear case to be considered later we will add a quadratic regularization term $\frac{q}{2}\|\hat{x}\|^2$, makes the trajectory turn back towards x_0 when the quadratic term dominates the piecewise linear components, which for $e = 1$ will effectively limit the search domain. We will indicate the presence of this regularization term by the prefix prox and thus refer for example to prox-linear or prox-piecewise-linear functions.

6 Closed form solution in prox-linear case

Now we reconsider the task of computing the savvy ball trajectory to the sum of an affine function and a proximal term in the Euclidean norm of the form

$$\varphi(x) = \mu + g^\top x + \frac{q}{2}\|x\|^2 \quad \text{with } q \geq 0. \quad (26)$$

Obviously this kind of simple model is both convex and homogeneous. The latter property holds if $q = 0$ with degree $d = 1$ with respect to all $\dot{x} \in \mathbb{R}^n$ and if $q \neq 0$ with degree $d = 2$ with respect to its unique global minimizer $x_* = -g/q$.

Theorem 6.1. For $\varphi(x)$ as above (x_0, \dot{x}_0) with $c < \varphi_0 = \varphi(x_0)$ and $\|\dot{x}_0\| = 1$ the solution of the initial value problem for the ODE

$$\ddot{x}(t) = - \left[I - \dot{x}(t) \dot{x}(t)^\top \right] \frac{\nabla \varphi(x(t))}{[\varphi(x(t)) - c]} \quad (27)$$

is given for small nonnegative t by the circle

$$x(t) = x_0 + \frac{\sin(\omega t)}{\omega} \dot{x}_0 + \frac{1 - \cos(\omega t)}{\omega^2} \ddot{x}_0 \quad (28)$$

where

$$\ddot{x}_0 = \left[I - \dot{x}_0 \dot{x}_0^\top \right] \frac{(g + qx_0)}{(\varphi_0 - c)} \quad \text{and} \quad \omega = \|\ddot{x}_0\|. \quad (29)$$

If ω and thus \ddot{x}_0 vanish because \dot{x}_0 is colinear with the gradient $g + qx_0$, then the circle degenerates to the straight line $x(t) = x_0 + t\dot{x}_0$.

Moreover the function value along the trajectory is given by

$$\varphi(x(t)) = \varphi_0 + \left[(g + qx_0)^\top \dot{x}_0 \right] \frac{\sin(\omega t)}{\omega} + \left[q - \omega^2(\varphi_0 - c) \right] \frac{(1 - \cos(\omega t))}{\omega^2}, \quad (30)$$

In the straightline case $\omega = 0$ the trigonometric coefficients on the RHS reduce to t and $\frac{1}{2}t^2$

Proof. Assuming $\omega \neq 0$ we get differentiating $x(t)$ twice

$$\dot{x}(t) = \cos(\omega t) \dot{x}_0 + \frac{1}{\omega} \sin(\omega t) \ddot{x}_0 \quad \text{and} \quad \ddot{x}(t) = \cos(\omega t) \ddot{x}_0 - \omega \sin(\omega t) \dot{x}_0, \quad (31)$$

which obviously ensures that the initial conditions $\dot{x}(0) = \dot{x}_0$, $\ddot{x}(0) = \ddot{x}_0$ hold and that $\|\dot{x}(t)\| = 1$, $\|\ddot{x}(t)\| = \omega$ as well as $\dot{x}(t)^\top \ddot{x}(t) = 0$ for all $t \geq 0$. When $\ddot{x}_0 = 0$ the trigonometric quotients reduce continuously to t and 0 , respectively as claimed so that $\dot{x}(t) = \dot{x}_0$ and $\ddot{x}(t) = \ddot{x}_0 = 0$. To get the function value we apply Taylor's theorem for the quadratic function $\varphi(x)$ with Hessian

qI so that

$$\begin{aligned}
\varphi(x(t)) &= \varphi_0 + \nabla\varphi(x_0)^\top(x(t)x_0) + \frac{1}{2}\|x(t)x_0\|^2 \\
&= \varphi_0 + (g + qx_0)^\top \left[\frac{\sin(\omega t)}{\omega} \dot{x}_0 + \frac{(1 - \cos(\omega t))}{\omega^2} \ddot{x}_0 \right] \\
&+ \frac{1}{2} \left[\left(\frac{\sin(\omega t)}{\omega} \right)^2 + \left(\frac{\cos(\omega t)}{\omega} \right)^2 - 2 \frac{\cos(\omega t)}{\omega^2} + \frac{1}{\omega^2} \right] \\
&= \varphi_0 + (g + qx_0)^\top \dot{x}_0 \frac{\sin(\omega t)}{\omega} + \left[(g + qx_0)^\top \ddot{x}_0 + q \right] \frac{(1 - \cos(\omega t))}{\omega^2}.
\end{aligned}$$

This implies the asserted equality since by definition of \ddot{x}_0

$$\begin{aligned}
(g + qx_0)^\top \ddot{x}_0 &= -(g + qx_0)^\top \left[I - \dot{x}_0 \dot{x}_0^\top \right] \frac{(g + qx_0)}{(\varphi_0 - c)} \\
&= -(g + qx_0)^\top \left[I - \dot{x}_0 \dot{x}_0^\top \right] \left[I - \dot{x}_0 \dot{x}_0^\top \right] \frac{(g + qx_0)}{(\varphi_0 - c)} = -\omega^2(\varphi_0 - c).
\end{aligned} \tag{32}$$

The remainder of the proof consists in substituting the expressions for $x(t)$, $\dot{x}(t)$ and $\varphi(x(t))$ into the right hand side of

$$a(t) \equiv - \left[I - \dot{x}(t) \dot{x}(t)^\top \right] [g + qx(t)] / [\varphi(x(t)) - c] \tag{33}$$

and checking that the resulting acceleration $a(t)$ is indeed identical to $\ddot{x}(t)$ as given in (31).

Firstly we note for the numerator on the right hand side that

$$(g + qx_0) = (\varphi_0 - c) [\dot{x}_0 \dot{x}_0^\top (g + qx_0) - \ddot{x}_0] \in \text{span}(\dot{x}_0, \ddot{x}_0)$$

and hence

$$g + qx(t) = (g + qx_0) + q \frac{\sin(\omega t)}{\omega} \dot{x}_0 + q \frac{(1 - \cos(\omega t))}{\omega^2} \ddot{x}_0 \in \text{span}(\dot{x}_0, \ddot{x}_0) \tag{34}$$

Thus we see from (34) that the vector $a(t)$ like $\dot{x}(t)$ and $\ddot{x}(t)$ is always an element of the two dimensional subspace spanned by \dot{x}_0 and \ddot{x}_0 . For every $t \geq 0$ the pair $(\dot{x}(t), \ddot{x}(t)/\omega)$ forms an orthogonal basis of this plane so that $a(t) = \ddot{x}(t)$ is equivalent to $\dot{x}(t)^\top \ddot{x}(t) = 0 = \dot{x}(t)^\top a(t)$ and $\ddot{x}(t)^\top \ddot{x}(t)/\omega = \omega = \ddot{x}(t)^\top a(t)/\omega$. All but the very last equality are obvious. To show it we

compute using again (32) to eliminate \ddot{x}_0

$$\begin{aligned}
& \ddot{x}(t)^\top (g + qx(t)) \\
&= [\cos(\omega t)\ddot{x}_0\omega \sin(\omega t)\dot{x}_0]^\top \left[(g + qx_0) + \frac{g}{\omega}\sin(\omega t)\dot{x}_0 + \frac{g}{\omega^2}(1 - \cos(\omega t))\ddot{x}_0 \right] \\
&= -\omega(g + qx_0)^\top \dot{x}_0 \sin(\omega t) + (g + qx_0)^\top \ddot{x}_0 \cos(\omega t) - q \cos(\omega t)(1 - \cos(\omega t)) - q \sin^2(\omega t) \\
&= -\omega(g + qx_0)^\top \dot{x}_0 \sin(\omega t)\omega^2(\varphi_0 - c)(\cos(\omega t) - 1)\omega^2(\varphi_0 c) - q(1 - \cos(\omega t)) \\
&= -(\varphi(x(t)) - c)\omega^2
\end{aligned}$$

where we have used the expression (37) for the function value. Thus we get finally

$$\ddot{x}(t)^\top a(t)/\omega = -\ddot{x}(t)^\top (g + qx(t))/[\omega(\varphi(x(t)) - c)] = \omega$$

which completes the proof. \square

When the function (26) is valid on all of \mathbb{R}^n we get a full circle except in the special case where it degenerates to a straight line due to \dot{x}_0 pointing exactly along or against the the negative gradient $-\nabla\varphi(x_0)$. The last radial escape possibility is the only case where the trajectory runs off to infinity, whether or not the target can actually be obtained. Provided $q > 0$ the target level set is always the ball

$$T \equiv \left\{ x \in \mathbb{R}^n : \|x + g/q\|^2 \leq 2q(c - b)/q + \|g/q\|^2 \right\} \quad (35)$$

which may of course be empty. The unconstrained minimizer of the prox-linear function is then given by

$$x_* = -g/q \quad \text{with} \quad \varphi(x_*) = \varphi_0 - \frac{1}{2q}\|g + qx_0\|^2 = \mu - \frac{1}{2q}\|g\|^2. \quad (36)$$

when $q = 0$ so that φ is in fact linear the ball becomes the hemisphere

$$T \equiv \left\{ x \in \mathbb{R}^n : g^\top x \leq c - \mu \right\}$$

which is always nonempty if $g \neq 0$. In the rather uninteresting degenerate case $g = 0$ the target level set is empty if $c < b$ and equal to the whole of \mathbb{R}^n otherwise. Whenever T is nonempty and we do not have radial escape as defined above, all other trajectory reach the target, i.e.

the boundary ∂T in a perpendicular fashion. Strictly speaking the ODE (27) has a singularity where $\varphi(x) = c$ so that the solution can be extended in a continuous fashion into the interior of the target level in several ways. One of the infinitely many ways is the continuation of the circle, another one is the transition to steepest descent $\dot{x} = -\nabla\varphi(x)/\|\nabla\varphi(x)\|$, which would in this special case simply follow a straight line to the global minimizer $x_* = -g/q$ which is the center of the target ball. When exactly $\varphi(x_*) = c$ all circular trajectories contain the global minimizers directly, so that no continuation is necessary. In the case of a hemisphere with $r'_0 > 0$ the straight line would run off to infinity with φ becoming arbitrarily negative. These observations are of course consistent with the Propositions 1 and 2 in the previous two sections.

If T is empty because $c < \varphi(x_*)$ the trajectory will be a complete circle with the undesirable global minimizer in its interior, except in the highly unlikely radial escape case. We will have to design heuristic measures to detect this unproductive circling and reduce the target to a more realistic level. Whenever T is nonempty and we do not have radial escape there exists a value t_0 that solves the trigonometric equation

$$c = \varphi_0 + \left[(g + qx_0)^\top \dot{x}_0 \right] \frac{\sin(\omega t_0)}{\omega} + \left[q - \omega^2(\varphi_0 - c) \right] \frac{(1 - \cos(\omega t_0))}{\omega^2} \quad (37)$$

if $\omega \neq 0$ and otherwise the simple quadratic

$$c = \left[(g + qx_0)^\top \dot{x}_0 \right] t_0 + \frac{q}{2} t_0^2.$$

Most of the time the point $x(t_0)$ will lie outside the current polyhedron as defined in the next section and we can use t_0 as an upper bound in searching for the exit point in the next subsection. If either equation from has no positive solution we set $t_0 = \infty$.

7 The Generalized Abs-Normal and Abs-Linear Forms

All objective functions in machine learning and other applications are evaluated by a sequence of smooth intrinsic functions, arithmetic operations, and the nonsmooth element abs, which can be used to represent min and max as well. Such an evaluation procedure can be formally

interpreted as the generalized abs-normal form

$$\text{Min} f(x, z) \quad \text{s.t.} \quad z = F(x, z, h) \quad \text{and} \quad h = |z| \quad (38)$$

where

$$f : \mathbb{R}^{n+s+s} \mapsto \mathbb{R} \quad \text{and} \quad F : \mathbb{R}^{n+s+s} \mapsto \mathbb{R}^s .$$

Here we must require that the matrices

$$M \equiv \frac{\partial F}{\partial z} \in \mathbb{R}^{s \times s} \quad \text{and} \quad L \equiv \frac{\partial F}{\partial h} \in \mathbb{R}^{s \times s}$$

are strictly lower triangular. That means we can compute for any x the piecewise smooth functions $z(x)$, which finally yields the objective

$$\varphi(x) = f(x, z(x)) : \mathbb{R}^n \mapsto \mathbb{R} .$$

For simplicity we have assumed here that $h(x) = |z(x)|$ does not occur explicitly in the response function f so that all nonsmoothness is incorporated in what we will sometimes call the state equation $z = F(x, z, |z|)$. On the other hand we have a slight generalization of the usual abs-normal form [18] where z does not occur directly as arguments of f and F . As a motivation for allowing z itself to occur as an argument of F and f , it was noted in [13] that for repeated applications of the maximum to multi-component vectors using the nonsymmetric form

$$\max(u, v) = u + \frac{1}{2}[z + |(|z)|] \quad \text{with} \quad z = v - u \quad (39)$$

generates matrices L and M that are quite sparse. In contrast the repeated application of the symmetric form

$$\max(u, v) = \frac{1}{2}[u + v + |(|z)|] \quad \text{with} \quad z = v - u \quad (40)$$

tends to fill in the matrix L in the standard form where $M \equiv 0$.

Given code for evaluating (38) every AD tool will be able to compute for given x and

z the vectors y and the partitioned Jacobian

$$\frac{\partial[F-z, f]}{\partial[x, z, h]} \equiv \begin{bmatrix} Z & M-I & L \\ a^\top & b^\top & 0 \end{bmatrix} \in \mathbb{R}^{(s+1) \times (n+s+s)}.$$

Given the generalized abs-linear form evaluated at some consistent point (\hat{x}, \hat{z}) we get the piecewise linear model

$$\begin{aligned} z &= \hat{x} + Z(x - \hat{x}) + M(z - \hat{z}) + L(|z| - |\hat{z}|) \\ &= \underbrace{(\hat{x} - Z\hat{x} - M\hat{z} - L|\hat{z}|)}_{=c} + Zx + Mz + L|z|, \end{aligned} \quad (41)$$

$$\begin{aligned} y &= \hat{y} + a^\top(x - \hat{x}) + b^\top(z - \hat{z}) \\ &= \underbrace{(\hat{y} - a^\top\hat{x} - b^\top\hat{z})}_{=\mu} + a^\top x + b^\top z. \end{aligned} \quad (42)$$

where the constant shift μ can be set to zero without loss of generality as we are doing optimization. Note that we can unambiguously evaluate the piecewise linear functions $z(x)$ and $y(x)$ for any $x \in \mathbb{R}^n$ by forward substitution using the triangular structure of L and M .

The fundamental property on which our successive piecewise linearization approach is based is the generalized Taylor approximation

$$|\varphi(x) - \varphi(\hat{x}) - \Delta\varphi(\hat{x}; x - \hat{x})| \leq \frac{\gamma}{2} \|x - \hat{x}\|^2$$

where $\Delta\varphi(\hat{x}; x - \hat{x}) \equiv y(x) - \mu$ with y the piecewise linear approximation defined above. We will generally refer to a quadratic regularization term in the Euclidean norm $\|\cdot\| = \|\cdot\|_2$ as the proximal term and label functions including this term accordingly.

The key advantage of the piecewise linearization is that it allows us to deal with the combinatorial aspect of nonsmoothness more or less explicitly. More specifically, the full domain \mathbb{R}^n is decomposed into polyhedra, which can be identified by the signature vector and matrix

$$\sigma = \sigma(x) \equiv \text{sgn}(z(x)) \in \{-1, 0, +1\}^s \quad \text{and} \quad \Sigma \equiv \Sigma(x) = \text{diag}(\sigma(x)) \in \mathbb{R}^{s \times s}$$

as a function of the piecewise linear $z(x)$. The inverse images

$$P_\sigma \equiv \{x \in \mathbb{R}^n : \sigma(x) = \sigma\}$$

are pairwise disjoint, relatively open polyhedra. Using the partial order of the signatures given by

$$\tilde{\sigma} < \sigma \iff \tilde{\sigma}_i \sigma_i \leq \sigma_i^2 \quad \text{for } i = 1 \dots s,$$

we can define the essential closures

$$\bar{P}_\sigma \equiv \{x \in \mathbb{R}^n : \sigma(x) < \sigma\},$$

which are no longer disjoint and whose inclusion ordering corresponds exactly to the partial ordering $<$ of the signatures such that

$$\bar{P}_\sigma \subset \bar{P}_{\tilde{\sigma}} \iff \sigma < \tilde{\sigma}.$$

Hence, we see that $\hat{x} = 0$ with $\hat{\sigma} = \sigma(\hat{x})$ belongs exactly to the essential closures \bar{P}_σ for which $\sigma > \hat{\sigma}$. Consequently, we find for some open ball $B_\rho(\hat{x})$

$$B_\rho(\hat{x}) = \left\{ \bigcup_{\sigma > \hat{\sigma}} P_\sigma \right\} \cap B_\rho(\hat{x}).$$

Here, the σ on the right hand side can be restricted to be definite, i.e., only have nonzero components $\sigma_i = \pm 1$, which will be denoted by $0 \notin \sigma$. Within each \bar{P}_σ , we have $|z| = \Sigma z$ so that one can solve the equality constraint Eq. (41) for z to obtain the affine function

$$z(x) = (I - L\Sigma)^{-1}(c + Zx) \quad \text{for } x \in \bar{P}_\sigma.$$

Note here that due to the strict lower triangularity of L the unit lower triangular matrix $(I - L\Sigma)^{-1}$ is for any σ well defined and its elements are polynomial in the entries of L . For definite signatures $\sigma \neq 0$ the elements $x \in \bar{P}_\sigma$ are exactly characterized as solutions of the system of inequalities

$$\Sigma(I - L\Sigma)^{-1}(c + Zx) = (\Sigma - L)^{-1}(c + Zx) \geq 0.$$

If there is an $x \in \bar{P}_\sigma$ with definite signature $\sigma(x) \neq 0$ then the polyhedron \bar{P}_σ has a nonempty interior. The converse needs not be true in the presence of degeneracy. From duality theory it is known that either: \bar{P}_σ has a nonempty interior (in which case we call it full-dimensional), or the rows of $(\Sigma - L)^{-1}Z$ have a vanishing convex combination such that

$$\lambda^\top (\Sigma L)^{-1} Z = 0 \quad \text{with} \quad 0 \leq \lambda \neq 0.$$

Obviously this can be checked by standard Linear Optimization techniques. If $\dim(\bar{P}_\sigma) = n$ we have by (42) the function value

$$y_\sigma = \mu + a^\top x + b^\top \Sigma (IL\Sigma)^{-1} (c + Zx), \quad (43)$$

and correspondingly the gradient

$$g_\sigma = a^\top + b^\top \Sigma (IL\Sigma)^{-1} Z = a^\top + b^\top (\Sigma L)^{-1} Z, \quad (44)$$

where the last equality relies on definiteness, i.e., $0 \notin \sigma$, so that $\det(\Sigma) \pm 1$. This explicit gradient expression is crucial for our numerical integration procedure.

8 Piecewise Numerical Integration

In [24] an explicit integrator with a third order local truncation was suggested and tested. The results were not entirely convincing. Of course we are not really interested in the solution trajectory itself but only where it leads us to in the medium run. However, especially when going uphill it is important that the integration is sufficiently exact such that the trajectory does not turn back prematurely. The ability to climb uphill occasionally is essential for any nonlocal search procedure. Moreover, we have to deal with kinks in the function φ and thus jumps in its gradient. So rather than following the ODI on the original φ we perform the savvy ball method on its piecewise linearization, where the trajectory can be expressed as a sequence of circle segments.

An objective of the form (26) is not of practical interest, since the global minimizer x_* can be computed directly without any significant effort. Instead we are interested in the case

of prox-abs-linear forms as an assumed upper bounding approximation of a piecewise smooth function in abs-normal form. If the starting point x_0 lies in the interior of a polyhedron and one chooses any c and x_0 , the resulting circle will usually leave the polyhedron unless it reaches the level set or stays inside \bar{P}_σ .

In the smooth case like for example on MNIST with a single layer neural model, the maximal step t_* will always be given by the root t_0 of the equation (37) if that exists. That value depends very strongly on the chosen q and generally may be too large. We have therefore introduced an additional restriction, namely that the change in the direction \dot{x} may not exceed a certain angle α so that always $\omega t_* \leq \alpha < 2\pi$. In our calculations we typically set $\alpha \sim 2\pi/30$ which corresponds to 12 degrees. Also in the smooth case one can reset the reference point \dot{x} to the current point after each step because the piecewise linearization just amounts to linearization, i.e. normal differentiation. In the nonsmooth case that is not feasible because one might repeatedly run into the same gradient discontinuity.

Note that all computations in the Savvy Ball method are of order $(n+s)^2$, whereas the local minimizer ALMIN involves matrix factorizations with a computational cost of order $(n+s)^3$. This is even more true for the application of mixed integer linear optimization packages like GUROBI, which can be adopted to minimizing (42) subject to (41) since the constraints can be written using bilinear products of real and binary variables [25]. Like SALMIN Savvy Ball, is on the other hand dependent on the ℓ_2 norm and thus strongly affected by linear variable transformations or diagonal rescalings. Fortunately, the free parameters in an abs-linear prediction model and in particular a neural network seem to have naturally a quite homogeneous scaling.

9 Experimental Results

9.1 Classification Problem: The MNIST Dataset for Digit Recognizer

In this paper, the Savvy Ball Method has been tested on the MNIST dataset which is a classic source to test classification algorithms. The MNIST digit database is the most used sample for supervised learning task such as character recognition and pattern recognition. It consists of 70000 images of digits from 0 to 9. All these black and white digit images are size normalized,

and centered with fixed dimensions of 28×28 pixels as described in 1. The dataset is split into the classifier training sample, that contains 60000 images, and the remainder belonging to the classifier testing sample.

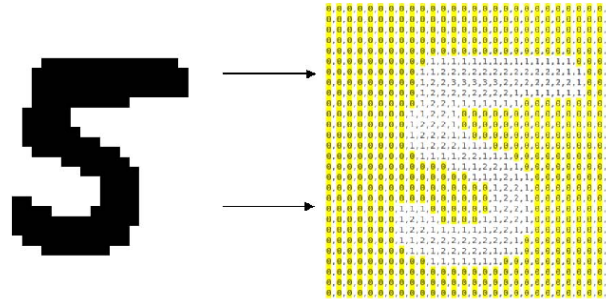
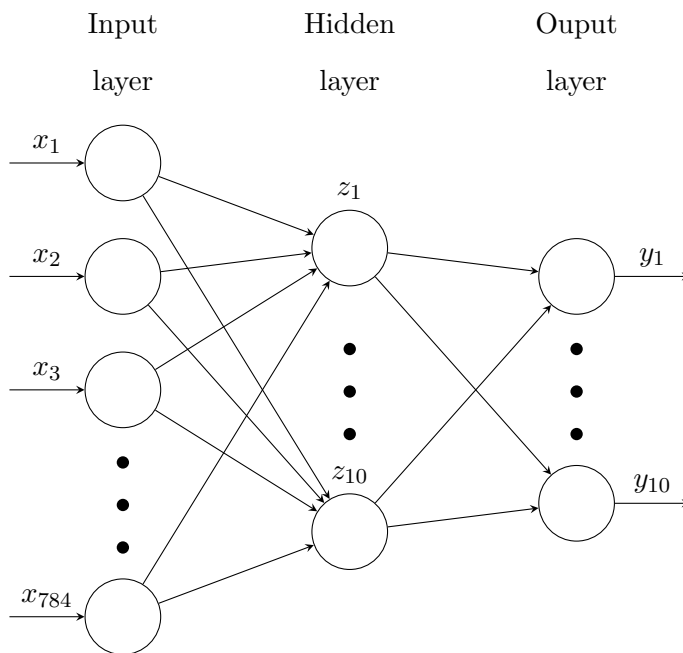


Figure 1: Here we can observe how the pattern recognition task works using MNIST samples. That is a real-valued square matrix $M_{28 \times 28}$ [26]

For each digit image the feature value $x \in X = [0, 1]^{784} \subset \mathbb{R}^{784}$ describes the pixel value associated to the grayscale image with a range of 0 to 255 or 0 to 65535 depending on 8-bit or 16-bit data value. Moreover, the correspondant label $y \in Y$ that represent the itself digits $Y = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. We consider the MNIST sample divided into two sets corresponding to each learning stage: the training dataset D_{60000} and the testing dataset D_{10000} .

We consider an abs-linear prediction function for single-layer case using the Savvy Ball method. Hence, the Neural network model consider for the numerics has the following structure



where the hidden layer is computed by $z = Wx + b$ and the output layer is given by $y = \text{smax}(z)$.

Now, for the classification task we used the loss function given by

$$\varphi(W, b) = \sum_{k=1}^{60000} -\log(\text{smax}(Wx_k + b)_{l_k})$$

where l_k makes reference to the k^{th} label and smax is the softmax activation function, which is given by

$$y_j = \text{smax}(z)_j = \frac{\exp(z_j - z_{max})}{\sum_{i=1}^{10} \exp(z_i - z_{max})}$$

where $z_{max} \approx \max_{1 \leq i \leq 10} z_i$. Here z_{max} is only introduced to avoid numerical overflow in the exponential evaluations and can be set to zero mathematically or any number similar to z_{max} .

We can check that the gradient of the loss function with respect to z is given by

$$\begin{aligned} \frac{\partial}{\partial z_j}(-\log y_\ell) &= \frac{\partial}{\partial z_j}(-\log(\text{smax}(z)_\ell)) \\ &= \frac{\partial}{\partial z_j}(-z_\ell + \log(\text{den})) \\ &= \begin{cases} -1 + y_\ell & \text{if } \ell = j \\ y_\ell & \text{if } \ell \neq j \end{cases} \end{aligned}$$

$$\implies \nabla_z(-\log y_\ell) = y - e_\ell \text{ where } e_\ell \text{ is the } \ell - \text{th unit vector.}$$

Applying the chain rule we can then compute the derivative of the k -th loss with $\ell = l_k$ as

$$\nabla_b(-\log(\text{smax}(Wx_k + b)_\ell)) = \bar{y}_k - e_\ell \quad \text{and} \quad \nabla_W(-\log(\text{smax}(Wx_k + b)_\ell)) = (\bar{y}_k - e_\ell)x_k^\top$$

where $\bar{y}_k = Wx_k + b$. Since the \bar{y}_k are already available from the forward evaluation of the loss values one only has to do 60000 rank-one updates of the derivative with respect to $[W, b]$. This requires about the same effort as the computing the 60000 matrix vector products $Wx_k + b$ so that the empirical risk and its gradient are about twice as expensive as computing the risk function by itself. This complexity ratio is consistent with the theory of the reverse mode of automatic or algorithmic differentiation.

As we can see the tasks of evaluating for each sample point the loss contribution and

the gradient contribution are completely independent except for the additive accumulation of these contributions. Thus we have 60000 parallel tasks whose results can be combined by a so-called reduce operation. This structure could best be exploited by an MPI implementation, but because we have a shared memory situation it is more convenient to utilize OpenMP. At the time of writing we cannot report conclusive results on the size of the speed-up that can be obtained.

An even better speed-up can be expected if one utilizes the fact that all 60000 feature vectors x_k are multiplied by the same matrix W so that we have in fact the product of a 10×785 matrix with a 785×600000 matrix. For such numerical linear algebra operations very effective Basic Linear Algebra Subprograms (BLAS) have been developed and implemented on all major computing platforms. The BLAS interface are provided for the three levels

- **Level 1:** Encompasses linear algebra subroutines for operations depending only on vectors (e.g. $y = \alpha x + y$).
- **Level 2:** Consists of linear algebra functions for computing expressions of the form matrix-vector multiplication (e.g. $y = \alpha Ax + \beta y$).
- **Level 3:** A set of functions for computing operations like matrix-matrix multiplication (e.g. $Z = \alpha WX + \beta Z$).

We have obtained run-time reductions of some 40% using the BLAS 2 Routine for the individual products $Wx_k + b$. Unfortunately, our efforts to implement BLAS 3 were not yet successful.

9.2 Numerical Results

For the single layer model learning on MNIST it is well known that steepest descent and stochastic gradient can reach an accuracy of some 91%. This is achieved essentially in one monotonic descent sweep. Applying the Savvy algorithm we can do a little bit better, namely reaching 92% accuracy. This means that after training the model on the training set it yields fails only on 8% of the test samples to classify the digits properly. As one can see in the figure 2 after a more or less monotonic descent over some 400 steps, there is an extended search that involves periods of uphill searches. That is essential for global optimization. It is also typical that after the optimal result is obtained, further efforts to lower the risk value are made but of course not successful. Eventually the calculation has to be terminated when the computing resources or the users patience have run out.

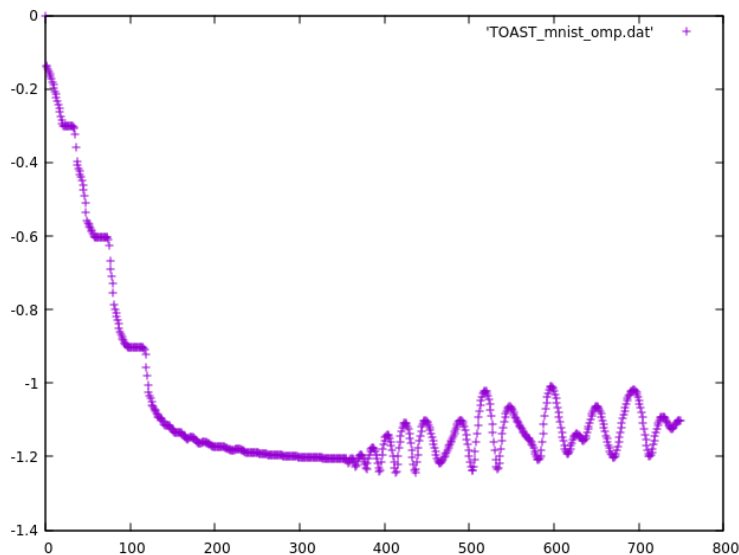


Figure 2: The figure shows how an OMP implementation of the Savvy Ball method (TOAST) speed up the convergence for less number of iterations for the same MNIST dataset.

The nearly horizontal stretches in the figure indicate periods where the current target value has been nearly obtained. It is then reduced to a half, which leads to further reduction possibly after a transitional period of growth. The halving strategy works reasonably well as we know that the globally minimal value should be close to zero. So far other strategies for adjusting the target have lead to similar results and reducing the sensitivity e below 1 also did not make a big difference. The parallelized version should of course generate the same convergence curve but using less wall clock time.

10 Summary and Conclusion

From the point of view of Non-smooth Analysis, we approached the study of the single-layer ANN case as a globally minimization problem for piecewise smooth objective functions. We considered a supervised learning task where the ANN looks for a predictor to fit the training sample and then to predict data based on the parameters adjustment. Here the value to minimize is the empirical risk.

For the non-smooth analysis, we consider the Clarke's generalized gradient which is well-defined from the assumption that the objective function $\varphi(x)$ is Lipschitz continuous. Then the steepest descent differential inclusion given by (2) has at least one absolutely continuous solution trajectory by Filippov's theory. In terms of global optimization, the motion modeled in (3) guarantees an optimal convergence rate, but its behavior is exactly the opposite of what one wants. Therefore, we developed a method based on the Polyak's heavy ball method associated to equation (3) which we called Savvy Ball method. This method is given by the equation (4) which satisfies the conditions of the Filippov Theorem as shown in Section 2. The attribute *savvy ball* suggests that its behavior is controlled by the *target value* and the *sensitivity parameter*.

Moreover, in Section 3, we show up the background needed to avoid the optimizer's house of horror observed in [5]. Here, we remarked the fact that the definition of Clarke's generalized derivative is equivalent to the concept of sub-gradient and Hadamard derivative when φ is Lipschitz continuous and convex. In section 4, we provide a detailed analysis to the convex case where we bring conditions to guarantee a convergence rate. Here the curvature $x''(t)$ may tend to infinity where we could not find an example where this scenario holds, so the question remains to be explored. We did not address the question of what happens if the target set is empty and unreachable. In such situation, we could either gradually increase the target value or reduce the sensitivity parameter, but this cannot happen when φ is convex.

Since for a global optimization method getting away from undesirable stationary points or local minimizers is crucial, we consider the nonconvex case with homogeneous center points in Section 5. In section 6, we consider the task of computing the savvy ball trajectory to the sum of an affine function and a proximal term in the Euclidean norm where it is both convex and homogeneous. This result is an alternative proof which was presented by [5] where they show that the initial value problem for the ODE given by (27) has a solution given by the circle

(28). In section 7, we consider the abs-normal and abs-linear form where we observed that the piecewise linearization allow us to deal with the combinatorial aspect of nonsmoothness through the decomposition of the domain into polyhedra.

10.1 Conclusions

From the numerical simulations, we appreciate that the code performed serially reaches a local minima with less than 800 iterations where the CPU run-time was 2808.4348 seconds. We tried to implement a parallel code but we noticed that parallelization is not a so easy task. However, we was able to code a parallel region using Open MP which shows up an optimized performance of 0.69379 % for 17 iterations. This was the number of iterations that holds an optimization for the CPU run-time, since we observed a decay in the way the paralleled region was setted up. It means that probably the time used for the communication between the threads is more than the processing time. We suggest the use of another libraries such as Open MPI where the directions for passing information and synchronization between threads is given explicitly. This hopefully will speed up the performance. Moreover, it remains to be investigated from which conditions the Savvy ball trajectories are unique in the convex case.

References

- [1] S.-A. N. Alexandropoulos, P. M. Pardalos, and M. N. Vrahatis, “Dynamic search trajectory methods for global optimization,” *Annals of Mathematics and Artificial Intelligence*, vol. 88, pp. 3 – 37, 2019.
- [2] A. Griewank, “Generalized descent of global optimization,” *Journal of Optimization Theory and Applications*, vol. 34, pp. 11–39, 05 1981.
- [3] J. A. Snyman, “A new and dynamic method for unconstrained minimization,” 1982.
- [4] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” 1964.
- [5] A. Rojas and A. Griewank, “A target oriented averaging search trajectory and its application to artificial neural network training,” 2019.
- [6] A. Clifford, *Artificial Intelligence: An Illustrated History: From Medieval Robots to Neural Networks*. Sterling, 2019.
- [7] Y. G. Petalas, D. K. Tasoulis, and M. N. Vrahatis, “Dynamic search trajectory methods for neural network training,” in *ICAISC*, 2004.
- [8] A. Griewank and A. Rojas, “Treating artificial neural net training as a nonsmooth global optimization problem,” in *LOD*, 2019.
- [9] F. Clarke, *Optimization and Nonsmooth Analysis*. SIAM, 1990.
- [10] A. F. Filippov, “Differential equations with discontinuous righthand sides,” in *Mathematics and Its Applications*, 1988.
- [11] A. F. Filippov, “Classical solutions of differential equations with multi-valued right-hand side,” 1967.
- [12] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms I*. Springer, 1993.
- [13] A. Griewank and A. Walther, “Finite convergence of an active signature method to local minima of piecewise linear functions,” *Optimization Methods and Software*, pp. 1–21, 12 2018.

- [14] R. A and A. S. B, “Introduction to optimization,” in *Optimization Software, Publications Division*, 1987.
- [15] J. A. Snyman and L. P. Fatti, “A multi-start global minimization algorithm with dynamic search trajectories,” *Journal of Optimization Theory and Applications*, vol. 54, pp. 121–141, 1987.
- [16] J. Snyman and S. Kok, “A reassessment of the snyman–fatti dynamic search trajectory method for unconstrained global optimization,” *Journal of Global Optimization*, vol. 43, pp. 67–82, 03 2009.
- [17] L. M. Rios and N. V. Sahinidis, “Derivative-free optimization: a review of algorithms and comparison of software implementations,” *Journal of Global Optimization*, vol. 56, pp. 1247–1293, 2013.
- [18] A. Griewank, “On stable piecewise linearization and generalized algorithmic differentiation,” *Optimization Methods and Software*, vol. 28, no. 6, pp. 1139–1178, 2013.
- [19] A. Griewank, “On stable piecewise linearization and generalized algorithmic differentiation,” *Optimization Methods and Software*, vol. 28, pp. 1139–1178, 2013.
- [20] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Wiley, 1983.
- [21] P. M. Floudas, Christodoulos A. ; Pardalos, *Encyclopedia of Optimization*. Springer Science Business Media, 2001.
- [22] S. Arora, N. Cohen, N. Golowich, and W. Hu, “A convergence analysis of gradient descent for deep linear neural networks,” *CoRR*, vol. abs/1810.02281, 2018.
- [23] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *SIAM Review*, vol. 60, pp. 223–311, 2018.
- [24] A. Griewank, “Generalized descent for global optimization,” *Journal of Optimization Theory and Applications*, vol. 34, pp. 11–39, 1981.
- [25] T. Achterberg, “Products of Variables in Mixed Integer Programming.” Gurobi Webinar: [https://gurobi.com > uploads > 2019/07](https://gurobi.com/uploads/2019/07), 2019.

- [26] A. J. Patel and T. V. Kalyani, “Support vector machine with inverse fringe as feature for mnist dataset,” *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, pp. 123–126, 2016.

A Savvy Ball method for the MNIST character recognition problem: Code

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 #include <time.h>
5 #include "mnist.h"
6
7 #define square(x) ((x)*(x))
8 #define sgn(x) (x>0? 1 :-1) // sign function
9 #define act(x) (x>0? x : 0) //activation function
10 #define actp(x) (x>0? 1 : 0) //activation derivative
11 #define max(x,y) (y>x? y: x)
12 #define min(x,y) (y<x? y: x)
13 #define firstsign(x,y,z) ((x !=0) ? sgn(x) : ((y != 0) ? sgn(y): sgn(z)))
14 FILE *fptre;
15 FILE *fptrt;
16 int m,mt,n,n1,d, d1;
17 int sw;
18 double targetred;
19 double one = 1;
20 double* x;
21 double* p;
22 double** X;
23 double** Z;
24 double** dZ;
25 int* Y;
26 double** W;
27 double** Wmin;
```

```
28 double** Wini;
29 double** dW;
30 double** ddW;
31 double** bW;
32 double* z;
33 double* bz;
34 double** Z;
35 double** M;
36 double** L;
37 double* a;
38 double* c;
39 double q; // for the benefit of CGS
40 double omega;
41 double e;
42 int l1=0;
43 int mnist=1;
44 double iprod;
45 double tnorm;
46 double gnorm;
47
48
49
50 void resetzero(int l, double* v)
51 { for(int i = 0; i < l; i++) v[i] = 0.0; }
52
53
54 int randint = 3;
55 int prime = 524287;
56 double myrandnumb() // between 0 and 1
57 { randint = (randint*7)%prime;
```

```

58 //     printf("\n randint %f", randint/(double)prime );
59     return randint/(double)prime;}
60
61 int setsample() // allocating memory of features,labels, layers and its de
62 { Y = (int*)malloc(m*sizeof(int)); // scalar label
63   X = (double**)malloc(m*sizeof(double*)); // feature vectors
64   Z = (double**)malloc(m*sizeof(double*)); // middle layer
65   dZ = (double**)malloc(m*sizeof(double*)); // middle layer derivative
66   *X = (double*)malloc(n1*m*sizeof(double));
67   *Z = (double*)malloc(d1*m*sizeof(double));
68   *dZ = (double*)malloc(d1*m*sizeof(double));
69   load_mnist();
70   for(int i = 0; i < m; i++ )
71   { X[i] = *X + i*n1;
72     Z[i] = *Z + i*d1;
73     dZ[i] = *dZ + i*d1;
74     for(int j=0; j<n; j++)X[i][j] = train_image[i][j];
75     X[i][n] = 1;
76     Y[i] = train_label[i];
77   };
78   return 0;
79 };
80
81 void printmatrix(int m, int n,double** A){
82   for (int i=0;i<m;i++)
83   {printf("\n row %i ",i);
84     for(int j=0;j<n;j++)
85       printf(" %f ", A[i][j]);
86   };
87 }

```

```
88
89 double dot(int l, double* a, double* b) //Inner product evaluation
90 { double val;
91     val = 0;
92     for(int i=0;i<l;i++) val += a[i]*b[i]; //parallelization could be needed
93     return val;
94 };
95
96 double norm(int l, double* a) //Inner product evaluation or euclidean norm
97 { double valmax=0;
98     for(int i=0;i<l;i++) valmax = max(valmax, fabs(a[i]));
99     if(valmax==0) return 0;
100    double val=0;
101    for(int i=0;i<l;i++) val += square(a[i]/valmax);
102    return sqrt(val)*valmax;
103 };
104
105 void softmax(int l, double* b, double* e) // b and e can coincide
106 { double sum = 0;
107     double maxi=0;
108     for(int i=0; i<l; i++) maxi=max(maxi, b[i]);
109     for(int i=0; i<l; i++) sum += exp(b[i]-maxi);
110     for(int i=0; i<l; i++) e[i]= exp(b[i]-maxi)/sum;
111 }
112
113
114
115 void saxpy(int l, double* a, double q, double* b) // sum of a vector and a scalar
116 { for(int i=0;i<l;i++) b[i] += q*a[i];
117 };
```

```
118
119 void scale(int l, double* a, double q, double* b)// scalar product
120 { for(int i=0;i<l;i++) b[i] = q*a[i];
121 };
122
123
124 int pred(double* x,double* z, int l) // prediction function, y last component
125 {   for (int i = 0; i < d; i++) z[i] = dot(n1,W[i],x);
126     softmax(d, z, z);
127     z[d] = -log(z[l]);
128     return 0;
129 };
130
131
132 int bpred(double* x, double* z, int l) //incremental reverse of pred
133 {   z[l]-=1;
134     for(int i=0; i < d; i++){
135         saxpy(n1, x, z[i], bW[i]); // parallel region must be implemented*
136     };
137     z[l]-=1;
138     return 0;
139 }
140
141 int b2pred(double* x, double* z, int l) //combination of pred and bpred
142 {   for (int i = 0; i < d; i++) z[i] = dot(n1,W[i],x); //has been integrated
143     softmax(d, z, z);
144     z[d] = -log(z[l]);
145     z[l]-=1;
146     for(int i=0; i < d; i++) saxpy(n1, x, z[i], bW[i]);
147     z[l]+=1;
```



```
148     return 0;
149 }
150
151 int emprisk(double* risk) //emp risk penalized by the proximal term
152 {   /**risk = 0.5*q*square(norm(n1*d,*W)); //proximal term
153     *risk=0;
154     for (int k = 0; k< m; k++ )    //depends globally on W
155     { pred(X[k],Z[k],Y[k]);    // global z serves as workspace
156         *risk += Z[k][d]; //cross entropy
157     };
158     return 0;
159 };
160
161 double accuracy() // counts how often we are wrong on the test set
162 {
163     int count = 0;
164     x[n] = 1;
165     for (int k = 0; k< mt; k++ )
166     { int l = test_label[k];
167         for(int i= 0;i< n ;i++)
168             x[i] = test_image[k][i];
169         pred(x,z,l);    // global z serves as workspace
170         int wrong = 0;
171         for(int i= 0;i < 10;i++)
172             if (z[i] > z[l]) wrong = 1 ; //label has not maximal probability
173         if(wrong) count++;
174     };
175     return count/((double)mt);
176 };
177
```

```

178 int bemprisk(double* risk) // empirical risk and gradient evaluation//////////
179 { *risk=0;
180     double scale = 0; // could always be zero mathematically
181     resetzero(n1*d, *bW);
182     for (int k = 0; k< m; k++ )
183     { int l = Y[k];
184         x = X[k];
185         double sum = 0;
186         for (int i = 0; i< d; i++)
187         { z[i] = exp(dot(n1,W[i],x)-scale);
188             sum += z[i];};
189         z[l] -= sum;
190         for(int i=0; i < d; i++)
191             {z[i] /= sum;
192                 saxpy(n1, x, z[i], bW[i]);};
193         *risk += -log(1+z[l]);
194         scale += max(log(sum),-scale*0.1); // presumably this helps with the
195     }
196     return 0;
197 };
198
199 // This is for the global method
200 void setcircle(double den) // compute the tangent and radial vector
201 {
202     // scale(d*n1,*dW,1.0/tnorm,*dW); // (re)normalize the tangent
203     // double gnorm = norm(d*n1,*bW);
204     if(den <=0) /// should never happen
205     { omega = 0;
206         resetzero(d*n1,*ddW); //
207         printf("\n now we are following straight lines");

```

```

208     scale(d*n1, *bW, -1/gnorm, *dW); // tangent = normalized steepest descent
209     tnorm = 1;
210 }
211 else
212 { tnorm = norm(d*n1,*dW); //norm of the tangent
213   iprod = dot(d*n1,*bW,*dW)/square(tnorm);
214   scale(n1*d,*dW,iprod/den,*ddW); // set second derivative to
215   saxpy(n1*d,*bW,-1/den,*ddW);
216 //     check = dot(n1*d, *ddW, *dW);
217   omega = norm(n1*d,*ddW); // normalize second derivative in Euclid
218   tnorm = norm(d*n1,*dW);
219   if(omega != 0) scale(n1*d,*ddW,1/omega,*ddW); //ddW is either zero or
220 };
221 };
222
223 double trigsolve(double omega, double zbar, double zhat, double ztil, double* tea)
224 {double sigma = firstsign(zbar, zhat, ztil); // possible here but does is di
225   double tea = *teast; double tau = tea*omega;
226   if (sigma < 0) { zbar *= sigma; zhat*= sigma; ztil*= sigma;}
227   double ztest = zbar - *teast*(fabs(zhat)-*teast*omega*min(0,ztil)/2);
228   if (ztest >= 0) return tea; // No change in upper bound since test failed
229   if(omega == 0){ // steepest descent case below target or by accident
230     if(zbar*zhat < 0) tea = -zbar/zhat; // second derivative term drops out
231     return tea;
232   }
233   else { zbar*= omega;} // ztil has like already has one omega in denominator
234   {double rho = sqrt(square(zhat)+square(ztil));
235     double zbplzt;
236     zbplzt = zbar+ztil;
237     if(fabs (zbplzt)<= rho)

```

```
238     {   double delta = atan2(zhat,ztil);
239         double tautil = acos(zbplzt/rho);
240         if ((- delta - tautil > 0) && (zbar > 0.00000001) ) tau = - delta
241         else if ( tautil - delta > 0.0000001 ) tau = tautil - delta;
242         else {tau = 2*M_PI - delta - tautil;}
243         double err = zbar+ zhat*sin(tau) + ztil*(1-cos(tau));
244         double slope = zhat*cos(tau) + ztil*sin(tau);
245         if(fabs(err) >= 0.00001) printf("\n omega %e root test tau %e res %e\n");
246     };
247     return tau/omega;
248 };
249 };
250
251 // This is for the global method
252
253
254 int main(){
255     double y, el, elt, eta;
256     n = SIZE;
257     n1 = n+1;
258     srand(2019);
259     printf("feature dimension: %i \n",n);
260     d = 10;
261     q=0;
262     e = 1;
263     d1 = d+1;
264     printf("layer size: %i \n",d);
265     m = NUM_TRAIN;
266     mt = NUM_TEST;
267     printf("sample size: %i \n",m);
```

```

268     int meth = 3;
269     printf("method: %i \n",meth);
270     sw = m*d1;
271     eta = 2*M_PI/60;
272     targetred = 0.5 ;
273     printf("angle bound: %f sensitivity %f targetred %f \n",eta, e, targetred);
274     int maxit =750;
275     printf("maxit: %i \n",maxit);
276     z = (double*)calloc(d,sizeof(double)); //intermediate layer
277     bz = (double*)calloc(d,sizeof(double)); //adjoints of intermediates
278     x = (double*)malloc(n1*sizeof(double)); //input data
279     //xb = (double*)calloc(n1,sizeof(double)); //input gradient
280     p = (double*)malloc(d*sizeof(double)); //fixed output weights
281     W = (double**)malloc(d*sizeof(double*)); //Weight matrix
282     Wmin = (double**)malloc(d*sizeof(double*)); //Weight matrix
283     Wini = (double**)malloc(d*sizeof(double*)); //Weight matrix
284     dW = (double**)malloc(d*sizeof(double*)); //Weight matrix tangent
285     ddW = (double**)malloc(d*sizeof(double*)); //Weight matrix curvature
286     bW = (double**)malloc(d*sizeof(double*)); //adjoined weight matrix
287     *W = (double*)calloc(d*n1,sizeof(double)); // continuous allocation vector
288     *Wini = (double*)calloc(d*n1,sizeof(double)); // continuous allocation vector
289     *dW = (double*)calloc(d*n1,sizeof(double)); // continuous allocation vector
290     *ddW = (double*)calloc(d*n1,sizeof(double)); // continuous allocation vector
291     *bW = (double*)calloc(d*n1,sizeof(double)); // contiguous allocation vector
292     *Wmin = (double*)calloc(d*n1,sizeof(double)); // continuous allocation vector
293
294     fptre = fopen("./eltdat.txt","w");
295     for( int i = 0; i < d; i++) // allocation of weights and
296     { W[i] = *W+n1*i;
297       Wmin[i] = *Wmin+n1*i;

```

```

298     Wini[i] = *Wini+n1*i;
299     dW[i] = *dW+n1*i;           // only needed for TOAST
300     ddW[i] = *ddW+n1*i;        //      "
301     bW[i] = *bW+n1*i;          //      "
302 };
303 for( int j = 0; j < n; j++) x[j] = (2*(double)rand())/RAND_MAX-1;
304     x[n] = 1; //allocation of single sample poin
305
306 // Verification of adjoints by devided differences
307
308     setsample(); //Initialize the training set of m samples with same random
309     for (int i = 0; i < n1*d; i++)
310         (*W)[i] = (*Wini)[i] = myrandnumb()- 0.5 ;
311     pred(x,z,1);
312     y =z[d]; //prediction function evaluation
313     printf("\n prediction value %f \n \n ",y); //print prediction value
314     bpred(x,z,1); //adjoint prediction function evaluation
315     x[1] += 0.01;
316     pred(x,z,1); //check x-derivatives against divided difference
317     printf("%f, %f xerror \n \n ",xb[1],(z[d]-y)/0.01);
318     x[1] -= 0.01;
319     W[1][0] += 0.01;
320     pred(x,z,1); //check W-derivative against devided differences
321     printf("%f, %f Werror \n \n ",bW[1][0],(z[d]-y)/0.01);
322     W[1][0] -= 0.01;
323     W[0][n] += 0.01;
324     pred(x,z,1); // check W-derivative against devided differences
325     printf("%12.7f, %12.7f berror \n \n ",bW[0][n],(z[d]-y)/0.01);
326     emprisk(&el);
327     double el0 = el;

```

```

328     printf(" lossvalue %f, \n",el);/// empirical risk
329     bemprisk(&el);
330     printf(" lossvalue using bemprisk %f, \n",el);
331     emprisk(&elt);    // check W-derivative against devided differences
332 // End checking against divided difference
333 // Begin learning
334     if(meth==3)// This is for TOAST
335     {
336         double mul1, mul2;
337         double taust = 0;
338         double teast = 0;
339         double target = 0 ;    // not active currently
340         double targetol = 0.00000001;
341         double t = 0; // length of trajectory
342         double elmin = 1/0.0;
343         int it, bestit = -1;
344         for(it =0;it < maxit; it++){
345             bemprisk(&el); // Compute the function value and gradients
346             gnorm = norm(n1*d, *bW);
347             if(it%10==0) printf("it %i, emprisk %12.4f , target %12.4f , gnorm
348 teastar %12.4f \n", it, el/m, target/m, gnorm/m, taust/eta, teast);
349             if(it%1==0) {fprintf(fptre,"%i, %f \n",it,log(min(1,el/el0))/log(1
350                 { elmin = el; bestit = it;
351                 for (int i = 0; i < n1*d; i++)
352                     (*Wmin)[i] = (*W)[i]; };
353             if(it == 0) // initialize tangent to perturbed negative gradient
354             { target = el0/2;
355                 for (int i = 0; i < d*n1 ; i++){*dW)[i] = - (*bW)[i]*(1 + 0.1*

```

```

//initialize tangent to perturb SD
356         tnorm = norm(n1*d,*dW);
357         //printf(" initial gradient size  %18.12f \n ", tnorm);
358     };
359     if(el<target +targetol){
360         target *= targetred;
361         printf("\n it %i target reached and reduced el %18.12e target %18.12e\n", it, el, target);
362         for (int i = 0; i < d*n1 ; i++) (*dW)[i] = - (*bW)[i]*(1 + 0.1*targetred);
//initialize tangent to perturb SD
363         tnorm = norm(n1*d,*dW);
364         if(target<0.0001) exit(50);
365     };
366     // Compute the circle
367     scale(d*n1,*dW,1.0/tnorm,*dW);
368     tnorm = 1;
369     e = 1.0;
370     setcircle((el-target)/e); //
371     // compute targetea
372     double zbar, zhat, ztil, targetea;
373     zbar = (el-target);
374     zhat = iprod;
375     ztil = 0.0;
376     double teastold = teast;
377     if(omega == 0)
378     {   printf("it %i, omega equal to zero \n", it);
379         targetea = fabs((target-el)/zhat);    // go to mimimizer which
380     }
381     else{
382         ztil = q/omega - zbar*omega;
383         teast = 2*M_PI/omega;    // computing targetea

```



```

384         targetea = trigsolve(omega, zbar, zhat, ztil, &teast);
385     };
386     teast = targetea;
387     if(it) teast = min(teast, teastold*3);
388     taust = min(teast*omega, eta);
389     if( taust == targetea*omega) printf(" Level set in reach %18.12f, ",
390     t += teast;
391     if(taust == 2*M_PI){
392         printf("Adjust c \n");
393         exit(1);
394     };
395     // Update the point and tangent
396     if(omega == 0){
397         saxpy(n1*d,*dW,teast,*W);
398     } // straight line
399     else{
400         saxpy(n1*d,*dW,sin(taust)/(omega*tnorm),*W);
401         saxpy(n1*d,*ddW,(1-cos(taust))/omega,*W);
402         scale(n1*d,*dW,cos(taust)/tnorm,*dW);
403         saxpy(n1*d,*ddW,sin(taust),*dW);
404         if(fabs(check0-1)+fabs(check1-1)+fabs(check2-1)> 0.000001)
405             %18.12f, check1 %18.12f, omega %18.12e, check2
406             %18.12f, teast %f, \n ", check0, check1,omega, check2, teast);
407         printf("\n it %5d , e10 %5f, e1 %5f, target %5f, targetol %8.8f ",it,
408             );
409     };
410     printf("\n maxit reached elmin %18.12f bestit %i trajectory %f target %f
411     \n ", elmin/m, bestit, t, target/m);

```

```
412     fclose(fptre);
413     W = Wmin;
414     for (int i = 0; i < d*n1 ; i++) (*Wini)[i] -= (*Wmin)[i];
415         double dist = norm(n1*d, *Wini);
416         printf("failure rate %12.4f distance %12.4f \n",accuracy(), dist);
417     return 14;
418 }
419 };
```