



**UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA
EXPERIMENTAL YACHAY**

Escuela de Ciencias Matemáticas y Computacionales

**TÍTULO: A Target Oriented Averaging Search Trajectory and its
Application in Artificial Neural Networks**

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

Autor:

Rojas Jiménez Ángel Adrián

Tutor:

Dr. Oscar Guillermo Chang Tortolero

Urququí, Septiembre 2019

Urququí, 23 de septiembre de 2019

SECRETARÍA GENERAL
(Vicerrectorado Académico/Cancillería)
ESCUELA DE CIENCIAS MATEMÁTICAS Y COMPUTACIONALES
CARRERA DE MATEMÁTICA
ACTA DE DEFENSA No. UITEY-ITE-2019-00012-AD

En la ciudad de San Miguel de Urququí, Provincia de Imbabura, a los 23 días del mes de septiembre de 2019, a las 16:00 horas, en el Aula AI-102 de la Universidad de Investigación de Tecnología Experimental Yachay y ante el Tribunal Calificador, integrado por los docentes:

Presidente Tribunal de Defensa _____ Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.
Miembro No Tutor _____ Dr. ACOSTA ORELLANA, ANTONIO RAMON , Ph.D.
Tutor _____ Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

Se presenta el(la) señor(ita) estudiante ROJAS JIMENEZ, ANGEL ADRIAN, con cédula de identidad No. 0704302470, 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, con el objeto de rendir la sustentación de su trabajo de titulación denominado: **A Target Oriented Averaging SearchTrajectory and its application to Artificial Neural Network Training**, previa a la obtención del título de MATEMÁTICO/A.

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

Tutor _____ Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.

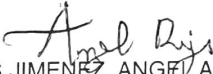
Y recibió las observaciones de los otros miembros del Tribunal Calificador, las mismas que han sido incorporadas por el(la) estudiante.


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, 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:

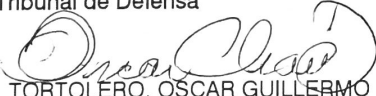
Tipo	Docente	Calificación
Tutor	Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.	9,5
Presidente Tribunal De Defensa	Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.	10,0
Miembro Tribunal De Defensa	Dr. ACOSTA ORELLANA, ANTONIO RAMON , Ph.D.	10,0

Lo que da un promedio de: **9.8 (Nueve punto Ocho)**, 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.


ROJAS JIMENEZ, ANGEL ADRIAN
Estudiante


Dr. PELUFFO ORDONEZ, DIEGO HERMAN , Ph.D.
Presidente Tribunal de Defensa


Dr. CHANG TORTOLERO, OSCAR GUILLERMO , Ph.D.
Tutor

Handwritten signature

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

Handwritten signature

MEDINA BRITO, DAYSY MARGARITA
Secretario Ad-hoc

Contents

1. Introduction and Motivation.	6
2. Preliminaries: Viewing Machine Learning as Optimization.	8
2.1. Learning from samples: a supervised approach.	8
2.2. Training with global optimization methods.	12
2.2.1. Abs-normal form.	15
2.2.2. Abs-linear form.	17
3. Supervised learning via artificial neural networks.	19
3.1. Optimizer’s house of horrors.	20
3.2. Academic learning problems.	22
4. Successive Abs-Linear Global Optimization (SALGO).	25
4.1. Generalized abs-normal form (GANF) and generalized abs-linear form (GALF).	25
4.2. SALGO approach.	26
5. Optimization strategies of SALGO.	28
5.1. Mixed Integer Linear Optimization (MILOP).	28
5.2. Target Oriented Averaging Search Trajectory (TOAST).	29
5.2.1. Generalization to piecewise smooth functions.	32
5.2.2. Numerical integration of TOAST given the prox-linear form.	35
5.2.3. Exact solution of TOAST for the prox-linear case.	36
5.3. SALGO-TOAST algorithm.	38
6. Experimental Results.	42
6.1. TOAST path.	42
6.2. Regression of Griewank function.	43
7. Conclusions.	46
References	47
Appendices	49
A. Further analysis.	49
A.1. Advantages of the GANF/GALF.	49
A.2. SALGO-TOAST inner loop for the multilayer case.	50
B. Codes.	54
B.1. SALGO-MILOP code in AMPL.	54

AUTORÍA

Yo, Ángel Adrián Rojas Jiménez, con cédula de identidad 0704302470, 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, Septiembre 2019.

Ángel Adrián Rojas Jiménez
C.I.:0704302470

AUTORIZACIÓN DE PUBLICACIÓN

Yo, Ángel Adrián Rojas Jiménez, con cédula de identidad 0704302470, 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.

Urcuquí, Septiembre 2019.

Ángel Adrián Rojas Jiménez
C.I.:0704302470

Resumen

El entrenamiento de redes neuronales artificiales usualmente involucran funciones objetivos no suaves a ser minimizadas. Este problema de optimización actualmente se resuelve evitando puntos de ensilladura y por tanto alcanzando un minimizador local o global. Optimizadores conocidos como Descenso por Gradiente (GD) o su versión estocástica llamada Método de Gradiente Estocástico (SG) dependen de la dirección de descenso más próximo. Análisis de convergencia de estos métodos dejan mucho que desear cuando se asume que convexidad y/o suavidad local (véase, [1][2]). Aunque en la práctica estos métodos iterativos funcionan bien a través de la estrategia de propagación de error hacia atrás, nosotros desarrollamos un método determinístico de optimización global llamado SALGO-TOAST. El acrónimo de SALGO describen las iniciales en inglés de la técnica **S**uccessive **A**bs-**L**inearization o Abs-Linearizaciones Sucesivas de la función objetivo y la tarea de **G**lobal **O**ptimization u Optimización Global sobre la aproximación mencionada. Lo último es realizado a través de nuestro método de trayectoria de búsqueda promediada orientada a objetivos o TOAST. Su nombre trata de explicar el comportamiento de las direcciones de búsqueda obtenidas en el método. De hecho, cada dirección de búsqueda está definida por un promedio de la dirección de descenso más próximo y orientado por un valor objetivo a ser alcanzado [4]. La diferencia principal de nuestra estrategia y el algoritmo de propogación de error hacia atrás es nuestra consideración de la no suavidad de la función de predicción. Implementamos TOAST al entrenamiento de redes neuronales artificiales con funciones de activación de visagra o ReLUs para solucionar el problema de aprendizaje de la regresión de la función Griewank y el reconocimiento de imágenes de dígitos. Los resultados son comparados con SG, GD y otro método determinísticco llamado Optimización Mixta Entero Lineal (MILOP). Este último método tiene una formulación preliminar y es la única con el que puede ser probado que un mínimo global ha sido obtenido [6].

Palabras clave: Abs-Linearización Sucesiva, Regularización Cuadrática, Forma Abs-Normal, Descenso Generalizado, Trayectoria de Búsqueda Dinámica.

Abstract

The training of artificial neural networks usually involves nonsmooth objective functions to be minimized. This optimization problem is currently solved just avoing saddle points and thus reaching a local and fortunately a global minimizer. Well-known optimizers like Gradient Descent (GD) or its stochastic version called Stochastic Gradient (SG) depend on the steepest descent direction. Convergence analysis of these methods leave a lot to be desired when is assumed local smoothness and/or (strong) convexity (see, [1][2]). Though, in practice, these iterative methods work well through the back-propagation algorithm [3], we develop a deterministic global optimization method called SALGO-TOAST. Here SALGO denotes the **S**uccessive **A**bs-**L**inearization technique of the objective function and the **G**lobal **O**ptimization task over that approximation. The latter task is given by our **T**arget **O**riented **A**veraging **S**earch **T**rajectory (TOAST). Its name try to describe the behavior of the search direction developed in the method. Indeed each direction is defined by an average of the stepest descent direction and oriented by a target value to be reached [4]. The main difference of our algorithm and backpropagation is that the latter does not consider the nonsmoothness of the prediction function. We implement our method to the training of the Rectifier ANNs [5] to solve the learning problem of the Griewank function regression and the digit-image recognition. The results are compared with SG, GD and another deterministic method called **M**ixed **I**nteger **L**inear **O**ptimization (MILOP). The last method has a preliminary formulation and is the only one able to reach a global minima [6].

Keywords: Successive Abs-Linearization, Quadratic Regularization, Abs-Normal Form, Generalized descent, Dynamic Search Trajectory.

1. Introduction and Motivation.

Machine Learning (ML) conceived with the ambitious goal to automatically reproduce the human learning. A first example of this was the Perceptron Algorithm [7], a model of an artificial neuron learning from samples. Nowadays the computational capacity allows us to apply the basics of the Perceptron Algorithm on a symbolic net of these artificial neurons which is well-known as Artificial Neural Networks (ANNs). ANN and all ML prediction models likewise employ an ML algorithm to solve an optimization problem (a.k.a. learning problem) through different learning stages. In general, the optimization is done over a non-smooth and non-negative objective function. This function depends on a set of parameters and is evaluated on a known dataset called *samples*. The ANN considers subsets of these samples to *train*, *validate*, or *test* itself. In *supervised learning*, the purpose is that the ANN can produce a prediction function or a predictor that fits the training sample and predicts similar data depending on the adjustments of its parameters. The researchers label the fitness error average the Empirical Risk (ER) where its minima are solutions of the optimization problem [1].

Nowadays, the literature presents outstanding results of the fitting power of the Rectifier ANN. That is the ANN using hinge activation functions called Rectifier Linear Units (ReLUs)(see, e.g. [3],[8]). However the training task of algorithmically choosing the ANN parameters, weights and shifts, in order to globally minimize the associated non-smooth objective function ER, turns out not to be simple in this case study. For example, let us briefly study the single-layer case of the Rectifier ANN with constant output weighting $p \in \{-1, 1\}^d$. The predictor

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

depends on a feature vector $\chi \in \mathbb{R}^n$, the corresponding label $y \in \mathbb{R}$, and the prediction function parameter weights $W \in \mathbb{R}^{d \times n}$, and the shifts $b \in \mathbb{R}^d$. Then the learning problem

$$\min_x \varphi(x) = \frac{1}{m} \sum_{k=1}^m |f(\chi_k; x) - y_k| \quad (2)$$

over a training sample of m pairs $(\chi_k, y_k)_{k=1}^m$ sets a complex scenario for any optimization method or optimizer. We describe this scenario in the following four observations:

1. **Nonsmoothness.** At all isolated local, and at least one global, minimizer of the ER, $\varphi(x)$ is not differentiable.
2. **Multi-modality.** There may be local minima with values higher than the globally minimal value of the ER.
3. **Singularity.** Reciprocal scaling of successive weights leaves the prediction values constant.
4. **Zero-Plateau.** For large negative b the prediction function f and the ER gradient $\nabla\varphi(x)$ vanish identically.

These four constitute an optimizer's house of horrors. Most of the users often neglect these aspects implementing ML algorithms and their respective optimizers as black boxes. Such a pragmatic approach contrasts with the convergence analysis of steepest descent variants under ideal assumptions like local smoothness and/or (strong) convexity (see, e.g. [2]). Nevertheless, we notice in some experiments that classical optimizers such as Stochastic Gradient Method (SG) and Gradient Descent (GD) may get stuck in saddle points and at some local minima.

The latter suggests that the training of the Rectifier ANN could be improved. Therefore our motivation is to enhance such a training by means of a novel (nonsmooth) optimizer. For a meaningful implementation, this optimizer should take care of the house of horrors and enforce implicit optimality conditions of the theory of nonsmooth optimization (see e.g. [9][10]).

This research focuses on two aspects of the training of the Rectifier ANN: the nonsmoothness and the global optimization of the correspondent ER. We formulate a global nonsmooth optimization approach called Successive Abs-Linear Global Optimization (SALGO). The aim of SALGO is to exploit the echeloned nonsmooth composition of hinge activation functions through a suitable representation of the ER namely Abs-Normal Form [11][12][13]. SALGO also allows for a family of different search strategies to find global minima. One of them is the Target Oriented Averaging Search Trajectory (TOAST) based on *the generalized descent* proposed in [4]. These tools are the mathematical foundation of our SALGO-TOAST algorithm and its application to the training of the Rectifier ANN. Summarizing, our algorithm performs two tasks: the Successive Abs-Linearization of the objective function ER and its Global Optimization using TOAST. These tasks are implemented in the outer and the inner loop of the algorithm. The reader can view our optimizer like the Gradient Momentum Method [14] rather than just: as a variant of steepest descent. Additionally, we develop another method called Mixed Integer Bilinear Optimization, MIBLOP. We exploit the multi-piecewise linearity with respect to the weights and shifts of the prediction function and thus the objective function. A preliminary formulation is given for just the one-layer ANN case. Unlike to SALGO-TOAST philosophy, MIBLOP is an iterative but not heuristic method that can be proven mathematically to reach a global minima of the objective function in a finite number of iterations.

This document is organized in six main chapters and one appendix. The chapter of preliminaries introduces main concepts of machine learning to formulate the optimization problem of our interest. At its last two subsections, we present the approximation technique of abs-linearization. We describe its properties like uniformly boundness and the resulting formulation of the piecewise smooth structure. The next chapter is the supervised learning via artificial neural networks. We present the mathematical formulation of the prediction function deduced by the multi-layer case of the Rectifier ANNs. In its two sections, we formally describe the optimizer's house of horrors and two academic learning problems: the regression of the Griewank function and the recognition of hand-written numbers. Afterwards, there are the chapters titled Successive Abs-Linearization and optimization strategies in SALGO approach. In both, we detail our strategies to solve the optimization problem of learning. Those are MIBLOP and TOAST. Finally, we end up with our experimental results but just for the one-layer case and the conclusions about our algorithms implemented. In the appendix, one can consult a further analysis for the multi-layer case of the SALGO-TOAST algorithm, the codes of the numerical simulation of the methods used, and more.

2. Preliminaries: Viewing Machine Learning as Optimization.

2.1. Learning from samples: a supervised approach.

Machine Learning (ML) is the scientific field that researches fundamental principles and develops algorithms capable of leveraging collected data to automatically produce accurate prediction functions applicable to similar data (see, e.g., [1][15][16]). Here a computer is in charge to preprocess such information to convert it into a useful dataset representation. In general, a vector or matrix of real numbers are behalf on the dataset so that the ML algorithm \mathcal{A} can operate. This dataset is called samples or prior knowledge where each data point is composed of its features and labels. We write a description of this terminology below.

- **Samples (Prior Knowledge):** The dataset \mathcal{K} of the pairs of known features and labels, i.e. $\mathcal{K} = \mathcal{X} \times \mathcal{Y}$, collected by an electronic device such that \mathcal{A} can operate, evaluate, or test itself with \mathcal{K} . For example, \mathcal{K} can be a set of voice notes, photos, or e-mails which are accompanied with their respective labels. These can be levels of soundness, sort of objects, and issues of content, respectively.
- **Features:** A subset of \mathcal{K} that are the attributes of each sample. The common representation is a real column vector $\chi \in \mathcal{X}$ where \mathcal{X} denotes the set of all possible features. \mathcal{X} is also known as the *input space*.
- **Labels:** Values or categories that corresponds to a sample accompanying its features. Depending on the learning task, the label is a real value or a real vector $y \in \mathcal{Y}$ where \mathcal{Y} is the set of all possible labels. \mathcal{Y} is also known as the *output space*.

So far we can say that the key ingredient of \mathcal{A} is the richness of the prior knowledge and its representation. This implies that the complexity of the data representation and the dimension of the samples determine the applicability of a particular ML algorithm. This observation is currently studied in "The Representation Learning", a field itself in the ML community [17]. Nevertheless, we assume a "good" representation of the *prior knowledge* so the capacity of the ML algorithm relies on its computational complexity and its memory usage. Both factors are discussed for SALGO-TOAST and MIBLOP algorithms. The question now is how we manipulate this knowledge. The following terminology describes the mathematical perspective of such an approach.

- **Prediction model:** A prediction model is said to be a representation of a family of functions \mathcal{F} that aims to estimate or *generalize* a data which has not been previously analyzed. Prediction models like *artificial neural networks* particularly represent a family of parametrized predictors that no longer depend on the input data. This kind of prediction functions are denoted by $f \equiv f(\chi; x)$ where $\chi \in \mathcal{X}$ and x is the only independent variable made up of prediction model parameters. Afterwards we establish the conditions under which \mathcal{A} uses f to achieve an approximate solution of the (supervised) learning problem.
- **Supervised learning problem:** Let us assume that each known sample $(\chi, y) \in \mathcal{K}$ is independently and identically distributed (i.i.d.) according to some fixed, but unknown distribution \mathcal{P} . Consider a family of parametrized prediction functions \mathcal{F} which is fixed and each $f \in \mathcal{F}$ is defined as

$$\begin{aligned} f : \Omega \subset \mathbb{R}^q &\rightarrow \bar{\mathcal{Y}} \\ (\chi; x) &\mapsto f(\chi; x) \end{aligned}$$

where the value $f(\chi; x)$ is called prediction and $\bar{\mathcal{Y}}$ does not necessarily coincide with the output space \mathcal{Y} . Taking $\mathcal{S} = (\chi_1, \dots, \chi_m) \subset \mathcal{X}$, we define a *concept* c , which is unknown in general, that pointwise maps \mathcal{S} to \mathcal{Y} . So the vector of labels is defined by $y = (c(\chi_1), \dots, c(\chi_m)) \in \mathcal{Y}$. Therefore the learning problem is based on how the prediction function f learns the (unknown) concept c . Mathematically, it is the selection of the optimal parameters x that minimize the *generalization error* (a.k.a. Expected Risk) between f and c over the prior knowledge \mathcal{K} .

Definition 2.1 (Generalization error). *For any prediction function $f \in \mathcal{F}$, a concept c evaluated in \mathcal{S} , and $(\chi, y = c(\chi))$ i.i.d. by \mathcal{P} , the generalization error $R : \mathcal{F} \rightarrow \mathbb{R}$ is given by*

$$R(f) = \mathbb{E}_{\chi \sim \mathcal{P}}[\ell(f(\chi; x), y)] \quad (3)$$

where $\ell : \bar{\mathcal{Y}} \times \mathcal{Y} \rightarrow \mathbb{R}$ is called the *loss function* that quantifies the discrepancy between the prediction $f(\chi; x)$ and the label y .

In other words, the generalization error estimates how poorly the prediction is on the infinity of future samples that are distributed by \mathcal{P} . So the learning problem can be mathematically written as a variational problem of the generalization error in terms of the functional f

$$\min_{f \in \mathcal{F}} R(f) \quad (4)$$

Unfortunately, (4) is not explicitly computable because the distribution \mathcal{P} is unknown. Instead we compute an approximation of (3) called *Empirical Risk* (ER).

Definition 2.2 (Empirical Risk). *For any $f \in \mathcal{F}$ and a given dataset $\mathcal{D} \subset \mathcal{K}$, we denote the empirical risk as $\varphi : \mathcal{F} \rightarrow \mathbb{R}$ given by*

$$\varphi(f, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(\chi, y) \in \mathcal{D}} \ell(f(\chi; x), y) \quad (5)$$

where $|\mathcal{D}|$ is the number of samples in \mathcal{D} . Since \mathcal{D} is fixed, note that φ just depends on the functional f .

This measure tells us how poorly the evaluation of the prediction is on average on the dataset \mathcal{D} . The **Empirical Risk Minimization (ERM) principle** says that one should find a predictor f^* that minimizes the Empirical Risk over \mathcal{D} , i.e.

$$f^* = \operatorname{argmin}_{f \in \mathcal{F}} \varphi(f, \mathcal{D}). \quad (6)$$

So we can afford a good estimate $\varphi(f^*, \mathcal{D})$ of the generalization error $R(f)$ [16].

Example 2.1 (Binary Classification.). *We are going to prove that the expected value of the ER evaluated in f^* is equal to the Generalization Error for the binary classification problem. Let us denote the concept c as the binary mapping $\chi \mapsto y \in \mathcal{Y} = \{0, 1\}$, and the loss function by*

$$\ell(x, y) = \mathbb{1}[f(\chi; x) \neq y], \text{ where } \mathbb{1}[E] = \begin{cases} 1, & \text{the event } E \text{ is true} \\ 0, & \text{the event } E \text{ is false} \end{cases}, \quad (7)$$

namely the indicator function. Since the samples are drawn i.i.d. and because of the linearity of the expectation value, one obtains the following equalities

$$\mathbb{E}_{\chi_i \sim \mathcal{P}}[\varphi(f^*, \mathcal{D})] = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbb{E}_{\chi_i \sim \mathcal{P}}[\mathbb{1}[f(\chi_i; x) \neq y_i]] = \frac{1}{|\mathcal{D}|} \sum \mathbb{E}_{\chi \sim \mathcal{P}}[\mathbb{1}[f(\chi; x) \neq y]]$$

for any $(\chi, y) \in \mathcal{K}$. Due to the arbitrariness of (χ, y) , the expected value repeats $|\mathcal{D}|$ times. Thus the desired equality

$$\mathbb{E}_{\chi \sim \mathcal{P}}[\varphi(f^*, \mathcal{D})] = R(f)$$

holds as the generalization error is the expected value of the loss function.

The construction of the proof of the above equality is nontrivial when the prediction model represents a family of complex prediction function, e.g. non-linear functions, to solve a multi-classification problem or a regression of a non-polynomial mapping. A further observation is the i.i.d choice of the sample points to evaluate the ER. Otherwise the computation of the ER would result in a biased estimate of the generalization error. For this purpose, the prior knowledge \mathcal{K} is split in three subsets: training sample, validation data, and test sample to be used in the following learning stages, see Figure 1:

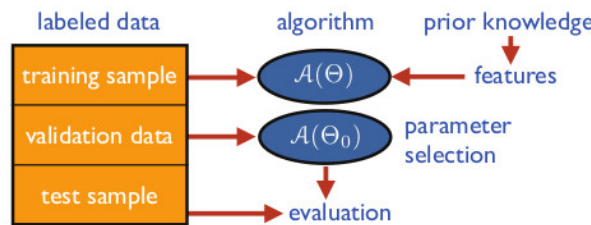


Figure 1: Learning stages of a supervised learning algorithm

- **Training stage:** Given a random subset $\mathcal{D}_{\text{TRAIN}} \subset \mathcal{K}$ called training sample, the training stage is the usage of an optimizer in order to implement the ERM principle using $\mathcal{D}_{\text{TRAIN}}$.
- **Validation stage:** Given a random subset $\mathcal{D}_{\text{VALID}} \subset \mathcal{K}$, called validation data, the solution of the optimizer is validated with a different dataset than $\mathcal{D}_{\text{TRAIN}}$. This stage is also utilized to adjust the hyperparameters Θ of \mathcal{A} if a substantial reduction of the ER is presented. Here the hyperparameters are free parameters of \mathcal{A} and not determined by \mathcal{A} itself, e.g. the stepsize of \mathcal{A} 's optimizer.
- **Test stage:** \mathcal{A} utilizes the last adjustment of the ML algorithm hyperparameters and prediction parameter to compute a new prediction. This evaluation is done over the test sample. Here the test sample is a random subset $\mathcal{D}_{\text{TEST}} \subset \mathcal{K}$. Unlike $\mathcal{D}_{\text{TRAIN}}$ or $\mathcal{D}_{\text{VALID}}$, $\mathcal{D}_{\text{TEST}}$ must never have been used during training or validation stages of \mathcal{A} .

Remark 2.3. *The hyperparameters can be automatically adjusted in which case the validation stage is not always necessary. This possibility will depend on the sophistication or generalization power of the prediction model and the ML algorithm.*

In summary, a family F of (parametrized) prediction functions and a loss mapping ℓ to evaluate the quality of the predictions are needed to attack the learning problem through \mathcal{A} . The missing component is to find a suitable method that solves the ERM principle described in (6). Since

this is an optimization problem, the ML community refers to this optimization method as an optimizer. Whence the optimization task is to search the optimal (hyper)parameters x, Θ of the functional f , and \mathcal{A} . The following are the steps to follow for solving the ERM principle and achieving an estimate of the solution of the supervised learning problem.

Algorithm 1 Supervised learning algorithm $\mathcal{A}(\Theta)$

Require: $\mathcal{K}, x, \Theta, f, \ell$

function SPLIT(\mathcal{K}) ▷ split the dataset of samples in three: $\mathcal{D}_{\text{TRAIN}}, \mathcal{D}_{\text{VALID}}, \mathcal{D}_{\text{TEST}}$
end function

while f^* does not sufficiently minimize $\varphi(f, \mathcal{D}_{\text{TRAIN}})$ **do**
 function OPTIMIZER(x, Θ, f, ℓ) ▷ The ERM principle solved by the optimizer
 end function

return f^*

end while ▷ The above loop is called the TRAINING of \mathcal{A}

evaluate $\varphi(f^*, \mathcal{D}_{\text{VALID}}) = \varphi^*$ ▷ Remember that φ is a functional

function ADJUST(Θ) $\equiv \Theta_0$ ▷ Adjustment of Θ to possibly enforce a reduction of the ER
end function

evaluate $\varphi(f^*, \mathcal{D}_{\text{VALID}}, \Theta_0) = \bar{\varphi}$

if $\varphi^* < \bar{\varphi}$ **then return** φ^*
else TRAIN again with Θ_0

end if ▷ This if-statement is called the VALIDATION stage of \mathcal{A}

return $\varphi(f^*, \mathcal{D}_{\text{TEST}})$ ▷ Compute the unbiased estimate of the GE. This is the TEST stage.

The variational problem (6) is simplified if one choose a fixed form of the parametrized functional f . Then the ER becomes a function dependent on just the predictor's parameters, i.e. $\varphi : \Omega \rightarrow \mathbb{R}$. This implies that, during the training stage, the optimizer tries to obtain x_* given by

$$x^* = \operatorname{argmin}_{x \in \Omega} \varphi(x) \equiv \operatorname{argmin} \varphi(f(\chi; x), \mathcal{D}_{\text{TRAIN}}). \quad (8)$$

Here x^* is denoted as the outcome of the training of the prediction model represented by f . However this problem could be NP-hard if the loss function is discontinuous or the prediction function is very nonlinear. For instance, the indicator loss function described in (7) is non-differentiable almost everywhere with respect to the parameters x . In [15][18], they argue that the minimization of the ER become possible if the surrogate loss function is convex and non-decreasing. The goal is to get an upper bound of the original error using a surrogate loss function. This observation is verified for the indicator function and its following convex non-decreasing surrogate functions. For any prediction f and any label $y \in \{0, 1\}$, the binary classification problem has indicator loss function values upper bounded by the following mappings

- Hinge loss: $\ell(f, y) = \max(0, f - y)$
- Exponential loss: $\ell(f, y) = \exp(f - y)$
- Logistic loss: $\ell(f, y) = \log_2(e^f - y)$
- Euclidean norm loss: $\ell(f, y) = (f - y)^2$
- Squared Hinge loss: $\ell(f - y) = (\max(0, f - y))^2$.

The multiclass classification problem, when the labels are two or more, holds the observation above.

We should note that Algorithm 1 is just useful for the supervised learning problem, but not for other approaches like semi-supervised, unsupervised, and reinforcement learning. Because the way to estimate the generalization error is different [19]. In unsupervised learning, there are no labels in the output space, so \mathcal{A} must generate a descriptive model of the input space rather than a prediction model. Tasks like clustering, dimensional reduction, ranking, and density estimation are examples of this kind of learning. Meanwhile a reinforcement learning algorithm needs a feedback called reward or reinforcement as input in order to take a new decision depending on all(or some) past decisions. For example, a reinforcement learning model is the Markov decision process. With a mix of all these approaches in just one algorithm, the goal is to construct the ultimate Machine Learning Algorithm [20][21].

In the next section we deep on the analysis of the training stage. This optimization task relies on that all of the prediction models construct a nonsmooth and nonnegative ER. We will check that the geometry and smoothness of the ER make of the optimization problem in (8) hard to solve. The latter is due to the non-uniqueness of its gradient and the existence of several stationary points high above the globally minimal value.

2.2. Training with global optimization methods.

We introduce our mathematical approach to the training of a specific family of parametrized prediction function, the Rectifier ANN. During the training stage, ML generally attempts to avoid getting stuck at saddle points to hopefully reach low local minima of the ER. But the difficulty of this task relies on the mathematical properties of the predictor and the loss function. For example, the predictor in (1) for the two variable-case is given by

$$f(\chi; W) \equiv p^\top \max(0, W\chi) \quad \text{with} \quad \chi, W \in \mathbb{R}^2. \tag{9}$$

So the choice of the loss function implies different mathematical geometries for the ER surface. If we consider the ER evaluated on one training sample point $(\chi, y) = (1, 1)$ and $p^\top = (1, -1)$, we obtain the following loss surfaces where $\varphi(x) \equiv \ell(f(\chi; x), y)$

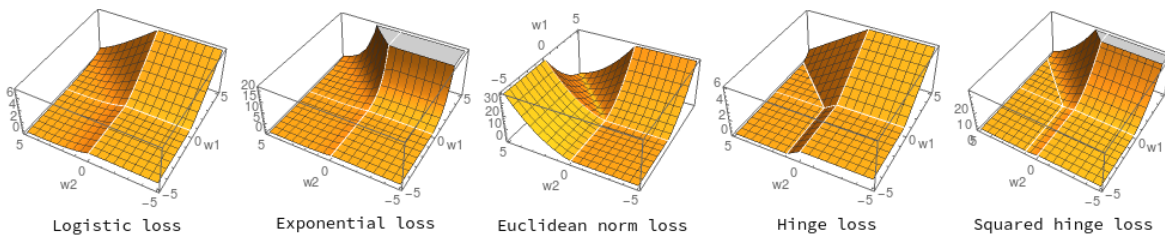


Figure 2: Loss surfaces of an one-layer ANN based on 2 neurons using each one of the loss functions listed in the last section

The ER consequently has a non-differentiable domain with zero Lebesgue measure even for a larger training sample. These few insights set up a realistic scenario about what the optimization methods should consider for mathematically and computationally guaranteeing a global minimizer. We extend our analysis by the following definitions, theorems, and examples.

Definition 2.4 (Global Optimization). *We say that global optimization is the design and numerical comparisons of heuristic procedures to locate low local minima of an objective function most successfully [4].*

This characterization emphasizes that, apart of space covering/filling techniques, all other strategies are heuristic, e.g. trajectory methods, random-search methods, and methods based on stochastic models [22][23]. The justification for using these methods is that, for more than two variables, space covering methods tend to exceed computational limitation, as they have to evaluate the function on a sufficiently dense grid to *cover* the search area. We suggest that the search for minimal values should reach the promising region in the parameter normed space $(\mathbb{R}^n, \|\cdot\|_p)$ with $n < \infty$. $\|\cdot\|_p$ denotes the l_p norm with $p \in \{1, 2, \infty\}$ given by

$$\|\cdot\|_p : v \in \mathbb{R}^n \mapsto \|v\|_p = \begin{cases} \sum_{i \in I} |v_i|, & \text{if } p = 1 \\ \sqrt{\sum_{i \in I} v_i^2}, & \text{if } p = 2 \\ \max(v_i)_{i \in I}, & \text{if } p = \infty \end{cases}$$

where I is the index that contains the i -th components of the vector v . Thus if the ER domain $\Omega \subset \mathbb{R}^n$ is a compact set then it is closed and bounded (see, [24]). Theoretically, the existence of a global minimum of a function defined in that normed space is assumed as follows.

Definition 2.5. A function $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ is called lower semi-continuous (l.s.c.) on a set Ω if for $x_0, x \in \Omega$

$$\varphi(x_0) \leq \liminf_{x \rightarrow x_0} \varphi(x)$$

Given the ER is a continuous mapping, it holds to be a lower semicontinuous function. Therefore the minimization of the ER is always possible because of the Weierstrass theorem.

Theorem 2.6 (Weierstrass). Given a l.s.c. function $\varphi : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$ and if Ω is a compact set, then φ attains a global minimizer

$$x_* = \operatorname{argmin}_{x \in \Omega} \varphi(x)$$

So one should design a strategy to attain the minimizer of the ER according to its geometrical and analytical properties. By definition, the ER might be convex as it is the average of the convex and non-decreasing surrogate loss function.

Definition 2.7 (Convex set and Convex function). We call a set $C \subset \mathbb{R}^n$ convex if for all real numbers $\lambda \in [0, 1]$

$$x, y \in C \Rightarrow \lambda x + (1 - \lambda)y \in C,$$

whereas a function $\ell : C \rightarrow \mathbb{R}$ is said to be convex if for any $x, y \in C$, where $C \subset \mathbb{R}^n$ is a convex set, and for all $\lambda \in [0, 1] \subset \mathbb{R}$ the relation

$$\ell(\lambda x + (1 - \lambda)y) \leq \lambda \ell(x) + (1 - \lambda)\ell(y)$$

holds.

Example 2.2. The surrogate loss functions: hinge, exponential, logistic, Euclidean norm, squared hinge loss are convex on \mathbb{R} .

However, this is not a guarantee that the ER is convex. In [Figure 2](#), the composition of a convex function and the subtraction of two other convex functions is not convex. Thus the learning problem cannot generally be classified as a convex optimization problem (see, [Table 1](#)). Therefore we extend to a more analytical rather than a geometric approach. We study the Lipschitz continuity of φ so that $\|\cdot\|$ denotes the Euclidean norm without loss of generality.

Table 1: Cases that a convex function maintains convexity. The (concave) hinge column refers to the composition between a convex function and the hinge function where the concave case denotes $\min(0, \ell)$.

Geometry	Scaling	Addition	Subtraction	Composition	Hinge	Concave hinge
Convex	Yes	Yes	No	No	Yes	No

Definition 2.8 (Lipschitz continuity). *A function $\varphi : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$ is called locally Lipschitz continuous over Ω if, for all $x_0 \in \Omega$, there exist $\varepsilon > 0$ and a constant $L > 0$ such that*

$$\forall x \in \Omega \cap B_\varepsilon(x_0) : |\varphi(x_0) - \varphi(x)| \leq L\|x_0 - x\|$$

where $B_\varepsilon(x_0) = \{x \in \mathbb{R}^n : \|x_0 - x\| < \varepsilon\}$ denotes the open ball around x_0 with radius ε . If there exists a choice of L , so that

$$\forall x_1, x_2 \in \Omega : |\varphi(x_1) - \varphi(x_2)| \leq L\|x_1 - x_2\|$$

holds, then we call φ globally Lipschitz continuous on Ω .

Furthermore, we denote $\varphi \in \mathcal{C}^{1,1}(\Omega)$ where the second superscript refers to the Lipschitz continuity of the first-order derivative of φ . The constant L is referred as the Lipschitz constant. The following statement establishes the relation between convexity and Lipschitz continuity.

Proposition 2.9. *Any convex function φ is Lipschitz continuous with some Lipschitz constant L on some open ball $B_\varepsilon(x_0)$ with $\varepsilon = \varepsilon(x_0) > 0$ for all $x_0 \in C \subset \mathbb{R}^n$.*

From now on we assume that the ER is Lipschitz continuous because the prediction function and the loss function studied before are too. But this is not true for the indicator function or any other discontinuous loss function. Then we can apply the following theorem.

Theorem 2.10 (Rademacher). *Suppose $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}$ is locally Lipschitz continuous on $\Omega \subset \mathbb{R}^n$, then there exists the Fréchet derivative*

$$\varphi'(x) \in \mathbb{R}^n \text{ with } \|\varphi'(x)\| \leq L(x) \quad \text{s.t.} \quad \lim_{s \rightarrow 0} \frac{\|\varphi(x+s) - \varphi(x) - \varphi'(x)s\|}{\|s\|} = 0$$

at all $x \in \Omega \setminus S$ where S is a singular set, i.e. S has Lebesgue measure zero.

Other important property to analyze is the ER smoothness. The term smooth refers to, for any order $d \geq 1$, a function being continuously differentiable and represented by the set $\mathcal{C}^d(\mathbb{R}^n)$ of real valued mappings. So if $\varphi \in \mathcal{C}^1(\mathbb{R}^n)$, the equality between the Fréchet derivative and the gradient $\nabla\varphi(x)$ holds. But, like in (9), ML often involves non-smooth predictors and loss functions of different d orders. This fact also corresponds to the nonsmooth of the ER because of continuity. So we need to generalize the concept of the derivative as follows:

Definition 2.11 (Limiting gradient). *For all $x \in \Omega$ we call*

$$\partial^L \varphi(x) \equiv \left\{ \lim_{k \rightarrow \infty} \varphi'(x_k) : x_k \rightarrow x \text{ and } x_k \notin S \right\}$$

the set of limiting gradients which are also called Bouligand derivative in the literature.

Definition 2.12 (Convex hull). *We define the convex hull of a set $V \subset \mathbb{R}^n$ as*

$$\text{conv } V = \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^k \lambda_i x_i, \sum_{i=1}^k \lambda_i = 1, x_i \in V, \lambda_i \geq 0, k > 0 \right\}.$$

Taking the convex hull of the set of the limiting gradient $\partial^L \varphi(x)$, we obtain Clarke's generalized derivative

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

Another popular generalized derivative is the one defined by the directional derivatives or Hadamard derivative.

Definition 2.13 (Hadamard derivative). *The Hadamard derivative of φ in the direction $d \in \mathbb{R}^n$ is denoted by*

$$\varphi'(x; d) = \lim_{t \rightarrow 0^+} \frac{1}{t} [\varphi(x + td) - \varphi(x)] \in \mathbb{R} \cup \{-\infty\}$$

provided the limit on the right hand side exists.

Even for Lipschitz functions this equality is not necessarily true in all direction $d \in \mathbb{R}^n$. However, Hadamard derivative does always exist on piecewise smooth functions, which are of our interest. Then the sub-gradient or generalized derivative of φ is defined as follows:

Definition 2.14 (Sub-gradient). *We call the sub-gradient of φ the convex set $\partial \varphi$ defined by the supporting function $\varphi'(x_0, \cdot)$, i.e.*

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

which is identical to the set

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

Then if φ is Lipschitz continuous and convex, the concept of subgradient and Clarke's generalized derivative coincide. In the nonconvex case $\partial \varphi$ may be empty, whereas $\partial^C \varphi(x) \neq \emptyset$ everywhere in the Lipschitz continuous case.

This classical generalized differentiation approach simply reduces to the computing of the slopes where the function is smooth for some piecewise differentiable functions (which are studied in the next subsection). For numerical purposes, any of these generalizations really give no indication whether there is nearby a jump/kink or the nonsmooth domain. This scenario encourages ML users to implement optimization methods like Gradient Descent or Stochastic Gradient without considering the nonsmoothness as the nondifferentiable domain is singular. The next subsection discusses the mathematical properties of this sort of functions constructing its abs-normal form.

2.2.1. Abs-normal form.

Definition 2.15 (Piecewise differentiable). *Let Ω be an open set and a locally Lipschitz continuous function $\varphi : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$ and $d \in \mathbb{N}$, we say that φ is d times piecewise differentiable if at any $x \in \Omega$ there exists a selection function $\varphi_\sigma(x) \in \mathcal{C}^d(\Omega)$ such that $\varphi(x) = \varphi_\sigma(x)$. Here σ belongs to some finite index set \mathcal{E} labeling the selection functions φ_σ .*

The majority of the objective functions associated to the ML prediction models is piecewise differentiable (see, Definition 2.15). This class is enough to represent compositions of all the surrogate loss functions, affine transformations, and the so-called activation functions presented in ANN. In [13], the authors emphasize compositions of nonlinear smooth elemental functions and the absolute value function $\text{abs}(x) = |x| = (|x_1|, \dots, |x_n|)$ which includes the convex or concave hinge function. One can successively number all arguments of the absolute value evaluations as switching (real or vector) variables $z \equiv \{z_i\}_{i=1}^s$. Under these observations, we define the following subset of piecewise differentiable functions.

Definition 2.16 (Composite piecewise smooth functions.). *The set of functions $\varphi : \Omega \subset \mathbb{R}^n \rightarrow \mathbb{R}$ that can be defined by an abs-normal form*

$$z = \Lambda(x, |z|) \quad (10)$$

$$\varphi(x) = \lambda(x, |z(x)|), \quad (11)$$

where $\Lambda : \Omega \times \mathbb{R}_+^s \mapsto \mathbb{R}^s$ and $\lambda : \Omega \times \mathbb{R}_+^s \mapsto \mathbb{R}$ with $\Omega \times \mathbb{R}_+^s \subset \mathbb{R}^{n+s}$ such that $\lambda, \Lambda \in \mathcal{C}^d(\Omega \times \mathbb{R}_+^s)$, is said to be composite piecewise differentiable. We denote this set by $\mathcal{C}_{abs}^d(\Omega)$ for any $d \in \mathbb{N}$.

Here, the switching variables can only influence each other in an echeloned manner, that is, z_i influences z_j only if $i < j$. In consequence, for any $x \in \Omega$, the evaluation of the piecewise smooth value $z(x)$ is unique. Nevertheless, the set of composite piecewise differentiable functions does not include every piecewise differentiable function.

Example 2.3. *Consider the predictor f defined in (9) and using the squared hinge loss function to construct the following ER on the training sample $\mathcal{D} \equiv \mathcal{D}_{TRAIN}$*

$$\varphi(W) = \frac{1}{|\mathcal{D}|} \sum_{(\chi, y) \in \mathcal{D}} [\max(0, (f(\chi, W) - y))]^2.$$

Then the abs-normal form of the ER is

$$\begin{aligned} (z_1, z_2) &= (W\chi, \frac{1}{2}p^\top(W\chi + \text{abs}(z_1)) - y) \\ \varphi(z_1, z_2) &= \frac{1}{16|\mathcal{D}|} \sum_{(\chi, y) \in \mathcal{D}} (p^\top(z_1 + \text{abs}(z_1)) - y + \text{abs}(z_2))^2 \end{aligned}$$

where $\text{abs}(\cdot)$ is the componentwise mapping of the absolute value function.

One can identify the combinatorial structure of the nonsmooth function φ by the signature vector

$$\sigma = \sigma(x) \equiv \text{sgn}(z(x)) \in \{-1, 0, +1\}^s \quad (12)$$

and the signature matrix

$$\Sigma = \Sigma(x) = \text{diag}(\sigma(x)) \in \mathbb{R}^{s \times s}. \quad (13)$$

Then we can define the selection function

$$\varphi_\sigma(x) \equiv \lambda(x, \Sigma z(x)) \text{ s.t. } z(x) = \Lambda(x, \Sigma z(x))$$

where $\varphi \equiv \varphi_\sigma$ is smooth on each signature σ . This fact verifies that every composite piecewise smooth function is piecewise differentiable.

The abs-normal form of any composite piecewise differentiable function leads us to implement a new set-valued mapping of gradients relevant for the optimization of φ . First, we take advantage of the explicit formulation of the selection smooth functions φ_σ .

Definition 2.17 (Active selections.). *The selection function φ_σ is said to be active at $\hat{x} \in \Omega$ if the coincidence set $M_\sigma = \{x \in \Omega : \varphi(x) = \varphi_\sigma(x)\}$ contains \hat{x} . Moreover φ_σ is called essentially active if \hat{x} is in the closure of the interior of M_σ . Finally, it is called conically active if the tangent cone of M_σ at \hat{x} has a nonempty interior. The index sets of the correspondingly active selection function indexes is characterized by the inclusion chain $\mathcal{E}_c(\hat{x}) \subset \mathcal{E}_e(\hat{x}) \subset \mathcal{E}_a(\hat{x}) \subset \mathcal{E}$.*

Lemma 2.1 (Scholtes [25]). *For any piecewise smooth function φ , the limiting gradient is the span of the essentially active gradients, i.e.*

$$\partial^L \varphi(\hat{x}) = \bigcup_{\sigma \in \mathcal{E}_e} \{\nabla \varphi_\sigma(\hat{x})\}.$$

Therefore, for the index of the active selection functions, the following order of the set-valued mappings

$$\emptyset \neq \partial^K \varphi(\hat{x}) \equiv \bigcup_{\sigma \in \mathcal{E}_c(x)} \{\nabla \varphi_\sigma(\hat{x})\} \subset \partial^L \varphi(\hat{x}) \subset \bigcup_{\sigma \in \mathcal{E}_a(x)} \{\nabla \varphi_\sigma(\hat{x})\}$$

holds. Here ∂^K refers to the conical derivative. It can be shown that, for a given a local minimizer \hat{x} , $0 \in \text{conv}\{\partial^K \varphi(\hat{x})\}$. This is a stronger requirement than the Clarke stationarity condition [9]. But the computational verification of this condition is not easy. We approximate this condition using a strategy of linearization and local minimization described in [11],[13].

2.2.2. Abs-linear form.

Definition 2.18 (Semi-smoothness). *A function $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m$ is called semi-smooth at a point x if it holds*

$$\sup_{M \in \partial \varphi(x+s)} \|\varphi(x+s) - \varphi(x) - M \cdot s\| = o(\|s\|)$$

Also, φ is said to be strong semi-smooth if the above equality holds for $O(\|s\|^2)$. Both properties are maintained by elemental function compositions.

Since the abs-normal of objective functions are strongly semi-smooth, their generalized gradients satisfy the backward approximation property for a fixed \hat{x} as follows

$$\varphi(x) - \varphi(\hat{x}) - g^\top (x - \hat{x}) = O(\|x - \hat{x}\|^2) \quad \text{for all } g \in \partial^C \varphi(x). \quad (14)$$

Applying the classical first order Taylor expansion to the abs-normal form in (10) and (11), one obtains the general smooth Taylor expansion for the reference point \hat{x} given by

$$\begin{bmatrix} z - \hat{z} \\ \varphi - \hat{\varphi} \end{bmatrix} = \begin{bmatrix} Z & L \\ a^\top & b^\top \end{bmatrix} \begin{bmatrix} x - \hat{x} \\ |z| - |\hat{z}| \end{bmatrix} + O \left(\begin{array}{l} \|x - \hat{x}\|^2 \\ \|z - \hat{z}\|^2 \end{array} \right) \quad (15)$$

with $\hat{z} = z(\hat{x})$. We call (15) the abs-linear form (ALF) of $\varphi(x)$ at \hat{x} . Here the matrices Z, L are given by

$$L \equiv \frac{\partial \Lambda(x, |z|)}{\partial |z|} \in \mathbb{R}^{s \times s}, \quad Z \equiv \frac{\partial \Lambda(x, |z|)}{\partial x} \in \mathbb{R}^{s \times n} \quad (16)$$

and the vectors a, b are given by

$$a = \frac{\partial \lambda(x, |z|)}{\partial x} \in \mathbb{R}^n, \quad b = \frac{\partial \lambda(x, |z|)}{\partial |z|} \in \mathbb{R}^s. \quad (17)$$

Z, L, a, b are evaluated at the reference point (\hat{x}, \hat{z}) . Because of the echeloned dependence between the z_i , L is a lower triangle matrix and so one can easily check by induction that $\|z - \hat{z}\| =$

$O(\|x - \hat{x}\|)$. Therefore we can obtain the approximation of any composite piecewise differentiable function as follows

$$\Delta\varphi(\hat{x}; \Delta x) \equiv a^\top x + b^\top |z| = \varphi(x) - \varphi(\hat{x}) + O(\|\Delta x\|^2) \quad (18)$$

where

$$z \equiv \hat{z} - L|\hat{z}| + Z\Delta x + L|z|. \quad (19)$$

Since the ER in (5) has an abs-normal form then it has an abs-linear form. A useful remark is that $\Delta\varphi$ is piecewise linear with respect to x .

Definition 2.19 (Piecewise linear function). *A function $\varphi : \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}$ is called piecewise linear if it is piecewise differentiable according to the Definition 2.15 and all its selection functions are linear or affine. We denote this set by $L_{abs}(\mathbb{R}^n)$.*

Moreover, the signature vector (12) and signature matrix (13) carry over the abs-linearization of φ . Actually from the signature vector and using the partial order of the signatures $\sigma = \text{sgn}(z)$ given by

$$\sigma < \tilde{\sigma} \Leftrightarrow \tilde{\sigma}_i \sigma_i \leq \tilde{\sigma}_i^2, \quad \text{for } i = 1 \dots s \quad (20)$$

we can partition the whole domain \mathbb{R}^n into polyhedra

$$P_\sigma \equiv \{x \in \mathbb{R}^n : \sigma(x)\}. \quad (21)$$

Here their essential closures

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

satisfy the inclusion

$$P_\sigma \subset \bar{P}_\sigma.$$

This inclusion is equivalent to the condition in (20). Therefore we can solve the equality constraint of the abs-linear form in (19) over each polyhedra P_σ . Using the equality $|z| = \Sigma z$ for all $x \in \bar{P}_\sigma$, we obtain the affine function

$$z(x) = (I - L\Sigma)^{-1}(c + Zx) \text{ where } c = \hat{z} + L|\hat{z}|. \quad (22)$$

Here the matrix $(I - L\Sigma)^{-1}$ is well defined for any σ because of the strict lower triangularity of L . If one restricts σ to $\{-1, +1\}^s$, we obtain P_σ with non-empty interior and made up of the solutions of the system of inequalities

$$|z(x)| = \Sigma(I - L\Sigma)^{-1}(c + Zx) = (\Sigma - L)^{-1}(c + Zx) \geq 0,$$

if we assume $\hat{x} = 0$ without loss of generality.

Considering the last analysis of the abs-linear form, one can therefore compute the active gradients g_σ of $\Delta\varphi$ given by

$$g_\sigma = a^\top + b^\top (\Sigma - L)^{-1} Z. \quad (23)$$

These gradients can be replaced in (14) and the equality can be still held. Consequently we have a more descriptive and computable gradient to define optimality conditions of a piecewise smooth function. In fact, the following equality holds

$$\emptyset \neq \partial_x^K \varphi(x)|_{x=\hat{x}} \equiv \partial_{\Delta x}^L \Delta\varphi(\hat{x}, \Delta x)|_{\Delta x=0} \subset \partial_x^L \varphi(x)|_{x=\hat{x}} \quad (24)$$

where

$$\partial_{\Delta x}^L \Delta\varphi(\hat{x}, \Delta x)|_{\Delta x=0} = \bigcup_{0 \neq \sigma > \hat{\sigma}} \{a^\top + b^\top (\Sigma - L)^{-1} Z\}.$$

In other words, the conical derivative of the original model φ is the limiting gradient of its local piecewise linearization $\Delta\varphi$. We use this result in our optimization approach called SALGO and in its respective underlying algorithms: SALGO-TOAST and MIBLOP.

3. Supervised learning via artificial neural networks.

Before we study the SALGO approach, we detail the training stage of an ML algorithm that will depend on the prediction model given by an ANN. This model is made up of interconnected neurons that are arranged in layers. These hidden layers are between an input and a resulting prediction while their intermediate connections represent smooth or nonsmooth transformations of the input. In general, the whole network is studied as the compositions of activation functions and affine transformation dependent on the so-called weights and shifts and evaluated in the input. Particularly, we consider a fully connected feedforward ANN, that is, a net graph with the same number of connections as per number of neuron pairs from adjacent layers.

Mathematically we describe this sort of ANNs as an interconnected graph composed of $l + 2$ layers. Here the 0-th layer is the input data or features $\chi_k \in \mathbb{R}^n$ and the $(l + 1)$ -th layer represents the prediction $f(\chi_k, x) \in \mathbb{R}$. However, there will be cases where the dimension of the label is a vector instead of a real number. Both cases are implemented in section 5.3. Moreover each layer has a finite number of neurons which describes the dimensionality of each intermediate variable. We denote this dimension by d^i for the i -th layer. Ultimately this ANN is composed of l affine transformations given by $x = (W, b, p)$ and also of activation functions that (in)activate (i.e. active if the mapping evaluations differ from 0, or inactive otherwise) the affine mapping.

Remark 3.1. *The superscript is reserved to denote the layer which the transformation, parameter, or dimension correspond to.*

Here the affine mappings for each i -th layer are given by

$$a^{(i)} : \mathbb{R}^m \rightarrow \mathbb{R}^s$$

$$x^{(i)} = (W^{(i)}, b^{(i)}) \mapsto a(\chi; x) = W^{(i)}\chi + b^{(i)}$$

and represent the connections between each neuron of the adjacent ANN intermediate layers. Meanwhile the activation function is a componentwise mapping given by

$$h : \mathbb{R}^s \rightarrow \mathbb{R}^s$$

$$v \mapsto h(v) = (h(v_1), \dots, h(v_n))$$

that acts on the affine transformation values.

If we have a real-valued prediction function, the output of the prediction model is computed as a linear combination of the entries of the l -th layer neurons. We represent this mapping by the weighting vector $p \in \{-1, +1\}^{d^l}$, i.e. p has components of ones with different signs. This data-based model only depending on the hinge activation function is called Rectifier ANN. So we formulate our prototypical prediction model as follows:

For a fixed feature $\chi \in \mathbb{R}^n$, the prediction function is given by

$$(\chi; x) \in \mathbb{R}^q \mapsto \mathbb{R} \ni f(\chi; x) = p^\top \Phi(x) \quad (25a)$$

where $q = \sum_{i=0}^l d^{i+1}(d^i + 1)$, and Φ is a mapping made up of compositions and defined by

$$\Phi : x \in \mathbb{R}^q \mapsto \mathbb{R} \ni \Phi(x) = h^l(a^l(\dots h^1(a^1(x^1)) \dots)) \quad (25b)$$

where $h(\cdot) = (\max(0, \cdot), \dots, \max(0, \cdot))$. We select this model because of the investigations done in [8],[5],[26]. Under their claims, we can approximate any continuous function to any accuracy. Notice that the prediction function is multi-pieceswise linear function w.r.t. the parameters $x \in \mathbb{R}^q$ and pieceswise linear w.r.t. the features χ .

Definition 3.2 (Multi-piecewise linear function). *Consider a real-valued function*

$$f = f(x_1, \dots, x_{j-1}, x_j, x_{j+1}, \dots, x_q).$$

It is said that f is a multi-piecewise linear function if it is piecewise linear with respect to x_j when x_i are constant with $i \in \{1, \dots, q\} - \{j\}$.

The piecewise linear functions maintain their geometric form under the transformations described in Table 1. Since the set of piecewise linear functions $L_{\text{abs}}(\mathbb{R}^n)$ are closed under addition, (min)maximization, and composition. This is not true for piecewise quadratics and other generalizations. We extend this characterization of Rectifier ANN when the prediction function is a vector mapping. In that case, we define $f \equiv \Phi$ where Φ is defined in (25b). Thus each of its components f_i is a piecewise linear function with respect to $x_i = (W_i, b_i)$.

Example 3.1. *Consider the prediction model for the one-layer case in (25b), so the associated ER φ evaluated over one training sample pair $(\chi, y) = (1, (1, 1))$ is given by*

$$\varphi(W) = |\max(0, w_1) - 1| + |\max(0, w_2) - 1| = \|f(W) - y\|_1.$$

Here $x = W \in \mathbb{R}^2$ with the weights $W = (w_1, w_2)$ as the inhomogeneity shift $b = (0, 0)$. One can see that the lower bound of φ is clearly zero and applying some basic algebra its global minimizer is at the point $(w_1, w_2) = (1, 1)$. Also, fixing one of the weights, one hinge function is preserved maintaining the piecewise linearity of φ over the other weight.

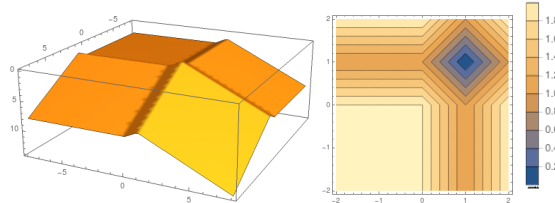


Figure 3: Left: The graph of φ turned upside down with a global minimizer at $(1,1)$. Right: The contours of φ that induce a certain polyhedral decomposition of the domain with respect to each piece, where φ is smooth.

The example 3.1 also illustrates that variations of steepest descent methods may get stuck at local minimizers of the associated ER. In the next subsection, we study this possible phenomenon based on the following four simple observations of the empirical risk: nonsmoothness, multimodality, singularity, and zero-plateau. We emphasize again that these features constitute an optimizer’s house of horrors.

3.1. Optimizer’s house of horrors.

We recall the learning problem for an ANN with a single intermediate layer of d nodes and hinge activation [6] given in (1) as already mentioned in the introduction. This problem can be mathematically described by

$$\min_x |f(\chi; x) - y| \text{ where } f \text{ is given in (25a) with } x \equiv (W, b, p) \quad . \quad (26)$$

Here $\chi \in \mathbb{R}^n$ is a feature vector from a training set, $y \in \mathbb{R}$ the corresponding label, $W \in \mathbb{R}^{d \times n}$ the weight matrix in a vector form between the sample input and the intermediate layer, $b \in \mathbb{R}^d$

the shift inhomogeneity and $p \in \mathbb{R}^d$ the output weight vector.

The empirical risk objective to be minimized is the average of the losses $|f(\chi; x) - y|$ for $(\chi, y) \in \mathbb{R}^{n+1}$ ranging over a training set. In other words we are looking for an optimal l_1 fit over a set of sample points rather than just minimizing the discrepancy at one particular point. This summation does not effect the properties of the objective with respect to the optimization variables $x = (W, b, p) \in \mathbb{R}^{d \times (n+2)}$ to be discussed here. Firstly we observe that due to the positive homogeneity of the hinge function for a scalar $0 < \rho \in \mathbb{R}$

$$\begin{aligned} f(\chi; \rho W, \rho b, p/\rho) &= (p/\rho)^\top \max(0, (\rho W)\chi + (\rho b)) \\ &= p^\top \max(0, W\chi + b) = f(\chi; W, b, p) . \end{aligned}$$

This invariance implies immediately that all local and global minimizers cannot be isolated and their Hessians must be **singular** if they exist at all.

Of course a similar reciprocal scaling of weights and shifts can be applied between successive layers in more general networks leaving the resulting prediction function invariant and thus the risk function minimization problem singular. To remove the **singularity** in the single layer case considered here, we could fix the components of the output weight vector to be -1 or $+1$ so that the prediction function would effectively be split into

$$f(\chi; W_+, W_-, b_+, b_-) = \underbrace{e_+^\top \max(0, W_+\chi + b_+)}_{=f_+(\chi; W_+, b_+)} - \underbrace{e_-^\top \max(0, W_-\chi + b_-)}_{=f_-(\chi; W_-, b_-)}$$

where e_+ and e_- denote vectors of ones in suitable dimensions n_- and n_+ . If we choose $n_+ = n_- = n$ we have twice as many weights and shifts and all possible prediction functions of the original model can be reproduced. Even for this extremely simple model we have universality in the sense that every continuous function can be approximated up to an arbitrarily small absolute error $\varepsilon > 0$ as shown by Yarotsky in [5]. Of course the necessary size $2n$ of the intermediate layer will grow rapidly as ε becomes small. We also note that in this normalized single layer case the prediction function $f(\chi; x)$ and the averages of the losses $|f(\chi; x) - y|$ are piecewise linear w.r.t to each variable in $x = (W, b)$. Hence the learning problem is a global multi-piecewise linear optimization problem so that for any Cartesian basis vector e_j the function $f(\chi; x_j + te_j) - y$, and thus its absolute value is piecewise linear in t . Then the result below follows easily because the multi-piecewise linearity of the objective function verifies its **multi-modal** nature on each kink.

Proposition 3.3 (Minimizers are generically **nonsmooth**).

Suppose a locally Lipschitz continuous function $\varphi : \mathbb{R}^q \mapsto \mathbb{R}$ has a nonempty bounded level set $\{x \in \mathbb{R}^q : \varphi(x) \leq \varphi(x_0)\}$ for some $x_0 \in \mathbb{R}^q$ and is multi-piecewise linear as defined above. Then φ has global minimizers where it is not differentiable, which also applies at all geometrically isolated local minimizers.

Proof. Say x_* is a global minimizer of φ so that clearly $t_* = 0$ is a global minimizer of $\varphi_j(t) = \varphi(x_* + te_j)$ for any $j = 1 \dots q$. Then t_* typically represents a kink, i.e $\varphi_j(t)$ is linear on two intervals $[t_* - \delta, t_*]$ and $[t_*, t_* + \delta]$ but not their union. That means that the left and right directional derivative of $\varphi_j(t)$ differ at t_* so that neither φ_j is differentiable at $t_* = 0$ nor φ is differentiable at x_* . Now, if $t_* = 0$ does not represent a kink in the sense above, it must belong to the interior of a maximal interval $[t_{lo}, t_{hi}] \subset \mathbb{R}$ on which φ_j is linear and, in fact, constant. Otherwise $t_* = 0$ could not be a global minimizer. Because φ is assumed to have a bounded level set, the interval $[t_{lo}, t_{hi}]$ must also be bounded. So its two endpoints represent global minimizers

of the univariate function φ_j and thus the multi-variate φ , at which the former has a kink and the latter is therefore not differentiable. If we have a geometrically isolated-local minimizer it follows immediately that $t_* = 0$ must be a kink, which concludes the proof. \square

In general, a point x_* which is *coordinate minimal* in that all univariate restrictions $\varphi_j(t) = \varphi(x_* + te_j)$ are first order minimal need not be first order minimal for $\varphi(x)$ itself. This is true even for piecewise linear φ , like for example the lemon squeezer described in [12]. However, there is some hope that the special case of ANN, where the variables represent weights-coordinate minimality, might have some implications for the behavior of φ in the full space.

Of course, the observation that machine learning problems are not everywhere differentiable is not new. However, it is often suggested that this nonsmoothness is only transitional and does not matter in the end, which we believe to be unduly optimistic, even if one uses a smooth loss function. In the book [9] several classes and many individual examples of nonsmooth problems are described and locally optimized by bundle and other general purpose nonsmooth optimization methods.

The essential message of the above observations is that all global and even local minimizers of multi-piecewise linear functions are nonsmooth. Theoretically, and with some luck, this problem can be overcome by smoothing of the activation function. However, it seems rather difficult to find the right smoothing scale such that the algorithm behavior is markedly improved without the objective function and its minimizers being altered significantly. In the next chapter, we pursue the strategy of accounting for the nonsmoothness and its combinatorial consequences explicitly. Near a kink steepest descent and all other algorithms designed for smooth problems are likely to chatter back and forth across the kink severely limiting their ability to move tangentially along a valley towards a minimizer that also has directions with smooth growth of second order. It was shown in [12] that under the Linear Independent Kink Qualification such a so-called \mathcal{V} - \mathcal{U} decomposition always exists. The usual recipe for dealing with this chattering or zig-zagging problem is to successively reduce the step-size at a suitable rate. Loosely speaking, the stepsize, also known as learning rate, must be smaller than twice the local Lipschitz constant, which is of course difficult to determine.

There is a fourth difficulty, namely that for all bounded χ and sufficiently large negative b the prediction function $f(\chi; x)$ and its gradients with respect to all variables will vanish identically, so that gradient based methods including stochastic variants can not move away at all. We denote this phenomenon as the **zero-plateau**. In conclusion, we note that from a classical optimization point of view of ANN training problems represent the worst of two worlds, namely nonsmoothness and various singularities. In some other contexts these two properties can be traded off against each other, but here they occur jointly in a generic fashion. This means that the frequently presented convergence analyses [1],[2] for variants of steepest descent assuming smoothness and strong convexity simply do not apply to ANNs with piecewise linear activation. To deal with such problems we pursue the strategy of piecewise linearization described in the next chapter.

3.2. Academic learning problems.

For computational purposes, we highlight two *supervised learning* tasks: classification and regression. Each approach is presented by its respective academic learning problem (see, Chapter 2.1 to recall terminology).

Classification: Handwritten-digits from MNIST database

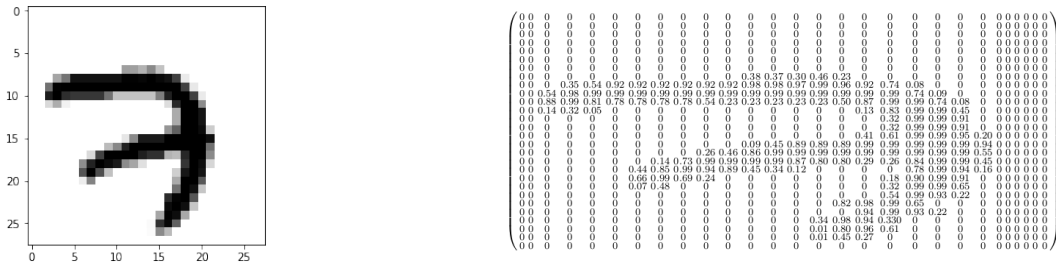


Figure 4: On the left, there is the raw version of a sample of a hand-written *seven* from the MNIST database in a 28×28 pixel image. On the right, a real square matrix $\mathcal{M}_{28 \times 28}$. Each entry corresponds to the color of each pixel in the left image.

The MNIST database provides images of isolated black and white handwritten digits from 0 to 9 [27]. There are in total 70000 images like Figure 4. The size of these images are normalized and centered in a two-dimensional image (28 pixels in each direction). For each hand-written digit, the feature χ belongs to the input space $\mathcal{X} = [0, 1]^{784}$ that describes the unitary grayscale of each pixel, and the correspondent label $y \in \mathcal{Y}$, a vector with discrete components representing 0 each digit category, i.e. the output space $\mathcal{Y} = \{0, 1, \dots, 8, 9\}$. This dataset comes already split in three categories to develop each learning stage: the training sample \mathcal{D}_{55000} , the validation data \mathcal{D}_{5000} , and the test sample \mathcal{D}_{10000} .

The standard loss function for the multiclass classification is the indicator function defined in Example 2.1

$$\ell(f(\chi; x), y) = \mathbb{1}[f(\chi; x) \neq y]$$

where f is the parametrized prediction function. However, this loss function is not useful for any optimizer since it is almost discontinuous everywhere. Therefore the optimizer should implement a continuous surrogate function to measure the misclassification loss.

Regression: the bivariate Griewank function

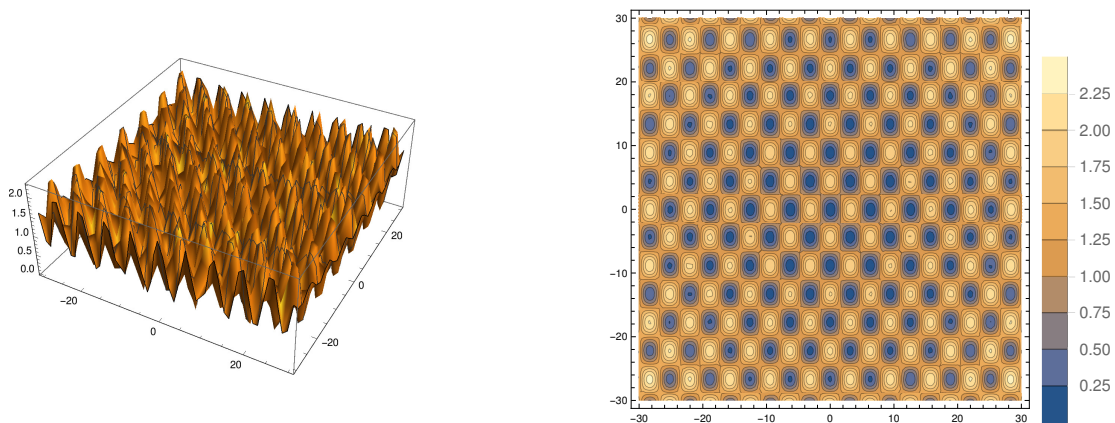


Figure 5: The bivariate Griewank function. The left image is the graph of c , formally written as $\text{graph}(c) = \{(\chi, c(\chi)) : \chi \in \mathcal{X} \subset \mathbb{R}^2\}$. The right one shows different level sets l of c taking values $\rho = \{0.25, 0.50, \dots, 2.00, 2.25\}$, i.e. $l(c) = \{\chi \in \mathcal{X} : c(\chi) = \rho\}$

The Griewank function is a real valued mapping $c : \mathbb{R}^n \rightarrow \mathbb{R}$ given by

$$c(\chi) = \sum_{k=1}^n \frac{\chi_k^2}{4000} - \prod_{k=1}^n \cos\left(\frac{\chi_k}{\sqrt{k}}\right) + 1 \quad (27)$$

where n is the order of the function or the number of components that the vector $\chi = (\chi_1, \dots, \chi_n)$ has (see, e.g [4]). The reader can verify that $c \in \mathcal{C}^2(\mathbb{R}^n)$. For this problem, we use the l_1 norm as our loss function to measure the distance between the prediction values and the Griewank function. In our experimental results, we study a high dimensional case of this function.

4. Successive Abs-Linear Global Optimization (SALGO).

First, let us generalize the abs-normal form in Definition 2.16 and the abs-linear form given in (15) for any objective function in $\mathcal{C}_{\text{abs}}^d(\mathbb{R}^n)$ with $d \geq 1$.

4.1. Generalized abs-normal form (GANF) and generalized abs-linear form (GALF).

We have studied that Rectifier ANNs and ML objective functions are compositions of a sequence of arithmetic operations, smooth intrinsic functions and the nonsmooth elemental functions abs, min and max. Mathematically we can interpret such an evaluation procedure as the generalized abs-normal form

$$\min \varphi(x) \equiv \lambda(x, z, v) \quad \text{s.t.} \quad z = \Lambda(x, z, v) \quad \text{and} \quad v = \text{abs}(z) \quad (28)$$

where

$$\lambda : \mathbb{R}^{q+s+s} \mapsto \mathbb{R} \quad \text{and} \quad \Lambda : \mathbb{R}^{q+s+s} \mapsto \mathbb{R}^s$$

This is a slight generalization of the usual abs-normal form (see, subsection 2.2.1) where z does not occur directly as arguments of φ . As usual, we must require that the matrices

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

are strictly lower triangle. That means we can compute for any x the piecewise smooth functions $z(x)$ and $v(x) = \text{abs}(z(x))$ which finally yields the objective

$$\varphi(x) = \lambda(x, z(x), v(x)) : \mathbb{R}^q \mapsto \mathbb{R}$$

As a motivation for allowing z itself to occur as an argument of Λ and λ , it was observed in [28] that encoding the maximum in the nonsymmetric form

$$\max(u, \tilde{u}) = u + \frac{1}{2}[z + v] \quad \text{with} \quad z = \tilde{u} - u \quad (29)$$

generates in a repeated application to compute maxima of vectors matrices M and L that are quite sparse. In contrast the repeated application of the symmetric form

$$\max(u, \tilde{u}) = \frac{1}{2}[u + \tilde{u} + \text{abs}(z)] \quad \text{with} \quad z = \tilde{u} - u \quad (30)$$

gives us the matrix L in the standard form when $M \equiv 0$, i.e. v is not defined.

Given a code for evaluating (28), every AD tool for given \hat{x} and \hat{z} will be able to compute the partitioned Jacobian as follows

$$\frac{\partial[\Lambda - z, \lambda]}{\partial[x, z, v]} \equiv \begin{bmatrix} Z & M - I & L \\ a^\top & b^\top & c^\top \end{bmatrix} \in \mathbb{R}^{(s+1) \times (q+s+s)}.$$

Hopefully, this can be done in sparse matrix formats so it does not matter that M , and especially L may have many completely vanishing columns [29]. Therefore we obtain the piecewise linear

model

$$\begin{aligned} z &= \dot{z} + Z(x - \dot{x}) + M(z - \dot{z}) + L(|z| - |\dot{z}|) \\ &= \underbrace{(\dot{z} - Z\dot{x} - M\dot{z} - L|\dot{z}|)}_{=\dot{d}} + Zx + Mz + L|z|, \end{aligned} \quad (31)$$

$$\begin{aligned} \Delta\varphi(\dot{x}, x - \dot{x}) &= \dot{\varphi} + a^\top(x - \dot{x}) + b^\top(z - \dot{z}) + c^\top(|z| - |\dot{z}|) \\ &= \underbrace{(\dot{\varphi} - a^\top\dot{x} - b^\top\dot{z} - c^\top|\dot{z}|)}_{=\dot{\mu}} + a^\top x + b^\top z + c^\top |z|. \end{aligned} \quad (32)$$

where the constant shift $\dot{\mu}$ does not sometimes consider in the piecewise linear model $\Delta\varphi$ since we are doing optimization. Note that due to the triangularity structure of M and L we can unambiguously evaluate the piecewise linear functions $z(x)$ and $\Delta\varphi(\dot{x}, x - \dot{x})$. Besides, (31) allows us to solve the equality constraint in the following manner

$$(I - M - L\Sigma)z = \dot{d} + Zx \quad (33)$$

which implies

$$z = (I - M - L\Sigma)^{-1}(\dot{d} + Zx), \quad (34)$$

if we assume that x belongs to a fixed polyhedron P_σ or σ has a fixed sign of 1. Therefore one can uniquely represent every successive abs-linearization of the original model when we replace (34) in (32), i.e.

$$\Delta\varphi(\dot{x}, x - \dot{x}) = \dot{\mu} + a^\top x + b^\top (I - M - L\Sigma(z(x)))^{-1}(\dot{d} + Zx)z + c^\top \Sigma(I - M - L\Sigma)^{-1}(\dot{d} + Zx)z. \quad (35)$$

The advantages of these generalizations can be studied with the following trouble-maker example.

Example 4.1. For the composite piecewise differentiable function $\varphi : \mathbb{R}^s \rightarrow \mathbb{R}$ given by

$$\varphi(x) = \max_{0 \leq i \leq s} (a_i^\top x - \beta_i) \quad \text{with } a_i \in \mathbb{R}^n, \beta_i \in \mathbb{R} \quad \text{for } 0 \leq i \leq s,$$

the sparsity of the L matrix between the GALF and ALF is notorious as shown in the Appendix A.1.

It is of our interest to compute explicitly the active gradient of the piecewise linearization for optimization purposes. Therefore the gradient of the piecewise linear approximation for a fixed σ is

$$g_\sigma = a^\top + b^\top (I - M - L\Sigma)^{-1}Z + c^\top (\Sigma - M - L)^{-1}Z. \quad (36)$$

One can compare the similarity between the GALF and the ALF active gradients shown in (23). In the next sections, we use the GANF and thus the GALF of φ to minimize it accordingly the SALGO approach and its subsequent algorithms.

4.2. SALGO approach.

As we have discussed earlier, the "nonsmoothness" of the ER leads to a "house of horrors" for optimizers such as Stochastic Gradient and Gradient Descent. Those observations lead us to question whether the training could be improved. In this section, we propose our own approach to deal with this optimization task and the subsequent house of horrors. We call it SALGO which stands for Successive Abs-Linear Global Optimization.

The aim of SALGO is to solve the optimization problem

$$\min_{x \in \Omega} \varphi(x). \quad (37)$$

exploiting the objective function nonsmoothness. Let us recall the piecewise linearization form and its uniform quadratic error w.r.t. the original model. This observation relies on the inequality

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

where the piecewise linear approximation $\Delta\varphi(\hat{x}; x - \hat{x})$ and $\hat{\mu}$ are defined in (32) and (35). In this sense, we add up the quadratic error, which is from now on called proximal term, to the piecewise linearization $\Delta\varphi$ to construct the following function form.

Definition 4.1 (Prox-linear form). *The prox-linear form of a composite piecewise smooth function φ is given by*

$$\phi(\hat{x}, \Delta x) = g^\top x + \nu + \frac{\gamma}{2} \|\Delta x\|^2 \quad (38)$$

where g^\top is defined in (36) and

$$\nu = \hat{\mu} + b^\top (I - M - L\Sigma)^{-1} \hat{d} + c^\top \Sigma (I - M - L\Sigma)^{-1} \hat{d}$$

The goal of the prox-linear form is to smooth the original model on each P_σ where piecewise linearization was constructed. Since this procedure is possible in the whole decomposition of the objective function domain, one can compute succesively as many abs-linearizations and thus prox-linear forms as the optimization method requires to reach a global minimizer. We denote x_k the iterate in which the optimizers updates its search. Under this approach, we formulate the following iterative procedure that describes SALGO in essence.

$$x_{k+1} = x_k + \operatorname{argmin}_{\Delta x} \left\{ \Delta\varphi(\hat{x}; \Delta x) + \frac{\gamma_k}{2} \|\Delta x\|^2 \right\} = x_k + \operatorname{argmin}_{\Delta x} \left\{ \phi_k(\hat{x}; \Delta x) \right\}, \quad (39)$$

where ϕ_k is identified by its constant proximal term γ_k . The right choice of γ_k will determine if the minima of the novel approximation ϕ_k is close enough to the ones of φ_σ . We call this procedure SALGO. The next question arises wheter the optimization strategy over the prox-linear form is able to generate a finite sequence $\{x_k\}_{k \in \mathbb{N}}$ that has a cluster point x_* and all the cluster points are Clarke stationary or even better. We attack this task with two different search strategies: Mixed Integer Bilinear Optimization (MIBLOP), and Target Oriented Averaged Search Trajectory (TOAST). Here MIBLOP is the only strategy that can guarantee a global minimizer of the local model meanwhile TOAST is based on the empirical results.

5. Optimization strategies of SALGO.

5.1. Mixed Integer Linear Optimization (MILOP).

The minimization of a function in abs-linear form can be written as the task

$$\min a^\top x + b^\top z + c^\top \Sigma z \quad s.t. \quad z = d + Zx + Mz + L\Sigma z \quad \text{and} \quad \Sigma z \geq 0, \quad (40)$$

where the elements $\sigma_i \in \{-1, 1\}$ of the diagonal matrix $\Sigma = \text{diag}(\sigma)$ are binary variables. Equation (40) is a Mixed Integer Bilinear Optimization Problem (MIBLOP). It can be transformed [30] into a Mixed Integer Linear Optimization (**MILOP**), provided we have a uniform bound γ on the components of $|z|$. We may then even add a proximal term and get the MILOP

$$\min_{(x,z,v,\sigma) \in \mathbb{R}^{q,s,s,s}} \left(a^\top x + b^\top z + c^\top h + \frac{\gamma}{2} \|x\|^2 \right) \quad s.t. \quad z = Zx + Mz + Lh, \quad (41)$$

$$-h \leq z \leq h \quad \text{and} \quad h + \gamma(\sigma - e) \leq z \leq -h + \gamma(\sigma + e), \quad (42)$$

which can be solved by very effective modern solvers like Gurobi.

However, there is the slight problem that for learning we actually want to minimize not with respect to x but with respect to the coefficients

$$\alpha = (Z, M, L, J, N, Y) \in \mathbb{R}^{(s \times q, s \times s, s \times s, m \times q, m \times s, m \times s)} \simeq \mathbb{R}^{s(q+s-1)+m(q+2s)}.$$

Whereas in general we have only multi-piecewise linearity w.r.t. x , we have noted that in the single layer case with fixed weighting, this problem is also piecewise linear and it can be rewritten as the mixed bilinear programming problem

$$\begin{aligned} \min_x \varphi(x) &= \frac{1}{m} \sum_{k=1}^m u_k \quad s.t. \quad z_k = W\chi_k + b, \quad g_k = \frac{1}{2} p^\top (z_k + h_k) - y_k, \quad (43) \\ \mu_k g_k = u_k &\geq 0 \in \mathbb{R}, \quad \text{and} \quad h_k = \Sigma_k z_k \geq 0 \in \mathbb{R}^n. \end{aligned}$$

where the $\mu_k \in \{-1, 1\}$ and the diagonal elements $\sigma_k \in \{-1, 1\}^s$ of the matrices Σ_k are binary variables. The bilinear terms can be replaced in the usual fashion by the system of linear inequalities for $k = 1 \dots m$

$$\begin{aligned} -u_k \leq g_k \leq u_k &\quad \text{and} \quad u_k + \delta_k(\mu_k - 1) \leq g_k \leq -u_k + \delta_k(\mu_k + 1) \in \mathbb{R}, \\ -h_k \leq z_k \leq h_k &\quad \text{and} \quad h_k + \gamma_k(\sigma_k - e) \leq z_k \leq -h_k + \gamma_k(\sigma_k + e) \in \mathbb{R}^d. \end{aligned}$$

where $\delta_k \in \mathbb{R}$ must be an upper bound on the $g_k \in \mathbb{R}$ and $\gamma_k \in \mathbb{R}^s$ on all components of $z_k \in \mathbb{R}^s$. Here e represents the corresponding vector of ones. Explicitly, let us denote $i \in \{1, \dots, s\}$ and $k \in \{1, \dots, m\}$ where s and m are the numbers of intermediate values and of training sample points, respectively. Due to the equality constraints in (43), we obtain the upper bound $\gamma_{k,i}$ for each component of the vector z_k .

$$z_{k,i} \leq \|W^{(i)}\|_\infty \|\chi_k\|_1 + \|b\|_\infty =: \gamma_{k,i} \quad (44)$$

Finally, considering γ_k as the vector of $\gamma_{k,i}$, we define the upper bound δ_k of the real number g_k as follows

$$g_k \leq \|y\|_\infty + \|p\|_1 \|\gamma_k\|_\infty =: \delta_{k,i} \quad (45)$$

Preliminary results of this method were achieved using the AMPL interface (see, [subsection B.1](#)). We discuss the experimental implementation in section 6. Remarkly, in contrast to the following heuristic approach, MILOP is guaranteed to locate global minimizer of the piecewise linear model.

5.2. Target Oriented Averaging Search Trajectory (TOAST).

As a second optimization method we consider *the generalized descent* proposed in [4]. But we avoid that terminology in order to not invite association with the concept of generalized derivative. We interpret it like a search trajectory rather than a variant of steepest descent. Since this search is oriented to reach a target value and defined by an average of the steepest descent direction, we use the acronym TOAST referring to Target Oriented Averaging Search Trajectory.

First we introduce the mathematical derivation of TOAST from the calculus of variations. Let us consider two distinct points y_0, y_1 contained in a simply connected compact set $\Omega \subset \mathbb{R}^n$, and the set \mathcal{R} of all rectifiable curves on Ω , i.e. continuous curves with finite length. Let $y \in \mathcal{R}$ such that it is parametrized over the nonnegative real-valued interval $[\tau_0, \tau_1]$. We denote $y(\tau_0) = y_0$ and $y(\tau_1) = y_1$. We also assume that the real-valued objective function $\varphi \in \mathcal{C}^2(\mathbb{R}^n)$. Finally, we denote by

$$d \equiv \min_{v \in \Omega}(\varphi(v)) \text{ and } D \equiv \max_{v \in \Omega}(\varphi(v)),$$

$l(y)$ the length of the curve y and ds its infinitesimal measure. For $c \in \mathbb{R}$ such that $c < d$, we define the functional $q : \mathcal{R} \rightarrow \mathbb{R}$ as

$$q(y) = \int_{y_0}^{y_1} \frac{ds}{(\varphi(y(\tau)) - c)}.$$

Then q is continuous and positive, and the inequality:

$$\frac{l(y)}{(D - c)} \leq q(y) \leq \frac{l(y)}{(d - c)}$$

holds. Consequently, there exists an infimum

$$q_0 = \inf_{y \in \mathcal{R}} q(y)$$

and a minimizing sequence $y_j \in \mathcal{R}$ such that

$$q_0 = \lim_{j \rightarrow \infty} q(y_j).$$

We additionally normalize the independent parameter τ such that for all y_j we have $ds = l(y_j) \cdot d\tau$. In this sense, we can bound the length $l(y_j)$ as follows

$$q_1 = \max_{j \in \mathbb{N}} q(y_j) \geq q(y_j) \geq \frac{l(y_j)}{(D - c)}, \text{ for all } j.$$

Furthermore, y_j satisfies the uniformly Lipschitz condition over τ , i.e.

$$\|y_j(\tau + \Delta\tau) - y_j(\tau)\| \leq \int_{y_j(\tau)}^{y_j(\tau + \Delta\tau)} ds \leq l(y_j) \cdot \Delta\tau \leq \Delta\tau \cdot q_1(D - c).$$

Definition 5.1. A family of functions \mathcal{F} is uniformly equicontinuous if

$$\forall y \in \mathcal{R}, \forall \varepsilon > 0, \exists \delta > 0 : \|\tau_0 - \tau_1\| < \delta \Rightarrow \|y(\tau_0) - y(\tau_1)\| < \varepsilon$$

Theorem 5.2 (Arzela-Ascoli). Any uniformly bounded equicontinuous sequence of functions in $\mathcal{C}(I)$ has a subsequence that converges uniformly.

The Arzela-Ascoli theorem proves the existence of a limit curve $x = x(\tau)$, which is continuous with respect to τ , rectifiable, and almost everywhere differentiable. Let us assume that x is second order continuously differentiable. To make the x independent of affine transformations, we use an arbitrary invertible positive definite matrix \mathcal{H} to redefine its infinitesimal length so that

$$ds = (\dot{x}(\tau)\mathcal{H}^{-1}\dot{x}(\tau))^{\frac{1}{2}}d\tau, \quad \text{where} \quad \dot{x}(\tau) \equiv \frac{dx(\tau)}{d\tau}$$

For $0 \leq \tilde{\tau} \leq \tau \leq 1$, we then obtain

$$q(x) = \int_0^\tau \frac{(\dot{x}^\top(\tilde{\tau})\mathcal{H}^{-1}\dot{x}(\tilde{\tau}))^{\frac{1}{2}}}{\varphi(x(\tilde{\tau})) - c} d\tilde{\tau}$$

Therefore the following variational necessary condition holds replacing $F(\tilde{\tau}, x(\tilde{\tau}), \dot{x}(\tilde{\tau})) = \frac{(\dot{x}^\top(\tilde{\tau})\mathcal{H}^{-1}\dot{x}(\tilde{\tau}))^{0.5}}{\varphi(x(\tilde{\tau})) - c}$.

Proposition 5.3 (DuBois-Reymond equality). *If $x \in \mathcal{R}$ is an extremal of the functional q , then the following equality holds by first order variation:*

$$F_{\dot{x}}(\tau, x(\tau), \dot{x}(\tau)) = C + \int_0^\tau F_x(\tilde{\tau}, x(\tilde{\tau}), \dot{x}(\tilde{\tau}))d\tilde{\tau}$$

where $C \in \mathbb{R}$ is constant.

Because x is differentiable, the differential-integral equation below has a solution.

$$\frac{\mathcal{H}^{-1}\dot{x}}{(\varphi(x) - c)(\dot{x}^\top\mathcal{H}^{-1}\dot{x})^{\frac{1}{2}}} = \frac{\mathcal{H}^{-1}\dot{x}(0)}{(\varphi(0) - c)(\dot{x}_0^\top\mathcal{H}^{-1}\dot{x}_0)^{\frac{1}{2}}} - \int_0^\tau \frac{\nabla\varphi(\tilde{\tau})(\dot{x}(\tilde{\tau})^\top\mathcal{H}^{-1}\dot{x}(\tilde{\tau}))^{\frac{1}{2}}}{(\varphi(\tilde{\tau}) - c)^2} d\tilde{\tau}$$

Remark 5.4. *Here the subscripting of x and \dot{x} means partial differentiation w.r.t those variables. Also the functional F is given by*

$$F(\tilde{\tau}, x(\tilde{\tau}), \dot{x}(\tilde{\tau})) = \frac{(\dot{x}^\top(\tilde{\tau})\mathcal{H}^{-1}\dot{x}(\tilde{\tau}))^{\frac{1}{2}}}{\varphi(x(\tilde{\tau})) - c}.$$

Then its partial derivatives are

$$\begin{aligned} \frac{\partial F}{\partial x} &= -\frac{(\dot{x}^\top\mathcal{H}^{-1}\dot{x})^{-\frac{1}{2}}}{(\varphi(x) - c)^2} \nabla\varphi(x) \\ \frac{\partial F}{\partial \dot{x}} &= \frac{\mathcal{H}^{-1}\dot{x}}{(\varphi(x) - c)(\dot{x}^\top\mathcal{H}^{-1}\dot{x})^{\frac{1}{2}}} \\ \frac{\partial^2 F}{\partial \dot{x} \partial \dot{x}} &= \frac{1}{(\varphi(x) - c)(\dot{x}^\top\mathcal{H}^{-1}\dot{x})^{\frac{1}{2}}} \left(\mathcal{H}^{-1} - \frac{\mathcal{H}^{-1}\dot{x}\dot{x}^\top\mathcal{H}^{-1}}{\dot{x}^\top\mathcal{H}^{-1}\dot{x}} \right) \end{aligned}$$

Because $ds = l(x)d\tau = (\dot{x}^\top\mathcal{H}^{-1}\dot{x})^{\frac{1}{2}}d\tau$ we find that

$$\frac{\dot{x}}{(\varphi(x) - c)l(x)} = \frac{\dot{x}(0)}{(\varphi(0) - c)l(x)} - \mathcal{H} \int_0^\tau \frac{\nabla\varphi(x(\tilde{\tau}))l(x)}{(\varphi(x(\tilde{\tau})) - c)^2} d\tilde{\tau}. \quad (46a)$$

All functions on the right hand side are assumed to be differentiable. Therefore we obtain the second order ordinary differential equation

$$\frac{\ddot{x}}{(\varphi(x) - c)} - \frac{\dot{x}\dot{x}^\top \nabla \varphi(x)}{(\varphi(x) - c)^2} = -\frac{\mathcal{H} \cdot \nabla \varphi(x) \cdot l^2(x)}{(\varphi(x) - c)^2}. \quad (46b)$$

Finally, with a change of variable, we set:

$$t = (l(x))^{-1} \cdot \tau, \quad x' \equiv \frac{dx}{dt} = \dot{x} \cdot (l(\hat{x}))^{-1}, \quad x'' = \frac{d^2x}{dt^2} = \ddot{x} \cdot (l(\hat{x}))^{-2}.$$

This yields the equation

$$x'' = -(\mathcal{H} - x'x'^\top) \frac{\nabla \varphi(x)}{(\varphi(x) - c)}$$

Remark 5.5. Observe that $\frac{\partial^2 F}{\partial \dot{x} \partial \dot{x}}$ is positive semidefinite so that the Legendre condition is always satisfied.

Theorem 5.6 (Legendre's condition). *A necessary condition, for the functional*

$$q(x) = \int_0^1 F(t, x(t), \dot{x}(t)) dt, \quad x(0) = x_0, x(1) = x_1$$

to have a minimum evaluated on the curve $x \equiv x(t)$, is that the inequality

$$F_{\dot{x}, \dot{x}} \geq 0$$

be satisfied at every point of the curve. In other words, $F_{\dot{x}, \dot{x}} \equiv F_{x, \dot{x}}(t, x(t), \dot{x}(t))$ is a square matrix that must be positive semidefinite.

Thus no other additional condition on x is needed to satisfy the above minimal variational condition for q . The search trajectory x is the solution of the second order differential equation. We can define $\mathcal{H} = I$, i.e. the identity matrix in $\mathcal{M}^{n \times n}$. So

$$x''(t) = -(I - x'x'^\top(t)) \frac{\nabla \varphi(x(t))}{\varphi(x(t)) - c}, \quad \|x'(0)\| = 1 \quad (47)$$

for an arbitrary initial condition $(x(0), \dot{x}(0)) = (x_0, \dot{x}_0) \in \mathbb{R}^n \times \mathbb{R}^n$. Additionally, the unitary norm of $\dot{x}(t)$ is satisfied for all $t > 0$.

Remark 5.7. Afterwards the derivatives of the solution $x(t)$ w.r.t the time $t > 0$ will be denoted by the dot notation.

Here the method parameter c is the *target value* that one wants to reach from above at every specific stage of the optimization procedure. Besides, we define the *target level* as the closure of the level set defined by c

$$\partial L_c(\varphi) = \{x \in \mathbb{R}^n : x = \varphi^{-1}(c)\}. \quad (48)$$

so that the level set $L_c = \{x \in \mathbb{R}^n : x \leq \varphi^{-1}(c)\}$ is denoted as the *target sub-level* $\varphi^{-1}(-\infty, c)$. Let us denote by $t_c < \infty$ the supremum of all t for which $x(t)$ is defined. Then the trajectory must reach the target level L_c , i.e.

$$\lim_{t \rightarrow t_c} x(t) = x(t_c) = x_c \in \varphi^{-1}(c) \quad (49)$$

Otherwise the trajectory may be infinite and never even come close to the target level. As can be seen from (46a), the TOAST direction is given by an average of the steepest descent direction of the objective function $-\nabla\varphi(x(t))$ weighted by the reciprocal of the difference between the current evaluation $\varphi(x(t))$ and the real target value c . Actually, as long as the current evaluation of $\varphi(x(t))$ is high above the target value, the reciprocal will be small and thus the search direction will be more or less constant and hopefully ignore small local wiggles of the objective function. Once $\varphi(x(t))$ gets closer to c , the adjustment of \dot{x} towards the steepest descent direction will be more drastic. Provided that the gradient does not vanish, \dot{x} reduces to the steepest descent direction when $\varphi(x) - c$ tends to zero, i.e.

$$\lim_{t \rightarrow t_c} \dot{x} = -\frac{\nabla\varphi(x_c)}{\|\nabla\varphi(x_c)\|} \quad (50)$$

When $t_c < \infty$ we can define the extended system

$$\max[0, \varphi(x) - c]\ddot{x} + (I - \dot{x}\dot{x}^\top)\nabla\varphi(x) = 0 \quad (51)$$

for any initial point mentioned above (see, [4]).

Once the target value has been reached and if it is not satisfactory, it can be lowered to a more ambitious level. For the ANN training, since the ER $\varphi \geq 0$, we wish to get down as close to zero as possible. One may find the target value to be unattainable during the minimization calculation. Then one has to move the target value up towards the values that have already been attained during the current run. This adjustment of c is the main heuristic aspect of the proposed method and will be discussed later.

Most observations described above are still valid in the equation (47) when we add a new parameter called the *sensitivity parameter* $e > 0$, namely

$$\ddot{x}(t) = -e(I - \dot{x}\dot{x}^\top(t))\frac{\nabla\varphi(x(t))}{\varphi(x(t)) - c}, \quad \|\dot{x}(0)\| = 1, \quad (52)$$

where its integrated equivalent is

$$\frac{\dot{x}(t)}{[\varphi(x(t)) - c]^e} = \frac{\dot{x}_0}{[\varphi_0 - c]^e} - e \int_0^t \frac{\nabla\varphi(x(\tau))}{[\varphi(x(\tau)) - c]^{e+1}} d\tau. \quad (53)$$

In the original paper it was suggested that e should be selected as the reciprocal of the growth rate of $\varphi(x) - \varphi(\hat{x})$ in the catchments of its local minimizers \hat{x} such that

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

Hence we can think of $1/e$ as an average degree of positive homogeneity. This would suggest the choice $e = \frac{1}{2}$ for essentially quadratic and otherwise smooth functions, and $e = 1$ for functions with linear growth. We apply now this result when the objective function is piecewise smooth functions so that it can be abs-linearized.

5.2.1. Generalization to piecewise smooth functions.

Under conditions of machine learning the smoothness assumption on φ (see, [2]) is only likely to be satisfied on the generally nonlinear coincidence sets M_σ defined by the GANF of φ (see, Definition 2.17). The corresponding theoretical observations given in the last section need to be

spliced together. This would mean the implementation of the second-order differential equation (47) over each M_σ with $\varphi \in \mathcal{C}_{\text{abs}}^d(\mathbb{R}^n)$ and $d \geq 2$. We carry out this task taking advantage of the φ nonsmoothness through its successive abs-linearization on each M_σ .

This approach is based on the approximation of the gradient information of the original model ER by its GALF and, consequently, by its prox-linear form ϕ (as shown in (24)). This implies that the gradient of the right hand side satisfies that $\nabla\phi \in L_{\text{abs}}(\mathbb{R}^n, \mathbb{R}^n)$. Because \mathbb{R}^n is a finite dimensional space, we obtain that $\nabla\phi \in \mathcal{C}_{\text{abs}}^{\infty,1}(\mathbb{R}^n, \mathbb{R}^n)$. So each smooth piece of the prox-linear form in (38) is defined over each coincidence polyhedral set $M_\sigma = P_\sigma$ which are defined by different proximal terms γ_k and reference points \hat{x} . Therefore we use the TOAST formulation on the prox-linear form ϕ of the objective function φ . Here the sensitivity parameter $e = 1$ based on the positive homogeneity of the abs-linearization holds when x is close enough to the reference point \hat{x} . At this homogeneity neighborhood, the proximal term can be closed enough to zero so that the 2-degree homogeneity of $\|\Delta x\|^2$ would not affect the following characterization of the prox-linear form $\phi(\hat{x}; x) \equiv \Delta\varphi(\hat{x}, \Delta x) + \frac{\gamma}{2}\|\Delta x\|^2$:

$$\phi(\hat{x} + \rho(x - \hat{x})) - \phi(\hat{x}) = \rho [\phi(x) - \phi(\hat{x})] \quad \text{for } \rho > 0 \quad (54)$$

and

$$\phi(x) - \phi(\hat{x}) \sim \|x - \hat{x}\|. \quad (55)$$

Hopefully this characterization holds when the trajectory arrives at some boundary of P_σ where the piecewise linearization term was constructed. Here the distance $r(t) = \|x(t) - \hat{x}\|$ satisfies

$$\frac{\dot{r}(t)r(t)}{[\phi(x(t)) - c]^e} = \frac{\dot{r}_0 r_0}{[\phi_0 - c]^e} + \int_0^t \frac{(1-e)[\phi(x(\tau)) - \phi(\hat{x})] + (\phi(\hat{x}) - c)}{[\phi(x(\tau)) - c]^{e+1}} d\tau. \quad (56)$$

Since $e = 1$ it is clear that the right hand side is monotonically falling or growing if the constant $\phi(\hat{x}) - c$ is negative or positive, respectively. Thus we get the following result for any trajectory that moves at some time t towards \hat{x} in that $\dot{x}(t)^\top(\hat{x} - x_0) = 2\dot{r}(t)r(t) < 0$. Without loss of generality we may assume that this happens at the initial time $t = 0$.

Proposition 5.8 (convergence/divergence in homogeneous case).

Suppose (55) holds almost everywhere along the trajectory $x(t)$ defined by (52) with $e = 1$ and $\dot{r}_0 < 0$. Then $x(t)$ must approach the target sub-level $\phi^{-1}(-\infty, c)$ if $\phi(\hat{x})$ is desirable, i.e. at or below the target value c . If $\phi(\hat{x})$ is undesirable, i.e. above the target value c , then the trajectory either reaches the target value somewhere else or $r(t) = \|x(t)\|$ diverges monotonically towards infinity.

Proof. First let us consider the desirable case $\phi(\hat{x}) \leq c$. If the target level was not approached $\dot{r}(t)$ would by (56) be negative and bounded away from zero which leads to a contradiction since $r(t)$ cannot become negative. Hence we are left with the undesirable case $\phi(\hat{x}) > c$. If the feasible set is not approached then the trajectory $x(t)$ is well defined for all $t \geq 0$. If $r(t)$ was bounded above the integral on the right hand side of (56) would go to plus infinity. Hence $\dot{r}(t)r(t)$ in the numerator of the left hand side would also have to go to infinity, which is a contradiction. Hence $r(t)$ must be unbounded which means that its derivative $\dot{r}(t)$ at some t_1 and thus by (56) all subsequent $t \geq t_1$ must be positive. This constitutes together with the unboundedness monotonic divergence towards infinity, which completes the proof. \square

The last implies that, once the search trajectory $x(t)$ enters a ball of radius r_0 about some \hat{x} in which $\phi(x)$ is homogeneous with respect to \hat{x} in the sense of (54), $x(t)$ does reach the target

level if $\phi(\hat{x}) \leq c$ and otherwise it either leaves the ball altogether or finds another point in the ball below the target value. The last situation cannot arise if \hat{x} is a local minimizer of ϕ . Hence we see that local, homogeneous minimizers attract the trajectory if they are desirable and repulse it if they are undesirable, i.e. above the target level. With and without the local minimality of \hat{x} we have excluded the highly undesirable possibility of the trajectory circling within the ball forever without reaching the target. Of course this may still happen when ϕ does not satisfy the rather stringent homogeneity assumption (54). Nevertheless, since the prox-linear form is positive homogenous one can be optimistic that the selective behavior of the trajectory will work in many situations. The proximal term will effectively limit the search region. Since $e = 1$ makes the trajectory turn back towards x_0 when the proximal term dominates the piecewise linear components. Thus, barring severe degeneracies, the TOAST philosophy can still be applied in the piecewise-smooth case. Of course, the finitely many transitions must be handled numerically.

Finally, the prox-linear case also gives an analytical observation about the target level to be reached. Since the equality $\phi(x) = c$, i.e.

$$g^\top x + \nu + \frac{\gamma}{2} \|x - \hat{x}\|^2 = c,$$

must hold, where \hat{x} is the reference point, we multiply by $\frac{2}{\gamma}$ and reorder the terms to obtain

$$x^\top (x + \frac{2}{\gamma}g - 2\hat{x}) = \frac{2}{\gamma}(c - \nu) - \|\hat{x}\|^2.$$

Adding up $\|\frac{1}{\gamma}g - \hat{x}\|^2$ in both sides,

$$x^\top (x + \frac{2}{\gamma}g - 2\hat{x}) + \|\frac{1}{\gamma}g - \hat{x}\|^2 = \frac{2}{\gamma}(c - \nu) - \|\hat{x}\|^2 + \|\frac{1}{\gamma}g - \hat{x}\|^2.$$

The equality above is equivalent to

$$\|x + \frac{1}{\gamma}g - \hat{x}\|^2 = \frac{2}{\gamma}(c - \nu) - \|\hat{x}\|^2 + \|\frac{1}{\gamma}g - \hat{x}\|^2. \tag{57}$$

Thus the target level is a circle with radius equal to the squared root of the left hand side in (57). Notice that it can be empty if the root is imaginary.

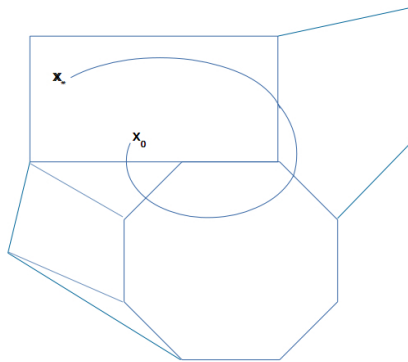


Figure 6: TOAST graph defined by the prox-linear form $\phi : \mathbb{R}^2 \rightarrow \mathbb{R}$ starting at x_0 and ending at x_* . Here x_* can be a local minimum above the target value c or a point of the target level $\varphi^{-1}(c)$. The disjunctive situation relies on the condition of desirability of $\varphi(\hat{x})$ w.r.t. c .

5.2.2. Numerical integration of TOAST given the prox-linear form.

Let us implement the numerical integrator of Mathematica 11.3 to solve our initial value problem (47) w.r.t. the prox-linear form (see, 97) for the two-dimensional case, i.e.

$$\phi(x_1, x_2) = [g_1, g_2]^T(x_1, x_2) + \nu + \frac{\gamma}{2}(x_1^2 + x_2^2),$$

with an arbitrary selection of g, ν, c and x denoted by $(x_1, x_2) \in \mathbb{R}^2$. So the initial value problem is rewritten as the following system

$$\ddot{x}_1 = -\left(\frac{1}{h}\right)(\dot{x}_2^2(g_1 + \gamma x_1) - \dot{x}_1 \dot{x}_2(g_2 + \gamma x_2)) \tag{58}$$

$$\ddot{x}_2 = -\left(\frac{1}{h}\right)(\dot{x}_1^2(g_2 + \gamma x_2) - \dot{x}_1 \dot{x}_2(g_1 + \gamma x_1)) \tag{59}$$

for the initial condition $(x(0), \dot{x}(0)) = (x_0, \dot{x}_0)$, where $h \equiv h(x_1, x_2) = \phi(x_1, x_2) - c$.

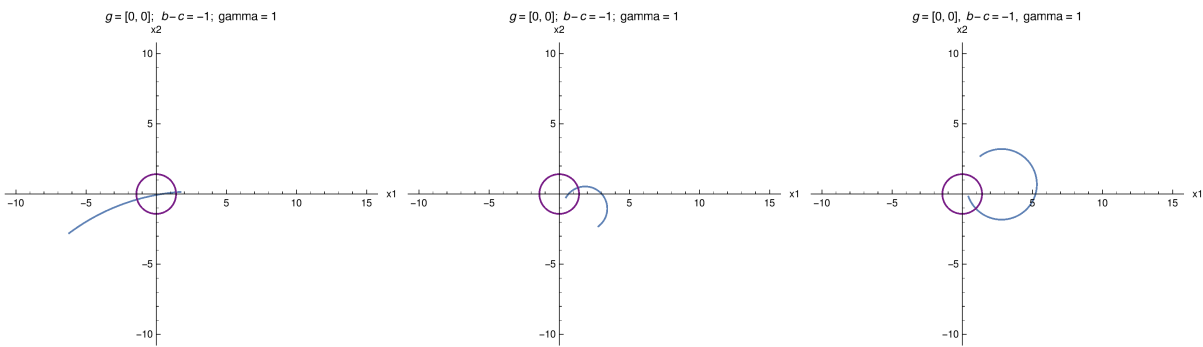


Figure 7: Here $g = [0, 0], \gamma = 1$ and $b - c = -1$. From left to right the initial conditions are: $(x_1(0), x_2(0)) = (-\frac{\sqrt{2}}{2} - 5.5, \frac{\sqrt{2}}{2} - 3.5), (-\frac{\sqrt{2}}{2} + 3.5, \frac{\sqrt{2}}{2} - 3.01), (-\frac{\sqrt{2}}{2} + 2, \frac{\sqrt{2}}{2} + 2)$

The figures above show no problems w.r.t. the trajectory and its goal to reach the target level. We can say that the selection of the initial conditions allows the trajectory to start in the homogeneity neighborhood. Further, since the minimum of this prox-linear form is given by $\gamma(x_1, x_2) = (0, 0)$, then the origin is its actual minimizer. Since $x_* = (0, 0)$ is desirable, i.e. $x_* \leq \varphi^{-1}(c)$, the numerical simulation of the trajectory satisfies the convergence proposition described in the last subsection. Indeed, the results suggest that $x(t)$ can be circular segments.

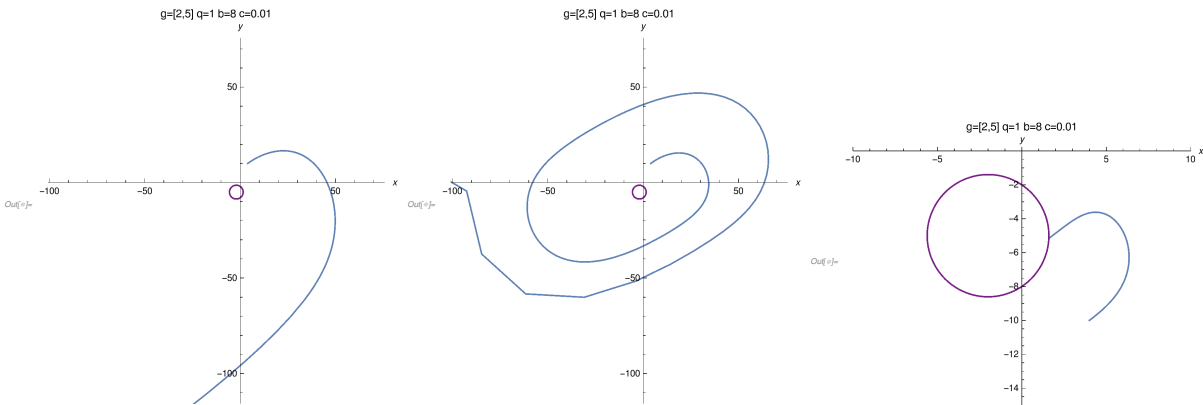


Figure 8: Here $g = (2, 5), \nu = 8, c = 0.01$ and $\gamma = 1$. From left to right the initial conditions are: $(x_1(0), x_2(0)) = (4, 10), (4, 10), (4, -10)$

Unlike the last prox-linear form, we obtain with a nonzero g more difficulties to recognize. Here the minimum is given by $x_* = -g$ which is desirable as the right hand side in (57) is 13. However, this would imply a contradiction between the first two results and the last one. In this case, the numerical integration seems worthless as their initial conditions are symmetric w.r.t. x_1 -axis. This symmetry would suggest that the homogeneity neighborhood includes both initial conditions, which implies that the trajectory should converge to the target level.

Theoretically, the cases mentioned in the convergence proposition are studied further in [4]. As we will see in the following subsection the exact solutions are circle trajectories so that the numerical integration results appear worthless. This conviction is due to the failing on simple prox-linear formulation. So far we just assumed global smoothness in the search space which is not our requirement for the successive prox-linear forms.

Remark 5.9. *The lilac contour is the circular target level given in (57). All the experiments have the initial tangent $\dot{x}_0 = (0.8, 0.6)$, which has unit norm.*

5.2.3. Exact solution of TOAST for the prox-linear case.

In a first stage, we construct TOAST for the novel approximation of the piecewise smooth real valued function φ . Since the piecewise linear model $\Delta\varphi(\hat{x}, \Delta x) \in \mathcal{C}_{\text{abs}}^{1,1}(\mathbb{R}^n)$, then its prox-linear form $\phi \in \mathcal{C}_{\text{abs}}^{\infty,1}(\mathbb{R}^n)$ due to the proximal term $\frac{\gamma}{2}\|\Delta x\|^2$. In this sense, we satisfy the assumption of smoothness in TOAST formulation in its polyhedral domain. The following theorem gives the exact form of the search trajectory saying that the prox-linear form is our new objective function.

Theorem 5.10. *With x_0, \dot{x}_0 and $\phi(x)$ is defined by the prox-linear form as shown in (38), and a constant target value $c < \phi_0$, the solution of the initial-value problem in (47) is given by the circle segment*

$$x(t) = x_0 + \frac{\sin(\omega t)}{\omega} \dot{x}_0 + \frac{1 - \cos(\omega t)}{\omega^2} \ddot{x}_0 \quad (60)$$

where

$$\ddot{x}_0 = -\frac{1}{h_0}(g + \gamma x_0 - g^\top \dot{x}_0 \dot{x}_0 - \gamma \dot{x}_0 x_0 \dot{x}_0) \quad (61)$$

and

$$h(x) = \phi(x) - c \text{ and } \omega = \|\ddot{x}_0\| \quad (62)$$

Proof. We claim that the ODE

$$h\ddot{x} = -g - \gamma x + \dot{x}^\top g \dot{x} + \gamma(\dot{x}^\top x) \dot{x} \quad (63)$$

has solution with formula shown in (60). First, the ansatz with trigonometric form holds the initial conditions

$$\dot{x}(t) = \cos(\omega t) \dot{x}_0 + \frac{\sin(\omega t)}{\omega} \ddot{x}_0, \quad \dot{x}(0) = \dot{x}_0, \quad (64)$$

$$\ddot{x}(t) = \cos(\omega t) \ddot{x}_0 - \omega \sin(\omega t) \dot{x}_0, \quad \ddot{x}(0) = \ddot{x}_0. \quad (65)$$

One can also verify the orthogonality between the initial tangent and the curvature of the trajectory $x(t)$

$$\ddot{x}_0^\top \dot{x}_0 = 0 \quad (66)$$

from the assumption on \ddot{x}_0 (see, (61)).

This assumption additionally implies that

$$\omega^2 h_0^2 = -(\gamma x_0^\top \dot{x}_0 + g^\top \dot{x}_0)^2 + \|g\|_2^2 + \gamma^2 \|x_0\|_2^2 + 2\gamma g^\top x_0 \quad (67)$$

and

$$-\omega^2 h_0 = g^\top \ddot{x} + \gamma x_0^\top \ddot{x}_0. \quad (68)$$

Now we manipulate the right hand side of the equality in (63). Using (65) and (66), the term $\dot{x}^\top g \dot{x}$ is rewritten as

$$\dot{x}^\top g \dot{x} = \left(\frac{\sin^2(\omega t)}{\omega^2} g^\top \ddot{x}_0 + \frac{\cos(\omega t) \sin(\omega t)}{\omega} g^\top \dot{x}_0 \right) \ddot{x}_0 + \left(\frac{\sin(\omega t) \cos(\omega t)}{\omega} g^\top \ddot{x}_0 + \cos^2(\omega t) g^\top \dot{x}_0 \right) \dot{x}_0. \quad (69)$$

The same replacement in \dot{x} but in $\gamma \dot{x}^\top x \dot{x}$ and using (67) to obtain

$$\begin{aligned} \gamma \dot{x}^\top x \dot{x} = & \gamma \left(\frac{-(1-\cos(\omega t)) + \sin^2(\omega t)}{\omega^2} \ddot{x}_0^\top x_0 + \frac{\sin(\omega t) \cos(\omega t)}{\omega} \dot{x}_0^\top x_0 + \frac{\sin^2(\omega t)}{\omega^2} \right) \ddot{x}_0 \\ & + \gamma \left(\frac{\sin(\omega t) \cos(\omega t)}{\omega^2} \ddot{x}_0^\top x_0 + \cos^2(\omega t) \dot{x}_0^\top x_0 - \frac{\sin(\omega t)(1-\cos(\omega t))}{\omega} \right) \dot{x}_0. \end{aligned} \quad (70)$$

Now the left hand side in (63) is equivalent to

$$h\ddot{x} = (g^\top x + \nu - c + \frac{\gamma}{2} \|x\|_2^2) (\cos(\omega t) \ddot{x}_0 - \omega \sin(\omega t) \dot{x}_0) \quad (71)$$

where

$$\|x\|_2^2 = \|x_0\|_2^2 + 2 \left(\frac{1-\cos(\omega t)}{\omega^2} + \frac{\sin(\omega t)}{\omega} x_0^\top \dot{x}_0 + \frac{1-\cos(\omega t)}{\omega^2} x_0^\top \ddot{x}_0 \right). \quad (72)$$

However, (71) and (72) imply that

$$\begin{aligned} h\ddot{x} = & \left(h_0 \cos(\omega t) + \frac{\sin(\omega t) \cos(\omega t)}{\omega} g^\top \dot{x}_0 + \frac{(1-\cos(\omega t)) \cos(\omega t)}{\omega^2} g^\top \ddot{x}_0 \right) \ddot{x}_0 \\ & - \left(h_0 \omega \sin(\omega t) + \sin^2(\omega t) g^\top \dot{x}_0 + \frac{\sin(\omega t)(1-\cos(\omega t))}{\omega} g^\top \ddot{x}_0 \right) \dot{x}_0 \\ + \gamma & \left(\frac{(1-\cos(\omega t)) \cos(\omega t)}{\omega^2} + \frac{\sin(\omega t) \cos(\omega t)}{\omega} x_0^\top \dot{x}_0 + \frac{(1-\cos(\omega t)) \cos(\omega t)}{\omega^2} x_0^\top \ddot{x}_0 \right) \ddot{x}_0 \\ - \gamma & \left(\frac{(1-\cos(\omega t)) \sin(\omega t)}{\omega} + \sin^2(\omega t) x_0^\top \dot{x}_0 + \frac{\sin(\omega t)(1-\cos(\omega t))}{\omega} x_0^\top \ddot{x}_0 \right) \dot{x}_0. \end{aligned} \quad (73)$$

From (69),(70),(73) we simplify and obtain the following equality for (63)

$$\begin{aligned} -g - \gamma x_0 = & -\gamma \left(\frac{1-\cos(\omega t)}{\omega^2} x_0^\top \ddot{x}_0 \right) \ddot{x}_0 - \gamma \left(x_0^\top \ddot{x}_0 + \frac{\sin(\omega t)}{\omega} x_0^\top \dot{x}_0 \right) \dot{x}_0 \\ & + \left(h_0 \cos(\omega t) - \frac{1-\cos(\omega t)}{\omega^2} g^\top \ddot{x}_0 \right) \ddot{x}_0 - \left(h_0 \omega \sin(\omega t) + g^\top \dot{x}_0 + \frac{\sin(\omega t)}{\omega} g^\top \ddot{x}_0 \right) \dot{x}_0. \end{aligned} \quad (74)$$

By assumption over \ddot{x}_0 in (61), the equality above turns to be

$$\begin{aligned} h_0 \ddot{x}_0 = & -\gamma \left(\frac{1-\cos(\omega t)}{\omega^2} x_0^\top \ddot{x}_0 \right) \ddot{x}_0 - \gamma \left(\frac{\sin(\omega t)}{\omega} x_0^\top \dot{x}_0 \right) \dot{x}_0 \\ & + \left(h_0 \cos(\omega t) - \frac{1-\cos(\omega t)}{\omega^2} g^\top \ddot{x}_0 \right) \ddot{x}_0 - \left(h_0 \omega \sin(\omega t) + \frac{\sin(\omega t)}{\omega} g^\top \ddot{x}_0 \right) \dot{x}_0. \end{aligned} \quad (75)$$

Reordering and dividing by h_0 ,

$$(1-\cos(\omega t)) \ddot{x}_0 = -\omega \sin(\omega t) \dot{x}_0 - \left(\frac{1-\cos(\omega t)}{\omega^2 h_0} g^\top \ddot{x}_0 + \gamma \frac{1-\cos(\omega t)}{\omega^2 h_0} x_0^\top \ddot{x}_0 \right) \ddot{x}_0 - \left(\frac{\sin(\omega t)}{\omega h_0} g^\top \ddot{x}_0 + \gamma \frac{\sin(\omega t)}{\omega h_0} x_0^\top \dot{x}_0 \right) \dot{x}_0 \quad (76)$$

Using (68), the equality (76) is equivalent to

$$(1 - \cos(\omega t)) \ddot{x}_0 = -\omega \sin(\omega t) \dot{x}_0 + (1 - \cos(\omega t)) \ddot{x}_0 + \omega \sin(\omega t) \dot{x}_0 \Leftrightarrow \vec{0} = \vec{0} \quad (77)$$

In other words, for any initial condition (x_0, \dot{x}_0) , (76) is satisfied by our ansatz. Thus our trigonometric solution solves the initial-value problem. \square

The difficult situations arise: when we hit two or more faces at the same time, i.e. two or more $z_i(x(t))$ want to change sign at the same time; or when the trajectory has transitions between polyhedra with a zero transition angle to the common polyhedral boundary. Though our preliminary formulation just assume any of these cases do not occur. Under these conditions called unique transversality, we ratify the uniqueness of TOAST. Also one can prove that the union of each piece of the trajectory is $\mathcal{C}_{\text{abs}}^{\infty,1}(\mathbb{R}^n)$. The latter is denoted by $\{\phi_k\}_{k=1}^{K<\infty}$ and called the prox-abs-linear form.

A consequence of the proposition is the circle-shaped search trajectory given by its constant distant with respect to the following center

$$\|x(t) - \left(x_0 - \frac{1}{\omega^2}\ddot{x}_0\right)\| = \frac{1}{\omega}. \quad (78)$$

Here $x(t)$ takes itself $t = \frac{2\pi}{\omega}$ units to complete a loop. Notice that if x_0 is colinear to the gradient $g + \gamma x_0$, then ω, \ddot{x}_0 vanish and the circle degenerates to the straight line $x(t) = x_0 + t\dot{x}_0$. Otherwise the function value along the circle trajectory is given by

$$\varphi(x(t)) = \varphi_0 + \frac{\sin(\omega t)}{\omega} \left[(g + \gamma x_0)^\top \dot{x}_0 \right] + \frac{1 - \cos(\omega t)}{\omega^2} \left[\gamma - \omega^2(\varphi_0 - c) \right] \quad (79)$$

where the trigonometric quotients on the right hand side reduce to t and $\frac{1}{2}t^2$ in the straightline case $\omega^2 = 0$.

5.3. SALGO-TOAST algorithm.

We call SALGO-TOAST algorithm to the implementation of SALGO approach described in (39) and the search strategy proposed in TOAST for the prox-linear case. Each task is formulated in the outer and inner loop of the algorithm. A remarkable fact of its inner loop is that we should generate a strategy to compute the most nearby polyhedra boundary that TOAST intersects under the suitable conditions mentioned at the end of the last subsection. The following pseudo code describes the two loops of the algorithm:

1. Form piecewise linearization $\Delta\varphi$ of objective φ at the current iterate \hat{x} and estimate the proximal coefficient γ , set $x_0 = \hat{x}$,
2. Set the initial tangent \dot{x}_0 and $\sigma = \text{sgn}(z(x_0))$.
3. Compute and follow circular segment $x(t)$ in P_σ .
4. Determine minimal t_* where $\varphi(x(t_*)) = c$ or $x_* = x(t_*)$ lies on the boundary of P_σ with some $P_{\tilde{\sigma}}$.
5. If $\varphi(x_*) \leq c$, lower c or go to step (1) with $\hat{x} = x_*$ or terminate.
6. Else, set $x_0 = x_*$, $\dot{x}_0 = \dot{x}(t_*)$, $\sigma = \tilde{\sigma}$ and continue with step (3).

Step (1) represent the successive computation of the prox-linear forms which result in the prox-abs-linear form $\{\phi_k\}_{k=1}^{K<\infty}$ on its union. If the starting point x_0 lies in the interior of a polyhedron and one chooses any target value c , the resulting circle segment will usually leave the polyhedron unless it reaches the target level or stays inside P_σ completing the circle. In this section we will address the problem of distinguishing these three cases in step (5) and in particular computing the value of t for which the circle leaves the current polyhedron. This procedure is called the

inner loop of the TOAST strategy.

Under the assumption of Theorem 5.10 with $\Sigma = \text{diag}(\sigma)$ definite so that $\Sigma = \Sigma^{-1}$, we know that $x(t) \in P_\sigma$ exactly as long as the s components of

$$z(t) = d + Zx(t) + Mz(t) + L\Sigma z(t) \iff z(t) = (I - M - L\Sigma)^{-1}(d + Z(x(t)))$$

do not change their sign. Substituting (60) for $x(t)$ and multiplying by Σ we get

$$\Sigma z(t) = \bar{z} + \hat{z} \frac{\sin(\omega t)}{\omega} + \tilde{z} \frac{(1 - \cos(\omega t))}{\omega^2} \quad (80)$$

$$= \bar{z} + t\hat{z} + \frac{1}{2}t^2\tilde{z} + O(\omega^2 t^3) = \bar{z} + t\hat{z} + O(\omega t^2) \quad (81)$$

$$\geq \bar{z} - t|\hat{z}| + \frac{1}{2}t^2 \min(0, \tilde{z}) \quad (82)$$

where

$$\bar{z} = (\Sigma - M\Sigma - L)^{-1}(d + Zx_0) \geq 0,$$

$$\hat{z} = (\Sigma - M\Sigma - L)^{-1}(Z\dot{x}_0),$$

$$\tilde{z} = (\Sigma - M\Sigma - L)^{-1}(Z\ddot{x}_0) = O(\omega).$$

In principle we now have to calculate for each $i = 1 \dots s$ the bound

$$t_i = \sup\{0 \leq t : \sigma_i z_i(t) \geq 0\} \in [0, \infty] \quad (83)$$

so that $\sigma_i z_i(t)$ is nonnegative on the interval $[0, t_i]$ $i = 1 \dots s$. Then we obtain the limiting step size as the minimum

$$t_* = t_{i_*} = \min_{0 \leq i \leq s} t_i \in [0, \infty].$$

where we have included the value t_0 at which the target c is or would be reached as discussed in the previous subsection. Notice that the trajectory must really leave P_σ and cannot just tangentially graze the boundary. In that case $\sigma_i z_{i_*}(t)$ would stay nonnegative in the neighborhood of t_* so that t_{i_*} would be bigger than t_* in contradiction to its definition.

Theoretically t_* can be infinite when $\omega = 0$ or satisfy the following equalities

$$t_i = \begin{cases} -\bar{z}_i/\hat{z}_i & \text{if } \sigma_i \bar{z}_i > 0 > \sigma_i \hat{z}_i \\ 0 & \text{if } \sigma_i \bar{z}_i = 0 > \sigma_i \hat{z}_i \\ \infty & \text{if } \sigma_i \bar{z}_i = 0 \leq \sigma_i \hat{z}_i \end{cases}.$$

If $\omega > 0$ and $t_* \geq 2\pi/\omega$ the full circle is contained in the current polyhedron. That can only happen if we start in the interior rather than entering transversally. In any case the trajectory must then circle around the unconstrained minimizer

$$x_* = -g/\gamma \quad \text{with} \quad \varphi(x_*) = \varphi_0 - \frac{1}{2\gamma} \|g + \gamma x_0\|^2 = \mu - \frac{1}{2\gamma} \|g\|^2. \quad (84)$$

Even though this local minimizer must lie above the target level it would again seem natural to go to it and restart with a new search direction and possibly lower target level.

To compute the exit value t_* defined in (83) we need to solve a sequence of trigonometric equations starting with (79) and continuing with (80). The current estimate \tilde{t}_* can be lowered by each new t_i but this needs only computed at all if the lower bound (81) is negative at the current

\tilde{t}_* . In practice this means that the following somewhat expensive solution of the trigonometric equations is only applied very rarely.

The exact solution τ_i of the equation $z_i(\tau/\omega) = 0$ is the nonnegative root of the equation

$$0 = \frac{(\bar{z}_i + \tilde{z}_i)}{\sqrt{\hat{z}_i^2 + \tilde{z}_i^2}} + \frac{(\hat{z}_i \sin(\tau_i))}{\sqrt{\hat{z}_i^2 + \tilde{z}_i^2}} - \frac{(\tilde{z}_i \cos(\tau_i))}{\sqrt{\hat{z}_i^2 + \tilde{z}_i^2}}$$

which is equivalent to

$$\gamma_i \equiv \frac{(\bar{z}_i + \tilde{z}_i)}{\sqrt{\hat{z}_i^2 + \tilde{z}_i^2}} = \cos(\tau_i + \delta_i)$$

where $\delta_i \in (-\pi, \pi]$ is uniquely defined by

$$\sin(\delta_i) = \frac{\hat{z}_i}{\sqrt{\hat{z}_i^2 + \tilde{z}_i^2}} \quad \text{and} \quad \cos(\delta_i) = \frac{\tilde{z}_i}{\sqrt{\hat{z}_i^2 + \tilde{z}_i^2}} .$$

In the C math library this angle can be computed directly as $\delta_i = \mathbf{atan2}(\hat{z}_i, \tilde{z}_i)$. There exists a first positive root τ_i if and only if $|\gamma_i| \leq 1$. Then we can compute

$$\tau_i = \tilde{\tau}_i - \delta_i \quad \text{with} \quad \tilde{\tau} = \arccos(\gamma_i) \in [0, \pi] .$$

To actually obtain the smallest positive τ_i we may have to make some adjustments. Because the cosine is an even function, the only other solution in $[-\pi, \pi]$ is given by the smaller $\tau_i = -\delta_i - \tilde{\tau}_i$, which may still be greater than δ_i . On the other hand the next larger solution is given by $\tau_i = 2\pi - \tilde{\tau}_i - \delta_i$. In other words we have to take τ_i as the first positive value of the ordered sequence

$$-\delta_i - \tilde{\tau}_i \leq \tilde{\tau}_i - \delta_i \leq 2\pi - \tilde{\tau}_i - \delta_i .$$

Hence to check out the first positive root τ_i of $z_i(\tau/\omega) = 0$ we have to evaluate one square root and two inverse trigonometric functions, a considerable amount of work, which rarely needs to be performed.

We also have to remember the i_* representing the last index $i \in \{0 \dots s\}$ for which the minimal value $\tilde{\tau}_*$ was decremented to τ_* , which means that the switching variable z_{i_*} and thus σ_{i_*} changes sign in the transition to the neighboring polyhedron. The corresponding exit time is of course given by $t_* = \tau_*/\omega$. We may then switch into the next polyhedron P_σ defined by $\sigma = \sigma - 2e_{i_*}\sigma_{i_*}$. Even if τ_* is attained at several indices the new polyhedron will be different and resetting t to zero we can progress to the new starting point

$$\begin{aligned} x_0 &= x_0 + (\ddot{x}_0 + \omega \sin(\tau_*)\dot{x}_0 - \cos(\tau_*)\ddot{x}_0) / \omega^2 \\ \dot{x}_0 &= \cos(\tau_*)\dot{x}_0 + \sin(\tau_*)\ddot{x}_0 / \omega . \end{aligned}$$

Moreover, we have to update the absolute term ν and the gradient g according to the formulae (32) and (36). Note that the new $\Sigma = \text{diag}(\sigma)$ differs from its previous value only in the sign of one of its diagonal elements so that we have a rank-one change that is extremely sparse. Hence ν and g can be updated cheaply using low rank linear algebra techniques. After that \ddot{x}_0 and its norm ω can be computed according to the formula (47). Notice that the sequential computing of g through the (generalized)abs-linear form is equivalent to the backpropagation algorithm [31]. The big difference is the consideration and procedures after its computation like the construction of the prox-abs-linear form and its optimization. Also, the computational cost of g is the same

as the function evaluation of the ER when its dimensionality and the intermediate (or switching) values satisfy certain relation described in [29].

Note that all computations are at most of order $(n+s)^2$, whereas the local minimizer SALMIN involves matrix factorizations with a computational cost of order $(n+s)^3$. On the other hand SALGO is strongly dependent on the Euclidean norm and thus significantly effected by variable rescalings or more general linear transformations. Via the proximal term $\frac{\gamma}{2}\|x\|^2$ the current version of SALMIN is also effected and furthermore the pivot selection is like in any active set type method scaling dependent. Fortunately, the weights and shifts in a neural network seem to have a natural and quite homogeneous scaling.

6. Experimental Results.

6.1. TOAST path.

The next experiment shows how TOAST is generated by SALGO-TOAST algorithm for the training of the prototypical two-dimensional Rectifier ANN described in (9). As the parameter space is \mathbb{R}^2 , it is possible to plot the ER surface. Here the loss function is the l_1 -norm. That is the sum of the absolute discrepancy values between the predictions and the labels of the training sample.

At each iteration the algorithm produces an update of the parameters in which the predictor evaluates. Thus SALGO-TOAST algorithm produces a sequence of iteratives. Along each evaluation, a path over the surface is traced according to the TOAST circle segments (see, (79)). The TOAST capacity of searching is notorious as the path "climbs" or "falls" in order to look for the locally minimal value of the prox-abs-linear form. We should recall that each iterate depends on the imposed target value c of search.

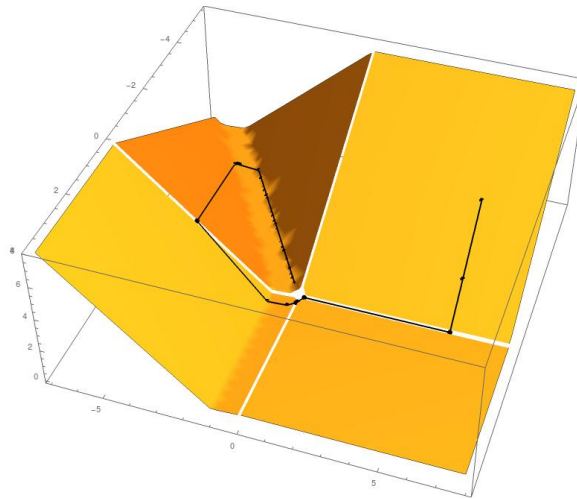


Figure 9: Nonlinear regression of the univariate Griewank function. Here the training sample is $\mathcal{D}_{\text{TRAIN}} = ((-1.33005, -1.1395), (1.39886, -0.016524))$ or just univariate Griewank function evaluations.

The starting point is located at the fourth quadrant of the Cartesian plane. There the objective function is evaluated and the target value is set to that evaluation. This first target is reached in one steepest descent step still within the initial polyhedron. Again the target is halved so the search continues till the edge of the zero-plateau. From there the trajectory reaches a point closed to a corner of intersection of polyhedra. Hopefully the trajectory would not be affected and still holding the unique transversality condition. The outcome over this plateau is a circle segment. However, our drawing tool connects the two iterates by a straight line. The next iterates come close to the global minimizer but passed it by, due to the ambitious target level imposed by the target value. The latter means that the target is unachievable because it is empty or the other cases analyzed in Proposition 5.8. After some iterates, the trajectory leaves the vicinity of the global minimizer and climbs up a hill until it enters into the neighbor valley. There it cannot find lower values of the last lowest one reached. Finally it returns in a meandering fashion towards the vicinity of the global minimizer.

This behavior shows clearly that our successive halving strategy needs modification so that the target is gradually moved upwards when it is not achievable over a long stretch of the search trajectory. Also the ER surface describes consistently the house of horrors of the learning problem (see, section 3.1).

6.2. Regression of Griewank function.

We simply minimize the averaged losses $|f(w; x) - y(x)|$ defined in (26) for the Griewank function [4]

$$y = c(\chi) = 1 + \frac{1}{40000} \sum_{k=1}^n \chi_k^2 - \prod_{k=1}^n \cos\left(\frac{\chi_k}{\sqrt{k}}\right)$$

over m different sample points $\chi \in \mathbb{R}^n$ chosen uniformly at random in the cube $[-8, 8]^n$. The results reported are for $n = 4$, the number of nodes $d = 10$ and $m = 20$ sample points. A comparison is done among the optimizers based on steepest descent (SD), stochastic gradient (SG), SALGO-TOAST algorithm, and MIBLOP. The next figures represent numerical results showing the ER evaluations with the same initial parameter setup.

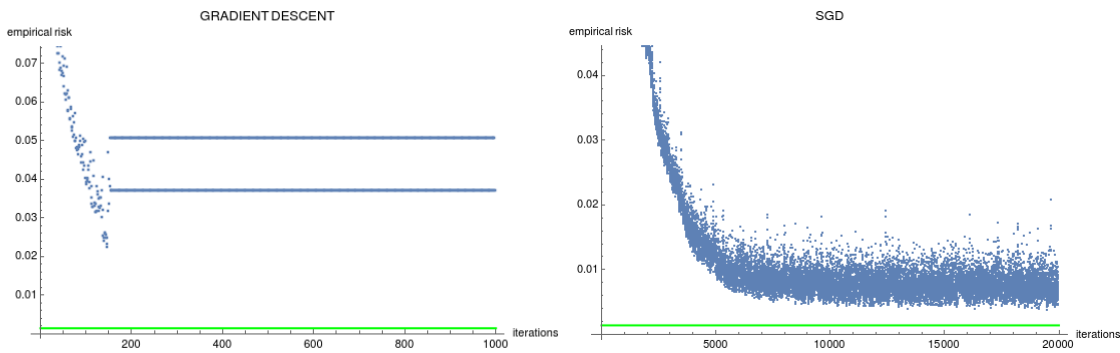


Figure 10: Nonlinear regression of the forth–th order Griewank function through steepest descent and Stochastic Gradient.

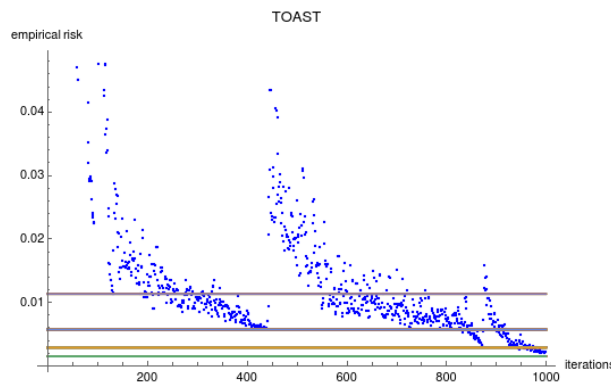


Figure 11: SALGO-TOAST implementation

In figures 10(a), 10(b), 11 we let TOAST and steepest descent run for a 1000 steps and stochastic gradient for 20000. Both steepest descent and stochastic gradient were run with a constant stepsize, which clearly is supoptimal. In Figure 10(a), the steepest descent method

got stuck after some 150 steps, oscillating between values of about 0.038 and 0.06. The stepsize was chosen as 0.003 also for stochastic gradient. The latter method reaches its lowest values of about 0.006 after about 10000 steps. TOAST reaches a lower value of about 0.002 at the end of the 1000 steps, when it still is going down. The horizontal lines show the target levels obtained by halving the target value reached. It is noteworthy that after the later halvings the TOAST trajectory does some significant hill climbing to eventually reach the lower values required by the new target. In this sense we have a true global optimization method.

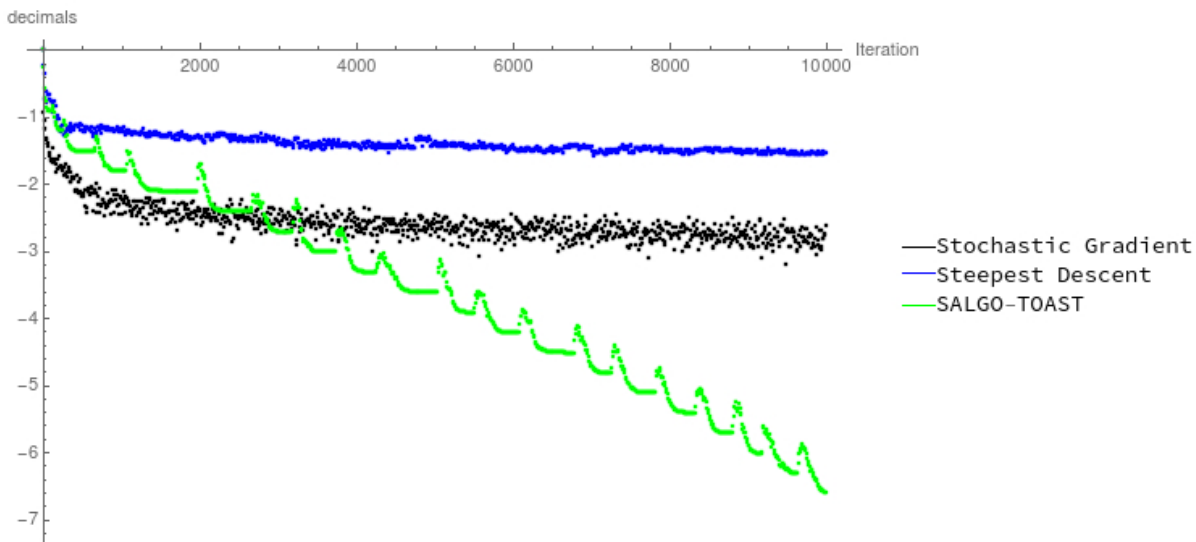


Figure 12: Decimal digits gained by four methods on single layer regression problem.

Now we compare these methods with MILOP strategy. Its results are given by 220 binary variables and equations, 484 real variables, and 880 equality constraints. Larger choices of n , d and m can be explore for our AMPL model using the solver Gurobi via the NEOS server. The good news is that the MILOP system was solved exactly with an objective value of zero so that the regression problem reduced actually to an interpolation problem. The bad news is that Gurobi reached this result by solving 2529 branch and bound nodes using a total number of 95309 Simplex iterations. Though a direct complexity comparison appears difficult, it seems certain that each of them is much more costly than evaluating the empirical risk and its gradient by back propagation. That is the cost of each steepest descent iteration of which we allowed 10000. We allowed the same number of iterations to TOAST, whose steps are a little more costly than those of steepest descent. Each step of stochastic gradient costs about one m -th of the cost for steepest descent. Because the latter we allowed a total of 20000 iterations for SG. At every 200-th iteration of SG we evaluated and plotted the full empirical risk and computed the minimum of all these values over the trajectory. For the other methods we plotted every tenth value but took the minimal value over all iterations (see, Figure 12). The numbers in the second and the third row of the following table give the minimal value and the iteration counter at which it was achieved, respectively.

method	TOAST	MILOP	SD	SG
minvalue	0.0000006	0.0000000	0.069400	0.0017503
iteration	9999	195309	7036	189400

Here the stepsize for SG and GD are anymore constant. SD and SG were run with the step

sizes given respectively by 0.35 and 0.01 divided by the square root of the iterations count. The vertical axis in Figure 12 represents the decimal logarithm of the ratio between the current and the initial empirical risk. As one can see the coordinate search gets stuck in the neighborhood of a local minimizer after some 3000 iterations. Except for Gurobi TOAST achieved the lowest value of $6 \cdot 10^{-6}$, which involved halving the target about 19 times. Every time the trajectory picks up speed and goes uphill for a while before dropping down to the desired lower value. The average rate of convergence appear to be linear, i.e. what in machine learning is sometimes called exponential. Steepest descent and stochastic gradient appear to get stuck or at least slow down near a local minimizer or saddle point. Their performance most probably can be improved by a more sophisticated step size selection heuristic.

7. Conclusions.

The advantages of SALGO-TOAST algorithm are the following. The construction of the Abs-Normal Form of the ER is not a difficulty. Our code already defines the prox-linear form of the ER. Our algorithm also can train an ANN with any architecture, that is, an arbitrary number of neurons and synapses. Unlike steepest descent, our algorithms compute by itself the stepsize of the optimizer. This step size is given by TOAST length in each polyhedron where the approximation of the ER is smooth. Another advantage is that stopping criterium of the training is given by the target level imposed. The user can define a targeting strategy and a bound of search since the ER minima are nonnegative. For instance, we show results using the half of the initial ER and halving it as many times as it can be reached. Our stop criteria is defined by the user as an arbitrary tolerance given by the difference between the lowest minimum reached and the last target level.

Further improvements concern partially the linear algebra implementation. For example, the fact that at each polyhedral boundary the weight matrices undergo only a rank one change can be used to reduce the linear algebra cost per step by a factor equal to the feature dimension. That is still higher than the cost of stochastic gradient by a factor equal to the average layer size.

So far we have applied TOAST only to formulations with an l_1 -norm loss function, which introduces a lot of nonsmoothness. In fact the large majority of the polyhedral boundaries crossed were generated by sign changes in the difference $f(\chi; x) - y$. Therefore one might expect a better performance of TOAST when the loss is quantified by one of the smooth alternatives.

Another aspect that requires further development is the strategy for adjusting the target level c . The simple halving strategy applied so far leads in some cases to an overly ambitious target just below the actual global minimizer and thus an infinitely long search trajectory.

Of course, our simple minded scalar implementation must be vectorized and optimized for parallel architectures and Graphical Processing Units. Only then a meaningful comparison with steepest descent and stochastic gradient as implemented in such packages as TensorFlow and PyTorch will be possible. We have already verified the applicability of TOAST to the MNIST dataset.

References

- [1] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.
- [2] S. Arora, N. Cohen, N. Golowich, and W. Hu, “A convergence analysis of gradient descent for deep linear neural networks,” *arXiv preprint arXiv:1810.02281*, 2018.
- [3] A. Géron, *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. ” O’Reilly Media, Inc.”, 2017.
- [4] A. O. Griewank, “Generalized descent for global optimization,” *Journal of optimization theory and applications*, vol. 34, no. 1, pp. 11–39, 1981.
- [5] D. Yarotsky, “Error bounds for approximations with deep relu networks,” *Neural Networks*, vol. 94, pp. 103–114, 2017.
- [6] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.
- [7] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain.” *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [8] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” *arXiv preprint arXiv:1611.03530*, 2016.
- [9] A. Bagirov, N. Karmitsa, and M. M. Mäkelä, *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer, 2014.
- [10] A. Walther and A. Griewank, “Characterizing and testing subdifferential regularity for piecewise smooth objective functions,” *SIAM Journal on Optimization* 29123:1473-1501, Tech. Rep., 2017.
- [11] A. Griewank, “On stable piecewise linearization and generalized algorithmic differentiation,” *Optimization Methods and Software*, vol. 28, no. 6, pp. 1139–1178, 2013.
- [12] 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, 2018.
- [13] —, “Beyond the oracle: Benefits of piecewise differentiation,” *Numerical Nonsmooth Optimization*.
- [14] N. Loizou and P. Richtárik, “Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods,” *arXiv preprint arXiv:1712.09677*, 2017.
- [15] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.
- [16] V. Vapnik, *The nature of statistical learning theory*. Springer science & business media, 2013.

- [17] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [18] T. Zhang *et al.*, "Statistical behavior and consistency of classification methods based on convex risk minimization," *The Annals of Statistics*, vol. 32, no. 1, pp. 56–85, 2004.
- [19] J. Buhmann, *Empirical risk approximation: An induction principle for unsupervised learning*. Citeseer, 1998.
- [20] P. Domingos, *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books, 2015.
- [21] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.
- [22] I. Diener, "Trajectory methods in global optimization," in *Handbook of Global optimization*. Springer, 1995, pp. 649–668.
- [23] A. Törn and A. Žilinskas, *Global optimization*. Springer, 1989, vol. 350.
- [24] E. Kreyszig, *Introductory functional analysis with applications*. wiley New York, 1978, vol. 1.
- [25] S. Scholtes, *Introduction to Piecewise Differentiable Functions*. Springer, 2012.
- [26] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in neural information processing systems*, 2014, pp. 2924–2932.
- [27] C. J. B. Yan LeCun, Corina Cortes. (2011) The mnist database of handwritten digits. [Online]. Available: <http://http://yann.lecun.com/exdb/mnist/>
- [28] S. Fiege, A. Walther, K. Kulshreshtha, and A. Griewank, "Algorithmic differentiation for piecewise smooth functions: a case study for robust optimization," *Optimization Methods and Software*, vol. 33, no. 4-6, pp. 1073–1088, 2018.
- [29] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Siam, 2008, vol. 105.
- [30] A. Gupte, S. Ahmed, M. Cheon, and S. Dey, "Solving mixed integer bilinear problems using milp formulations," *SIAM Journal on Optimization*, vol. 23, no. 2, pp. 721–744, 2013. [Online]. Available: <https://doi.org/10.1137/110836183>
- [31] P. J. Werbos, *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*. John Wiley & Sons, 1994, vol. 1.
- [32] B. Mordukhovich and M. Sarabi, "Critical multipliers in variational systems via second-order generalized differentiation," *Mathematical Programming*, vol. 4, no. 2, 2017.

Appendices

A. Further analysis.

A.1. Advantages of the GANF/GALF.

In parts of the nonsmooth analysis literature, particular attention is devoted to the convex case (see e.g. [32]). It is well understood that any globally defined convex piecewise linear function is continuous and can be represented in the simple max form

$$\varphi(x) = \max_{0 \leq i \leq s} (a_i^\top x - \beta_i) \quad \text{with} \quad a_i \in \mathbb{R}^n, \beta_i \in \mathbb{R} \quad \text{for} \quad 0 \leq i \leq s$$

Obviously its evaluation in one way or another must involve s comparisons. We will consider the simple loop

$$\gamma_0 = a_0^\top x - \beta_0, \quad \gamma_i = \max(\gamma_{i-1}, a_i^\top x - \beta_i) \quad \text{for} \quad 1 \leq i \leq s, \quad y = \gamma_s$$

In terms of absolute values the loop can be rewritten as

$$\gamma_i = a_i^\top x - \beta_i + \frac{1}{2}z_i + \frac{1}{2}|z_i| \quad \text{with} \quad z_i = \gamma_{i-1} - a_i^\top x + \beta_i$$

Expressing γ_i in terms of z_{i+1} we obtain for $i = 0 \dots s-1$

$$z_{i+1} = -\tilde{a}_{i+1}^\top x + \tilde{\beta}_{i+1} + \frac{1}{2}z_i + \frac{1}{2}|z_i| \quad \text{with} \quad \tilde{a}_{i+1} = a_{i+1} - a_i \quad \text{and} \quad \tilde{\beta}_{i+1} = \beta_{i+1} - \beta_i$$

With $L \in \mathbb{R}^{s \times s}$ the square matrix, whose only nonzero entries are in the $s-1$ subdiagonal positions with constant value $\frac{1}{2}$, we obtain in matrix vector form

$$z = Lz + \tilde{b} - \tilde{A}x + L|z| \quad \text{with} \quad \tilde{b} = (\tilde{\beta}_i)_{1 \leq i \leq s} \in \mathbb{R}^s \quad \text{and} \quad \tilde{A} = (\tilde{a}_i^\top)_{1 \leq i \leq s} \in \mathbb{R}^{s \times n}$$

To arrive at the abs-normal form we have to solve for z using the lower triangular inverse

$$I + \tilde{L} \equiv (I - L)^{-1} = (c_{ij})_{1 \leq j \leq s}^{1 \leq i \leq s} \quad \text{with} \quad c_{ij} = \begin{cases} (\frac{1}{2})^{i-j} & \text{if } i \geq j \\ 0 & \text{else} \end{cases}$$

Using $(I - L)^{-1}L = (I - L)^{-1}(L - I + I) = (I - L)^{-1} - I = \tilde{L}$ we obtain finally the standard triangular system

$$z = (I + \tilde{L})(\tilde{b} - \tilde{A}x) + \tilde{L}|z|$$

for the switching variable vector z .

The objective function takes the somewhat complicated form

$$\varphi(x) = a_s^\top x - \beta_s + \frac{1}{2}e_s^\top \left[(I + \tilde{L})(\tilde{b} - \tilde{A}x) + \tilde{L}|z| \right] + \frac{1}{2}|z_s| = \alpha + a^\top x + b^\top |z|$$

with

$$\alpha = \frac{1}{2}e_s^\top (I + \tilde{L})\tilde{b} - \beta_s, \quad a^\top = a_s^\top - \frac{1}{2}e_s^\top (I + \tilde{L})\tilde{A}, \quad b^\top = \frac{1}{2}e_s^\top (I + \tilde{L})$$

because we have to include z_s as a contribution to $\gamma_s = y$. Given the simplicity of the original function definition, the abs-normal form we derived is surprisingly complicated. More specifically, since \tilde{L} is triangular but otherwise dense the naive evaluation in the final form requires an effort of order s^2 , whereas the original problem can be quite sparse depending on the structure of A of course. It may also suffer a lot of fill-in in the transition to \tilde{A} . One aspect that is similarly

troubling is that from the final procedure it is no longer clear that all the z_i like the γ_i are convex functions of the input vector x . This is really a serious objection. One ad hoc possibility to improve things is to generalize the abs-normal representation of the switching variables to

$$z = Zx + Mz + Lv + c \text{ where } c = \hat{z} + L\hat{v} + M\hat{z} \text{ with } v = \text{abs}(z)$$

where $M = \frac{\partial \Lambda(x, z, v)}{\partial z}$ a triangular matrix whose inverse must have by definition only positive entries.

A.2. SALGO-TOAST inner loop for the multilayer case.

From now on we consider the global optimization of the ER based on a more general Rectifier ANN discussed in Chapter 3. In each i -th layer, the affine transformations depends on the weight matrices $W^{(i)}$ and shifts $b^{(i)}$. Due to the isomorphism between matrices and vector spaces, we consider the weight matrix in its vector form, i.e.

$$W^{(i)} = (W_1^{(i)}, \dots, W_j^{(i)}, \dots, W_{d_i}^{(i)})^\top$$

where $W_j^{(i)} \in \mathbb{R}^{d_{i-1}}$, is the j -th row of the matrix $W^{(i)}$. Also the shifts are denoted in its vector form, i.e.

$$b^{(i)} = (b_1^{(i)}, \dots, b_j^{(i)}, \dots, b_{d_i}^{(i)})^\top \in \mathbb{R}^{d_i}.$$

Therefore an element of the parameter space is given by

$$x = \left(W_1^{(1)}, \dots, W_{d_1}^{(1)}, b^{(1)}, W_1^{(2)}, \dots, W_{d_2}^{(2)}, b^{(2)}, \dots, W_1^{(l)}, \dots, W_{d_l}^{(l)}, b^{(l)} \right) \in \mathbb{R}^q$$

and, for all $1 \leq i \leq l$,

$$x^{(i)} = (W^{(i)}, b^{(i)}) \in \mathbb{R}^{d_i(d_{i-1}+1)}.$$

For a given feature $\chi \in \mathbb{R}^{d_0}$, the prediction function f is given by the following mapping

$$x \in \mathbb{R}^q \mapsto f(\chi; x) \equiv \Phi(x) \quad (85)$$

where $\Phi(x)$ is defined in (25b). Then the ER can be defined over the training dataset $(\chi_k, y_k) \in \mathbb{R}^{m \times (d_0 + d_{l+1})}$ as follows

$$\varphi(x) = \frac{1}{m} \sum_{i=1}^m |f(\chi_k; x) - y_k| \quad (86)$$

So the training of the predictor is given by the optimization problem:

$$\min_{x \in \mathbb{R}^q} \varphi(x) \equiv \Lambda(x, z, v) \quad \text{s.t.} \quad z = \lambda(x, z, v) \quad \text{and} \quad v = \text{abs}(z) \quad (87)$$

where the ER is written in its GANF as we describe below.

Let us define the switching vector $z \in \mathbb{R}^s$ with $s = \sum_{i=1}^{l+1} d_i$ that contains each intermediate switching vector variable of the ANN, which are denoted by $z^{(i)} \in \mathbb{R}^{d_i}$. Explicitly, z is written

as:

$$z = \begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(l)} \\ z^{(l+1)} \end{bmatrix} = \begin{bmatrix} z_1^{(1)} \\ \vdots \\ z_{d_1}^{(1)} \\ z_1^{(2)} \\ \vdots \\ z_{d_2}^{(2)} \\ \vdots \\ z_1^{(l)} \\ \vdots \\ z_{d_l}^{(l)} \\ z_1^{(l+1)} \\ \vdots \\ z_{d_{l+1}}^{(l+1)} \end{bmatrix} = \begin{bmatrix} W_1^{(1)} \chi_k + b_1^{(1)} \\ \vdots \\ W_{d_1}^{(1)} \chi_k + b_{d_1}^{(1)} \\ \frac{1}{2}(W_1^{(2)}(z^{(1)} + v^{(1)}) + b_1^{(2)}) \\ \vdots \\ \frac{1}{2}(W_{d_2}^{(2)}(z^{(1)} + v^{(1)}) + b_{d_2}^{(2)}) \\ \vdots \\ \frac{1}{2}(W_1^{(l)}(z^{(l-1)} + v^{(l-1)}) + b_1^{(l)}) \\ \vdots \\ \frac{1}{2}(W_{d_l}^{(l)}(z^{(l-1)} + v^{(l-1)}) + b_{d_l}^{(l)}) \\ \frac{1}{2}(z_1^{(l)} + v_1^{(l)}) + y_k^1 \\ \vdots \\ \frac{1}{2}(z_{d_l}^{(l)} + v_{d_l}^{(l)} + y_k^{d_l}) \end{bmatrix} \quad (88a)$$

One should notice that each switching variable recursively depends on the last intermediate switch. The last equality defines the constraint $z = \lambda(x, z, v)$. Replacing z in the ER formula (86), $\varphi(x)$ turns out to depend on justm, $z^{(l+1)}$ as follows:

$$\varphi(x) = \|z^{(l+1)} - y_k\|_1 \equiv \frac{1}{2} \|z^{(l)} + v^{(l)}\|_1 = \frac{1}{2} (z^{(l)} + v^{(l)})^\top \mathbf{1} \text{ with } \mathbf{1} \in \mathbb{R}^{d_{l+1}}. \quad (88b)$$

The right hand side results from the nonnegativity of $\frac{1}{2}(z^{(l)} + v^{(l)})$ components. Notice that $d_l = d_{l+1}$ due to componentwise mapping of the hinge function. Since we have verified that the ER for the multilayer case has a GANF, then its GALF is described by the following gradients:

$$M \equiv \frac{\partial \Lambda}{\partial z} \in \mathbb{R}^{s \times s} \quad ; \quad L \equiv \frac{\partial \Lambda}{\partial v} \in \mathbb{R}^{s \times s} \quad ; \quad Z \equiv \frac{\partial \Lambda}{\partial x} \in \mathbb{R}^{s \times q}.$$

As Λ is a vector function, we express it by $l + 1$ component functions Λ_i in terms of each intermediate switching vector variable $z^{(i)}$, i.e.

$$\Lambda_i(x, z, v) = z^{(i)}, \quad \forall 1 \leq i \leq l + 1$$

Due to the echeloned dependence between the switching vector variables, we obtained Z, M, L as follows:

$$Z = \left(\frac{\partial \Lambda_i}{\partial x^{(j)}} \right)_{\forall 1 \leq i, j \leq l+1} \quad (89a)$$

where

$$\frac{\partial \Lambda_i}{\partial x^{(j)}} = \begin{cases} \left(\chi_k^\top \otimes I_{d_1}, I_{d_1} \right)_{d_1 \times d_1(d_0+1)}, & \text{if } i = j = 1 \\ \left(\frac{1}{2}(z^{(i-1)\top} + v^{(i-1)\top}) \otimes I_{d_i}, I_{d_i} \right)_{d_i \times d_i(d_{i-1}+1)}, & \forall 1 \leq i = j < l \\ 0, & \forall 1 \leq i \neq j \leq l + 1 \end{cases}$$

and I_{d_i} is the identity matrix of d_i^2 components.

$$M = \left(\frac{\partial F_i}{\partial z^{(j)}} \right)_{\forall 1 \leq i \leq l+1}, \quad \frac{\partial F_i}{\partial z^{(j)}} = \begin{cases} \frac{1}{2} W^{(i)} & \forall 1 \leq j = i - 1 < l + 1 \\ 0 & \forall 1 \leq j \neq i - 1 < l + 1 \\ \frac{1}{2} I_{d_i} & i = l + 1, j = l \\ 0 & i = l + 1, j \neq l \end{cases} \quad (89b)$$

$$L = \left(\frac{\partial F_i}{\partial v^{(j)}} \right)_{\forall 1 \leq i \leq l+1}, \quad \frac{\partial F_i}{\partial v^{(j)}} = \begin{cases} \frac{1}{2}W^{(i)} & \forall 1 \leq j = i-1 < l+1 \\ 0 & \forall 1 \leq j \neq i-1 < l+1 \\ \frac{1}{2}I_{d_i} & i = l+1, j = l \\ 0 & i = l+1, j \neq l \end{cases} \quad (89c)$$

At this point, we should mention that each 0 in our matrices does not have the same dimensions. Under these above computation is easily verify the lower triangle form of L, M . Furthermore,

$$a = \mathbf{0} \in \mathbb{R}^q \quad c = b = [\mathbf{0}, \mathbf{1}] \in \mathbb{R}^s \quad \text{with} \quad \mathbf{0} \in \mathbb{R}^{\bar{s}}, \mathbf{1} \in \mathbb{R}^{d_{l+1}} \quad \text{and} \quad \bar{s} = \sum_{i=1}^l d_i \quad (90)$$

We obtain so the GALF of our ER given by

$$z = (I - M - L\Sigma)^{-1} \dot{d} + (I - M - L\Sigma)^{-1} Zx \quad (91a)$$

$$\Delta\varphi(\dot{x}, \Delta x) = \dot{\mu} + b^\top [z + v] \quad (91b)$$

where

$$\dot{\mu} = \dot{\varphi} - b^\top (\dot{z} + \dot{w}) \quad \text{and} \quad \dot{d} = \dot{z} + Z\dot{x} - M\dot{z} - L\dot{v} \quad \text{and} \quad \dot{z} = z(\dot{x}), \dot{v} = v(\dot{x})$$

Remark A.1. The matrices Z, L, M in (89a), (89b), (89c) are evaluated at the base point \dot{x} . Besides the procedure of abs-linearization is equivalent to the backpropagation algorithm.

Remember that the full domain \mathbb{R}^n is now decomposed into polyhedra P_σ as in (21) which can be identified by the signature vector σ and signature matrix Σ given by

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

Since $L = M$ and thus $b = c$, we obtain the piecewise linear model replacing (91a) in (91b) and given by

$$\Delta\varphi(\dot{x}, \Delta x) = \dot{\mu} + b^\top (\Sigma + I) ((I - M - L\Sigma)^{-1} \dot{d} + (I - M - L\Sigma)^{-1} Zx). \quad (92)$$

Because Σ is a diagonal matrix $L\Sigma$ is a block strictly lower triangle which is the same structure as L . Afterwards, we denote by $L_\sigma = L\Sigma$ that contains the active components of L on its nonzero entries. Therefore $I_s - M - L_\sigma$ is block lower triangle, invertible, and given by

$$(I_s - M - L_\sigma)^\top = \begin{pmatrix} I_{d_1} & -\frac{1}{2}\bar{W}^{(2)\top} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & I_{d_2} & -\frac{1}{2}\bar{W}^{(3)\top} & 0 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & 0 & \ddots & \ddots & 0 & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & 0 & I_{d_i} & -\frac{1}{2}\bar{W}^{(i)\top} & 0 & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & 0 & \ddots & \ddots & 0 & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & 0 & I_{d_{l-1}} & -\frac{1}{2}\bar{W}^{(l-1)\top} & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & 0 & I_{d_{l-1}} & -\frac{1}{2}\bar{W}^{(l)\top} & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & I_{d_l} & -\frac{1}{2}\bar{I}_{d_l} & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 0 & 0 & I_{d_l} \end{pmatrix} \quad (93)$$

where $\bar{W}^{(i)} = W^{(i)}(I_{d_i} + \Sigma_{d_i})$ and $\bar{I}_{d_l} = I_{d_l}(I_{d_l} + \Sigma_{d_l})$.

Proposition A.2. Consider $\mathbf{r} = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_{l+1})^\top \in \mathbb{R}^{s \times 1}$ where $r_i \in \mathbb{R}^{d_i}$. Let us define $\mathbf{r}^\top = b^\top (I_s + \Sigma)(I_s - M - L_\sigma)^{-1}$ with b defined in (90) and $I_s - M - L_\sigma$ defined in (93). Then the linear system

$$(I_s - M - L_\sigma)^\top r = [\mathbf{0}, \mathbf{1} + \sigma_{l+1}]^\top, \quad \text{where } \sigma_{l+1} = \mathbf{1}^\top \Sigma_{d_{l+1}} \quad (94)$$

has a **unique** solution given by

$$r = [r_1, \dots, r_{l-1}, \frac{1}{2}(\mathbf{1} + \sigma_{l+1} + \sigma_l + \sigma_l \cdot \sigma_{l+1}), \mathbf{1} + \sigma_{l+1}]^\top \quad (95)$$

where

$$\forall 1 \leq i \leq l-1: \quad r_i = \frac{1}{2^{l-i+1}} \left(\prod_{j=i+1}^l \bar{W}^{(j)\top} \right) [\mathbf{1} + \sigma_{l+1} + \sigma_l + \sigma_l \cdot \sigma_{l+1}] \quad (96)$$

Proof. The uniqueness is given by (93) since all the diagonal values of $I_s - M - L_\sigma$ are ones. Because of the bidiagonal block structure of $(I_s - M - L_\sigma)$, the solution is given by

$$r_{l+1} = I_{d_{l+1}} [\mathbf{1} + \sigma_{l+1}]^\top = [\mathbf{1}_{d_{l+1}} + \sigma_{l+1}]^\top$$

$$r_l = \frac{1}{2} \bar{I}_{d_l} r_{l+1} = \frac{1}{2} ([\mathbf{1} + \sigma_{l+1}]^\top + [\sigma_l \cdot \mathbf{1}_{d_{l+1}} + \sigma_l \cdot \sigma_{l+1}]^\top) = \frac{1}{2} [\mathbf{1} + \sigma_{l+1} + \sigma_l + \sigma_l \cdot \sigma_{l+1}]^\top$$

Then, by recursivity and I_{d_i} invertibility, we obtain explicitly r_i for all $1 \leq i \leq l-1$ the equality in (95). \square

Proposition A.2 provides an explicit formula for our multi-piecewise linear model in (92) using (95) at any base point \hat{x} . That is

$$\Delta\varphi(\hat{x}, \Delta x) = \hat{\mu} + \mathbf{r}^\top \hat{d} + \mathbf{r}^\top Zx. \quad (97)$$

Under this formulation, we proceed to the construction of the prox-abs-linear form of the ER and the inner loop of the TOAST strategy suggested in subsection 5.3.

B. Codes.

B.1. SALGO-MILOP code in AMPL.

```

param n integer; #number of nodes
param m integer; # number of samples
param d integer; #feature dimension
set Samples={1..m};
set Nodes={1..n};
set Dim={1..d};

param x{Samples, Dim}=Uniform(-8, 8); #features, real vectors
param x_norm{k in Samples}= sum{j in Dim}abs(x[k,j]);
param y{k in Samples}=1+((1/4000)*sum{j in Dim}x[k,j]*x[k,j])
    -(product{j in Dim}cos(x[k,j]/sqrt(j+1))); #labels
param y_norm = max{k in Samples} abs(y[k]);

var W{Nodes, Dim}, := Uniform(-0.5, 0.5);
var W_norm;
var b{Nodes}, :=Uniform(-0.5, 0.5);
var b_norm ;

param p {i in Nodes} = 2*(i mod 2)-1;
param p_norm = sum{i in Nodes} abs(p[i]);

var SIGMA{Samples, Nodes} binary;
var sigma{k in Samples, i in Nodes}= 2*SIGMA[k, i]-1; #sign of switching variables
var z{Samples, Nodes}; #switching variables
var h{Samples, Nodes}>=0; #nonnegative switching
var sigma_nn{k in Samples, i in Nodes}=1+sigma[k, i]; #translation by unit vector
var sigma_np{k in Samples, i in Nodes}=-1+sigma[k, i]; #translation by negative unit vector

var g{Samples}; #discrepancy
var u{Samples}>=0; #loss evaluations
var MU{Samples} binary;
var mu{k in Samples}=2*MU[k]-1;#sign of loss evaluation

param gamma{Samples, Nodes};
param delta{Samples}; #upper bounds of discrepancy

minimize Prediction: (1/m)*(sum {k in Samples} u[k]);
s.t. z_equa {k in Samples, i in Nodes}: z[k, i]= sum{j in Dim}(W[i,j]*x[k,j]+b[j]);
s.t. g_equa {k in Samples}: g[k]=0.5*(sum{i in Nodes}p[i]*(z[k, i]+h[k, i]))-y[k];

```


#inequality constraints

```
s.t. z1_constraints {k in Samples, i in Nodes}: -h[k, i] <= z[k, i];
s.t. z2_constraints {k in Samples, i in Nodes}: z[k, i] <= h[k, i];
s.t. z3_constraints {k in Samples, i in Nodes}: z[k, i] <= gamma[k, i]*(sigma_nn[k, i]) - h[k, i];
s.t. z4_constraints {k in Samples, i in Nodes}: z[k, i] >= h[k, i] + gamma[k, i]*(sigma_np[k, i]);

s.t. g11_constraints {k in Samples}: -u[k] <= g[k];
s.t. g12_constraints {k in Samples}: g[k] <= u[k];
s.t. g21_constraints {k in Samples}: g[k] <= -u[k] + delta[k]*(mu[k] + 1);
s.t. g22_constraints {k in Samples}: g[k] >= u[k] + delta[k]*(mu[k] - 1);
```