This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Delaunay-Triangulation-Based Learning with Hessian Total-Variation Regularization

MEHRSA POURYA, ALEXIS GOUJON, and MICHAEL UNSER, *FELLOW, IEEE*

[1]Biomedical Imaging Group, École polytechnique fédérale de Lausanne, 1015 Lausanne, Switzerland

CORRESPONDING AUTHOR: Mehrsa Pourya (e-mail: mehrsa.pourya@epfl.ch).

This work was supported in part by the European Research Council (ERC Project FunLearn) under Grant 101020573 and in part by the Swiss National Science Foundation, Grant 200020_184646/1.

ABSTRACT Regression is one of the core problems tackled in supervised learning. Neural networks with rectified linear units generate continuous and piecewise-linear (CPWL) mappings and are the state-of-the-art approach for solving regression problems. In this paper, we propose an alternative method that leverages the expressivity of CPWL functions. In contrast to deep neural networks, our CPWL parameterization guarantees stability and is interpretable. Our approach relies on the partitioning of the domain of the CPWL function by a Delaunay triangulation. The function values at the vertices of the triangulation are our learnable parameters and identify the CPWL function uniquely. Formulating the learning scheme as a variational problem, we use the Hessian total variation (HTV) as a regularizer to favor CPWL functions with few affine pieces. In this way, we control the complexity of our model through a single hyperparameter. By developing a computational framework to compute the HTV of any CPWL function parameterized by a triangulation, we discretize the learning problem as the generalized least absolute shrinkage and selection operator. Our experiments validate the usage of our method in low-dimensional scenarios.

INDEX TERMS Regression, Sparsity, CPWL, Simplicial splines, Generalized LASSO

I. Introduction

Supervised learning entails finding a function $f : \mathbb{R}^d \to \mathbb{R}$ using a set of M training data points $\{\boldsymbol{x}_m\}_{m=1}^M \subset \mathbb{R}^d$ and their target values $\{y_m\}_{m=1}^M \subset \mathbb{R}$. The function f should approximate the target values at the data points $y_m \approx f(\boldsymbol{x}_m)$ and generalize well to new inputs [1]. In a variational framework, the learning problem is formalized as the optimization task

$$\min_{f\in\mathcal{X}} \sum_{m=1}^{M} \mathrm{E}(f(\boldsymbol{x}_m), y_m) + \lambda \mathcal{R}(f), \tag{1}$$

where $\mathcal{X}$ is the function search space, $\mathrm{E} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ is a convex loss function controlling the data-fitting error, and $\mathcal{R} : \mathcal{X} \to \mathbb{R}$ is the regularizer. The hyperparameter $\lambda \geq 0$ adjusts the contribution of the regularizer. Regularization is used to promote functions with desirable structures such as sparsity and to reduce overfitting [2], [3].

In order to make (1) tractable, the search space $\mathcal{X}$ is usually expressed as a parametric space. For instance, linear regression—the simplest model—reduces the learning problem to the search for $\boldsymbol{a} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ such that $f(\boldsymbol{x}) = \boldsymbol{a}^\mathsf{T}\boldsymbol{x} + b$ [4]. This model is very well understood, but it is severely lacking expressivity. Another approach to the learning problem is founded on the theory of reproducing-kernel Hilbert spaces (RKHS) [5], [6]. If the search space in (1) is the reproducing-kernel Hilbert space $\mathcal{X} = \mathcal{H}(\mathbb{R}^d)$ with kernel $\mathrm{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and $\mathcal{R}(f) = \|f\|_{\mathcal{H}}^2$ where $\|\cdot\|_{\mathcal{H}}$ is the Hilbert-space norm, then the RKHS representer theorem states that the solution of (1) admits the closed-form

$$f(\cdot) = \sum_{m=1}^{M} a_m \mathrm{k}(\cdot, \boldsymbol{x}_m) \tag{2}$$

for some coefficients $(a_m) \in \mathbb{R}^M$ [7]. There, the problem is recast into a solvable linear model. Moreover, by using radial-basis functions as kernels, one can approach continuous functions as closely as desired [8]–[10]. While kernel methods have nice theoretical properties, they are outperformed by deep neural networks, which have state-of-the-art performance in many applications [11]–[14].

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

A. From Neural Networks to Continuous and Piecewise-Linear Models

A neural network of depth L forms a mapping

$$x \mapsto (\boldsymbol{\sigma}_L \circ \boldsymbol{f}_{\boldsymbol{\theta}_L} \circ \boldsymbol{\sigma}_{L-1} \circ \cdots \circ \boldsymbol{\sigma}_l \circ \boldsymbol{f}_{\boldsymbol{\theta}_l} \circ \cdots \circ \boldsymbol{\sigma}_1 \circ \boldsymbol{f}_{\boldsymbol{\theta}_1})(\boldsymbol{x}), \quad (3)$$

where $\boldsymbol{f}_{\boldsymbol{\theta}_l} : \mathbb{R}^{d_{l-1}} \to \mathbb{R}^{d_l}$ is a learnable affine mapping parameterized by $\boldsymbol{\theta}_l$ and $\boldsymbol{\sigma}_l$ is the nonlinearity (a.k.a. "activation function") performed in the lth layer. Among all possible activation functions, the most common choice and usually the best in terms of performance is the rectified linear unit $\mathrm{ReLU}(x) = \max(x, 0)$ [15]. It is known that a ReLU network produces continuous and piecewise-linear (CPWL) functions [16]. Conversely, any CPWL function can be described by a ReLU network [17]. The CPWL model is universal, in the sense that it can approximate any continuous mapping [18].

A major drawback of letting deep networks learn a CPWL function is the lack of their interpretability, in the sense that the effect of each parameter on the generated mapping is not understood explicitly [19]. As such, it is hard to find a regularization scheme for the training of networks that would have an explicit geometric effect on the learned mapping [20]. For instance, weight decay (the most popular regularizer) somehow regulates the magnitude of the network parameters, but its impact on the final mapping is hard to understand [21]. Recent research links the effect of weight decay in shallow ReLU networks to the sparsity of the mapping in certain latent spaces [22]–[24]. However, shallow networks are not capable of representing all CPWL functions [25]. To the best of our knowledge, there is no comprehensive variational interpretation for deep networks.

B. Variational Learning of CPWL Functions

Linear regression is interpretable but has almost no expressivity. Deep neural networks, by contrast, have excellent expressivity but are difficult to interpret. In this paper, we try to reconcile the two approaches by proposing a scheme that locally resembles linear regression and is therefore interpretable. Our model maintains expressivity through the underlying CPWL construction. This is achieved by investigating CPWL models that are solutions of a variational problem with a regularization term that explicitly promotes certain structures on the generated mapping. Such models are well understood in the one-dimensional scenarios [26], [27]. To that end, one rewrites Equation (1) with $f : \mathbb{R} \to \mathbb{R}$ and the second-order total variation $\mathrm{TV}^{(2)}(f)$ as the regularizer. Then, by restricting the search space to functions with a bounded second-order total variation, the extreme points of the solution set are necessarily of the form

$$f : x \mapsto ax + b + \sum_{k=1}^{K} d_k (x - \tau_k)_+, \quad (4)$$

where $a, b \in \mathbb{R}, \boldsymbol{d} = (d_k) \in \mathbb{R}^K$, $K < M$, and $(\tau_k) \in \mathbb{R}^K$. By inspection of (4), f is indeed a CPWL function [28]. For such solutions, the regularization has the simple closed-form $\mathrm{TV}^{(2)}(f) = \|\boldsymbol{d}\|_{\ell_1}$. As the ℓ_1-norm promotes sparsity, a regularization by the second-order total variation will promote CPWL functions with few knots. The natural multidimensional generalization of $\mathrm{TV}^{(2)}$ is the Hessian total-variation (HTV) seminorm [29] whose origin can be traced back to [30]. Since the null space of HTV is composed of affine mappings, this regularization favors solutions that are predominantly affine; typically, CPWL functions with few pieces. Hence, HTV regularization allows direct control over the complexity of the learned mapping. This concept was put into practice in [31], albeit in the restricted setting $d = 2$ (two input variables). The CPWL functions in [31] are parameterized through box splines on a hexagonal lattice. The extension of this uniformly gridded framework to higher dimensions is computationally restricted due to the exponential growth of the number of grid points with the dimension.

Our framework addresses those limitations: (i) it is theoretically applicable to any number of dimensions and for any type of (uniform or nonuniform) triangulation; and (ii) it can be used in practice for intermediate dimensions d up to 9, through our implementation. The present contributions are as follows.

1) Use of a flexible parameterization of CPWL functions: We partition the input domain by performing a Delaunay triangulation on a set of data-adaptive grid points; the function values at the grid points are our learnable parameters and identify the CPWL functions uniquely. In fact, any d-dimensional CPWL function can be described exactly by such a parameterization [32].
2) Development of a tractable scheme for computing HTV: For the mentioned CPWL model, we show that HTV is the ℓ_1-norm applied to a linear transformation of the grid-point values. We present the details of the computation of this transformation for any set of grid points in any dimension.
3) Efficient algorithm to solve the HTV learning problem: We develop an iterative algorithm based on a two-step fast iterative shrinkage-thresholding algorithm (FISTA) to solve the training optimization problem [33]. Our algorithm accommodates any choice of grid points.

We then validate the performance of our method, which we refer to as DHTV, through experimental results.

C. Roadmap

Our paper is organized as follows: In Section II, we introduce the mathematical tools that we need to develop our method. In Section III, we describe our CPWL parameterization, explain the procedure to calculate its HTV, and derive the generalized least-absolute-shrinkage-and-selection-operator (LASSO) formulation of our learning problem and

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

our proposed algorithm for solving it. Finally, we present our experimental results in Section IV.

II. Preliminaries

A. Delaunay Triangulation

In the d-dimensional space $\mathbb{R}^d$, the convex hull of $d+1$ affinely independent points forms a polytope known as a d-simplex, or simplex for short. These simplices are indeed triangles and tetrahedrons in 2- and 3-dimensional spaces. A triangulation of a set $\mathbf{X} \subset \mathbb{R}^d$ of points is the partition of their convex hull into simplices such that any two simplices intersect either at a face joint or not at all. Also, the triangulation vertices, referred to as grid points, are exactly the points themselves. We consider two simplices as neighbors if their intersection is a facet, which is a $(d-1)$-simplex. In general, a triangulation of a set of points is not unique. A celebrated triangulation method is the Delaunay triangulation.

Definition 1 (Delaunay Triangulation)**.** *For a set $\mathbf{X}$ of points in $\mathbb{R}^d$, a Delaunay triangulation is the triangulation $DT(\mathbf{X})$ such that no point in $\mathbf{X}$ is inside the circum-hypersphere of any simplex in $DT(\mathbf{X})$.*

Simply put, the Delaunay triangulation partitions the convex hull of points into well-formed simplices. Specifically for $d = 2$, it is known that the Delaunay triangulation maximizes the minimal angle of the triangles of the triangulation and avoids skinny triangles. Similar optimal properties exist in higher dimensions [34]. In addition, there exist computational methods that produce Delaunay triangulations in any dimension [35], [36].

B. Continuous and Piecewise-Linear Functions

Definition 2 (CPWL function)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is continuous and piecewise-linear if*

1. *it is continuous;*
2. *its domain $\Omega = \bigcup_n P_n$ can be partitioned into a set of non-overlapping polytopes P_n over which it is affine, with $f|_{P_n}(\boldsymbol{x}) = \boldsymbol{a}_n^T \boldsymbol{x} + b_n$.*

The gradient of the function over each polytope or, equivalently, each linear region P_n, is $\boldsymbol{\nabla} f|_{P_n}(\boldsymbol{x}) = \boldsymbol{a}_n$. We denote the intersection facet of two neighboring polytopes P_n and P_k as $L_{n,k}$. The $(d-1)$-dimensional volume of the intersection is denoted by $\mathrm{Vol}_{d-1}(L_{n,k})$. For $d = 2$ and $d = 3$, this volume corresponds to length and area, respectively. Finally, we define $\boldsymbol{u}_{n,k} \in \mathbb{R}^d$ as the unit vector that is normal to $L_{n,k}$. We follow these notations throughout the paper.

Alternatively, any CPWL function can be defined by a triangulation and the values of the function at its vertices, which we refer to as the simplicial parameterization of CPWL functions. The Authors in [37] use a similar parameterization. This parameterization yields a Riesz basis, which guarantees a unique and stable link between the model parameters and the CPWL function [32]. In Figure 1, we show an example of the domain of an arbitrary CPWL function and a possible triangulation of it.

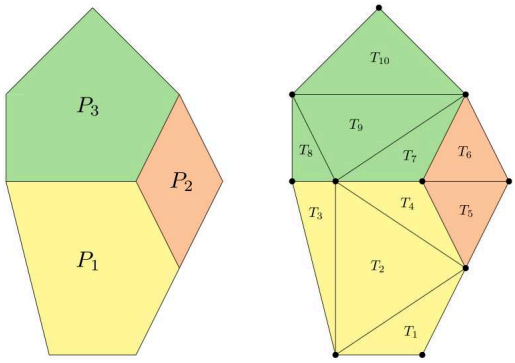


FIGURE 1: Domain of a two-dimensional CPWL function. Each patch corresponds to one linear region (left) and its partition into simplices through a Delaunay triangulation (right).

C. Hessian Total Variation

1) Generalized Hessian Matrix

The Hessian matrix of a twice-differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is defined as

$$\mathrm{H}\{f\} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}. \tag{5}$$

One can extend this definition to generalized functions (distributions) by using the notion of weak partial derivatives. This enables us to define the Hessian of CPWL functions even though they are not twice-differentiable in the classical sense. Using the Hessian matrix, we have that $\mathrm{D}^2_{\boldsymbol{u}} f(\boldsymbol{x}) = \boldsymbol{u}^T \mathrm{H}\{f\}(\boldsymbol{x}) \boldsymbol{u}$, where $\mathrm{D}^2_{\boldsymbol{u}} f$ is the second-order directional derivative of f along the direction $\boldsymbol{u}$. A symmetric Hessian matrix has a complete set of eigenvalues and eigenvectors which form an orthogonal basis for $\mathbb{R}^d$. Consequently, the second-order directional derivative along the eigenvector $\boldsymbol{v}_q$ of the Hessian is its associated eigenvalue λ_q, with $\mathrm{D}^2_{\boldsymbol{v}_q} f(\boldsymbol{x}) = \lambda_q$ for $q \in 1, \ldots, d$. If we use the eigenvectors of the Hessian to represent the direction $\boldsymbol{u} = \sum_{q=1}^d t_q \boldsymbol{v}_q$, then we have that $\mathrm{D}^2_{\boldsymbol{u}} f(\boldsymbol{x}) = \sum_{q=1}^d t_q^2 \lambda_q$. This means that at each point $\boldsymbol{x} \in \mathbb{R}^d$ of the domain, the second-order directional derivatives along the eigenvectors of the Hessian fully characterize the second-order derivatives of f along any direction.

2) Schatten p-Norm

The Schatten p-norm $\|\cdot\|_{\mathcal{S}_p}$ of a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ for $p \in [1, +\infty]$ is defined as the ℓ_p-norm of its singular values given

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

by

$$\|\mathbf{A}\|_{\mathcal{S}_p} \coloneqq \begin{cases} \left(\sum_{k=1}^{d} {\sigma_k}^p\right)^{\frac{1}{p}}, & 1 \leq p < +\infty \\ \max_k \sigma_k, & p = +\infty, \end{cases} \tag{6}$$

where $(\sigma_1, \ldots, \sigma_d)$ are the singular values of $\mathbf{A}$. In this paper, we focus on the $\mathcal{S}_1$-norm and its dual $\mathcal{S}_\infty$.

3) Hessian Total Variation

If f is twice differentiable with a symmetric Hessian at $\boldsymbol{x} \in \mathbb{R}^d$, then $\|\mathrm{H}\{f\}(\boldsymbol{x})\|_{\mathcal{S}_1}$ is given by the ℓ_1-norm of the second-order directional derivatives along the eigenvectors of the Hessian matrix. This measure provides a local characterization of the second-order variation of f. Hence, the total variation of the mapping $\cdot \mapsto \|\mathrm{H}\{f\}(\cdot)\|_{\mathcal{S}_1}$ is a reasonable generalization of the second-order total variation for multidimensional functionals. This is referred to as the Hessian total variation. If $\mathrm{H}\{f\} \in L_1(\mathbb{R}^d; \mathbb{R}^{d\times d})$, then HTV is simply defined as

$$\mathrm{HTV}(f) = \left\| \|\mathrm{H}\{f\}\|_{\mathcal{S}_1} \right\|_{L_1} = \int_{\mathbb{R}^d} \sum_{k=1}^{d} \sigma_k(\mathrm{H}\{f\}(\boldsymbol{x}))\mathrm{d}\boldsymbol{x}. \tag{7}$$

In particular, by using the HTV as a regularizer, we promote configurations where most of the singular values of the Hessian are zero throughout the function domain due to the sparsity effects of the ℓ_1- and L_1-norms. In that manner, CPWL functions are of special interest as their second-order directional derivatives vanish almost everywhere. Specifically, the gradient of a CPWL function f admits the explicit form

$$\boldsymbol{\nabla} f(\boldsymbol{x}) = \sum_n \boldsymbol{a}_n \mathbb{1}_{P_n}(\boldsymbol{x}), \tag{8}$$

where the indicator function $\mathbb{1}_{P_n}$ is equal to one inside the simplex P_n and zero elsewhere. Hence, the second-order directional derivatives vanish everywhere except on the boundaries of the domain polytopes. There, the generalized Hessian matrix of the CPWL functions exhibits delta-like distributions. While (7) helps us understand the effect of HTV, it is not applicable to CPWL functions which are not twice-differentiable in the classical sense. To accommodate for this latter type of functions, we need to adopt a more permissive distributional definition of HTV. Specifically, we define

$$\mathrm{HTV}(f) = \|\mathrm{H}\{f\}\|_{\mathcal{S}_1,\mathcal{M}}, \tag{9}$$

where H denotes the Hessian in the sense of distributions. The mixed norm $\|\cdot\|_{\mathcal{S}_1,\mathcal{M}}$ for any matrix-valued distribution $W \in \mathcal{S}'(\mathbb{R}^d; \mathbb{R}^{d\times d})$ is

$$\|\mathrm{W}\|_{\mathcal{S}_1,\mathcal{M}} \coloneqq \sup\{\langle W, F\rangle : F \in \mathcal{S}(\mathbb{R}^d; \mathbb{R}^{d\times d}), \|F\|_{\mathcal{S}_\infty} \leq 1\}. \tag{10}$$

This formulation enables us to derive a closed form for the HTV of a CPWL function f as

$$\mathrm{HTV}(f) = \sum_{(n,k)\in\mathcal{N}} \left|\boldsymbol{u}_{n,k}^T(\boldsymbol{a}_k - \boldsymbol{a}_n)\right| \mathrm{Vol}_{d-1}(L_{n,k}) \tag{11}$$

$$= \sum_{(n,k)\in\mathcal{N}} \|\boldsymbol{a}_k - \boldsymbol{a}_n\|_2 \,\mathrm{Vol}_{d-1}(L_{n,k}), \tag{12}$$

where the set $\mathcal{N}$ contains all unique pairs of neighboring polytope indices (see [29] for more details). Equation (12) tells us that, for CPWL functions, HTV penalizes the change of slope at neighboring domain polytopes by the volume of their intersection. For a purely affine mapping, the HTV is zero. Otherwise, it increases with the number of affine pieces.

HTV admits the following properties [29, Appendix C]:

$$\begin{aligned} &1)\quad \mathrm{HTV}\left(f(\mathbf{U}\,\cdot)\right) = \mathrm{HTV}(f), \quad \mathbf{U} \in \mathbb{R}^{d\times d} : \mathbf{U}^T\mathbf{U} = \mathbf{I}; \\ &2)\quad \mathrm{HTV}\left(f(\alpha\,\cdot)\right) = |\alpha|^{2-d}\,\mathrm{HTV}(f), \qquad \forall \alpha \in \mathbb{R}; \\ &3)\quad \mathrm{HTV}\left(f(\cdot - \boldsymbol{x}_0)\right) = \mathrm{HTV}(f), \qquad \forall \boldsymbol{x}_0 \in \mathbb{R}^d. \end{aligned} \tag{13}$$

In other words, HTV is invariant to translation and rotation while it is covariant with scaling.

III. Methods

Let us perform a Delaunay triangulation on the set $\mathbf{T}_g = \{\boldsymbol{\tau}_k\}_{k=1}^{N_g} \subset \mathbb{R}^d$ of grid points. We denote the resulting simplices by $\mathcal{T} = \{T_n\}_{n=1}^{N_s}$. We define the CPWL functions f_{Hull} inside the convex hull $\mathrm{Hull}(\mathbf{T}_g)$ by designating the members of $\mathcal{T}$ as their linear regions. Then, the functions are parameterized as

$$f_{\mathrm{Hull}}(\boldsymbol{x}) = \sum_{k=1}^{N_g} c_k s_k(\boldsymbol{x}) \tag{14}$$

with expansion coefficients $c_k \in \mathbb{R}$ and basis functions s_k. The basis s_k is the hat function attached to the kth vertex. It is is given by

$$s_k(\boldsymbol{x}) = \begin{cases} \beta_{k,l}(\boldsymbol{x}), & \boldsymbol{x} \in T_{k,l} \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

$T_{k,l}$ is the l-th simplex that contains the vertex k and $\beta_{k,l}(\boldsymbol{x})$ is the barycentric coordinate of the point $\boldsymbol{x}$ inside the simplex $T_{k,l}$ with respect to vertex k [32]. The basis functions s_k are continuous and form a Riesz basis for the space. From the definition of barycentric coordinates, we know that $s_k(\boldsymbol{\tau}_k) = 1$ and that $s_k(\boldsymbol{\tau}_n) = 0, n \neq k$. We illustrate an example of this basis function in Figure 2 for $d = 2$. Consequently, the c_k in (14) are given by $c_k = f_{\mathrm{Hull}}(\boldsymbol{\tau}_k)$. The item of relevance is that the function f_{Hull} is uniquely identified by $\{(\boldsymbol{\tau}_k, c_k)\}_{k=1}^{N_g}$. An example of such a function is illustrated in Figure 3 (a). In our work, the grid points are immutable and $\boldsymbol{c} = (c_k)$ is our vector of learnable parameters. These parameters are the sampled values of the function at the vertices of the triangulation. Hence, their effect on the generated mapping is known, which makes our

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

model directly interpretable. We can also write the function f_{Hull} as

$$f_{\text{Hull}}(\boldsymbol{x}) = \boldsymbol{\gamma}_{\boldsymbol{x}}^T \boldsymbol{c}, \tag{16}$$

where $\boldsymbol{\gamma}_{\boldsymbol{x}} = (\boldsymbol{s}_k(\boldsymbol{x}))_{k=1}^{N_g}$. The importance of this representation is that it expresses the function value as a linear combination of the grid-point values $\boldsymbol{c}$. In each simplex, the only nonzero basis functions are the ones defined over the vertices of that simplex. This implies that there are at most $(d+1)$ nonzero values in the vector $\boldsymbol{\gamma}_{\boldsymbol{x}}$.

A. Forward Operator

Given the set of training $\{(\boldsymbol{x}_m, y_m)\}_{m=1}^M$ data, we choose grid points $\mathbf{T}_g$ such that all data points are inside $\text{Hull}(\mathbf{T}_g)$. Then, the function f_{Hull} is evaluated at data points as

$$\boldsymbol{f} = \mathbf{H}\boldsymbol{c}, \tag{17}$$

where $\boldsymbol{f} = (f_m)_{m=1}^M$ and $f_m = f_{\text{Hull}}(\boldsymbol{x}_m)$. The matrix $\mathbf{H} \in \mathbb{R}^{M \times N_g}$ is referred to as the forward operator and is a mapping between the grid-point values $\boldsymbol{c}$ and the values of the function $\boldsymbol{f}$ at the data points. If we represent the forward matrix as

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{h}_1^T \\ \vdots \\ \boldsymbol{h}_{N_g}^T \end{bmatrix}, \tag{18}$$

then, by (16), each row of $\mathbf{H}$ can be written as $\boldsymbol{h}_m = \boldsymbol{\gamma}_{\boldsymbol{x}_m}$. This implies that $\mathbf{H}$ is a sparse matrix with at most $(d+1)$ nonzero entries per row.

B. Regularization Operator

To determine the Hessian total variation of the function f_{Hull} through (11), we need to calculate three quantities for each pair of neighbor simplices in $\mathcal{T}$; the gradient difference of the affine pieces over those pairs; the unit normal of their intersection; and, finally, their intersection volume.

Setup: Assume that $T_A, T_B \in \mathcal{T}$ are two neighboring simplices and that the sets of indices of their vertices are $\mathcal{V}_A$ and $\mathcal{V}_B$, with $|\mathcal{V}_A| = |\mathcal{V}_B| = d+1$. We denote the vertices of their intersection by $\mathcal{V}_{A,B} = \mathcal{V}_A \cap \mathcal{V}_B$. There are exactly d common vertices between two neighboring simplices, so that $|\mathcal{V}_{A,B}| = d$. We assume that $\tilde{\boldsymbol{\tau}}_0, \tilde{\boldsymbol{\tau}}_{d+1}$ and $\tilde{c}_0, \tilde{c}_{d+1}$ are the coordinates and values at the vertices indexed by members of $\mathcal{V}_A \setminus \mathcal{V}_{A,B}$ and $\mathcal{V}_B \setminus \mathcal{V}_{A,B}$. Also, we denote $\tilde{\boldsymbol{\tau}}_l$ and $\tilde{c}_l$ the coordinate and value of the vertex indexed by the l-th smallest member of $\mathcal{V}_{A,B}$ for $1 \le l \le d$. We show an example of this setup in Figure 3 (b).

Theorem 1 (Gradient difference)**.** *Let $\nabla f_{\text{Hull}}|_{T_A}(\boldsymbol{x}) = \boldsymbol{a}_A$ and $\nabla f_{\text{Hull}}|_{T_B}(\boldsymbol{x}) = \boldsymbol{a}_B$. Their difference can be expressed as the linear combination of grid points values $\mathbf{c}$ given by*

$$\boldsymbol{a}_A - \boldsymbol{a}_B = \mathbf{G}_{A,B}\boldsymbol{c}, \tag{19}$$

where

$$\mathbf{G}_{A,B} = \begin{bmatrix} \mathbf{G}_A\mathbf{1} & \mathbf{G}_B - \mathbf{G}_A & -\mathbf{G}_B\mathbf{1} \end{bmatrix} \mathbf{W}_{A,B},$$

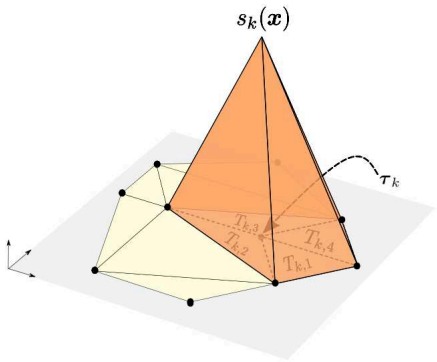


FIGURE 2: The kth vertex of the triangulation with coordinate $\boldsymbol{\tau}_k$ is included in 4 domain simplices $\{T_{k,l}\}_{l=1}^4$ over which the associated simplicial basis (hat) function $s_k(\boldsymbol{x})$ is illustrated.

$$\mathbf{G}_A = \begin{bmatrix} (\tilde{\boldsymbol{\tau}}_0 - \tilde{\boldsymbol{\tau}}_1)^T \\ \vdots \\ (\tilde{\boldsymbol{\tau}}_0 - \tilde{\boldsymbol{\tau}}_d)^T \end{bmatrix}^{-1}, \mathbf{G}_B = \begin{bmatrix} (\tilde{\boldsymbol{\tau}}_{d+1} - \tilde{\boldsymbol{\tau}}_1)^T \\ \vdots \\ (\tilde{\boldsymbol{\tau}}_{d+1} - \tilde{\boldsymbol{\tau}}_d)^T \end{bmatrix}^{-1}. \tag{20}$$

There, the symbol $\mathbf{1}$ represents the vector $(1)_{k=1}^d$ and $\mathbf{W}_{A,B} = [w_{p,k}]_{(d+2)\times N_g}$ is a sparse binary matrix such that $w_{p,k} = 1$ if and only if $\tilde{\boldsymbol{\tau}}_p = \boldsymbol{\tau}_k$.

Proof:

Since f_{Hull} is affine over the simplices T_A and T_B, we have that

$$\begin{cases} (\tilde{\boldsymbol{\tau}}_0 - \tilde{\boldsymbol{\tau}}_p)^T \boldsymbol{a}_A = \tilde{c}_0 - \tilde{c}_p, & p = 1, ..., d \\ (\tilde{\boldsymbol{\tau}}_{d+1} - \tilde{\boldsymbol{\tau}}_p)^T \boldsymbol{a}_B = \tilde{c}_{d+1} - \tilde{c}_p, & p = 1, ..., d. \end{cases} \tag{21}$$

Putting all equations together, we obtain that

$$\boldsymbol{a}_A = \begin{bmatrix} (\tilde{\boldsymbol{\tau}}_0 - \tilde{\boldsymbol{\tau}}_1)^T \\ \vdots \\ (\tilde{\boldsymbol{\tau}}_0 - \tilde{\boldsymbol{\tau}}_d)^T \end{bmatrix}^{-1} \begin{bmatrix} \tilde{c}_0 - \tilde{c}_1 \\ \vdots \\ \tilde{c}_0 - \tilde{c}_d \end{bmatrix} = \mathbf{G}_A \begin{bmatrix} \tilde{c}_0 - \tilde{c}_1 \\ \vdots \\ \tilde{c}_0 - \tilde{c}_d \end{bmatrix}$$

$$= \tilde{c}_0 \mathbf{G}_A \mathbf{1} - \mathbf{G}_A \begin{bmatrix} \tilde{c}_1 \\ \vdots \\ \tilde{c}_d \end{bmatrix}. \tag{22}$$

By analogy, we have that

$$\boldsymbol{a}_B = \tilde{c}_{d+1} \mathbf{G}_B \mathbf{1} - \mathbf{G}_B \begin{bmatrix} \tilde{c}_1 \\ \vdots \\ \tilde{c}_d \end{bmatrix}. \tag{23}$$

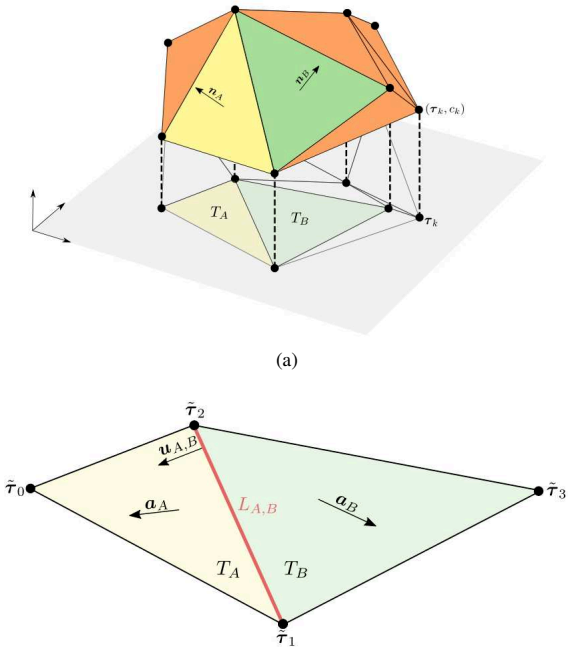
Next, we write the difference of $\boldsymbol{a}_A$ and $\boldsymbol{a}_B$ as

$$\boldsymbol{a}_A = \begin{bmatrix} \mathbf{G}_A\mathbf{1} & -\mathbf{G}_A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \vdots \\ \tilde{c}_{d+1} \end{bmatrix}, \tag{24}$$

$$\boldsymbol{a}_B = \begin{bmatrix} \mathbf{0} & -\mathbf{G}_B & \mathbf{G}_B\mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \vdots \\ \tilde{c}_{d+1} \end{bmatrix}, \tag{25}$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104



(a)

(b)

FIGURE 3: (a) Example of a CPWL function f_{Hull} defined through a triangulation. Over each simplex T_A, one can identify the function f_{Hull} by the plane that passes through the points $\{(\boldsymbol{\tau}_k, f_{\text{Hull}}(\boldsymbol{\tau}_k) = c_k)\}_{\boldsymbol{\tau}_k \in T_A}$. The vector $\boldsymbol{n}_A$ corresponds to the normal vector of that plane. (b) Top view of the domain simplices T_A and T_B.

where $\mathbf{0} = (0)_{k=1}^{d}$. Hence,

$$\boldsymbol{a}_A - \boldsymbol{a}_B = \begin{bmatrix} \mathbf{G}_A\mathbf{1} & \mathbf{G}_B - \mathbf{G}_A & -\mathbf{G}_B\mathbf{1} \end{bmatrix} \begin{bmatrix} \tilde{c}_0 \\ \vdots \\ \tilde{c}_{d+1} \end{bmatrix} \tag{26}$$

and, from the definition of $\mathbf{W}_{A,B}$, we know that

$$\begin{bmatrix} \tilde{c}_0 \\ \vdots \\ \tilde{c}_{d+1} \end{bmatrix} = \mathbf{W}_{A,B}\boldsymbol{c}. \tag{27}$$

From (26) and (27), we finally obtain (19). ■

Theorem 2 (Determination of a normal vector)**.** *The unit normal $\boldsymbol{u}_{A,B}$ of the intersection of T_A and T_B is given by*

$$\boldsymbol{u}_{A,B} = \frac{\mathbf{N}_{[1:d,1]}}{\left\|\mathbf{N}_{[1:d,1]}\right\|}, \tag{28}$$

where

$$\mathbf{N} = \begin{bmatrix} \tilde{\boldsymbol{\tau}}_0^T & 1 \\ \tilde{\boldsymbol{\tau}}_1^T & 1 \\ \vdots & \vdots \\ \tilde{\boldsymbol{\tau}}_d^T & 1 \end{bmatrix}^{-1} \tag{29}$$

and where the slicing operator $[1:d,1]$ returns the first d elements of the first column of $\mathbf{N}$.

Proof:
The intersection of T_A and T_B is the facet opposite to the vertex with coordinate $\tilde{\boldsymbol{\tau}}_0$ in the simplex T_A. We show that $\boldsymbol{z}_0 = \mathbf{N}[1:d,1]$ is perpendicular to that facet and, hence, to the intersection. From the definition of the inverse, we know that

$$\begin{bmatrix} \tilde{\boldsymbol{\tau}}_0^T & 1 \\ \tilde{\boldsymbol{\tau}}_1^T & 1 \\ \vdots & \vdots \\ \tilde{\boldsymbol{\tau}}_d^T & 1 \end{bmatrix} \mathbf{N} = \mathbf{I}. \tag{30}$$

Hence, if we write the matrix $\mathbf{N}$ as

$$\mathbf{N} = \begin{bmatrix} \boldsymbol{z}_0 & \cdots & \boldsymbol{z}_d \\ -b_0 & \cdots & -b_d \end{bmatrix} \tag{31}$$

for some $\{\boldsymbol{z}_k\}_{k=0}^{d} \subset \mathbb{R}^d$ and $\{b_k\}_{k=0}^{d} \subset \mathbb{R}$, then we have that

$$\begin{bmatrix} \tilde{\boldsymbol{\tau}}_0^T & 1 \\ \vdots & \vdots \\ \tilde{\boldsymbol{\tau}}_d^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_0 & \cdots & \boldsymbol{z}_d \\ -b_0 & \cdots & -b_d \end{bmatrix} = \begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix}. \tag{32}$$

In particular, this implies that $\tilde{\boldsymbol{\tau}}_0^T \boldsymbol{z}_0 = 1 + b_0$ and $\tilde{\boldsymbol{\tau}}_m^T \boldsymbol{z}_0 = b_0$ for $m > 0$, so that

$$(\tilde{\boldsymbol{\tau}}_m - \tilde{\boldsymbol{\tau}}_l)^T \boldsymbol{z}_0 = b_0 - b_0 = 0, \tag{33}$$

for $m > 0, l \neq m$. Equation (33) implies that $\boldsymbol{z}_0$ is perpendicular to the simplex formed by $\tilde{\boldsymbol{\tau}}_1, \ldots, \tilde{\boldsymbol{\tau}}_d$, which is exactly the facet opposite to $\tilde{\boldsymbol{\tau}}_0$ and is the intersection of T_A and T_B. ■

Theorem 3 (Cayley–Menger determinant [38])**.** *The (d-1) dimensional volume* Vol *of the simplex formed by $\{\tilde{\boldsymbol{\tau}}_1, \ldots, \tilde{\boldsymbol{\tau}}_d\}$ is given by*

$$\text{Vol}^2 = \gamma \begin{vmatrix} 0 & \tilde{d}_{1,2} & \cdots & \tilde{d}_{1,d} & 1 \\ \tilde{d}_{2,1} & 0 & \cdots & \tilde{d}_{2,d} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{d}_{d,1} & \tilde{d}_{d,2} & \cdots & 0 & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{vmatrix}, \tag{34}$$

where $\tilde{d}_{k,l} = \|\tilde{\boldsymbol{\tau}}_k - \tilde{\boldsymbol{\tau}}_l\|^2$ and

$$\gamma = \frac{(-1)^d}{((d-1)!)^2 2^{d-1}}. \tag{35}$$

The intersection of T_A and T_B is the simplex formed by $\{\tilde{\boldsymbol{\tau}}_1, \ldots, \tilde{\boldsymbol{\tau}}_d\}$. Hence, by using Theorem 3, we can obtain the intersection volume $\text{Vol}_{A,B}$.

We define the matrix $\mathbf{R}_{A,B}$ as

$$\mathbf{R}_{A,B} = \text{Vol}_{A,B} \boldsymbol{u}_{A,B}^T \mathbf{G}_{A,B}. \tag{36}$$

From Theorems 1-3, we have that

$$\left|\boldsymbol{u}_{A,B}^T (\boldsymbol{a}_A - \boldsymbol{a}_B)\right| \text{Vol}_{d-1}(L_{A,B}) = \left|\mathbf{R}_{A,B}\boldsymbol{c}\right|. \tag{37}$$

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

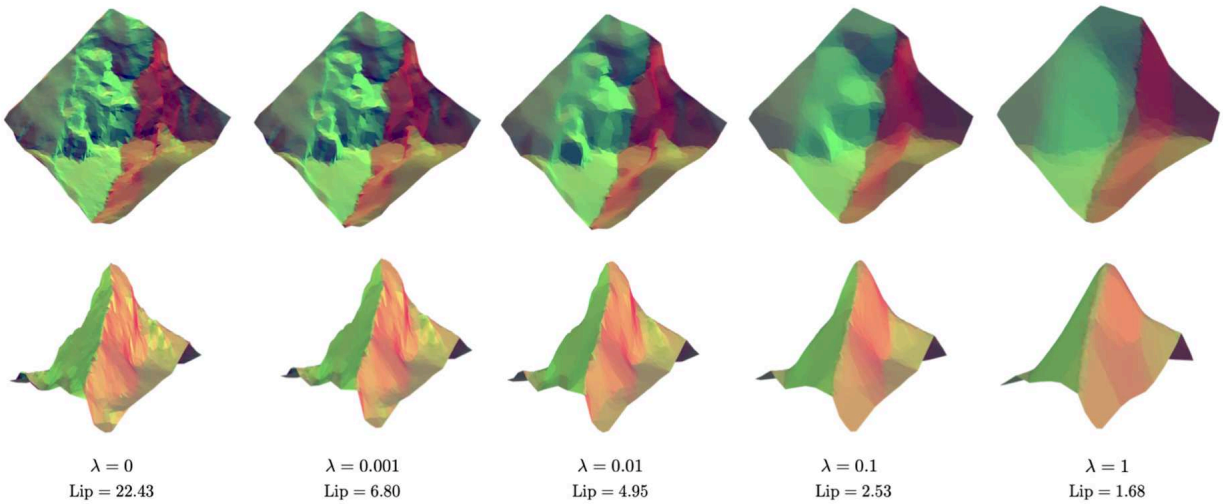


FIGURE 4: Effect of the regularization hyperparameter λ on the learned function for the evaluation map of the Matterhorn with our framework. Each column includes the results for the specified λ. The images in the first and second row give the top and side views of the learned mappings.

From (11), we then calculate the HTV of f_{Hull} as

$$\text{HTV}(f_{\text{Hull}}) = \sum_{(A,B)\in\mathcal{N}_\mathcal{T}} |\mathbf{R}_{A,B}\boldsymbol{c}| = \|\mathbf{L}\boldsymbol{c}\|_1 , \tag{38}$$

where $\mathcal{N}_\mathcal{T}$ is the set of all unique neighbor simplices in $\mathcal{T}$. In (38), $\mathbf{L}$ is a sparse matrix that is referred to as the regularization operator. It is of size $(|\mathcal{N}_\mathcal{T}|, N_g)$, and each of its rows corresponds to the HTV term associated with two neighbor simplices. Hence, there are at most $d+2$ nonzero elements at each row of $\mathbf{L}$.

C. Learning Problem

To learn the regressor $\hat{f}$ for some given data $\{\boldsymbol{x}_m, y_m\}_{m=1}^M$, we propose to solve the optimization problem

$$\hat{f} \in \underset{f\in\mathcal{X}_{\text{CPWL}}}{\arg\min} \sum_{m=1}^{M} \text{E}(f(\boldsymbol{x}_m), y_m) + \lambda \text{HTV}(f), \tag{39}$$

where $\mathcal{X}_{\text{CPWL}}$ represents the CPWL function search space. If we restrict the search space to CPWL functions f_{Hull} parameterized by a triangulation over a set of grid points $\mathbf{T}_g$, then we have that

$$\hat{f}_{\text{Hull}} \in \underset{f_{\text{Hull}}\in\mathcal{X}_{\mathbf{T}_g}}{\arg\min} \sum_{m=1}^{M} \text{E}(f_{\text{Hull}}(\boldsymbol{x}_m), y_m) + \lambda \text{HTV}(f_{\text{Hull}}). \tag{40}$$

Using the forward and regularization operators introduced in Section III.A and III.B and with $\boldsymbol{y} = (y_m)_{m=1}^M$, we rewrite (39) as

$$\hat{\boldsymbol{c}} \in \underset{\boldsymbol{c}\in\mathbb{R}^{N_g}}{\arg\min} \frac{1}{2}\|\boldsymbol{y} - \mathbf{H}\boldsymbol{c}\|_2^2 + \lambda \|\mathbf{L}\boldsymbol{c}\|_1 . \tag{41}$$

Algorithm 1 Iterations to solve (41)

Initialize : $\boldsymbol{c}_0 = \mathbf{0}, \boldsymbol{d}_0 = \mathbf{0}, t_k = 1$
for $k = 0 \to N_{\text{iter}_1}$ **do**
  $\boldsymbol{c}_{k+1} = \mathbf{Prox}_{\alpha_1,g}(\boldsymbol{d}_k - \alpha_1(\mathbf{H}^T\mathbf{H}\boldsymbol{d}_k - \mathbf{H}^T\boldsymbol{y}))$
  $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$
  $\boldsymbol{d}_{k+1} = \boldsymbol{c}_{k+1} + \frac{t_k-1}{t_{k+1}}(\boldsymbol{c}_{k+1} - \boldsymbol{c}_k)$
end for
return $\boldsymbol{c}_{k+1}$

Formulation (41) recasts the problem of finding $\hat{f}$ into a discrete problem of finding grid values $\hat{\boldsymbol{c}}$. It is generically referred to as the generalized LASSO in the literature [39], [40]. We have developed an iterative procedure based on FISTA to solve(41) [33]. We report its steps in Algorithm 1, where $g(\boldsymbol{z}) = \lambda\|\mathbf{L}\boldsymbol{z}\|_1$. We set the value of α_1 as the reciprocal of the largest eigenvalue of $\mathbf{H}^T\mathbf{H}$. The most critical calculation in Algorithm 1 is the evaluation of the proximal operator of g, which is done as

$$\mathbf{Prox}_{\alpha_1,g}(\boldsymbol{z}) = \underset{\boldsymbol{w}\in\mathbb{R}^{N_g}}{\arg\min} \left(\frac{1}{2}\|\boldsymbol{w}-\boldsymbol{z}\|_2^2 + \alpha_1\lambda\|\mathbf{L}\boldsymbol{w}\|_1\right). \tag{42}$$

In our particular setting, it is more efficient to consider the dual problem of (42), as expressed by

$$\hat{\boldsymbol{u}} \in \underset{\boldsymbol{u}\in\mathbb{R}^{|\mathcal{N}_\mathcal{T}|}}{\arg\min} \frac{1}{2}\left\|\boldsymbol{z} - \mathbf{L}^T\boldsymbol{u}\right\|_2^2 \text{ subject to } \|\boldsymbol{u}\|_\infty \le \alpha_1\lambda, \tag{43}$$

and to then invoke $\mathbf{Prox}_{\alpha_1,g}(\boldsymbol{z}) = \left(\boldsymbol{z} - \mathbf{L}^T\hat{\boldsymbol{u}}\right)$ [39]. We use another FISTA to solve (43). We report its steps in Algorithm 2. There, α_2 is set to the reciprocal of the largest eigenvalue

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

Algorithm 2 Computation of $\mathbf{Prox}_{\alpha_1,g}(\boldsymbol{z})$

Initialize : $\boldsymbol{u}_0 = \mathbf{0}, \boldsymbol{v}_0 = \mathbf{0}, t_k = 1$
for $k = 0 \to N_{\text{iter}_2}$ **do**
  $\boldsymbol{u}_{k+1} = \mathbf{Clip}(\boldsymbol{v}_k - \alpha_2(\mathbf{L}\mathbf{L}^T\boldsymbol{v}_k - \mathbf{L}\boldsymbol{z}), \alpha_1\lambda)$
  $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$
  $\boldsymbol{v}_{k+1} = \boldsymbol{u}_{k+1} + \frac{t_k-1}{t_{k+1}}(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k)$
end for
return $\left(\boldsymbol{z} - \mathbf{L}^T\boldsymbol{u}_{k+1}\right)$

of $\mathbf{L}^T\mathbf{L}$, while the function $\mathbf{Clip}(\boldsymbol{a}, \lambda)$ is defined as

$$(\mathbf{Clip}(\boldsymbol{a}, \lambda))_k = \begin{cases} -\lambda, & a_k < -\lambda \\ a_k, & |a_k| \leq \lambda \\ \lambda, & a_k > \lambda. \end{cases} \tag{44}$$

If we choose the grid points of the triangulation as the data points, then the forward matrix $\mathbf{H}$ is the identity. In this case, the solution of (41) is unique and equal to $\hat{\boldsymbol{c}} = \mathbf{Prox}_{1,g}(\boldsymbol{y})$, which is directly made accessible using Algorithm 2.

D. Final Regressor

The solution of (39) gives us the grid values $\hat{\boldsymbol{c}}$ that define uniquely the CPWL function $\hat{f}_{\text{Hull}}$. However, the domain of the definition of this function is restricted to the convex hull of the grid points. The most intuitive way to extend $\hat{f}_{\text{Hull}}$ is to use the notion of nearest neighbors and assign the value of the closest grid points to the points outside the convex hull. However, the mapping generated by using nearest neighbors is piecewise-constant outside the convex hull, which does not generate a global CPWL relation. To overcome this issue, we define our final CPWL function $\hat{f}_{\text{CPWL}}$ over $\mathbb{R}^d$ as

$$\hat{f}_{\text{CPWL}}(\boldsymbol{x}) = \begin{cases} \hat{f}_{\text{Hull}}(\boldsymbol{x}), & \boldsymbol{x} \in \text{Hull}(\mathbf{X}_g) \\ \hat{f}_{\text{Hull}}(\Phi(\boldsymbol{x})), & \text{otherwise}, \end{cases} \tag{45}$$

where Φ is the orthogonal projection of the point $\boldsymbol{x}$ onto the convex hull of the grid points. The projection on the convex hull can be formulated as a quadratic optimization problem [41]. It can be shown that the solution to this problem corresponds to CPWL functions so that our final regressor is guaranteed to be globally CPWL [42].

E. Lipschitz Constant of the Final Regressor

The Lipschitz constant of a mapping is the maximal modulus of the rate of change in the output relative to a change in the input. It measures the stability of the mapping. The Lipschitz constant of a continuous and almost-everywhere differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is known to be

$$\text{Lip}(f) = \operatorname*{ess\,sup}_{\boldsymbol{x}\in\mathbb{R}^d}(\|\boldsymbol{\nabla} f(\boldsymbol{x})\|_2), \tag{46}$$

which is the supremum of the gradient magnitudes. By plugging in (8), we find that

$$\text{Lip}(\hat{f}_{\text{Hull}}) = \max_{A\in\mathcal{T}}(\|\boldsymbol{a}_A\|_2). \tag{47}$$

Our final regressor $\hat{f}_{\text{CPWL}}$ is the composition of the projection on the convex hull Φ and $\hat{f}_{\text{Hull}}$. Since the projection onto a convex set is non-expansive [43], we have that $\text{Lip}(\Phi) = 1$. Moreover, we infer that

$$\text{Lip}(\hat{f}_{\text{CPWL}}) = \max_{A\in\mathcal{T}}(\|\boldsymbol{a}_A\|_2). \tag{48}$$

Hence, we can compute the exact Lipschitz constant of our introduced regressor. Through our experiments, we are going to show that regularization with HTV favors mappings with low Lipschitz constants.

IV. Experiments

In this section, unless stated otherwise, we choose the grid points as the training data points. In case of duplicate data points, we keep only one of them and let its target value be the average of the target values of the duplicates. This results in the forward operator $\mathbf{H}$ being identity and enables us to solve the learning problem with **Algorithm 2**. To have comparable ranges in each dimension of the input space, we standardize each feature. For the Delaunay triangulation, we use Scipy, which can safely handle data up to dimension $d = 9$ [44]. Our codes are available on the GitHub repository[1].

A. Effect of regularization

The purpose of our first experiment is to illustrate the effect of regularization. To that end, we consider the elevation map $f : \mathbb{R}^2 \to \mathbb{R}$ of the iconic Swiss mountain Matterhorn. We sample randomly 4800 points from the domain of the function[2]. Then, the mountain is reconstructed from the sampled data using our framework (DHTV). The result of the DHTV learning is reported in Figure 4 for several values of the regularization hyperparameter λ. When λ increases, we see that the number of affine pieces of the final mapping decreases. Interestingly, the regularizer tends to omit local fluctuations while preserving the main ridges. We also report the Lipschitz constant of each mapping. As expected, the Lipschitz constant decreases with an increase of λ.

B. Comparison with [31]

In two dimensions, we validate our framework by comparing it to [31] (HTV-Box). We learn a $\mathbb{R}^2 \to \mathbb{R}$ mapping using data from a face-height map[3] corrupted with additive Gaussian noise. We use 15000 training points. For our framework, we propose two alternative ways to define the grid: (i) we place grid points on a (128×128) hexagonal lattice (DHTV-Hex); or, (ii) we set them as the training data points (DHTV-Id). For fair comparisons, we also set the size of the HTV-Box lattice to (128×128). DHTV-Hex and HTV-Box have the same triangulation; therefore, they induce the same CPWL parameterization. To solve the learning task for DHTV-Hex and DHTV-Id, we use Algorithm 1

[1] https://github.com/mehrsapo/DHTV/
[2] https://lpdaac.usgs.gov/products/astgtmv003/
[3] https://www.turbosquid.com/3d-models/3d-male-head-model-1357522/

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

TABLE 1: Comparison with [31]

	N_g	Train MSE	Test MSE	HTV
HTV-Box [31]	16384	5.515×10^{-6}	2.372×10^{-6}	24.36
DHTV-Hex	16384	5.516×10^{-6}	2.372×10^{-6}	23.69
DHTV-Id	15000	4.298×10^{-6}	2.434×10^{-6}	27.31

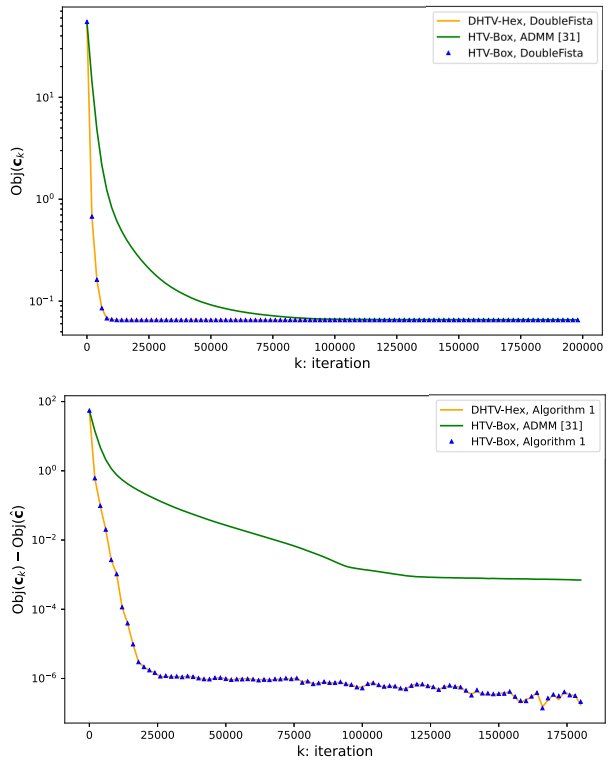


FIGURE 5: Loss of the learning task as implemented through Algorithm 1 versus the scheme used in [31].

with $\{n_{\text{iter1}} = 400,\ n_{\text{iter2}} = 500\}$ and $\{n_{\text{iter1}} = 1,\ n_{\text{iter1}} = 200000\}$, respectively. For HTV-Box, we use the alternating direction method of multipliers (ADMM) as in [31] with 200000 iterations.

In Table 1, we report the number N_g of parameters and the training and testing errors of each model for the validated regularization hyperparameter. We find that DHTV-Hex and HTV-Campos are equivalent, as expected, and that DHTV-Id has comparable performance in terms of the test error. However, we observe a difference in the HTV values. This difference for DHTV-Id is due to the different parameterization. For DHTV-Hex and HTV-Box, it originates from the faster convergence of Algorithm 1 in comparison with ADMM used in [31]. We illustrate this phenomenon in Figure 5, where we show that, if we use Algorithm 1 to solve the learning problem in [31], then DHTV-Hex and HTV-Box obtain the exact same final cost. Meanwhile, for a fixed number of iterations, ADMM yields a larger objective cost. In this figure, we have that $\text{Obj}(\boldsymbol{c}) = \frac{1}{2}\|\boldsymbol{y} - \mathbf{H}\boldsymbol{c}\|_2^2 + \lambda \|\mathbf{L}\boldsymbol{c}\|_1$, and $\hat{\boldsymbol{c}}$ is the solution with the lowest cost achieved across all methods.

C. Experiments on Real Data

For the experiments of this section, we use four different datasets suitable for regression analysis. The tasks there are to predict (i) the energy output of a power plant (Power-Plant), (ii) an average localization error (ALE), (iii) house prices (Housing with two features), and (iv) the autonomy of a car in miles per gallon (AutoMPG) based on some relevant input variables [45]–[48]. These datasets correspond to the learning of $f : \mathbb{R}^4 \to \mathbb{R}$, $f : \mathbb{R}^4 \to \mathbb{R}$, $f : \mathbb{R}^2 \to \mathbb{R}$, and $f : \mathbb{R}^7 \to \mathbb{R}$, respectively. The mappings contain 9569, 108, 20433, 398 samples, respectively. We compare four learning schemes: linear regression (LR), our framework (DHTV), neural networks (NN), and a kernel method with Gaussian radial-basis functions (RBF). For all learning schemes, we perform 30 independent runs. At each run, we split the data randomly into the train (70%), validation (15%), and test (15%) points. In our framework, at each run, we perform a grid search on the values of the regularization hyperparameter λ and retain the best λ in terms of the validation error. For the neural network, we try two fully connected networks: the first is a two-layer neural network (NN2) with 500 units in the hidden layer (253501 parameters); the second is a six-layer network (NN6) with 200 units per hidden layer (202201 parameters). To train the networks, we use the ADAM optimizer [49], a batch size of 1024, and 1000 epochs. We set the learning rate to 0.001 at the beginning and reduce it by a factor of 10 in the 400th, 600th, and 800th epochs. For these architectures (NN2 and NN6), we perform at each run a grid search on the weight-decay hyperparameter. We also perform a grid search on the hyperparameters of the RBF method and choose the model with the best validation error. To achieve a fair comparison of the HTV estimations of each model, we construct a random triangulation where the coordinates of these grid points follow a standard normal distribution in each dimension. We sample the final mapping of each model on the random grid points and calculate the HTV of the mapping using the sampled function values $\boldsymbol{c}_R$, the regularization operator of the random triangulation $\mathbf{L}_R$, and Formula (38). In addition, we define a metric of sparsity as $\frac{\||\mathbf{L}_R \boldsymbol{c}_R| \leq \epsilon\|_0}{\text{Number of rows of } \mathbf{L}_R} \times 100$. It corresponds to the percentage of almost-coplanar linear pieces over neighboring pairs of simplices. In practice, we set $\epsilon = 0.05$. We report in Table 2 the metrics averaged over 30 random splittings. The reported HTV is normalized by the mean HTV of the interpolation result or, equivalently, the generated mapping when $\lambda = 0$ in the DHTV framework. We observe that DHTV performs well in terms of prediction error in comparison with other methods, Our parameterization results in a mapping with a low fitting error and a good prediction power. Furthermore, our

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

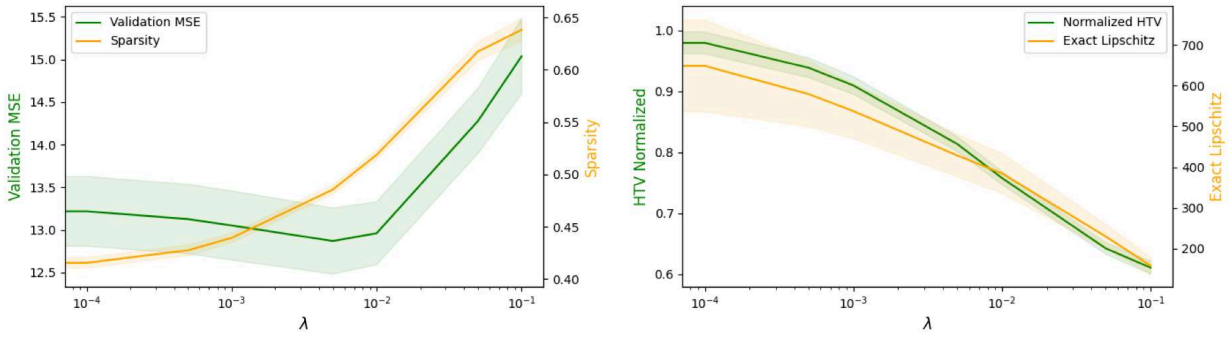


FIGURE 6: Effect of the regularization hyperparameter λ in the DHTV framework on: (i) average validation mean-squared error; (ii) sparsity; (iii) HTV; and (iv) Lipschitz constant for the PowerPlant dataset. The 90% confidence interval of each metric is depicted.

literal description of a CPWL function is easier to interpret than what is provided by a neural network. An important feature of our method is the control of the complexity of the model by the value of λ. In Figure 6, we show that tuning λ yields (i) sparser mappings while maintaining or even increasing the generalization performance, and (ii) more stable mappings in the sense of a smaller Lipschitz constant. In addition, as we can see in Table 2, the HTV and sparsity metrics are in correspondence with the prediction error of the various models, confirming that HTV is a good metric for the model complexity.

D. Discussion

Our numerical experiments showed that DHTV has a performance that is on par with that of with neural networks. The key advantages of our approach over the networks are as follows.

1) Stable parameterization: We use a simplicial parameterization where the effect of each parameter is local and stable in contrast to that of neural networks [32].
2) Global understanding of the mapping: Our model allows for a direct access to the linear regions of the mapping. In effect, this enables the computation of the exact Lipschitz constant. By contrast, the identification of the linear regions of ReLU neural networks is known to be NP-hard, while the same holds true for the determination of their Lipschitz constant [50].
3) Promotion of simplicity: The generated mappings are solutions to a variational optimization problem with HTV regularization that has an explainable geometric effect on the mapping. More precisely, HTV is a relaxation of the number of affine pieces of a CPWL function and, thus, favors simpler models.
4) Training with global-minima guarantees: Our optimization problem is convex. This is unlike the training of neural networks, which involves a highly non-convex optimization task.

TABLE 2: Comparisons of Metrics for different datasets

Data	Method	Train MSE	Test MSE	HTV	Sparsity
PowerPlant	LR	20.81	20.48	0.00	100%
	DHTV	1.338	**12.33**	0.82	48.4%
	NN2	9.497	13.23	1.85	36.2%
	NN6	6.882	13.22	1.57	31.7%
	RBF	11.43	14.04	1.90	29.8%
ALE	LR	0.0558	0.0581	0.00	100%
	DHTV	0.0220	0.0494	0.75	0.78%
	NN2	0.0171	**0.0460**	0.92	50.4%
	NN6	0.0130	0.0503	1.22	42.2%
	RBF	0.0131	0.0508	1.39	38.8%
Housing	LR	10128	10074	0.00	100%
	DHTV	1283.5	**2660.5**	0.81	27%
	NN2	4458.7	4615.8	1.73	9%
	NN6	2644.8	3230.8	1.65	11%
	RBF	4680.1	5150.0	3.56	17%
AutoMPG	LR	10.861	10.631	0.00	100%
	DHTV	1.168	**7.474**	0.79	83.8%
	NN2	3.813	7.500	0.86	84.0%
	NN6	4.182	8.375	1.04	81.5%
	RBF	2.772	7.958	0.74	82.8%

These four features make our method interpretable. Although our method can be theoretically used in any dimension, it is challenging to deploy for $d \geq 10$ in practice, mainly due to the construction of the Delaunay triangulation, which faces the curse of dimensionality. Neural networks, by contrast, can handle data in arbitrary high dimensions, which is one of the main reasons for their popularity.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

V. Conclusion

We have proposed a novel regression method referred to as Delaunay Hessian total variation (DHTV). Our approach provides continuous and piecewise-linear (CPWL) mappings—the same class of functions generated by ReLU networks. We employ a parameterization based on the Delaunay triangulation of the input space. This parameterization represents any CPWL function by its samples on a grid. Unlike deep networks, it is straightforward to understand the effect of the model parameters on the mapping, which makes the proposed model directly interpretable. We have formulated the learning process as a convex minimization task. The Hessian total variation (HTV) regularization was used to control the complexity of the generated mapping. HTV has an intuitive formula for CPWL functions which involves a summation over individual affine pieces. We have shown that the HTV of the proposed model is the ℓ_1-norm applied to a linear transformation of the grid-point values. This result has enabled us to recast the learning problem as the generalized least absolute shrinkage and selection-operator. By a clever choice of grid points, we use the fast iterative shrinkage-thresholding algorithm to solve the optimization problem. Our experiments show that the HTV regularizing leads to simple models while preserving the generalization power. In future works, we plan to investigate the removal of unnecessary grid points based on their contribution to the HTV, which could be used for mesh-simplification purposes.

VI. Acknowledgement

The authors would like to thank Shayan Aziznejad and Joaquim Campos for having fruitful discussions.

REFERENCES

[1] T. Hastie, R. Tibshirani, and J. Friedman, "Overview of supervised learning," in *The Elements of Statistical Learning*. Springer, 2009, pp. 9–41.

[2] Y. Tian and Y. Zhang, "A comprehensive survey on regularization strategies in machine learning," *Information Fusion*, vol. 80, pp. 146–166, 2022.

[3] B. Schlkopf, A. J. Smola, and F. Bach, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, 2018.

[4] J. Gross and J. Groß, *Linear Regression*. Springer Science & Business Media, 2003, vol. 175.

[5] H. Kadri, E. Duflos, P. Preux, S. Canu, and M. Davy, "Nonlinear functional regression: A functional RKHS approach," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010, pp. 374–380.

[6] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171 – 1220, 2008.

[7] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational Learning Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 416–426.

[8] R. Schaback, "A practical guide to radial basis functions."

[9] A. S. Miller, B. H. Blott, and T. K. hames, "Review of neural network applications in medical imaging and signal processing," *Medical and Biological Engineering and Computing*, vol. 30, no. 5, pp. 449–464, 1992.

[10] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *Journal of Machine Learning Research*, vol. 7, no. 95, pp. 2651–2667, 2006.

[11] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[12] K. Pentoś, J. T. Mbah, K. Pieczarka, G. Niedbała, and T. Wojciechowski, "Evaluation of multiple linear regression and machine learning approaches to predict soil compaction and shear stress based on electrical parameters," *Applied Sciences*, vol. 12, no. 17, p. 8791, 2022.

[13] K. J. Assi, K. M. Nahiduzzaman, N. T. Ratrout, and A. S. Aldosary, "Mode choice behavior of high school goers: Evaluating logistic regression and mlp neural networks," *Case studies on transport policy*, vol. 6, no. 2, pp. 225–230, 2018.

[14] D. Rajković, A. M. Jeromela, L. Pezo, B. Lončar, N. Grahovac, and A. K. Špika, "Artificial neural network and random forest regression models for modelling fatty acid and tocopherol content in oil of winter rapeseed," *Journal of Food Composition and Analysis*, vol. 115, p. 105020, 2023.

[15] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with ReLU activation function," *The Annals of Statistics*, vol. 48, no. 4, pp. 1875–1897, 2020.

[16] G. Montúfar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, p. 2924–2932.

[17] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," *arXiv preprint arXiv:1611.01491*, 2016.

[18] S. Park, C. Yun, J. Lee, and J. Shin, "Minimum width for universal approximation," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7*, 2021.

[19] F.-L. Fan, J. Xiong, M. Li, and G. Wang, "On interpretability of artificial neural networks: A survey," *IEEE Transactions on Radiation and Plasma Medical Sciences*, vol. 5, no. 6, pp. 741–760, 2021.

[20] B. Dherin, M. Munn, M. Rosca, and D. G. Barrett, "Why neural networks find simple solutions: The many regularizers of geometric complexity," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[21] G. Zhang, C. Wang, B. Xu, and R. Grosse, "Three mechanisms of weight decay regularization," in *International Conference on Learning Representations*, 2019.

[22] R. Parhi and R. D. Nowak, "Banach space representer theorems for neural networks and ridge splines," *J. Mach. Learn. Res.*, vol. 22, no. 1, jul 2022.

[23] M. Unser, "Ridges, neural networks, and the Radon transform," *arXiv preprint arXiv:2203.02543*, 2022.

[24] G. Ongie, R. Willett, D. Soudry, and N. Srebro, "A function space view of bounded norm infinite width ReLU nets: The multivariate case," *arXiv preprint arXiv:1910.01635*, 2019.

[25] R. Eldan and O. Shamir, "The power of depth for feedforward neural networks," in *Conference on learning theory*. PMLR, 2016, pp. 907–940.

[26] T. Debarre, J. Fageot, H. Gupta, and M. Unser, "B-spline-based exact discretization of continuous-domain inverse problems with generalized TV regularization," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4457–4470, 2019.

[27] T. Debarre, Q. Denoyelle, M. Unser, and J. Fageot, "Sparsest piecewise-linear regression of one-dimensional data," *Journal of Computational and Applied Mathematics*, vol. 406, p. 114044, 2022.

[28] M. Unser, "A unifying representer theorem for inverse problems and machine learning," *Foundations of Computational Mathematics*, vol. 21, no. 4, pp. 941–960, 2021.

[29] S. Aziznejad, J. Campos, and M. Unser, "Measuring complexity of learning schemes using Hessian-Schatten total-variation," *SIAM Journal on Mathematics of Data Science, in press, https://arxiv.org/pdf/2112.06209.pdf*, 2023.

[30] S. Lefkimmiatis, A. Bourquard, and M. Unser, "Hessian-based norm regularization for image restoration with biomedical applications," *IEEE Transactions on Image Processing*, vol. 21, no. 3, pp. 983–995, 2011.

[31] J. Campos, S. Aziznejad, and M. Unser, "Learning of continuous and piecewise-linear functions with Hessian total-variation regularization," *IEEE Open Journal of Signal Processing*, vol. 3, pp. 36–48, 2022.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

This article has been accepted for publication in IEEE Open Journal of Signal Processing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/OJSP.2023.3250104

[32] A. Goujon, J. Campos, and M. Unser, "Stable parametrization of continuous and piecewise-linear functions," *arXiv preprint arXiv:2203.05261*, 2022.

[33] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[34] V. T. Rajan, "Optimality of the Delaunay triangulation in $\mathbb{R}^d$," *Discrete & Computational Geometry*, vol. 12, no. 2, pp. 189–202, 1994.

[35] D.-T. Lee and B. J. Schachter, "Two algorithms for constructing a Delaunay triangulation," *International Journal of Computer & Information Sciences*, vol. 9, no. 3, pp. 219–242, 1980.

[36] P. Cignoni, C. Montani, and R. Scopigno, "Dewall: A fast divide and conquer Delaunay triangulation algorithm in Ed," *Computer-Aided Design*, vol. 30, no. 5, pp. 333–341, 1998.

[37] Y. Liu and G. Yin, "The Delaunay triangulation learner and its ensembles," *Computational Statistics & Data Analysis*, vol. 152, p. 107030, 2020.

[38] L. Blumenthal and B. Gillam, "Distribution of points in n-space," *The American Mathematical Monthly*, vol. 50, no. 3, pp. 181–185, 1943.

[39] R. J. Tibshirani and J. Taylor, "The solution path of the generalized LASSO," *The Annals of Statistics*, vol. 39, no. 3, pp. 1335–1371, 2011.

[40] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, p. 1–122, jan 2011.

[41] Z. Gabidullina, "The problem of projecting the origin of Euclidean space onto the convex polyhedron," *Lobachevskii Journal of Mathematics*, vol. 39, no. 1, pp. 35–45, 2018.

[42] J. Spjøtvold, P. Tøndel, and T. Johansen, "Continuous selection and unique polyhedral representation of solutions to convex parametric quadratic programs," *Journal of Optimization Theory and Applications*, vol. 134, no. 2, pp. 177–189, 2007.

[43] M. Kell, "Symmetric orthogonality and non-expansive projections in metric spaces," *manuscripta mathematica*, vol. 161, no. 1, pp. 141–159, 2020.

[44] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, 2020.

[45] P. Tüfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods," *International Journal of Electrical Power & Energy Systems*, vol. 60, pp. 126–140, 2014.

[46] A. Singh, V. Kotiyal, S. Sharma, J. Nagar, and C.-C. Lee, "A machine learning approach to predict the average localization error with applications to wireless sensor networks," *IEEE Access*, vol. 8, pp. 208 253–208 263, 2020.

[47] R. K. Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997.

[48] J. R. Quinlan, "Combining instance-based and model-based learning," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 236–243.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[50] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," *Advances in Neural Information Processing Systems*, vol. 31, pp. 3839–3848, 2018.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/