Mathematical Programming
https://doi.org/10.1007/s10107-023-02023-6

FULL LENGTH PAPER

Series A

Efficient Kirszbraun extension with applications to regression

Hananel Zaichyk[1] · **Armin Biess**[1] · **Aryeh Kontorovich**[1] · **Yury Makarychev**[2]

Received: 1 March 2022 / Accepted: 21 September 2023

Abstract

We introduce a framework for performing vector-valued regression in finite-dimensional Hilbert spaces. Using Lipschitz smoothness as our regularizer, we leverage Kirszbraun's extension theorem for off-data prediction. We analyze the statistical and computational aspects of this method—to our knowledge, its first application to supervised learning. We decompose this task into two stages: training (which corresponds operationally to smoothing/regularization) and prediction (which is achieved via Kirszbraun extension). Both are solved algorithmically via a novel multiplicative weight updates (MWU) scheme, which, for our problem formulation, achieves significant runtime speedups over generic interior point methods. Our empirical results indicate a dramatic advantage over standard off-the-shelf solvers in our regression setting.

Keywords Convex optimization · Quadratically constrained quadratic program · Kirszbraun extension · Regression

Mathematics Subject Classification 65K05 · 90C20 · 62J02

✉ Aryeh Kontorovich
karyeh@bgu.ac.il

Hananel Zaichyk
zaichyk@gmail.com

Armin Biess
abiess@bgu.ac.il

Yury Makarychev
yury@ttic.edu

[1] Ben-Gurion University of the Negev, Be'er Sheva, Israel

[2] Toyota Technological Institute at Chicago, Chicago, IL, USA

Published online: 07 December 2023

1 Introduction

Regression The classical problem of estimating a continuous-valued function from noisy observations, known as regression, is of central importance in statistical theory with a broad range of applications; see, for example, [20, 31]. When the target function is assumed to have a specific structure, the regression problem is termed *parametric* and the optimization problem is (effectively) finite-dimensional.[1] Linear regression, for example [30, chapter 10.3.1], is perhaps the simplest and most common type of parametric regression. When no structural assumptions concerning the target function are made, the regression problem is *nonparametric*. Informally, the main objective in the study of nonparametric regression is to understand the relationship between the regularity conditions that a function class might satisfy (e.g., Lipschitz or Hölder continuity, or sparsity in some representation) and its behavior vis-à-vis optimization and generalization. Most existing algorithms for regression either focus on the scalar-valued case or else reduce multiple outputs to several scalar problems [5], see the paragraph on related work, below in this section.

Convex optimization Many learning problems can be cast in the framework of convex optimization. In particular, regression naturally lends itself to this formulation. While some cases, such as linear regression, admit efficient closed-form solutions, this is not the case in general. Typically, convex optimization problems are solved via iterative methods up to a specified accuracy. One general approach is the interior-point methods, which, on problems with n variables and m constraints achieves a runtime of $O(\max\{n^3, n^2m, F\})$, where F is the cost of evaluating the first and second derivatives of the objective and the constraints [6].

Motivation and contribution The chief motivation of this work was to generalize the results of [17] to vector-valued output, which amounts to learning a regression problem between two vector spaces, via a *Lipschitz extension* technique. To our knowledge, this is the first application of this technique to multidimensional supervised learning.

Briefly, we are given a dataset $\{x_1, \ldots, x_n\}$ residing in $\mathbb{R}^a$ labeled by $\{y_1, \ldots, y_n\}$ in $\mathbb{R}^b$, as well as a "smoothness budget" captured by the constant $L > 0$.[2] The learning proceeds via two natural phases, which are formally described in Sect. 3. First is the training, or Empirical Risk Minimization (ERM) or yet *smoothing* phase, in which we modify the labels from $\{y_1, \ldots, y_n\}$ to $\{z_1, \ldots, z_n\}$ so as to minimize the distortion with respect to the original labels $\{y_i\}$ while ensuring that the data satisfies the L-Lipschitz constraint (1). Computationally, this procedure may be cast as a Quadratically Constrained Quadratic Program (QCQP) problem with bn variables and $O(n^2)$ constraints:

[1] Even when the problem is formally infinite-dimensional, such as with SVM, the Representer Theorem [25] shows that the solution is spanned by the finite sample.

[2] As explained in Sect. 3, there is no need to assume that L is given, as this hyper-parameter can be tuned via Structural Risk Minimization (SRM).

$$\begin{aligned} &\underset{\widetilde{\mathbf{Y}}}{\text{minimize}}\ \Phi(\mathbf{Y}, \widetilde{\mathbf{Y}}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2 \\ &\text{subject to } \|\tilde{y}_i - \tilde{y}_j\| \le L\|x_i - x_j\|, \qquad i, j \in [n]. \end{aligned} \tag{1}$$

The second phase is *prediction* or generalization on unseen data. We accomplish this via Kirszbraun extension of the smoothed data $(x_i, z_i)_{i\in[n]}$ to a new point x^*. Given a finite sequence $(x_i)_{i\in[n]} \subset X = \mathbb{R}^a$, its image $(y_i)_{i\in[n]} \subset Y = \mathbb{R}^b$ under some L-Lipschitz map $f : X \to Y$ (defined formally in Sect. 2), and a test point x^*, we wish to solve the following existence problem:

$$\begin{aligned} &\text{exist? } y^* \in \mathbb{R}^b \\ &\text{subject to } \left\|y^* - y_i\right\| \le L\left\|x^* - x_i\right\|, \quad i = 1, \ldots, m. \end{aligned}$$

Kirszbraun's Theorem [26] guarantees the existence of a solution, and our key contribution is an efficient algorithm for approximating it. Although general QCQP problems are not convex, our special instance is—and as such is, in principle, amenable to the standard convex optimization framework, such as interior point methods.

Attempts to numerically solve our optimization problem using Matlab's off-the-shelf solvers indicated that these were incapable of handling our framework. Even for relatively small problems, with a sample size of 200, the optimizer was not able to complete the run within 12 h. This is due to the number of constraints $m = O(n^2)$ for a sample of size n, runtime $O(\max\{n^3, n^2 m, F\}) \subset O(\max\{n^4, F\})$, as discussed above.

The lack of a dedicated solver for Kirszbraun extension motivated us to develop two new optimization algorithms. Both rely on the Multiplicative Weights Update (MWU) scheme, which is formally described in Sect. 3. We solve the smoothing problem, up to constant additive precision, in runtime $O(ma+m^{3/2}(\log m)^2 b)$ and the extension problem in runtime $O(na + nb \log n)$. When solving large-scale problems, even a modest improvement in the exponent of the solver's running time yields dramatic savings—and indeed, our improvement from $O(n^4)$ to $\tilde{O}(n^3)$ in the ERM phase and from $O(n^3)$ to $\tilde{O}(n)$ in the prediction phase produced palpable results.[3] The latter is of particular significance, since typically the ERM phase is performed once during "offline" training while extension is performed many times during "online" prediction. Thus, our main contributions are (a) on the statistical front to give a novel application of Kirszbraun extension to the problem of regression and (b) on the algorithmic front, to provide efficient algorithms for solving the optimization problems inherent in (a) as well as the more general class of QCQP problems we termed *Laplace's problem* in Sect. 3.1.

Our experiments in Sect. 4 show that our specialized algorithm significantly outperforms general-purpose QCQP solvers. Although our main contributions are algorithmic, a statistical analysis of our regression method is provided in "Appendix C".

Related work Previous approaches to vector-valued regression include ε-insensitive SVM with p-norm regularization [8], least-squares and MLE-based methods [23], and

[3] A further improvement via the use of spanners allows reducing the number of constraints m from $O(n^2)$ to $O(n)$ and hence the ERM runtime to $\tilde{O}(n^{3/2})$, as detailed in Sect. 3.1.

(for linear models) the Danzig selector [11]. According to a recent survey [5], existing methods essentially "transform the multi-output problem into independent single-output problems." Some approaches to multitask learning problems do exploit relations between the different tasks [10]. In econometrics, the decoupling of the outputs is made explicit in the Seemingly Unrelated Regressions (SUR) model [15, 18, 19]. These approaches, however, do not seem to encapsulate the need of a single vector output with possibly strong relations between its coordinates. In our approach, we devise a principled approach for leveraging the dependencies via Kirszbraun extension. The latter has previously been applied by [28] to dimensionality reduction (unsupervised learning), but to the best of our knowledge has not been used in the supervised learning setting. Very recently, a rather general regression setting studying mappings between two metric spaces was studied [13]; notably, the Lipschitz extension approach does not apply in such generality (as discussed therein). Additionally, the stringent condition of uniform Lipschitz smoothness of the regressor can be relaxed to a more forgiving average-case notion [4] and further to average Hölder smoothness [21].

Both of our problems (smoothing and extension) may be formulated as QCQP problems, whose most general form is

$$\begin{aligned} &\text{minimize } x^\top P_0 x + a^\top x \\ &\text{subject to } x^\top P_i x + a_i^\top x \le b_i, \quad i = 1, \ldots, m, \end{aligned}$$

where a, $a_{i\in[m]}$ and x are vectors, P_0, $P_{i\in[m]}$ are matrices, and the b_i are scalars. The general problem is NP-hard, but when all of the P_i are semi-definite, the problem is convex and can be solved in polynomial time [6]. QCQP problems are usually solved in practice using log-barrier or primal-dual interior-point methods. The running time of an optimization algorithm based on interior-point methods significantly depends on the problem at hand. Specifically, consider a problem with N variables and m constraints. In order to obtain a $(1+\varepsilon)$-approximate solution, the algorithm has to perform $\Theta(\sqrt{m}\log(1/\varepsilon))$ iterations in the worst case [33, Chapter 6]. In each iteration, the algorithm has to initialize and invert an $N \times N$ Hessian matrix (or equivalently solve a system of N linear equations with N variables). The time required to initialize the Hessian matrix is problem specific: while it is $O(mN^2)$ in the worst case, it is often significantly less than that. The Hessian matrix can be inverted in $O(N^\omega)$ time, where ω is the matrix multiplication exponent [9] (the best current upper bound on ω is 2.37286 [1]). However, to the best of our knowledge, all implementations used in practice perform this step in $O(N^3)$ time. That said, this step can be significantly sped up if the Hessian matrix has a special structure.

Our Multiplicative Weights Update (MWU) scheme is based on the framework of [2]. We include the relevant background and their results in "Appendix A" for completeness.

Main results We cast the general regression problem between Hilbert spaces as two QCQPs, and provide an efficient algorithm for each problem.

The problem setup, formalized in Sect. 3, involves a dataset of size n of vectors in an a-dimensional Euclidean space labeled by b-dimensional vectors. The *smoothing*

(also: training, regularization, denoising) problem (Sect. 3.1) is to perturb the labels so as to achieve the user-specified Lipschitz smoothness constraint while incurring a minimum distortion. This is a standard statistical technique, known as *regularization*, which prevents overfitting in prediction. Our Theorem 1 solves the smoothing optimization problem, up to a tolerance $\varepsilon \in (0, 1/2)$, in runtime

$$O(an^2 + bn^3(\log n)^2 \log(1/\varepsilon)/\varepsilon^{5/2}).$$

Next, we address the task of prediction (i.e., assigning a label to a test point). In Theorem 2, we accomplish this via ε-approximate Kirszbraun extension of the smoothed dataset, in runtime $O(an + bn(\log n)/\varepsilon^2)$. We also provide a data structure for answering Lipschitz extension queries, which is useful when a is small. The data structure can be constructed in time $2^{O(a)} n \log n$. Then it answers Lipschitz extension/prediction queries in per-query time

$$(1/\varepsilon)^{O(a)}(b + \log n).$$

In Sect. 4, we compare the performance of our MWU-based approach to a generic off-the-shelf interior-point based solver and report a significant runtime advantage, which allows processing larger samples and ultimately yields greater accuracy.

Finally, for completeness, in "Appendix C", we include a Rademacher-based analysis of the generalization error of our regression algorithm. Our MATLAB code is publicly available for reproducibility. [4]

2 Formal setup

Metric space and Lipshcitz function A metric space (X, d_X) is a set X equipped with a symmetric function $d_X : X^2 \to [0, \infty)$ satisfying $d_X(x, x') = 0 \iff x = x'$ and the triangle inequality. Given two metric spaces (X, d_X) and (Y, d_Y), a function $f : X \to Y$ is L-*Lipschitz* if $d_Y(f(x), f(x')) \leq L d_X(x, x')$ for all $x, x' \in X$; its *Lipschitz* constant $\|f\|_{\text{Lip}}$ is the smallest L for which the latter inequality holds. For any metric space (X, d_X) and $A \subseteq X$, the following classic *Lipschitz extension* result, essentially due to [29, 36], holds. If $f : A \to \mathbb{R}$ is Lipschitz (under the inherited metric of d_X on A) then there is an *extension* $f^* : X \to \mathbb{R}$ that coincides with f on A and $\|f\|_{\text{Lip}} = \|f^*\|_{\text{Lip}}$. A *Hilbert space* H is a vector space (in our case, over $\mathbb{R}$) equipped with an inner product $\langle \cdot, \cdot \rangle : H^2 \to \mathbb{R}$, which is a positive-definite symmetric bilinear form, which is further complete in the metric $d_H(x, x') := \sqrt{\langle x - x', x - x' \rangle}$ (in the sense of Cauchy sequences converging to points in H, see e.g. [34] for background). Under the definition of a metric space (X, d_X), for every $x \in X$ we can define $\text{Ball}(x, r) = \{x' \in X : d_X(x, x') < r\}$. For positive integers n, the set $\{1, \ldots, n\}$ will be denoted by $[n]$. We use standard graph-theoretic definitions:

[4] https://github.com/HananZaichyk/Kirszbraun-extension.

an undirected graph $G = (V, E)$ is defined by a finite set of vertices V and edges $E \subset V^2$; see, e.g., [7] for reference.

Kirszbraun theorem Kirszbraun [26] proved that for two Hilbert spaces (X, d_X) and (Y, d_Y), and f mapping $A \subseteq X$ to Y, there is an extension $f^* : X \to Y$ such that $\|f\|_{\text{Lip}} = \|f^*\|_{\text{Lip}}$. This result is in general false for Banach spaces whose norm is not induced by an inner product [32].

Learning problem We assume a familiarity with the abstract agnostic learning framework and refer the reader to [30] for background. Our approach will be applied to learn a mapping between two Hilbert spaces, (X, d_X) and (Y, d_Y). We assume a fixed unknown distribution P on $X \times Y$ and a labeled sample $(x_i, y_i)_{i\in[n]}$ of input–output examples. The *risk* of a given mapping $f : X \to Y$ is defined as $R(f) = \mathbb{E}_{(x,y)\sim P}[d_Y(f(x), y)]$; implicit here is our designation of the metric of Y as the loss function. Analogously, the empirical risk of f on a labeled sample is given by $\hat{R}_n(f) = n^{-1}\sum_{i\in[n]} d_Y(f(x_i), y_i)$. In this paper, we always take $X = \mathbb{R}^a$ and $Y = \mathbb{R}^b$, each equipped with the standard Euclidean metric. Uniform deviation bounds on $|R(f) - \hat{R}_n(f)|$, over all f with $\|f\|_{\text{Lip}} \leq L$ are given in "Appendix C".

3 Learning algorithm

Overview We follow the basic strategy proposed by [17] for real-valued regression. That is, we propose a learning paradigm based on Lipschitz functions. This is done in two phases *Smoothing* as described in Sect. 3.1 and *Lipschitz extension* as described in Sect. 3.2 In the smoothing phase, our Hypothesis class is

$$\mathscr{H}_L = \{h : X \to Y : h \text{ is } L\text{-Lipshchitz}\}$$

and the ERM process can be interpreted as picking the L-Lipschitz function $\hat{h}$ which fits best to the sample in terms of the Mean Squared Error. The Extension phase, which amounts to predicting the label of a new point, is simply assigning a value on a new point $\hat{h}(x^*)$ in a way that $\hat{h}$ will remain L-Lipschitz on $X \bigcup \{x^*\}$

We are given a labeled sample $(x_i, y_i)_{i\in[n]}$, where $x_i \in X := \mathbb{R}^a$ and $y_i \in Y := \mathbb{R}^b$. For a user-specified Lipschitz constant $L > 0$, we compute the (approximate) Empirical Risk Minimizer (ERM) $\hat{f} := \text{argmin}_{f\in F_L} \hat{R}_n(f)$ over $F_L := \{f \in Y^X : \|f\|_{\text{Lip}} \leq L\}$. (A standard method for tuning L is via Structural Risk Minimization (SRM): One computes a generalization bound $R(\hat{f}) \leq \hat{R}_n(\hat{f}) + Q_n(a, b, L)$, where $Q_n(a, b, L) := \sup_{f\in F_L} |R(f) - \hat{R}_n(f)| = O(L/n^{a+b+1})$, as derived in "Appendix C", and chooses $\hat{L}$ to minimize this. We omit this standard stage of the learning process.)

Predicting the value at a test point $x^* \in X$ amounts to Lipschitz-extending $\hat{f}$ from $\{x_i : i \in [n]\}$ to $\{x_i : i \in [n]\} \cup \{x^*\}$. Equivalently, the ERM stage may be viewed as a *smoothing* procedure, where $\tilde{y}_i := \hat{f}(x_i)$ and $(x_i, \tilde{y}_i)_{i\in[n]}$ is the *smoothed sample*—

which is then (approximately) Lipschitz-extended to x^*. We proceed to describe each stage in detail.

3.1 Smoothing

Problem statement We reformulate the ERM problem $\hat{f} = \operatorname{argmin}_{f \in F_L} \hat{R}_n(f)$ as follows. Given two sets of vectors, $(x_i, y_i)_{i\in[n]}$, where $x_i \in X := \mathbb{R}^a$ and $y_i \in Y := \mathbb{R}^b$, we wish to compute a "smoothed" version $\tilde{y}_i$ of the y_i's so as to

$$\begin{aligned} \underset{\widetilde{\mathbf{Y}}}{\text{minimize}}\ & \Phi(\mathbf{Y}, \widetilde{\mathbf{Y}}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2 \\ \text{subject to}\ & \|\tilde{y}_i - \tilde{y}_j\| \le L\|x_i - x_j\|, \qquad i, j \in [n], \end{aligned}$$

where $\Phi(\mathbf{Y}, \widetilde{\mathbf{Y}}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2$ is the distortion, and $\|\tilde{y}_i - \tilde{y}_j\| \le L\|x_i - x_j\|$ for all $i, j \in [n]$ are the Lipschitz constraints. Here, $\mathbf{Y} = (y_1, \ldots, y_n)$ and $\widetilde{\mathbf{Y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)$ (the columns of matrices $\mathbf{Y}$ and $\widetilde{\mathbf{Y}}$ are vectors $y_1, \ldots, y_n$ and $\tilde{y}_1, \ldots, \tilde{y}_n$, respectively). Notice that when we use the L_2 norm, this problem is a quadratically constrained quadratic program (QCQP). We refer to that problem as the *smoothing problem*.

To design an efficient algorithm for the smoothing problem, we consider a more general variant of this problem where we are given a graph $G = (\{1, \ldots, n\}, E)$, and the goal is to ensure that the Lipschitz constraints $\|\tilde{y}_i - \tilde{y}_j\| \le L\|x_i - x_j\|$ hold for all $(i, j) \in E$. We note that the smoothing problem corresponds to the case where E is the complete graph. Importantly, if the doubling dimension ddim X is low, we can solve the original problem by letting $([n], E)$ be a $(1+\varepsilon)$-stretch spanner;[5] then $m = n(1/\varepsilon)^{O(\text{ddim})}$ (this approach was previously used by [17]; see also [22, Section 8.2], who used a similar approach to compute the doubling constant). Our algorithm for Lipschitz Smoothing iteratively solves Laplace's problem (see Problem 1) in the graph G. We proceed to define this problem and present a closed-form formula for the solution.

Problem 1 (Laplace's problem) *We are given vectors* $\mathbf{Y} = \{y_i, i \in [n]\}$*, graph* G*, and additionally vertex weights* $\{\lambda_i \ge 0$ *(for* $i \in [n]\}$*) and edge weights* $\{\mu_{ij} \ge 0$ *(for* $(i, j) \in E)\}$*. Our goal is to find* $\widetilde{\mathbf{Y}} = \{\tilde{y}_i, i \in [n]\}$ *so as to*

$$\underset{\widetilde{\mathbf{Y}}}{\text{minimize}}\ \ \Psi(\mathbf{Y}, \widetilde{\mathbf{Y}}, \{\lambda_i\}, \{\mu_{ij}\}) \equiv \sum_{i=1}^{n} \lambda_i \|y_i - \tilde{y}_i\|^2 + \sum_{(i,j)\in E} \mu_{ij} \|\tilde{y}_i - \tilde{y}_j\|^2.$$

Let $\mathscr{L}$ be the Laplacian of $G = ([n], E)$ with edge weights μ_{ij}; that is, $\mathscr{L}_{ii} = \sum_{j:j\neq i} \mu_{ij}$ (diagonal entries), $\mathscr{L}_{ij} = -\mu_{ij}$ for $(i, j) \in E$, and $\mathscr{L}_{ij} = 0$ for $(i, j) \notin E$, $i \neq j$. Let $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$. We solve Laplace's problem using the standard

[5] A *spanner* of a graph is a sub-graph simultaneously enjoying sparsity and distance-preserving properties. The reader is referred to [17, 22] for the relevant background and in particular, the precise definition of a $(1+\varepsilon)$-stretch spanner and doubling dimension.

approach. We write the gradient of objective Ψ with respect to $\tilde{y}_i$ (assuming all other parameters are fixed):

$$\begin{aligned} 2\lambda_i(\tilde{y}_i - y_i) + 2\sum_{j:(i,j)\in E} \mu_{ij}(\tilde{y}_i - \tilde{y}_j) &= 2\left(\lambda_i + \sum_{j:(i,j)\in E} \mu_{ij}\right)\tilde{y}_i \\ &\quad - 2\sum_{j:(i,j)\in E} \mu_{ij}\tilde{y}_j - 2\lambda_i y_i \\ &= 2\left((\mathscr{L} + \Lambda)\,\tilde{Y}^\top\right)_i - 2(\Lambda Y^\top)_i. \end{aligned}$$

For the optimal solution $\tilde{Y}$, the gradient must be equal 0 for every i. Thus, the minimum is attained at:

$$(\mathscr{L} + \Lambda)\,\widetilde{\mathbf{Y}}^\top = \Lambda \mathbf{Y}^\top.$$

This equation can be solved separately for each of b rows of $\mathbf{Y}^\top$ using a nearly-linear equation solver for diagonally dominant matrices by [27] in total time $O(bm \log n \log(1/\varepsilon))$ (see also [35], which presented the first nearly-linear time solve for diagonally dominant matrices).

We solve the Lipschitz Smoothing problem via the multiplicative weights update algorithm LipschitzSmooth, presented below (see Algorithm 1). It was inspired by the algorithm for finding maximum flow using electrical networks by [12]. Notice that in the following analysis we prove that by defining λ and $\{\mu_{ij}\}$ as in Algorithm 1 LipschitzSmooth, solving Laplace's problem will give us an approximate solution to the *smoothing problem*

Analysis Let $\mathbf{Y}^*$ be the optimal solution to the Lipschitz Smoothing problem and and Φ_0 be a $(1+\varepsilon)$ approximation to the optimal value; that is,

$$\Phi(\mathbf{Y}, \mathbf{Y}^*) \le \Phi_0 \le (1+\varepsilon)\Phi(\mathbf{Y}, \mathbf{Y}^*)$$

(we assume that Φ_0 is given to the algorithm; note that Φ_0 can be found by binary search).

We use the multiplicative-weight update (MWU) method. Let

$$\begin{aligned} h_\Phi(\widetilde{\mathbf{Y}}) &= 1 - \sqrt{\frac{\Phi(\mathbf{Y}, \widetilde{\mathbf{Y}})}{\Phi_0}}, \\ h_{ij}(\widetilde{\mathbf{Y}}) &= 1 - \frac{\|\tilde{y}_i - \tilde{y}_j\|}{M\|x_i - x_j\|}, \qquad (i,j) \in E. \end{aligned}$$

Note that functions h_Φ and h_{ij} are concave.

Observe that $h_\Phi(\widetilde{\mathbf{Y}}^*) \ge 0$ and $h_{ij}(\widetilde{\mathbf{Y}}^*) \ge 0$ for every $(i,j) \in E$. On the other hand, if $h_\Phi(\widetilde{\mathbf{Y}}) \ge -\varepsilon$ and $h_{ij}(\widetilde{\mathbf{Y}}) \ge -\varepsilon$, then

$$\Phi(\mathbf{Y}, \widetilde{\mathbf{Y}}) \le (1+\varepsilon)^2 \Phi_0$$

Algorithm 1 LipschitzSmooth

Require: vectors $\mathbf{X} = x_1, \ldots, x_n \in \mathbb{R}^a$, $\mathbf{Y} = y_1, \ldots, y_n \in \mathbb{R}^b$, graph $G = ([n], E)$, M and Φ_0
Output: $\widetilde{\mathbf{Y}}$

let $\mathbf{Y} \equiv (y_1, \ldots, y_n)$
let $r_{ij} = L\|x_i - x_j\|$ for $(i, j) \in E$
let $w_{ij} = 1/(m+1)$ for $(i, j) \in E$, where $m = |E|$

let $w_\Phi = 1/(m+1)$
let $T = c_1 \lceil \frac{\sqrt{m} \ln n}{\varepsilon^2} \rceil$ (the number of iterations)
for for $t = 1$ to T **do**
  let L be the Laplacian of G with edge weights $\mu_{ij} = \frac{w_{ij} + \varepsilon/(m+1)}{r_{ij}^2}$
  let $\lambda = (w_\Phi + \varepsilon/(m+1))/\Phi_0$
  solve $(\lambda^{-1}\mathscr{L} + I)\,(\widetilde{\mathbf{Y}}^t)^\top = \mathbf{Y}^\top$ for $\widetilde{\mathbf{Y}}^t$
  update the weights: $w_\Phi = \left(1 + c_2\varepsilon\left(\sqrt{\frac{\Phi(\mathbf{Y}, \widetilde{\mathbf{Y}}^t)}{\Phi_0}} - 1\right)\right) w_\Phi$

$w_{ij} = \left(1 + c_2\varepsilon\left(\frac{\|\tilde{y}_i - \tilde{y}_j\|}{r_{ij}} - 1\right)\right) w_{ij}$ for every $(i, j) \in E$
  normalize the weights: $W = w_\Phi + \sum_{(i,j)\in E} w_{ij}$

$w_\Phi = w_\Phi / W$

$w_{ij} = w_{ij}/W$ for all $(i, j) \in E$

end for
return $\widetilde{\mathbf{Y}} = \frac{1}{T}\sum_{t=1}^{T} \widetilde{\mathbf{Y}}^t$

and $\|\tilde{y}_i - \tilde{y}_j\| \le (1+\varepsilon)L\|x_i - x_j\|$ for every $(i, j) \in E$.

In the Appendix, we describe the approximation oracle that we invoke in the MWU method.

Theorem 1 *There is an algorithm for the Lipschitz Smoothing problem that runs in time*

$$O\left(ma + m^{3/2}b(\log n)^2 \log(1/\varepsilon)/\varepsilon^{5/2}\right),$$

where $m = |E|$.

Proof Theorem 1 From Theorem 3.5 in [2], we get that the algorithm finds an $O(\varepsilon)$ approximate solution in $T = O\left(\frac{\sqrt{m/\varepsilon}\ln m}{\varepsilon^2}\right) = \left(\frac{\sqrt{m}}{\varepsilon^{5/2}}\right)$ iterations. Each iteration takes $O(bm \log n \log(1/\varepsilon))$ time (which is dominated by the time necessary to solve Laplace's problem); additionally, we spend time $O(am)$ to compute pairwise distances between points in X. □

3.2 Approximate Lipschitz extension

Problem statement Given a finite sequence $(x_i)_{i\in[n]} \subset X = \mathbb{R}^a$, its image $(y_i)_{i\in[n]} \subset Y = \mathbb{R}^b$ under some L-Lipschitz map $f : X \to Y$, a test point x^*,

Algorithm 2 OnePointExtension

Require: labeled sample $(x_i, y_i = f(x_i)) \subset (X \times Y)^n$, $\varepsilon \in (0, 1/2)$ query point $x^* \in X$, and upper bound $L \geq \|f\|_{\text{Lip}}$ **return** label y^*
let x° be the nearest neighbor of x^* among $x_1, \ldots, x_n$; $y^\circ = f(x^\circ)$; $d^\circ = \|x^\circ - x^*\|$
initialize weights $w_1^{(1)}, \ldots, w_n^{(1)}$ as follows: $w_i^{(1)} = 1/n$ for every i
let $d_i = \|x_i - x^*\|$ for every i
let $T = \lceil \frac{16 \ln n}{\varepsilon^2} \rceil$ (the number of iterations)
for $t = 1$ to T **do**
  let $P = \sum_{i=1}^n w_i^{(t)}/d_i^2$ and $p_i = \frac{w_i^{(t)}}{P d_i^2}$
  let $z_0 = \sum_{i=1}^n p_i y_i$ and $\Delta = \|z_0 - y^\circ\|$
  if $\Delta \leq L d^\circ$ **then** $z^{(t)} = z_0$ **else** $z^{(t)} = \frac{L d^\circ}{\Delta} z_0 + \frac{\Delta - L d^\circ}{\Delta} y^\circ$
  update the weights: $w_i^{(t+1)} = (1 + \frac{\varepsilon \|\tilde{y} - y_i\|}{8 L d_i}) w_i^{(t)}$ for every i
  normalize the weights: compute $W = \sum_{i=1}^n w_i^{(t)}$ and let $w_i^{(t+1)} = w_i^{(t)}/W$ for every i
end for
return $z = \frac{1}{T} \sum_{t=1}^T z^{(t)}$

and a precision parameter $\varepsilon \in (0, 1/2)$, we wish to compute $y^* = f(x^*)$ so that $\|y^* - y_i\| \leq (1+\varepsilon) L \|x^* - x_i\|$ for all $i \in [n]$. Our result is an efficient Algorithm 2 (called OnePointExtension) that achieves this:

Theorem 2 *The approximate Lipschitz extension algorithm* OnePointExtension *(Algorithm 2) has runtime* $O(na + nb \log n/\varepsilon^2)$.

We are also interested in the setting when the labeled sample $\{(x_i\ y_i)\}_i$ is fixed, and we compute Kirszbraun extensions y^* for multiple points x^*. In this setting, the query runtime—the time necessary to compute one extension—can be significantly improved if the dimension of X is small:

Theorem 3 *There is a data structure for the Lipschitz extension problem of memory size* $2^{O(a)}n$ *that can be constructed in time* $2^{O(a)}n \log n$. *Given a query point* x^* *and a parameter* $\varepsilon \in (0, 1/2)$, *one can compute* y^* *such that* $\|y^* - y_i\| \leq (1+\varepsilon) L \|x^* - x_i\|$ *for every* i *in time* $(1/\varepsilon)^{O(a)}(b + \log n)$.

Analysis We begin with an intuitive explanation. Consider a feasibility problem over a domain $\mathscr{P} \in \mathbb{R}^n$ in which one must determine whether there exists an $x \in \mathscr{P}$ satisfying the constraints $f_i(x) \geq 0$ for $i \in [m]$. This problem in the general case is NP-Hard. The Arora–Hazan–Kale result indicates that, under certain conditions, an approximate solution to the simpler problem, where the m constraints have been replaced by their convex combination $\sum_i w_i f_i(x) \geq 0$ can be leveraged to solve the original one. One makes repeated calls to the simpler-problem oracle and uses multiplicative weight updates to adjust the w. If a solution $x_i \in \mathscr{P}$ exists for all iterations, then $x^* = \frac{1}{T} \sum_{t=1}^T x^{(t)}$ will be an approximate solution to the original feasibility problem. The precise details in "Appendix A". In this section, we present our algorithm, show how our problem fits the paradigm of [2, Sec. 3.3.1, p. 137], show how our oracle solves the simpler problem, and prove that our algorithms solve the feasibility problems efficiently.

We analyze algorithm OnePointExtension (Algorithm 2) and prove Theorems 2 and 3 via the multiplicative update framework of [2]. In particular, we will invoke their Theorem 3.4, which, for completeness, is reproduced in "Appendix A" as Theorem 5. Let $\mathscr{P} = \text{Ball}(y^\circ, L\|x^\circ - x^*\|)$ (see algorithm OnePointExtension 2 for the definitions of x° and y°), and define $h_i(y) = 1 - \frac{\|y-y_i\|}{L\|x^*-x_i\|}$ for $y \in \mathscr{Y}$, $i \in \{1, \ldots, n\}$. Then the Lipschitz extension problem is equivalent to the following: find $y \in \mathscr{P}$ such that $h_i(y) \geq 0$ for all $i \in [n]$. Note that functions h_i are concave and thus the problem is in the form of (3.8) from [2]. We now bound the width of the problem, proving that $h_i(y) \in [-2, 1]$ for every $y \in \mathscr{P}$; that is, in the notation from [2], we show that every oracle for the problem is (ℓ, ρ)-bounded with $\ell = 1$ and $\rho = 2$. Observe that for every $y \in \mathscr{P}$ and every i, we have (i) $h_i(y) \leq 1$ as $\frac{\|y-y_i\|}{L\|x^*-x_i\|} \geq 0$ and (ii)

$$\begin{aligned} 1 - h_i(y) &= \frac{\|y - y_i\|}{L\|x^* - x_i\|} \leq \frac{\|y - y^\circ\| + \|y^\circ - y_i\|}{L\|x^* - x_i\|} \\ &\leq \frac{L\|x^\circ - x^*\| + L\|x^\circ - x_i\|}{L\|x^* - x_i\|} \\ &\leq \frac{2\|x^\circ - x^*\| + \|x^* - x_i\|}{\|x^* - x_i\|} \\ &= 1 + 2\frac{\|x^\circ - x^*\|}{\|x^* - x_i\|} \leq 3. \end{aligned}$$

Here, we used that $\|y - y^\circ\| \leq L\|x^\circ - x^*\|$ (which is true since $y \in \mathscr{P}$), $\|y^\circ - y_i\| \leq L\|x^\circ - x_i\|$ (which is true since f is L-Lipschitz), $\|x^\circ - x_i\| \leq \|x^\circ - x^*\| + \|x^* - x_i\|$ (the triangle inequality), and $\|x^* - x^\circ\| \leq \|x^* - x_i\|$ (which is true since x° is the point closest to x^* among all points $x_1, \ldots, x_n$). We conclude that $h_i(y) \in [-2, 1]$.

To apply Theorem 5, we design an oracle for the following problem:

Problem 2 *Given non-negative weights w_i that add up to 1, find $y \in \mathscr{P}$ such that*

$$\sum_{i=1}^{n} w_i h_i(y) \geq 0. \tag{2}$$

Note that Problem 2 has a solution, since (i) by the Kirzsbraun theorem there exists a Lipschitz extension y^* of f to point x^*, and (ii) y^* satisfies (2). Define auxiliary weights p_i as follows:

$$P = \sum_{i=1}^{n} \frac{w_i}{\|x^* - x_i\|^2} \quad \text{and} \quad p_i = \frac{w_i}{P\|x^* - x_i\|^2}.$$

Note that $\sum_i p_i = 1$. We now show that the oracle finds and outputs $z \in \mathscr{P}$ that minimizes $V(z) = \sum_{i=1}^n p_i\|z - y_i\|^2$. To this end, it first computes $z_0 = \sum_{i=1}^n p_i y_i$. Note that $V(z) = \|z - z_0\|^2 + \sum_{i=1}^n p_i\|z_0 - y_i\|^2$. Observe that the second term does not depend on z and the first term is minimized when z is the closest point to z_0 in $\mathscr{P}$. Thus, if $z_0 \in \mathscr{P}$, the algorithm sets $z = z_0$; otherwise, it sets z to be the point closest to z_0 in $\mathscr{P}$.

We derive a formula for z in the latter case. Note that $L\|x^\circ - x^*\| \le \|z_0 - y^\circ\|$ because $z_0 \notin \mathscr{P}$. Since $\mathscr{P}$ is a ball, point z is the intersection point of the sphere of radius $L\|x^\circ - x^*\|$ around y° (the boundary of $\mathscr{P}$) and segment $[z_0, y^\circ]$. Now we verify that z is given by the following formula

$$z = \frac{L\|x^\circ - x^*\|}{\|z_0 - y^\circ\|} z_0 + \left(1 - \frac{L\|x^\circ - x^*\|}{\|z_0 - y^\circ\|}\right) y^\circ.$$

Indeed, this z is a convex combination of points z_0 and y° and thus $z \in [z_0, y^\circ]$. Then,

$$\|z - y^\circ\| = \frac{L\|x^\circ - x^*\|}{\|z_0 - y^\circ\|} \cdot \|z_0 - y^\circ\| = L\|x^\circ - x^*\|.$$

We have verified that this z is the intersection point of the sphere of radius $L\|x^\circ - x^*\|$ around y° and segment $[z_0, y^\circ]$, as required.

We compute z on lines 6–8 of the algorithm. We verify that z satisfies condition (2). Write condition (2) for point $y = z$:

$$0 \le \sum_i w_i h_i(z) = \sum_i w_i \left(1 - \frac{||z - y_i||}{L||x^* - x_i||}\right) = 1 - \sum_i \frac{w_i ||z - y_i||}{L||x^* - x_i||}.$$

That is, we need to verify that $\sum_i \frac{w_i ||z-y_i||}{||x^*-x_i||} \le L$. Using that $\|y^* - y_i\|^2 / \|x^* - x_i\|^2 \le L^2$, we get

$$\begin{aligned}\sum_{i=1}^n w_i \frac{\|z - y_i\|}{\|x^* - x_i\|} &\le \left(\sum_{i=1}^n w_i \frac{\|z - y_i\|^2}{\|x^* - x_i\|^2} \sum_{i=1}^n w_i\right)^{1/2} \\ &\le \left(\sum_{i=1}^n w_i \frac{\|y^* - y_i\|^2}{\|x^* - x_i\|^2}\right)^{1/2} \le L.\end{aligned}$$

The first inequality is due to Cauchy–Schwarz, and the second holds since $V(z) \le V(y^*)$.

Proof of Theorem 2 From Theorem 5, we get that the algorithm finds a $1 + \varepsilon$ approximate solution in $T = \frac{8\rho\ell \ln m}{\varepsilon^2} = \frac{16 \ln m}{\varepsilon^2}$ iterations. Computing distances d_i takes $O(an)$ time, each iteration takes $O(bn)$ time. □

Proof of Theorem 3 Our key observation is that we can run the algorithm from Theorem 2 on a subset X' of X, which is sufficiently dense in X. Specifically, let x° be a $(1+\varepsilon)$-approximate nearest neighbor for x^* in X. Assume that a subset $X' \subset X$ contains x° and satisfies the following property: for every $x_i \in X \cap \text{Ball}(x^*, \|x^* - x^\circ\|/\varepsilon)$, there exists $x_j \in X'$ such that $\|x_j - x_i\| \le \varepsilon \|x^* - x_i\|$.

First, we prove that by running the algorithm on set X' as above, we get y^* such that $\|y_i - y^*\| \le (1 + O(\varepsilon))L\|x_i - x^*\|$ for all i. Then we describe a data structure that we use to find X' for a given query point x^* in time $(1/\varepsilon)^{O(a)} \log n$.

Lemma 1 *Algorithm from Theorem 2 finds y^* such that $\|y_i - y^*\| \le (1+O(\varepsilon))L\|x_i - x^*\|$ for all $x_i \in X'$.*

Proof Consider $x_i \in X$. First, assume that $x_i \in \text{Ball}(x^*, \|x^* - x^\circ\|/\varepsilon)$. Find $x_j \in X'$ such that $\|x_j - x_i\| \le \varepsilon\|x^* - x_i\|$. Then

$$\begin{aligned}\|y_i - y^*\| &\le \|y_i - y_j\| + \|y_j - y^*\| \\ &\le L\|x_i - x_j\| + (1+\varepsilon)L\|x_j - x^*\| \\ &\le L((2+\varepsilon)\|x_i - x_j\| + (1+\varepsilon)\|x_i - x^*\|) \\ &\le (1 + 3\varepsilon + \varepsilon^2)L\|x_i - x^*\|,\end{aligned}$$

Now assume that $x_i \notin \text{Ball}(x^*, \|x^* - x^\circ\|/\varepsilon)$. Then $\|x_i - x^*\| > \|x^* - x^\circ\|/\varepsilon$; that is, $\|x^* - x^\circ\| < \varepsilon\|x_i - x^*\|$. Thus,

$$\begin{aligned}\|y_i - y^*\| &\le \|y_i - y^\circ\| + \|y^\circ - y^*\| \\ &\le L\|x_i - x^\circ\| + (1+\varepsilon)L\|x^\circ - x^*\| \\ &\le L(\|x_i - x^*\| + (2+\varepsilon)\|x^\circ - x^*\|) \\ &\le L(\|x_i - x^*\| + (2+\varepsilon)\cdot\varepsilon\|x_i - x^*\|) \\ &= (1+\varepsilon)^2 L\|x_i - x^*\|.\end{aligned}$$

as required.

We can use a data structure $\mathscr{D}$ for approximate nearest neighbor search in X. We employ one of the constructions for low-dimensional Euclidean spaces by [3, 22], or [14]. Using $\mathscr{D}$, we can find a $(1+\varepsilon/3)$-approximate nearest neighbor of a point in $\mathbb{R}^a$ in time $(1/\varepsilon)^{O(a)}\log n$. Recall that we can construct $\mathscr{D}$ in $O(2^{O(a)}n\log n)$ time, and it requires $O(2^{O(a)}n\log n)$ space. Suppose that we get a query point x^*. We first find an approximate nearest neighbor x° for x^*. Let $r = \|x^\circ - x^*\|$. Take an $\varepsilon r/3$ net N' in the ball $\text{Ball}(x^*, r/\varepsilon)$. For every point $p \in N'$, we find an approximate nearest neighbor $x(p)$ in X (using $\mathscr{D}$). Let $X' = \{x(p) : p \in N'\} \cup \{x^\circ\}$. Consider $x_i \in \text{Ball}(x^*, r/\varepsilon) \cup X$. There is $p \in N'$ at distance at most $\varepsilon r/3$ from x_i. Let $x_j = x(p) \in X'$. Then

$$\|p - x_j\| \le (1+\varepsilon/3)\|x_i - p\| \le (1+\varepsilon/3)\varepsilon r/3$$

and

$$\begin{aligned}\|x_i - x_j\| &\le \|x_i - p\| + \|p - x_j\| \\ &\le \varepsilon r/3 + (1+\varepsilon/3)\varepsilon r/3 \\ &\le (2+\varepsilon/3)\varepsilon r/3 \le 3\|x^* - x_i\|,\end{aligned}$$

as required. The size of X' is at most the size of N', which is $(1/\varepsilon)^{O(a)}$.

Finally, let us compute the total Lipschitz extension query time. We make $(1/\varepsilon)^{O(a)}$ nearest-neighbor queries, each taking time $(1/\varepsilon)^{O(a)}\log n$. Then we run the algorithm from Theorem 2 on the set X', this requires time

$$O(|X'|a + |X'|b \log |X'|/\varepsilon^2) = O((1/\varepsilon)^{O(a)}a + (1/\varepsilon)^{O(a)}b \log(1/\varepsilon)^{O(a)}) = O((1/\varepsilon)^{O(a)}ab \log(1/\varepsilon)) = O((1/\varepsilon)^{O(a)}b).$$

The total query time is $(1/\varepsilon)^{O(a)}(b + \log n)$.

Algorithm 3 MultiPointExtension

Require: vectors $x_1, \ldots, x_n, x_{n+1}, \ldots, x_{n+n'} \in \mathbb{R}^a$ and $y_1, \ldots, y_n \in \mathbb{R}^b$, graph $G = ([n+n'], E)$, and M **return** $\widetilde{\mathbf{Y}} = (y_{n+1}, \ldots, y_{n+n'})$

let $\mathbf{Y} \equiv (y_1, \ldots, y_n)$

let $r_{ij} = L\|x_i - x_j\|$ for $(i, j) \in E$

let $w_{ij} = 1/m$ for $(i, j) \in E$

let $T = c_1 \lceil \frac{\sqrt{m} \ln n}{\varepsilon^2} \rceil$ (the number of iterations)

for for $t = 1$ to T **do**

  let $\lambda_{ij} = \frac{w_{ij} + \varepsilon/m}{r_{ij}^2}$ for $(i, j) \in E$

  define $n' \times n$ times matrix K as:

$K_{ij} = \lambda_{i+n, j+n}$ if $(i + n, j + n) \in E$ and 0 otherwise.

  solve $L(\widetilde{\mathbf{Y}}^t)^\top = K\mathbf{Y}^\top$ for $\widetilde{\mathbf{Y}}^t$

  update the weights: $w_{ij} = \left(1 + c_2\varepsilon\left(\frac{\|y_i - y_j\|}{r_{ij}} - 1\right)\right) w_{ij}$ for every $(i, j) \in E$

  normalize the weights: $W = \sum_{(i,j)\in E} w_{ij}$

$w_{ij} = w_{ij}/W$ for all $(i, j) \in E$

end for

return $\widetilde{\mathbf{Y}} = \frac{1}{T}\sum_{t=1}^{T} \widetilde{\mathbf{Y}}^t$

3.3 Multi-point Lipschitz extension

Finally, we describe an algorithm for the Multi-point Lipschitz Extension. The problem is a generalization of the problem we studied in Sect. 3.2.

We are given a set of points $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^a$ and their images $Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^b$ under L-Lipschitz map f. Additionally, we are given a set $Z = \{x_{n+1}, \ldots, x_{n+n'}\} \subset \mathbb{R}^a$ and a set of edges E on $\{1, \ldots, n + n'\}$. We need to extend f to Z—that is, find $y_{n+1}, \ldots, y_{n+n'}$—such that $\|y_i - y_j\| \leq (1 + \varepsilon)L\|x_i - x_j\|$ for $(i, j) \in E$. We note that E may contain edges that impose Lipschitz constraints (i) between points in X and Z and (ii) between pairs of points in Z. Without loss of generality, we assume that there are no edges $(i, j) \in E$ with $1 \leq i, j \leq n$.

Theorem 4 *There is an algorithm for the Multi-Point Lipschitz Extension problem that runs in time*

$$O\left(ma + \frac{m^{3/2}(\log m)^2 b \log(1/\varepsilon)}{\varepsilon^{5/2}}\right),$$

where $m = |E|$.

The algorithm and its analysis are almost identical to those for the Lipschitz Smoothing problem. (see Theorem 1).

4 Experiments

To illustrate the utility of our framework—learning via Kirszbraun extension—we designed two simple non-linear regression problems, where the input and output are both scalars. We generated data points uniformly at random over $[-2\pi, 2\pi]$ on two cases: $f(x) = x^3$ and $f(x) = \sin(x)$. We gave our learning algorithms the labeled data sets $\{(x_i, f(x_i); i \in [1, \ldots, n]\}$, and evaluated their prediction performance using the average square loss function $\frac{1}{n}\sum_{i=1}^{n}(h(x_i) - f(x_i))^2$, where $h(x_i)$ is the output of our Kirszbraun Extension model for a data point x_i. To illustrate the advantage of the MWU-based optimization methods presented in Sect. 3 over a generic QCQP solver, we designed a second experiment for learning the same non-linear transformations via Kirszbraun extension, but using MATLAB's QCQP solver based on the interior-point algorithm (IntPt) for the smoothing and extension problem.

Setup We implemented Algorithms 1 and 2 in MATLAB, solving the regression problem via the Kirszbraun extension technique for both cases $f(x) = x^3$ and $f(x) = \sin(x)$. We considered the squared Euclidean distance as the loss function. We ran several tests on data sets of size 20, 100, 200, 500, and 1000 random points as the training set, and 100 test points in all experiments. We then re-implemented the algorithms for the *smoothing* and *extension* problems, this time using MATLAB's QCQP solver based on the interior-point algorithm (IntPt). For reproducibility, we used MATLAB's random seed 1 in all our runs. All the tests were conducted on the same Macbook Pro computer: MacBook Pro (16-inch, 2019), 2.6 GHz 6-Core Intel Core i7 processor, Memory 16 GB 2667 MHz DDR4.

Tables 1, 2, 3, 4 and 5 summarise the results for $f(x) = x^3$. The results for $f(x) = \sin(x)$ are showing the same basic pattern and were added to the Appendix. The "TL" entries in the tables indicate that the process was *too long* as it did not terminate in the time allotted for the experiment (12 h).

Table 1 ERM of the smoothing process

Algorithm	Avg. loss				
Training points	20	100	200	500	1000
MWU	247.94	0.33	0.31	0.31	0.36
IntPt	4.1e−18	46,023.79	353,691.64	TL	TL

Table 2 Cross validation running time over the smoothing process in seconds

Algorithm	Avg. loss				
Training points	20	100	200	500	1000
MWU	2.75	19.94	46.24	212.48	1243.23
IntPt	18.17	692.65	4087.66	TL	TL

Table 3 Running time in seconds of the Smoothing process

Algorithm	Avg. loss				
Training points	20	100	200	500	1000
MWU	0.09	0.70	1.63	8.07	45.14
IntPt	2.47	155.55	766.54	TL	TL

Table 4 Extension avg loss

Algorithm	Avg. loss				
Training points	100	200	500	1000	
MWU	1119.47	0.33	0.37	0.43	0.52
IntPt	3065.56	9475.26	9475.48	TL	TL

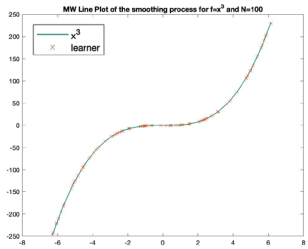
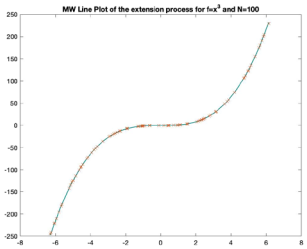
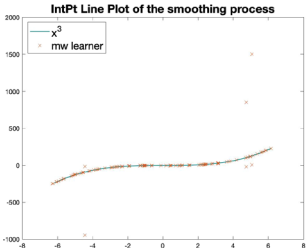
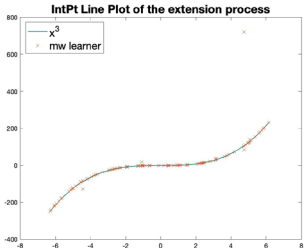
Table 5 Extension running time for a single point in the test set, in seconds

Algorithm	Avg. loss				
Training points	20	100	200	500	1000
MWU	0.05	0.001	0.002	0.005	0.009
IntPt	1.330	1.690	2.610	TL	TL

5 Discussion and conclusions

This work introduces a framework for performing regression between two Hilbert spaces based on Kirszbraun's extension theorem, along with statistical analysis for this method. This task is decomposed into two stages: Smoothing (which corresponds to the training) and prediction (which is achieved via Kirszbraun extension). Numerically solving our optimization problems has indicated a need for a more efficient solver for our optimization problems than generic off-the-shelf interior-point solvers. We introduced two optimization algorithms, one for the smoothing problem and one for the extension, both are solved algorithmically via novel MWU schemes. Both analysis and experiments show dramatic runtime improvement for both optimization problems, illustrating the practical utility of our algorithms. Our code is also provided for reproducibility and to facilitate deployment (Table 6).

Table 6 Visual comparison between our MWU-based algorithm (first row) and IntPt (MATLAB's) based algorithm (second row)

	Smoothing	Extension
MWU		
IntPt		

For $f = x^3$ and $N = 100$ random points. In all graphs, the blue line represents the ground truth function $f = x^3$ while the orange x symbols represent the estimation of the data points by the learned function. It is possible to see that while the MWU-based algorithm was able to fit both the training and test set to high accuracy, the IntPt method has several "heavy" outliers which increase significantly its average squared error

Acknowledgements AK was partially supported by the Israel Science Foundation (Grant No. 1602/19), the Ben-Gurion University Data Science Research Center, and an Amazon Research Award. HZ was an MSc student at Ben-Gurion University of the Negev during part of this research. YM was partially supported by NSF awards CCF-1718820, CCF-1955173, and CCF-1934843.

Appendix A: The Arora–Hazan–Kale result

Intuitive explanation of the result Consider a feasibility problem over a domain $\mathscr{P} \in \mathbb{R}^n$ in which you need to determine if there exists $x \in \mathscr{P}$ which satisfies finite set of constrains: $f_i(\mathbf{X}) \geq 0$ for all $i \in [m]$. This problem in the general case is NP-Hard. The Arora-Hazan-Kale result indicates that, under certain conditions, if we know how to solve approximately a simpler problem: $\exists ?\mathbf{X} \in \mathscr{P} : \sum_i w_i f_i(\mathbf{X}) \geq 0$ where $\sum_i w_i = 1$, by an algorithm which we call "Oracle", then we can run T iterations over the simpler problem, where in each iteration we update the weights $\{w_i\}_{i\in[m]}$ using the MWU framework, and solve the updated problem using the Oracle. If a solution $x_i \in \mathscr{P}$ exists for all iterations then $\mathbf{X}^* = \frac{1}{T}\sum_{t=1}^{T} \mathbf{X}^{(t)}$ will be an approximate solution to the feasible problem. The conditions and the definitions of the meaning in the intuitive explanation are given in "Appendix A". In this section, we present

our algorithm, show our problem fits the paradigm of [2, Sec. 3.3.1, p. 137], show our oracle solves the simpler problem (given in the algorithm) and proof that our algorithms solve the feasible problems efficiently. For completeness, we quote here the relevant definitions and results from [2, Sec. 3.3.1, p. 137].

Consider the following feasibility problem. Let $\mathscr{P} \in \mathbb{R}^n$ be a convex domain in $\mathbb{R}^n$ and $f_i : \mathscr{P} \to \mathbb{R}$ be concave functions for $i \in [m]$. The goal is to determine if there exists $x \in \mathscr{P}$ such that $f_i(\mathbf{X}) \geq 0$ for all $i \in [m]$:

$$\exists ?\mathbf{X} \in \mathscr{P} : \ \forall i \in [m] : \ f_i(\mathbf{X}) \geq 0, \tag{3}$$

The multiplicative weights update method of Arora, Hazan, and Kale [2] provides an algorithm that satisfies (3) approximately, up to an additive error of ε. We assume the existence of an oracle that given a probability distribution $\mathbf{w} = (w_1, w_2, \ldots, w_m)$ solves the following feasibility problem:

$$\exists ?\mathbf{X} \in \mathscr{P} : \ \sum_i w_i f_i(\mathbf{X}) \geq 0. \tag{4}$$

Definition 1 We say that oracle Oracle is (ℓ, ρ)-bounded if it always returns $x \in \mathscr{P}$ such that $f_i(\mathbf{X}) \in [-\rho, \ell]$ for all $\in [m]$. The width of the oracle is $\rho + \ell$.

Remark 1 Note that if $f_i(\mathbf{X}) \in [-\rho, \ell]$ for all $\mathbf{X} \in \mathscr{P}$ then every oracle for the problem is (ℓ, ρ)-bounded.

Definition 2 We say that oracle Oracle is ε-approximate if given $\mathbf{w} = (w_1, \ldots, w_m)$ it either finds a solution $\mathbf{X} \in \mathscr{P}$ such that $\sum_i w_i f_i \geq -\varepsilon$ for all $i \in [m]$ or correctly concludes that (4) has no feasible solution.

Consider the following algorithm.

Algorithm 4 Multiplicative Weights Update Algorithm

Require: functions $f_i(x)$ for $i \in [m]$, access to (ℓ, ρ)-bounded oracle Oracle
  return feasible solution $x^* \in \mathscr{P}$
  let $\eta = \frac{\varepsilon}{4\ell}$
  initialize weights $w_1^{(1)}, \ldots, w_n^{(1)}$ as follows: $w_i^{(1)} = 1/n$ for every $i \in [n]$
  let $T = \lceil \frac{8\rho\ell \ln m}{\varepsilon^2} \rceil$ (the number of iterations)
  for $t = 1$ to T **do**
    call Oracle with probability distribution $\mathbf{w} = w^{(i)}$.
      if the call fails, **return** "there is no feasible solution"
      else let $\mathbf{X}^{(t)}$ be the solution found by Oracle
    update the weights: $w_i^{(t+1)} = (1 - \eta f_i(\mathbf{X}^{(t)}))w_i^{(t)}$ for every i
    normalize the weights: compute $W = \sum_{i=1}^n w_i^{(t)}$ and let $w_i^{(t+1)} = w_i^{(t)}/W$ for every i
  end for
  return $\mathbf{X}^* = \frac{1}{T}\sum_{t=1}^T \mathbf{X}^{(t)}$

We now state theorems providing performance guarantees for Algorithm 4. Theorems 5 and 6 are for the cases where we have an exact and ε-approximate oracles, respectively.

Theorem 5 (Theorem 3.4 in [2], restated) *Let $\varepsilon > 0$ be a given error parameter. Suppose that there exists an (ℓ, ρ)-bounded* Oracle *for the feasibility problem (3) with $\ell \geq \varepsilon/2$. Then Algorithm 4 either*

- *solves problem (3) up to an additive error of ε; that is, finds a solution $\mathbf{X}^* \in \mathscr{P}$ such that $f_i(\mathbf{X}^*) \geq -\varepsilon$ for all $i \in [m]$,*
- *or correctly concludes that problem (3) is infeasible,*

making only $O(\ell\rho \log(m)/\varepsilon^2)$ calls to the Oracle*, with an additional processing time of $O(m)$ per call.*

Theorem 6 (Theorem 3.5 in [2], restated) *Let $\varepsilon > 0$ be a given error parameter. Suppose that there exists an (ℓ, ρ)-bounded $(\varepsilon/3)$-approximate* Oracle *for the feasibility problem (3) with $\ell \geq \varepsilon/2$. Consider Algorithm 4 with adjusted parameters $\eta = \frac{\varepsilon}{6\ell}$ and $T = \lceil \frac{18\rho\ell \ln m}{\varepsilon^2} \rceil$. Then the algorithm either*

- *solves problem (3) up to an additive error of ε; that is, finds a solution $\mathbf{X}^* \in \mathscr{P}$ such that $f_i(\mathbf{X}^*) \geq -\varepsilon$ for all $i \in [m]$,*
- *or correctly concludes that problem (3) is infeasible,*

making only $O(\ell\rho \log(m)/\varepsilon^2)$ calls to the Oracle*, with an additional processing time of $O(m)$ per call.*

Appendix B: Approximate oracle

This section is a continuation to Sect. 3.1, the analysis of the LipschitzSmooth algorithm in Theorem 1. To use the MWU method (see Theorem 6, Theorem 3.5 in [2]), we design an approximate oracle for the following problem.

Problem 3 *Given non-negative edge weights w_Φ and w_{ij}, which add up to 1, find $\widetilde{\mathbf{Y}}$ such that*

$$w_\Phi h_\Phi(\widetilde{\mathbf{Y}}) + \sum_{(i,j)\in E} w_{(i,j)} h_{(i,j)}(\widetilde{\mathbf{Y}}) \geq 0. \tag{5}$$

If Problem 3 has a feasible solution, the oracle finds a solution $\widetilde{\mathbf{Y}}$ such that

$$w_\Phi h_\Phi(\widetilde{\mathbf{Y}}) + \sum_{(i,j)\in E} w_{(i,j)} h_{(i,j)}(\widetilde{\mathbf{Y}}) \geq -\varepsilon. \tag{6}$$

Let $\mu_{ij} = \frac{w_{ij}+\varepsilon/(m+1)}{M^2\|x_i - x_j\|^2}$ and $\lambda_i = \lambda = (w_\Phi + \varepsilon/(m+1))/\Phi_0$. We solve Laplace's problem with parameters μ_{ij} and λ_i (see Sect. 1 and Line 9 of the algorithm). We get a matrix $\widetilde{\mathbf{Y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)$ minimizing

$$\lambda \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2 + \sum_{(i,j)\in E} \mu_{ij} \|\tilde{y}_i - \tilde{y}_j\|^2.$$

Consider the optimal solution $\tilde{y}_1^*, \ldots, \tilde{y}_n^*$ for Lipschitz Smoothing. We have

$$\lambda \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2 + \sum_{(i,j)\in E} \mu_{ij} \|\tilde{y}_i - \tilde{y}_j\|^2 \le \lambda \sum_{i=1}^{n} \|y_i - \tilde{y}_i^*\|^2 + \sum_{(i,j)\in E} \mu_{ij} \|\tilde{y}_i^* - \tilde{y}_j^*\|^2 \tag{7}$$

$$\le (w_\Phi + \varepsilon/(m+1)) + \sum_{(i,j)\in E} \left(w_{ij} + \frac{1}{m+1} \right) \frac{\|\tilde{y}_i^* - \tilde{y}_j^*\|}{M^2 \|x_i - x_j\|^2} \tag{8}$$

$$\le \left(w_\Phi + \sum_{(i,j)\in E} w_{ij} \right) + \varepsilon = 1 + \varepsilon. \tag{9}$$

We verify that $\widetilde{\mathbf{Y}}$ is a feasible solution for Problem 3. We have

$$\begin{aligned}
1 - \left(w_\Phi h_\Phi(\widetilde{\mathbf{Y}}) + \sum_{(i,j)\in E} w_{(i,j)} h_{(i,j)}(y) \right) &= w_\Phi (1 - h_\Phi) \\
&+ \sum_{(i,j)\in E} w_{(i,j)} (1 - h_{(i,j)}(y)) \\
&\le \sqrt{w_\Phi (1 - h_\Phi)^2 + \sum_{(i,j)\in E} w_{(i,j)} (1 - h_{(i,j)}(y))^2} \\
&= \sqrt{w_\Phi \frac{\Phi(\mathbf{Y}, \widetilde{\mathbf{Y}})}{\Phi_0} + \sum_{(i,j)\in E} w_{(i,j)} \frac{\|\tilde{y}_i - \tilde{y}_j\|^2}{M^2 \|x_i - x_j\|^2}} \\
&\le \sqrt{\lambda \Phi(\mathbf{Y}, \widetilde{\mathbf{Y}}) + \sum_{(i,j)\in E} w_{(i,j)} \mu_{ij} \|\tilde{y}_i - \tilde{y}_j\|^2} \\
&\le \sqrt{1 + \varepsilon} \le 1 + \varepsilon,
\end{aligned}$$

as required.

Finally, we bound the width of the problem. We have $h_\Phi(\widetilde{\mathbf{Y}}) \le 1$ and $h_{ij}(\widetilde{\mathbf{Y}}) \le 1$. Then, using (7), we get

$$(1 - h_\Phi(\widetilde{\mathbf{Y}}))^2 = \frac{1}{\Phi_0} \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2 \le \frac{1 + \varepsilon}{\lambda \Phi_0} \le \frac{(1+\varepsilon)(m+1)}{\varepsilon}.$$

Therefore, $-h_\Phi(\widetilde{\mathbf{Y}}) \le O(\sqrt{m/\varepsilon})$.

Similarly,

$$(1 - h_{ij}(\widetilde{\mathbf{Y}}))^2 = \frac{\|y_i - \tilde{y}_i\|^2}{M^2\|x_i - x_j\|^2} \leq \frac{1+\varepsilon}{\mu_{ij} \cdot M^2\|x_i - x_j\|^2} \leq \frac{(1+\varepsilon)(m+1)}{\varepsilon}.$$

Therefore, $-h_{ij}(\widetilde{\mathbf{Y}}) \leq O(\sqrt{m/\varepsilon})$.

Appendix C: Generalization bounds

Recall the following statistical setting of Sect. 3. We are given a labeled sample $(x_i, y_i)_{i\in[n]}$, where $x_i \in X := \mathbb{R}^a$ and $y_i \in Y := \mathbb{R}^b$. For a user-specified Lipschitz constant $L > 0$, we compute the (approximate) Empirical Risk Minimizer (ERM) $\hat{f} := \operatorname{argmin}_{f\in F_L} \hat{R}_n(f)$ over $F_L := \{f \in Y^X : \|f\|_{\text{Lip}} \leq L\}$. A standard method for tuning L is via Structural Risk Minimization (SRM): One computes a generalization bound $R(\hat{f}) \leq \hat{R}_n(\hat{f}) + Q_n(a, b, L)$, where $Q_n(a, b, L) := \sup_{f\in F_L} |R(f) - \hat{R}_n(f)| = O(L/n^{a+b+1})$, and chooses $\hat{L}$ to minimize this. In this section, we will derive the aforementioned bound.

Let $B_X \subset \mathbb{R}^k$ and $B_Y \subset \mathbb{R}^\ell$ be the unit balls of their respective Hilbert spaces (each endowed with the ℓ_2 norm $||\cdot||$ and corresponding inner product) and $\mathscr{H}_L \subset {B_Y}^{B_X}$ be the set of all L-Lipschitz mappings from B_X to B_Y. In particular, every $h \in \mathscr{H}_L$ satisfies

$$||h(x) - h(x')|| \leq L||x - x'||, \qquad x, x' \in B_X \subset X.$$

Let $\mathscr{F}_L \subset \mathbb{R}^{B_X \times B_Y}$ be the *loss class* associated with $\mathscr{H}_L$:

$$\mathscr{F}_L = \{B_X \times B_Y \ni (x, y) \mapsto f(x, y) = f_h(x, y) := ||h(x) - y||; h \in \mathscr{H}_L\}.$$

In particular, as every $f \in \mathscr{F}_L$ satisfies $0 \leq f \leq 2$.

Our goal is to bound the *Rademacher complexity* of $\mathscr{F}_L$. We do this via a covering numbers approach:

The empirical *Rademacher complexity* of a collection of functions $\mathscr{F}$, mapping some set
$\mathbf{A} = \{a_1, \ldots, a_n\} \subset A^n$ to $\mathbb{R}$ is defined by:

$$\hat{\mathscr{R}}(\mathscr{F}; A) = \mathbb{E}\left[\sup_{f\in\mathscr{F}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(a_i)\right]. \tag{10}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_m$ are independent random variables drawn from the Rademacher distribution: $\Pr(\sigma_i = +1) = \Pr(\sigma_i = -1) = 1/2$. Recall the relevance of Rademacher complexities to uniform deviation estimates for the risk functional $R(\cdot)$ [30, Theorem 3.1]: for every $\delta > 0$, with probability at least $1 - \delta$, for each $h \in \mathscr{H}_L$:

$$R(h(z)) \leq \hat{R}_n(h(z)) + 2\hat{\mathscr{R}}_n(\hat{\mathscr{F}}_L) + 6\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{11}$$

Define $Z = B_X \times B_Y$ and endow it with the norm $\|(x, y)\|_Z = \|x\| + \|y\|$; note that $(Z, \|\cdot\|_Z)$ is a Banach but not a Hilbert space. First, we observe that the functions in $\mathscr{F}_L$ are Lipschitz under $\|\cdot\|_Z$. Indeed, choose any $f = f_h \in \mathscr{F}_L$ and $x, x' \in B_X$, $y, y' \in B_Y$. Then

$$\begin{aligned}\left|f_h(x, y) - f_h(x', y')\right| &= \left|\|h(x) - y\| - \left\|h(x') - y'\right\|\right| \\ &\le \left\|(h(x) - y) - (h(x') - y')\right\| \\ &\le \left\|h(x) - h(x')\right\| + \left\|y - y'\right\| \\ &\le L\left\|x - x'\right\| + \left\|y - y'\right\| \\ &\le (L \vee 1)\left\|(x, y) - (x', y')\right\|_{\mathscr{Z}},\end{aligned}$$

where $a \vee b := \max\{a, b\}$. We conclude that any $f \in \mathscr{F}_L$ is $(L \vee 1) < (L + 1)$-Lipschitz under $\|\cdot\|_Z$.

Since we restricted the domain and range of $\mathscr{H}_L$, respectively, to the unit balls B_X and B_Y, the domain of $\mathscr{F}_L$ becomes $B_Z := B_X \times B_Y$ and its range is $[0, 2]$. Let us recall some basic facts about the ℓ_2 covering of the k-dimensional unit ball

$$\mathscr{N}(t, B_{\mathscr{X}}, \|\cdot\|) \le (3/t)^k;$$

an analogous bound holds for $\mathscr{N}(t, B_Y, \|\cdot\|)$. Now if $\mathscr{C}_X$ is a collection of balls, each of diameter at most t, that covers B_X and $\mathscr{C}_Y$ is a similar collection covering B_Y, then clearly the collection of sets

$$\mathscr{C}_Z := \{E = F \times G \subset \mathscr{Z} : F \in \mathscr{C}_X, G \in \mathscr{C}_Y\}$$

covers B_Z. Moreover, each $E \in \mathscr{C}_Z$ is a ball of diameter at most $2t$ in $(Z, \|\cdot\|_Z)$. It follows that

$$\mathscr{N}(t, B_Z, \|\cdot\|_Z) \le (6/t)^{2k}.$$

Finally, we endow F_L with the ℓ_∞ norm, and use a Kolmogorov–Tihomirov type covering estimate (see, e.g., [16, Lemma 4.2]):

$$\log \mathscr{N}(t, F_L, \|\cdot\|_\infty) \le (96(L + 1)/t)^{2k} \log(8/t).$$

Finally, we invoke a standard result bounding the Rademacher complexity in terms of the covering numbers via the so-called Dudley entropy integral [24],

$$\hat{\mathscr{R}}_n(F_L) \le \inf_{\alpha \ge 0}\left(4\alpha + 12\int_\alpha^\infty \sqrt{\frac{\log \mathscr{N}(t, F_L, \|\cdot\|_\infty)}{n}}\,dt\right). \tag{12}$$

The estimate in (12) is computed, e.g., in [16, Theorem 4.3]:

$$\hat{\mathscr{R}}_n(F_L; \mathscr{Z}) = O\left(\frac{L}{n^{1/(d+1)}}\right). \tag{13}$$

Table 7 ERM of the smoothing process

Algorithm	Avg. loss			
Training points	100	200	500	1000
MWU	5.5e−09	1.6e−08	3.7e−08	6.8e−08
IntPt	5.3e−16	5.9e 05	TL	TL

Table 8 Cross validation running time over the smoothing process in seconds

Algorithm	Avg. loss			
Training points	100	200	500	1000
MWU	5.19	11.38	66.56	335.06
IntPt	3508.40	3936.64	TL	TL

Table 9 Single* smoothing process running time in seconds

Algorithm	Avg. loss			
Training points	100	200	500	1000
MWU	0.78	1.81	8.18	48.51
IntPt	114.56	778.10	TL	TL

Table 10 Extension avg loss

Algorithm	Avg. loss			
Training points	100	200	500	1000
MWU	5.5e−09	1.1e−08	3.3e−08	7.3e−08
IntPt	0.29	0.83	TL	TL

Putting $d = k + \ell$ and combining (12) with (11) yields our generalization bound: with probability at least $1 - \delta$,

$$R(h(z)) \leq \hat{R}_n(h(z)) + 6\sqrt{\frac{\ln(2/\delta)}{2n}} + O\left(\frac{L}{n^{1/(d+1)}}\right). \tag{14}$$

Appendix D: Additional experiments

For completeness we add here the comparison of the results from the experiment for $f(x) = \sin(x)$ for $x \in [-2\pi, 2\pi]$ (Tables 7, 8, 9, 10, 11 and Fig. 1).

Table 11 Extension running time for a all test set in seconds

Algorithm	Avg. loss			
Training points	100	200	500	1000
MWU	0.002	0.003	0.006	0.009
IntPt	4.3164	1.824	TL	TL

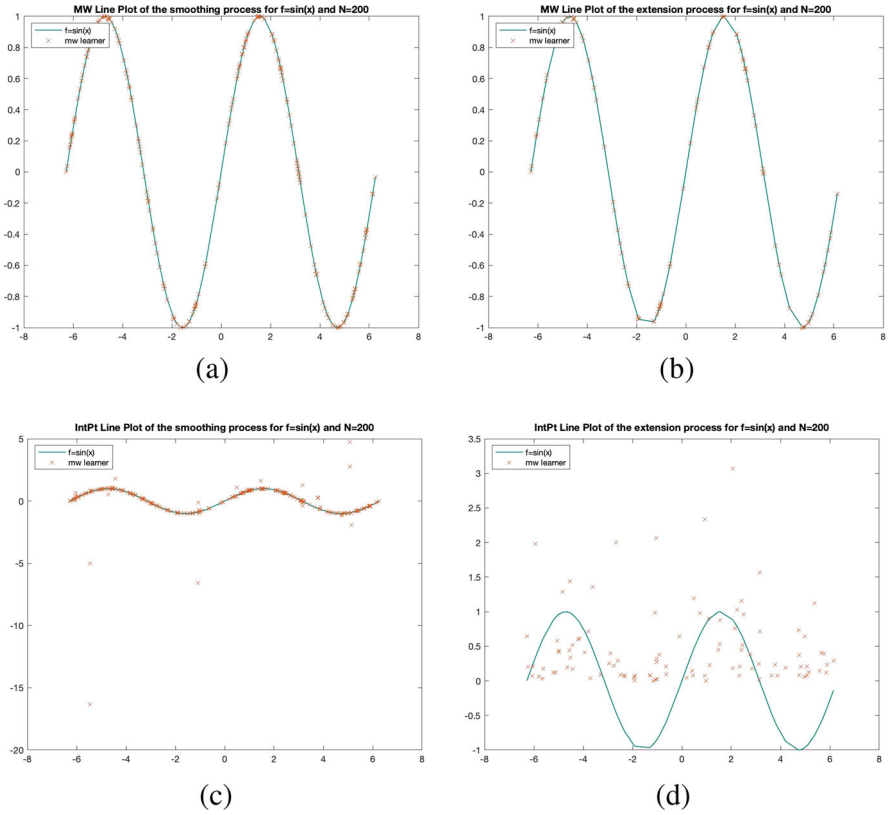


Fig. 1 **a–d** Demonstrate the comparison, where training size is 200 random points, and $f(x) = \sin(x)$. **a, b** Are the smoothing and extension phases implemented with our MWU based algorithm while **c–d** are the results of Matlab's IntPt implementation. In all graphs the line represents the ground truth function while the X represent the estimation by the learned function

References

1. Alman, J., Williams, V.V.: A refined laser method and faster matrix multiplication. In: Proceedings of the Symposium on Discrete Algorithms, pp. 522–539. SIAM (2021)
2. Arora, S., Hazan, E., Kale, S.: The multiplicative weights update method: a meta-algorithm and applications. Theory Comput. **8**(1), 121–164 (2012)
3. Arya, S., Mount, D.M., Netanyahu, N., Silverman, R., Wu, A.Y.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In: Symposium on Discrete Algorithms, pp. 573–582 (1994)

4. Ashlagi, Y., Gottlieb, L., Kontorovich, A.: Functions with average smoothness: structure, algorithms, and learning. In: Belkin, M., Kpotufe, S. (eds.) Conference on Learning Theory, COLT 2021, 15–19 August 2021, Boulder, Colorado, USA, PMLR, Proceedings of Machine Learning Research, vol. 134, pp. 186–236. http://proceedings.mlr.press/v134/ashlagi21a.html (2021)
5. Borchani, H., Varando, G., Bielza, C., Larrañaga, P.: A survey on multi-output regression. Wiley Interdiscip Rev Data Min Knowl Discov **5**(5), 216–233 (2015)
6. Boyd, S., Vandenberghe, L.: Convex Optimization. Information Science and Statistics. Cambridge University Press, Cambridge (2004)
7. Brualdi, R.A.: Introductory Combinatorics, 5th edn. Pearson Prentice Hall, Upper Saddle River (2010)
8. Brudnak, M.: Vector-valued support vector regression. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 1562–1569. IEEE (2006)
9. Bunch, J.R., Hopcroft, J.E.: Triangular factorization and inversion by fast matrix multiplication. Math. Comput. **28**(125), 231–236 (1974)
10. Caruana, R.: Multitask learning. Mach. Learn. **28**(1), 41–75 (1997)
11. Chen, S., Banerjee, A.: An improved analysis of alternating minimization for structured multi-response regression. In: Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS'18, pp. 6617–6628. Curran Associates Inc., USA (2018). http://dl.acm.org/citation.cfm?id=3327757.3327768
12. Christiano, P., Kelner, J.A., Madry, A., Spielman, D.A., Teng, S.H.: Electrical flows, Laplacian systems, and faster approximation of maximum flow in undirected graphs. In: Proceedings of Symposium on Theory of Computing, pp. 273–282 (2011)
13. Cohen, D.T., Kontorovich, A.: Learning with metric losses. In: Loh, P., Raginsky, M. (eds.) Conference on Learning Theory, 2–5 July 2022, London, UK, PMLR, Proceedings of Machine Learning Research, vol. 178, pp. 662–700 (2022). https://proceedings.mlr.press/v178/cohen22a.html
14. Cole, R., Gottlieb, L.A.: Searching dynamic point sets in spaces with bounded doubling dimension. In: Proceedings of the Symposium on Theory of Computing, pp. 574–583 (2006)
15. Davidson, R., MacKinnon, J.G., et al.: Estimation and Inference in Econometrics. OUP Catalogue (1993)
16. Gottlieb, L.A., Kontorovich, A., Krauthgamer, R.: Adaptive metric dimensionality reduction (extended abstract: ALT 2013). Theoretical Computer Science, pp. 105–118 (2016)
17. Gottlieb, L.A., Kontorovich, A., Krauthgamer, R.: Efficient regression in metric spaces via approximate Lipschitz extension. IEEE Trans. Inf. Theory **63**(8), 4838–4849 (2017)
18. Greene, W.H.: Econometric Analysis. Pearson Education India (2003)
19. Greene, W.H.: Econometric Analysis. William H. Greene (2012)
20. Györfi, L., Kohler, M., Krzyzak, A., Walk, H.: A Distribution-Free Theory of Nonparametric Regression. Springer, Cham (2006)
21. Hanneke, S., Kontorovich, A., Kornowski, G.: Near-optimal learning with average Hölder smoothness. CoRR arXiv:2302.06005 (2023)
22. Har-Peled, S., Mendel, M.: Fast construction of nets in low-dimensional metrics and their applications. SIAM J. Comput. **35**(5), 1148–1184 (2006)
23. Jain, P., Tewari, A.: Alternating minimization for regression problems with vector-valued outputs. In: Cortes, C., Lawrence, N.D., Lee, D.D., Sugiyama, M., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 28, pp. 1126–1134. Curran Associates, Inc. (2015). http://papers.nips.cc/paper/5820-alternating-minimization-for-regression-problems-with-vector-valued-outputs.pdf
24. Kakade, S., Tewari, A.: Dudley's theorem, fat shattering dimension, packing numbers. Lecture 15, Toyota Technological Institute at Chicago (2008). http://ttic.uchicago.edu/~tewari/lectures/lecture15.pdf
25. Kimeldorf, G.S., Wahba, G.: A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. Ann. Math. Stat. **41**(2), 495–502 (1970). https://doi.org/10.1214/aoms/1177697089
26. Kirszbraun, M.: Über die zusammenziehende und Lipschitzsche transformationen. Fundam. Math. **22**(1), 77–108 (1934)
27. Koutis, I., Miller, G.L., Peng, R.: A fast solver for a class of linear systems. Commun. ACM **55**(10), 99–107 (2012)
28. Mahabadi, S., Makarychev, K., Makarychev, Y., Razenshteyn, I.: Nonlinear dimension reduction via outer bi-lipschitz extensions. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 1088–1101. ACM (2018)

29. McShane, E.J.: Extension of range of functions. Bull. Am. Math. Soc. **40**(12), 837–842 (1934). https://projecteuclid.org:443/euclid.bams/1183497871
30. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations Of Machine Learning. The MIT Press, Cambridge (2012)
31. Nadaraya, E.A.: Nonparametric Estimation of Probability Densities and Regression Curves. Springer, Cham (1989)
32. Naor, A.: Metric embeddings and Lipschitz extensions (2015)
33. Nesterov, Y., Nemirovskii, A.: Interior-Point Polynomial Algorithms in Convex Programming. SIAM, Philadelphia (1994)
34. Rudin, W.: Principles of Mathematical Analysis. International Series in Pure and Applied Mathematics, 3rd edn. McGraw-Hill Book Co., New York (1976)
35. Spielman, D.A., Teng, S.H.: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: Proceedings of the Symposium on Theory of Computing, vol. 4 (2004)
36. Whitney, H.: Analytic extensions of differentiable functions defined in closed sets. Trans. Am. Math. Soc. **36**(1), 63–89 (1934). http://www.jstor.org/stable/1989708