Efficient Kirszbraun Extension with Applications to Regression

Hanan Zaichyk  
Ben-Gurion University  
zaichyk@post.bgu.ac.il

Armin Biess  
Ben-Gurion University  
armin.biess@gmail.com

Aryeh Kontorovich  
Ben-Gurion University  
karyeh@bgu.ac.il

Yury Makarychev  
Toyota Technological Institute at Chicago  
yury@ttic.edu

March 10, 2022

Abstract

We introduce a framework for performing regression between two Hilbert spaces. This is done based on Kirszbraun’s extension theorem, to the best of our knowledge, the first application of this technique to supervised learning. We analyze the statistical and computational aspects of this method. 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 a quadratic runtime improvement over the state of the art. Our empirical results indicate a dramatic improvement over standard off-the-shelf solvers in our setting.

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, Györfi et al. (2006); Nadaraya (1989). When the target function is assumed to have a specific structure, the regression problem is termed parametric and the optimization problem is finite-dimensional. Linear regression (Mohri et al., 2012, 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 (Borchani et al., 2015), see Related Work.

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 \( F \) id the cost of evaluating the first and second derivatives of the objective and the
constrains \cite{Boyd2004}.

**Motivation and contribution.** The chief motivation of this work was to generalize the
results of \cite{Gottlieb2017}, who provided efficient nonparametric regression methods
in the scalar-output case. Attempts to numerically solve our optimization problem, which
is naturally formulated as a Quadratically Constrained Quadratic Program (QCQP), via
state-of-the-art off the shelf solvers indicated that these are incapable of handling our
framework, even for relatively small data sets and dimensions. This limitation of QCQP
solvers motivated us to develop a specialized algorithm to solve the optimization problem
entailed by our regression setting.

In Section 4, we show that our specialized algorithm dramatically outperforms general-
purposed QCQP solvers; this algorithm, its theoretical analysis, and MATLAB code\cite{GitHub}
are the main contributions of this paper. We introduce a framework for performing regression
between two Hilbert spaces. This is done based on Kirszbraun’s extension theorem — to
the best of our knowledge, the first application of this technique to supervised learning.
This method directly exploits the metrics of the input and output spaces, which makes it
explicitly sensitive to the interaction among the output components. Although our main
contributions are algorithmic, a statistical analysis of our regression technique is provided
in Appendix C.

We formulate the regression problem in two stages: **smoothing** and **extension**, which
are formally described in Section 3. Roughly speaking, on a dataset of size \( n \) with \( a \) input
and \( b \) output dimensions, we formulate the smoothing problem as a Quadratically Con-
strained Quadratic Program (QCQP) problem with \( bn \) variables and \( O(n^2) \) constraints.
The extension problem is also formulated as a QCQP with \( O(n) \) variables and \( O((a+b)n) \)
constraints.

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. When solving large-scale problems, even a modest improvement in the
exponent yields dramatic runtime savings. We propose a Multiplicative Weight Update
(MWU) scheme to solve the smoothing problem, to a constant precision, in runtime
\( O(ma + m^3/2 \log^2 m) \) and the extension problem in runtime \( O(na + nb \log n) \).

**Related work.** Previous approaches to vector-valued regression include \( \varepsilon \)-insensitive
SVM with \( p \)-norm regularization \cite{Brudnak2006}, least-squares and MLE-based methods
\cite{Jain2015}, and (for linear models) the Danzig selector \cite{Chen2018}. According to a recent survey \cite{Borchani2015}, existing methods essentially
“transform the multi-output problem into independent single-output problems.” Some
approaches to multitask learning problems \cite{Caruana1997} exploit relations between
the different tasks. In econometrics, this decoupling of the outputs is made explicit in
the Seemingly Unrelated Regressions (SUR) model \cite{Davidson1993, Greene2003, Greene2012}. These approaches however, do not seem to encapsulate the need of a single vec-
tor output with possibly strong relations between its coordinates. In our approach, we
device a principled approach for leveraging the dependencies via Kirszbraun extension.
The latter has previously been applied by \cite{Mahabadi2018} to dimensionality re-

\cite{GitHub} available at \url{https://github.com/HananZaichyk/Kirszbraun-extension}
duction (unsupervised learning), but to the best of our knowledge has not been used in the supervised learning setting.

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

\[
\begin{align*}
\text{Minimize} & \quad x^\top P_0 x + a^\top x \\
\text{subject to} & \quad x^\top P_i x + a_i x \leq b_i, i = 1, \ldots, m
\end{align*}
\]

where \(a_i, a_i \in [m]\) 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 (Boyd and Vandenberghe 2004). The QCQP is usually solved in practice using log-barrier or primal-dual interior-point methods. The running time of an optimization algorithm based on the 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 (Nesterov and Nemirovskii 1994, 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 \(\omega\) is the matrix multiplication exponent (Bunch and Hopcroft 1974) (the best current upper bound on \(\omega\) is 2.37286 (Alman and Williams 2021)). 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 Weight Update (MWU) scheme is based on the framework of Arora et al. (2012). We include the relevant background and results in the Appendix for completeness.

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

The problem setup, formalized in Section 2, 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 (Section 3.2) 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 3.5 solves the smoothing optimization problem, up to a tolerance \(\varepsilon\), 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 3.1 we accomplish this via \(\varepsilon\)-approximate Kirszbraun extension of the smoothed dataset, in runtime \(O(an + bn(\log n)/\varepsilon^2)\). For small \(a\), an improvement is possible: a data structure can be constructed off-line at a (once) runtime cost of \(2^{O(a)n} \log n\) that allows to answer (multiple) future prediction queries in time

\[
(1/\varepsilon)^{O(a)b} \log n \log \log n.
\]

In Section 3 we compare the performance of our MWU-based approach to a state of the art interior-point based solver and report a significant runtime advantage, which allows to process larger samples and ultimately yields greater accuracy.

Finally, for completeness, in Section C we include a Rademacher-based analysis of the generalization error of our regression algorithm.
2 Formal setup.

**Metric space.** 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 McShane (1934); Whitney (1934), holds. If \(f : A \to \mathbb{R}\) is Lipschitz (under the inherited metric) 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; further, \(H\) is complete in the metric \(d_H(x, x') := \sqrt{\|x - x', x - x'\|}\).

**Kirszbraun theorem.** Kirszbraun (1934) 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 (Naor, 2015).

**Learning problem.** We assume a familiarity with the abstract agnostic learning framework and refer the reader to Mohri et al. (2012) 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 Section C.

3 Learning algorithm

**Overview.** We follow the basic strategy proposed by Gottlieb et al. (2017) for real-valued regression. 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) \(\hat{f} := \arg\min_{f \in F_L} \hat{R}_n(f)\) 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 \(R(f) \leq \hat{R}_n(f) + Q_n(a, b, L)\), 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 in Section C and chooses \(L\) 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 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 \(x^*\), and a precision parameter \(\varepsilon > 0\), we wish to compute \(y^* = f(x^*)\) so that \(\|y^* - f(x_i)\| \leq (1 + \varepsilon)L\|x^* - x_i\|\) for all \(i \in [n]\). Our first result is an efficient algorithm for achieving this:

Theorem 3.1. The approximate Lipschitz extension algorithm \(\text{OnePointExtension}\) has runtime \(O(na + nb \log n/\varepsilon^2)\).

The query runtime can be significantly improved if the dimension of \(X\) is moderate:

Theorem 3.2. There is a data structure for the Lipschitz extension problem of memory size \(O(2^{O(n)}n)\) that can be constructed in time \(O(2^{O(n)}n \log n)\). 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 \((1/\varepsilon)^{O(n)}b \log n \log \log n\).

Analysis. We analyze algorithm \(\text{OnePointExtension}\) and prove Theorems 3.1 and 3.2 via the multiplicative update framework of Arora et al. (2012). In particular, we will invoke their Theorem 3.4, which, for completeness, is reproduced in Section A as Theorem A.1. To simplify the notation, we assume (without loss of generality) that \(L := \|f\|_\text{Lip} = 1\). Let \(\mathcal{P} = \text{Ball}(y^o, \|x^o - x^*\|)\), and define \(h_i(y) = 1 - \frac{\|x^o - y\|}{\|x^o - x_i\|}\) for \(y \in \mathcal{Y}\) and \(i \in \{1, \ldots, n\}\). Then the Lipschitz extension problem is equivalent to the following: find \(y \in \mathcal{P}\) 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 Arora et al. (2012). We now bound the “width” of the problem, proving that \(h_i(y) \in [-2, 1]\) for every \(y \in \mathcal{P}\) (in the notation from Arora et al. (2012), we show that \(\ell \leq 1\) and \(\rho \leq 2\)). Observe that for every \(y \in \mathcal{P}\) and every \(i\), we have (i) \(h_i(y) \leq 1\) as \(\frac{\|y - y^o\|}{\|x^* - x_i\|} \geq 0\) and (ii)

\[
1 - h_i(y) = \frac{\|y - y^o\| + \|y^o - y_i\|}{\|x^* - x_i\|} \leq \frac{\|x^o - x^*\| + \|x^o - x_i\|}{\|x^* - x_i\|} \leq 2 + \frac{\|x^o - x^*\|}{\|x^* - x_i\|} \leq 3.
\]

Here, we used that \(\|y - y^o\| \leq \|x^o - x^*\|\) (which is true since \(y \in \mathcal{P}\)), \(\|y^o - y_i\| \leq \|x^o - x_i\|\) (which is true since \(f\) is 1-Lipschitz), and \(\|x^* - x^o\| \leq \|x^* - x_i\|\) (which is true since \(x^o\) 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 A.1 we design an oracle for the following problem:

Problem 3.3. Given non-negative weights \(w_i\) that add up to 1, find \(y \in \mathcal{P}\) such that

\[
\sum_{i=1}^{n} w_i h_i(y) \geq 0.
\]

Note that Problem 3.3 has a solution, since \(y^*\), the Lipschitz extension of \(f\) to \(x^*\) (whose existence is guaranteed by the Kirszbraun theorem), satisfies (1). Define auxiliary weights
$p_i$ and $q_i$ as follows:

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

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

The oracle finds and outputs $z \in \mathcal{P}$ 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 $V(z) = \|z - z_0\|^2 + \sum_{i=1}^{n} p_i \|z_0 - y_i\|^2$. Then, if $z_0 \in \mathcal{P}$, it sets $z = z_0$; otherwise, $z$ is set to be the point closest to $z_i$ in $\mathcal{P}$, which is

$$z = \frac{x_0 - x^*}{\|x_0 - y^0\|} z_0 + \left(1 - \frac{\|x_0 - x^*\|}{\|z_0 - y^0\|}\right) y^0. $$

This $z$ is computed on lines 6–8 of the algorithm. We verify that $z$ satisfies condition 1. Rewrite condition (1) in terms of weights $q_i$: $Q \sum_{i=1}^{n} q_i \|y_i - y_i\| \leq 1$. Using that

$$\|y^* - y_i\|^2 / \|x^* - x_i\|^2 \leq 1,$$

we get

$$Q \sum_{i=1}^{n} q_i \|z - y_i\| = \sum_{i=1}^{n} w_i \frac{\|z - y_i\|}{\|x^* - x_i\|} \leq \left(\sum_{i=1}^{n} w_i \frac{\|z - y_i\|^2}{\|x^* - x_i\|^2}\right)^{1/2} \left(\sum_{i=1}^{n} w_i\right)^{1/2} \leq \left(\sum_{i=1}^{n} w_i \frac{\|y^* - y_i\|^2}{\|x^* - x_i\|^2}\right)^{1/2} \leq 1. $$

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

**Proof. Proof of Theorem 3.1.** From Theorem A.1 we get that the algorithm finds a $1 + \varepsilon$ approximate solution in $T = \frac{8d \ln m}{\varepsilon^2} = \frac{16d \ln m}{\varepsilon}$ iterations. Computing distances $d_i$ takes $O(n)$ time, each iteration takes $O(\log n)$ time.

**Proof. Proof of Theorem 3.2 (sketch).** Our key observation is that we can run the algorithm from Theorem 3.1 on a subset $X'$ of $X$, which is sufficiently dense in $X$. Specifically, let $x^0$ be a $(1 + \varepsilon)$-approximate nearest neighbour for $x^*$ in $X$. Assume that a subset $X' \subset X$ contains $x^0$ and satisfies the following property: for every $x_i \in X \cap \text{Ball}(x^*, \|x^* - x^0\|/\varepsilon)$, there exists $x_j \in X'$ such that $\|x_j - x_i\| \leq \varepsilon \|x^* - x_i\|$.

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

1. Algorithm from Theorem 3.1 finds $y^*$ such that $\|y_i - y^*\| \leq (1 + O(\varepsilon))L \|x_i - x^*\|$ for all $x_i \in X'$. Consider $x_i \in X$. First, assume that $x_i \in \text{Ball}(x^*, \|x^* - x^0\|/\varepsilon)$. Find $x_j \in X'$ such that $\|x_j - x_i\| \leq \varepsilon \|x^* - x_i\|$. Then

$$\|y_i - y^*\| \leq \|y_i - y_j\| + \|y_j - y^*\| \leq L \|x_i - x_j\| + (1 + \varepsilon)L \|x_j - x^*\| \leq L((2 + \varepsilon) \|x_i - x_j\| + (1 + \varepsilon) \|x_i - x^*\|) \leq (1 + 3\varepsilon + \varepsilon^2)M \|x_i - x^*\|.$$
Finally, we describe an algorithm for the Multi-point Lipschitz extension.

**Algorithm 1 OnePointExtension**

**Require:** labeled sample \((x_i, y_i) \subset (X \times Y)^n, \varepsilon \in (0, 1/2)\) query point \(x^* \in X\), and upper bound \(L \geq \|f\|_{\text{lip}}\) **return** label \(y^*\)

1. let \(x^o\) be the nearest neighbor of \(x^*\) among \(x_1, \ldots, x_n\); \(y^o = f(x^o); d^o = \|x^o - x^*\|\)
2. initialize weights \(w_i^{(1)}, \ldots, w_i^{(1)}\) as follows: \(w_i^{(1)} = 1/n\) for every \(i\)
3. let \(d_i = \|x_i - x^*\|/L\) for every \(i\)
4. let \(T = \lceil \frac{16 \ln n}{\varepsilon^2} \rceil\) (the number of iterations)
5. for \(t = 1\) to \(T\) do
6. let \(P = \sum_{i=1}^{n} w_i^{(t)} / d_i^2\) and \(p_i = \frac{w_i^{(t)}}{P^{1/2}}\).
7. let \(z_0 = \sum_{i=1}^{n} p_i y_i\) and \(\Delta = \|z_0 - y^o\|\)
8. if \(\Delta \leq d^o\) then \(z^{(t)} = z_0\) else \(z^{(t)} = \frac{d^o}{\Delta} z_0 + \frac{\Delta - d^o}{\Delta} y^o\)
9. update the weights: \(w_i^{(t+1)} = (1 + \varepsilon/\Delta) w_i^{(t)}\) for every \(i\)
10. normalize the weights: compute \(W = \sum_{i=1}^{n} w_i^{(t)}\) and let \(w_i^{(t+1)} = w_i^{(t)} / W\) for every \(i\)
11. end for
12. return \(z = \frac{1}{T} \sum_{t=1}^{T} z^{(t)}\)

as required. Now assume that \(x_i \notin \text{Ball}(x^*, \|x^* - x^o\|/\varepsilon)\).

\[
\|y_i - y^o\| \leq \|y_i - y^o\| + \|y^o - y^*\| \\
\leq L\|x_i - x^o\| + (1 + \varepsilon) L\|x^o - x^*\| \\
\leq L(\|x_i - x^*\| + (2 + \varepsilon)\|x^o - x^*\|) \\
\leq (1 + \varepsilon)^2 L\|x_i - x^*\|.
\]

We use a data structure \(D\) for approximate nearest neighbor search in \(X\). We employ one of the constructions for low-dimensional Euclidean spaces, by either of Arya et al. (1994) or Har-Peled and Mendel (2006). Using \(D\), we can find a \((1 + \varepsilon/3)\)-approximate nearest neighbor of a point in \(\mathbb{R}^n\) in time \((1/\varepsilon)^{O(n)} \log n\). Recall that we can construct \(D\) in \(O(2^{O(n)} n \log n)\) time, and it requires \(O(2^{O(n)} n \log n)\) space. Suppose that we get a query point \(x^*\). We first find an approximate nearest neighbor \(x^o\) for \(x^*\). Let \(r = \|x^o - x^*\|\). 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 \(D\)). Let \(X' = \{x(p) : p \in N'\} \cup \{x^o\}\). Consider \(x_i \in \text{Ball}(x^*, r/\varepsilon)\). There is \(p \in X'\) at distance at most \(\varepsilon r/3\) from \(x_i\). Let \(x_j = x(p) \in X'\). Then

\[
\|p - x_j\| \leq (1 + \varepsilon/3)\|x_i - p\| \leq (1 + \varepsilon/3)\varepsilon r/3
\]

and

\[
\|x_i - x_j\| \leq \|x_i - p\| + \|p - x_j\| \\
\leq \varepsilon r/3 + (1 + \varepsilon/3)\varepsilon r/3 \\
\leq (2 + \varepsilon/3)\varepsilon r/3 \leq 3\|x^* - x_i\|,
\]
as required. The size of \(X'\) is at most the size of \(N'\), which is \((1/\varepsilon)^{O(n)}\).

**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 Section 3.1. We are given a set of points \(X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^n\) and their images
of matrices

Here, $Y$ are the Lipschitz constraints. There is an algorithm for the Multi-point Lipschitz Extension problem— that is, find $Z$ such that there are no edges $(i, j) \in E$. We reformulate the ERM problem as:

$\hat{f} := \text{argmin}_{f \in F_L} \hat{R}_n(f)$

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

$$\text{Minimize } \Phi(Y, \tilde{Y}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2$$

subject to $\|\tilde{y}_i - \tilde{y}_j\| \leq L\|x_i - x_j\|$ for all $i, j \in [n]$

$\Phi(Y, \tilde{Y}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2$ is the distortion, and $\|\tilde{y}_i - \tilde{y}_j\| \leq L\|x_i - x_j\|$ for all $i, j \in [n]$ are the Lipschitz constraints. Here, $Y = (y_1, \ldots, y_n)$ and $\tilde{Y} = (\tilde{y}_1, \ldots, \tilde{y}_n)$ (the columns of matrices $Y$ and $\tilde{Y}$ are vectors $y_1, \ldots, y_n$ and $\tilde{y}_1, \ldots, \tilde{y}_n$, respectively). Notice that

**Algorithm 2 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 $G = ([n], E)$, and $M \equiv (Y, \tilde{Y} = (y_{n+1}, \ldots, y_{n+n'})$

1. let $Y = (y_1, \ldots, y_n)$
2. let $r_{ij} = L\|x_i - x_j\|$ for $(i, j) \in E$
3. let $w_{ij} = 1/m$ for $(i, j) \in E$
4. let $T = c_1[\sqrt{\ln n}/2^2]$ (the number of iterations)
5. for $t = 1$ to $T$
6. let $\lambda_{ij} = w_{ij} \epsilon/m$ for $(i, j) \in E$
7. define $n' \times n$ times matrix $K$ as:
8. $K_{ij} = \lambda_{i+n, j+n}$ if $(i + n, j + n) \in E$ and 0 otherwise.
9. solve $L(Y^t)^\top = KY^\top$ for $Y^t$
10. update the weights: $w_{ij} = \left(1 + c_2 \epsilon \left(\frac{\|y_{i}-y_{j}\|}{r_{ij}} - 1\right)\right)w_{ij}$ for every $(i, j) \in E$
11. normalize the weights: $W = \sum_{(i,j) \in E} w_{ij}$
12. $w_{ij} = w_{ij}/W$ for all $(i, j) \in E$
13. end for
14. return $\tilde{Y} = \frac{1}{t} \sum_{t=1}^{T} Y^t$

$Y = \{y_1, \ldots, y_n\} \subset \mathbb{R}^b$ 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 $\|y_i - y_j\| \leq (1 + \epsilon)L\|x_i - x_j\|$ 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 3.4.** There is an algorithm for the Multi-point Lipschitz Extension problem that runs in time

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

where $m = |E|$.

The algorithm and its analysis are almost identical to those for the Lipschitz Smoothing problem. (see Theorem 3.5).

**3.2 Smoothing**

**Problem statement.** We reformulate the ERM problem $\hat{f} := \text{argmin}_{f \in F_L} \hat{R}_n(f)$ 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

$$\text{Minimize } \Phi(Y, \tilde{Y}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2$$

subject to $\|\tilde{y}_i - \tilde{y}_j\| \leq L\|x_i - x_j\|$, $i, j \in [n]$

$\Phi(Y, \tilde{Y}) := \sum_{i=1}^{n} \|y_i - \tilde{y}_i\|^2$ is the distortion, and $\|\tilde{y}_i - \tilde{y}_j\| \leq L\|x_i - x_j\|$ for all $i, j \in [n]$ are the Lipschitz constraints. Here, $Y = (y_1, \ldots, y_n)$ and $\tilde{Y} = (\tilde{y}_1, \ldots, \tilde{y}_n)$ (the columns of matrices $Y$ and $\tilde{Y}$ 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 consider a more general variant of this problem where we are given a set of edges $E$ on $\{1, \ldots, n\}$, and the goal is to ensure that the Lipschitz constraints $\|\tilde{y}_i - \tilde{y}_j\| \leq L\|x_i - x_j\|$ hold (only) for $(i, j) \in E$. The original problem corresponds to the case when $E$ is the complete graph, $(E_{ij} = L\|x_i - x_j\|)$. Importantly, if the doubling dimension $d_{\text{dim}} X$ is low, we can solve the original problem by letting $([n], E)$ be a $(1 + \varepsilon)$-stretch spanner; then $m = n(1/\varepsilon)^{O(d_{\text{dim}})}$ (this approach was previously used by Gottlieb et al. (2017); see also Har-Peled and Mendel (2006, Section 8.2), who used a similar approach to compute the doubling constant). Our algorithm for Lipschitz Smoothing iteratively solves Laplace’s problem in the graph $G$. We proceed to define this problem and present a closed-form formula for the solution.

**Laplace’s problem.** We are given vectors $\{y_i\}$, graph $G$, and additionally vertex weights $\lambda_i \geq 0$ (for $i \in [n]$) and edge weights $\mu_{ij} \geq 0$ (for $(i, j) \in E$), find $\tilde{y}_i$ so as to

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

Let $L$ be the Laplacian of $G = ([n], E)$ with edge weights $\mu_{ij}$; that is $L_{ii} = \sum_{j, j \neq i} \mu_{ij}$ and $L_{ij} = -\mu_{ij}$ for $i \neq j$. Let $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$. Then

$$(L + \Lambda) \tilde{Y}^T = \Lambda Y^T.$$ 

This equation can be solved separately for each of $b$ rows of $Y^T$ using an nearly-linear equation solver for diagonally dominant matrices by Koutis et al. (2012) in total time $O(bm \log n \log(1/\varepsilon))$ (see also the paper by Spielman and Teng (2004), which presented the first nearly-linear time solve for diagonally dominant matrices).

We solve the Lipschitz Smoothing problem via the multiplicative weight update algorithm **LipschitzSmooth**, presented below. It was inspired by the algorithm for finding maximum flow using electrical networks by Christiano et al. (2011).

**Analysis.** Let $Y^*$ be the optimal solution to the Lipschitz Smoothing problem and and $\Phi_0$ be a $(1 + \varepsilon)$ approximation to the optimal value; that is,

$$\Phi(Y, Y^*) \leq \Phi_0 \leq (1 + \varepsilon) \Phi(Y, Y^*)$$

(we assume that $\Phi_0$ is given to the algorithm; note that $\Phi_0$ can be found by binary search).

As in Section 3.1, we use the multiplicative-weight update (MWU) method. Let

$$h_{\Phi}(\tilde{Y}) = 1 - \sqrt{\frac{\Phi(Y, \tilde{Y})}{\Phi_0}},$$

$$h_{ij}(\tilde{Y}) = 1 - \frac{\|\tilde{y}_i - \tilde{y}_j\|}{M\|x_i - x_j\|}, \quad (i, j) \in E.$$ 

Note that functions $h_{\Phi}$ and $h_{ij}$ are concave.
Algorithm 3 LipschitzSmooth

Require: vectors $X = x_1, \ldots, x_n \in \mathbb{R}^a$, $Y = y_1, \ldots, y_n \in \mathbb{R}^b$, graph $G = ([n], E)$, $M$ and $\Phi_0$

Output: $\tilde{Y}$

1: let $Y \equiv (y_1, \ldots, y_n)$
2: let $r_{ij} = L\|x_i - x_j\|$ for $(i, j) \in E$
3: let $w_{ij} = 1/(m + 1)$ for $(i, j) \in E$, where $m = |E|$
4: let $w_{\Phi} = 1/(m + 1)$
5: let $T = c_1 \lceil \sqrt{\frac{\ln n}{\varepsilon^2}} \rceil$ (the number of iterations)
6: for $t = 1$ to $T$
7:    let $L$ be the Laplacian of $G$ with edge weights $\mu_{ij} = \frac{w_{ij} + \varepsilon/(m + 1)}{r_{ij}}$
8:    let $\lambda = (w_{\Phi} + \varepsilon/(m + 1))/\Phi_0$
9:    solve $(\lambda^{-1} L + I)(\tilde{Y}^t)^\top = Y^\top$ for $\tilde{Y}^t$
10:   update the weights: $w_{\Phi} = \left(1 + c_2 \varepsilon \left(\sqrt{\frac{\Phi(Y, \tilde{Y}^t)}{\Phi_0}} - 1\right)\right)w_{\Phi}$
11:   $w_{ij} = \left(1 + c_2 \varepsilon \left(\frac{\|y_i - \tilde{y}_j\|}{r_{ij}} - 1\right)\right)w_{ij}$ for every $(i, j) \in E$
12:   normalize the weights: $W = w_{\Phi} + \sum_{(i, j) \in E} w_{ij}$
13:   $w_{\Phi} = w_{\Phi}/W$
14:   $w_{ij} = w_{ij}/W$ for all $(i, j) \in E$
15: return $\tilde{Y} = \frac{1}{T} \sum_{t=1}^T \tilde{Y}^t$
Observe that $h_{\Phi}(\tilde{Y}^*) \geq 0$ and $h_{ij}(\tilde{Y}^*) \geq 0$ for every $(i, j) \in E$. On the other hand, if $h_{\Phi}(\tilde{Y}) \geq -\varepsilon$ and $h_{ij}(\tilde{Y}) \geq -\varepsilon$, then

$$\Phi(Y, \tilde{Y}) \leq (1 + \varepsilon)^2\Phi_0$$

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

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

**Theorem 3.5.** There is an algorithm for the Lipschitz Smoothing problem that runs in time

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

where $m = |E|$.

**Proof.** Proof of Theorem 3.5. From Theorem 3.5 in Arora et al. (2012), we get that the algorithm finds an $O(\varepsilon)$ approximate solution in $T = O\left(\frac{\sqrt{m/\varepsilon} \ln m}{\varepsilon^2}\right) = \frac{\sqrt{\pi}}{\varepsilon^{5/2}}$ 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 $X$.

4 Experiments

To illustrate the utility of our framework, we designed two simple non-linear transformation problems where the input and output are both scalars. Our data was generated uniformly at random over $[-2\pi, 2\pi]$ and evaluated the performance on two cases: $f(x) = x^3$ and $f(x) = \sin(x)$.

**Results.** In order to perform a fair, apples-to-apples comparison, we implemented both Algorithms 3 and 1 in Matlab, which standard, optimized QCQP solvers, and performed the regression problem via the Kirszbraun extension technique. We compared the results of this learning method when using our methods for the optimization problems (MWU) vs using Matlab’s QCQP solver based on the interior-point algorithm (IntPt). We considered the squared Euclidean distance as the loss function. We ran several tests using different size data sets of 20, 100, 200, 500, and 1000 random points as training set, and 100 test points in all experiments. For reproducibility, we’ve used Matlab’s random seed 1 in all our runs. All the tests where conducted on the same Macbook pro computer. The numeric comparison (Tables 1-5) shows undoubtedly supremacy of the MWU over the IntPt method both in efficiency and better learning. MWU method is able to optimize a data set of several thousands data points, while the IntPt based method could not complete its process in “reasonable” time (over 10 hours night run) with more than $N = 200$ training points. In terms of solving the learning problem, the MWU able to solve the QCQP problem and produce accurate smoothing and more accurate extensions functions as the data size grows. The IntPt method, on the other hand, able to meet all the constrains of the problem only with very small data set (less then 50 training points) which is insufficient data to for learning. Tables 1-5 shows that for 20 training points, the IntPt is able to train in 2.474 seconds and completely over fit the data set with 0 ERM, which leads to expected very poor generalization due to the size of the data (Table 5). On larger datasets the IntPt optimization fails to correctly solve the optimization problems with respect to all of the constrains. This result in several “heavy” outliers which affect
heavily on the average square error of both smoothing and extension phases as can be seen in tables 1-5. Table 6 shows a graphical comparison for both implementations when the training set has \( N = 100 \) points. The “heavy outliers” can be spotted easily on the graph, and explain why the same learning algorithm has such big differences when optimised with two different methods.

Tables 1-6 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 blank entries in the tables indicate that the process did not terminate in the time allotted (12 hours).

### Table 1: ERM of the smoothing process

| training points | 20   | 100  | 200  | 500  | 1000 |
|-----------------|------|------|------|------|------|
| Algorithm       |      |      |      |      |      |
| MWU             | 247.9405 | 0.3333 | 0.31581 | 0.31854 | 0.36143 |
| IntPt           | 4.1e-18 | 46023.7964 | 353691.64 | \n
### Table 2: Cross validation running time over the smoothing process in seconds

| training points | 20   | 100  | 200  | 500  | 1000 |
|-----------------|------|------|------|------|------|
| Algorithm       |      |      |      |      |      |
| MWU             | 2.7505 | 19.9428 | 46.2479 | 212.4875 | 1243.2395 |
| IntPt           | 18.1733 | 692.6523 | 4087.6655 | \n
### Table 3: Single* smoothing process running time in seconds

| training points | 20   | 100  | 200  | 500  | 1000 |
|-----------------|------|------|------|------|------|
| Algorithm       |      |      |      |      |      |
| MWU             | 0.092 | 0.7051 | 1.6326 | 8.0798 | 45.1497 |
| IntPt           | 2.474 | 155.5521 | 766.5433 | \n
### Table 4: Extension avg loss

| training points | 100 | 200 | 500 | 1000 |
|-----------------|-----|-----|-----|------|
| Algorithm       |     |     |     |      |
| MWU             | 1119.4705 | 0.3333 | 0.37267 | 0.43678 | 0.52797 |
| IntPt           | 3065.5698 | 9475.260 | 9475.4864 | \n
### 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 achieved via Kirszbraun extension). Numerically solving our
Table 5: Extension running time for all test points in seconds

| training points | Algorithm | Avg. loss  |
|-----------------|-----------|------------|
|                 | MWU       | 0.054      |
|                 | IntPt     | 1.3367     |
|                 |           | 0.0014903  |
|                 |           | 0.002826   |
|                 |           | 0.0051038  |
|                 |           | 0.0089853  |
|                 | 100       | 0.002826   |
|                 | 200       | 0.0051038  |
|                 | 500       | 0.0089853  |
|                 | 1000      | 0.012831   |

Table 6: Visual comparison between our MWU based algorithm (first row) and IntPt (Matlab’s) based algorithm (second row). 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 in high accuracy, the Intpt method has several “heavy” outliers which reduce significantly its average squared error.

Smoothing

|                  | MWU       | IntPt     |
|------------------|-----------|-----------|
|                  |           |           |
|                  |           |           |

Extension

|                  | MWU       | IntPt     |
|------------------|-----------|-----------|
|                  |           |           |
|                  |           |           |

Optimization problems has indicated a need for a more efficient solver for our optimization problems than off the shelf state-off-the-art 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 shows dramatically run time improvement for both optimization problems thus indicating that this algorithms are the main contribution off this work and are interest topic for future research on their own. Our code is also provided for reproducibility and to facilitate usage.

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.
References

Alman J, Williams VV (2021) A refined laser method and faster matrix multiplication. In: Proceedings of the Symposium on Discrete Algorithms, SIAM, pp 522–539

Arora S, Hazan E, Kale S (2012) The multiplicative weights update method: a meta-algorithm and applications. Theory of Computing 8(1):121–164

Arya S, Mount DM, Netanyahu N, Silverman R, Wu AY (1994) An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. In: Symposium on Discrete Algorithms, pp 573–582

Borchani H, Varando G, Bielza C, Larrañaga P (2015) A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5(5):216–233

Boyd S, Vandenberghe L (2004) Convex Optimization. Information Science and Statistics, Cambridge University press

Brudnak M (2006) Vector-valued support vector regression. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, IEEE, pp 1562–1569

Bunch JR, Hopcroft JE (1974) Triangular factorization and inversion by fast matrix multiplication. Mathematics of Computation 28(125):231–236

Caruana R (1997) Multitask learning. Machine learning 28(1):41–75

Chen S, Banerjee A (2018) An improved analysis of alternating minimization for structured multi-response regression. In: Proceedings of the 32Nd International Conference on Neural Information Processing Systems, Curran Associates Inc., USA, NIPS’18, pp 6617–6628, URL http://dl.acm.org/citation.cfm?id=3327757.3327768

Christiano P, Kelner JA, Madry A, Spielman DA, Teng SH (2011) Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In: Proceedings of Symposium on Theory of Computing, pp 273–282

Davidson R, MacKinnon JG, et al. (1993) Estimation and inference in econometrics. OUP Catalogue

Gottlieb LA, Kontorovich A, Krauthgamer R (2016) Adaptive metric dimensionality reduction (extended abstract: ALT 2013). Theoretical Computer Science pp 105–118

Gottlieb LA, Kontorovich A, Krauthgamer R (2017) Efficient regression in metric spaces via approximate lipschitz extension. IEEE Transactions on Information Theory 63(8):4838–4849

Greene WH (2003) Econometric analysis. Pearson Education India

Greene WH (2012) Econometric Analysis. William H. Greene

Györfi L, Kohler M, Krzyzak A, Walk H (2006) A distribution-free theory of nonparametric regression. Springer Science & Business Media

Har-Peled S, Mendel M (2006) Fast construction of nets in low-dimensional metrics and their applications. SIAM Journal on Computing 35(5):1148–1184
Jain P, Tewari A (2015) Alternating minimization for regression problems with vector-valued outputs. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) Advances in Neural Information Processing Systems 28, Curran Associates, Inc., pp 1126–1134, URL http://papers.nips.cc/paper/5820-alternating-minimization-for-regression-problems-with-vector-valued-outputs.pdf

Kirszbraun M (1934) Über die zusammenziehende und Lipschitzsche transformationen. Fundamenta Mathematicae 22(1):77–108

Koutis I, Miller GL, Peng R (2012) A fast solver for a class of linear systems. Communications of the ACM 55(10):99–107

Mahabadi S, Makarychev K, Makarychev Y, Razenshteyn I (2018) Nonlinear dimension reduction via outer bi-lipschitz extensions. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, ACM, pp 1088–1101

McShane EJ (1934) Extension of range of functions. Bull Amer Math Soc 40(12):837–842, URL https://projecteuclid.org:443/euclid.bams/1183497871

Mohri M, Rostamizadeh A, Talwalkar A (2012) Foundations Of Machine Learning. The MIT Press

Nadaraya EA (1989) Nonparametric estimation of probability densities and regression curves. Springer

Naor A (2015) Metric embeddings and lipschitz extensions

Nesterov Y, Nemirovskii A (1994) Interior-point polynomial algorithms in convex programming. SIAM, Philadelphia, PA

Spielman DA, Teng SH (2004) Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: Proceedings of the Symposium on Theory of Computing, vol 4

Whitney H (1934) Analytic extensions of differentiable functions defined in closed sets. Transactions of the American Mathematical Society 36(1):63–89, URL http://www.jstor.org/stable/1989708
A  The Arora-Hazan-Kale result

For completeness, we quote here verbatim (except for the numbering) the relevant definitions and results from (Arora et al. [2012] Sec. 3.3.1, p. 137).

Imagine that we have the following feasibility problem:

\[ \exists! \mathbf{x} \in \mathcal{P} : \forall i \in [m] : f_i(\mathbf{x}) \geq 0, \]  

(2)

where \( \mathcal{P} \in \mathbb{R}^n \) is a convex domain, and for \( i \in [m] \), \( f_i : \mathcal{P} \rightarrow \mathbb{R} \) are concave functions. We wish to satisfy this system approximately, up to an additive error of \( \varepsilon \). We assume the existence of an Oracle, which, when given a probability distribution \( \mathbf{p} = (p_1, p_2, \ldots, p_m) \) solves the following feasibility problem:

\[ \exists! \mathbf{x} \in \mathcal{P} : \sum_i p_i f_i(\mathbf{x}) \geq 0. \]  

(3)

An Oracle is said to be called \((\ell, \rho)\)-bounded if there is a fixed subset of constraints \( I \subseteq [m] \) such that whenever it returns a feasible solution \( \mathbf{x} \) to (2), all constraints \( i \in I \) take values \( \mathbf{x} \) in the range \([-\ell, \rho]\) on the point \( \mathbf{x} \), and all the rest take values in \([-\rho, \ell]\).

Theorem A.1 (Theorem 3.4 in Arora et al. [2012]). Let \( \varepsilon > 0 \) be a given error parameter. Suppose there exists an \((\ell, \rho)\)-bounded Oracle for the feasibility problem (2). Assume the \( \ell \geq \varepsilon / 2 \) then there is an algorithms which either solves the problem up to an additive error of \( \varepsilon \), or correctly concludes that the system \( \mathbf{s} \) 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.

B  Approximate oracle

To use the MWU method (see Theorem 3.5 in Arora et al. [2012]), we design an approximate oracle for the following problem.

Problem B.1. Given non-negative edge weights \( w_\Phi \) and \( w_{ij} \), which add up to 1, find \( \tilde{\mathbf{Y}} \) such that

\[ w_\Phi h_\Phi(\tilde{\mathbf{Y}}) + \sum_{(i,j) \in \mathcal{E}} w_{(i,j)} h_{(i,j)}(\tilde{\mathbf{Y}}) \geq -\varepsilon. \]  

(4)

Let \( \mu_{ij} = w_{ij} + \varepsilon/(m+1) \) and \( \lambda_i = \lambda = (w_\Phi + \varepsilon/(m+1))/\Phi_0 \). We solve Laplace’s problem with parameters \( \mu_{ij} \) and \( \lambda_i \) (see Section 3.2 and Line 9 of the algorithm). We get a matrix \( \tilde{\mathbf{Y}} = (\tilde{y}_1, \ldots, \tilde{y}_n) \) minimizing

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

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

\[ \lambda \sum_{i=1}^n \|y_i - \tilde{y}_i\|^2 + \sum_{(i,j) \in \mathcal{E}} \mu_{ij} \|\tilde{y}_i - \tilde{y}_j\|^2 \leq \lambda \sum_{i=1}^n \|y_i - \tilde{y}_i^*\|^2 + \sum_{(i,j) \in \mathcal{E}} \mu_{ij} \|\tilde{y}_i^* - \tilde{y}_j^*\|^2 \]

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

\[ \leq \left( w_\Phi + \sum_{(i,j) \in \mathcal{E}} w_{ij} \right) + \varepsilon = 1 + \varepsilon. \]
We verify that \( \tilde{Y} \) is a feasible solution for Problem B.1. We have

\[
1 - \left( w_\Phi(\tilde{Y}) + \sum_{(i,j) \in E} w_{i,j} h_{i,j}(y) \right) = w_\Phi(1 - h_\Phi) + \sum_{(i,j) \in E} w_{i,j} (1 - h_{i,j}(y))
\]

\[
\leq \sqrt{w_\Phi(1 - h_\Phi)^2 + \sum_{(i,j) \in E} w_{i,j} (1 - h_{i,j}(y))^2}
\]

\[
= \sqrt{\frac{\Phi(Y, \tilde{Y})}{\Phi_0} + \sum_{(i,j) \in E} w_{i,j} \frac{\|\tilde{y}_i - \tilde{y}_j\|^2}{M^2\|x_i - x_j\|^2}}
\]

\[
\leq \sqrt{\lambda \Phi(Y, \tilde{Y}) + \sum_{(i,j) \in E} w_{i,j} \mu_{i,j} \|\tilde{y}_i - \tilde{y}_j\|^2}
\]

\[
\leq \sqrt{1 + \varepsilon} \leq 1 + \varepsilon,
\]

as required.

Finally, we bound the width of the problem. We have \( h_\Phi(\tilde{Y}) \leq 1 \) and \( h_{i,j}(\tilde{Y}) \leq 1 \). Then, using (5), we get

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

Therefore, \( -h_\Phi(\tilde{Y}) \leq O(\sqrt{m/\varepsilon}) \).

Similarly,

\[
(1 - h_{i,j}(\tilde{Y}))^2 = \frac{\|y_i - \tilde{y}_i\|^2}{M^2\|x_i - x_j\|^2} \leq \frac{1 + \varepsilon}{\mu_{i,j} \cdot M^2\|x_i - x_j\|^2} \leq \frac{(1 + \varepsilon)(m + 1)}{\varepsilon}.
\]

Therefore, \( -h_{i,j}(\tilde{Y}) \leq O(\sqrt{m/\varepsilon}) \).

### C Generalization bounds

Let \( \mathcal{X} \subset \mathbb{R}^k \) and \( \mathcal{Y} \subset \mathbb{R}^\ell \) be the unit balls of their respective Hilbert spaces (each endowed with the \( \ell_2 \) norm \( \| \cdot \| \) and corresponding inner product) and \( \mathcal{H}_L \subset \mathcal{Y}^{\mathcal{X}} \) be the set of all \( L \)-Lipschitz mappings from \( \mathcal{X} \) to \( \mathcal{Y} \). In particular, every \( h \in \mathcal{H}_L \) satisfies

\[
\|h(x) - h(x')\| \leq L\|x - x'\|, \quad x, x' \in \mathcal{X}.
\]

Let \( \mathcal{F}_L \subset \mathbb{R}^{\mathcal{X} \times \mathcal{Y}} \) be the loss class associated with \( \mathcal{H}_L \):

\[
\mathcal{F}_L = \{ \mathcal{X} \times \mathcal{Y} \ni (x, y) \mapsto f(x, y) = f_h(x, y) := \|h(x) - y\|; h \in \mathcal{H}_L \}.
\]

In particular, every \( f \in \mathcal{F}_L \) satisfies \( 0 \leq f \leq 2 \).

Our goal is to bound the Rademacher complexity of \( \mathcal{F}_L \). We do this via a covering numbers approach.

The empirical Rademacher complexity of a collection of functions \( \mathcal{F} \) mapping some set \( Z_1, \ldots, Z_n \subset \mathbb{Z}^n \) to \( \mathbb{R} \) is defined by:

\[
\hat{\mathcal{R}}(\mathcal{F}; Z) = \mathbb{E}\left[ \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(Z_i) \right].
\]

(5)
Recall the relevance of Rademacher complexities to uniform deviation estimates for the risk functional $R(\cdot)$ [Mohri et al. 2012, Theorem 3.1]: for every $\delta > 0$, with probability at least $1 - \delta$, for each $h \in \mathcal{H}_L$:

\[
R(h(z)) \leq \hat{R}_n(h(z)) + 2\hat{R}_n(F_L) + 6\sqrt{\frac{\ln(2/\delta)}{2n}}. \tag{6}
\]

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

\[
|f_h(x, y) - f_h(x', y')| = \|h(x) - y - h(x') - y'\|
\leq \|h(x) - y - (h(x') - y')\|
\leq \|h(x) - h(x')\| + \|y - y'\|
\leq L \|x - x'\| + \|y - y'\|
\leq (L \lor 1) \|(x, y) - (x', y')\|_\mathcal{Z},
\]

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

Since we restricted the domain and range of $\mathcal{H}_L$, respectively, to the unit balls $B_X$ and $B_Y$, the domain of $\mathcal{F}_L$ becomes $B_{\mathcal{Z}} := B_{\mathcal{X}} \times B_{\mathcal{Y}}$ and its range is $[0, 2]$. Let us recall some basic facts about the $\ell_2$ covering of the $k$-dimensional unit ball

\[
\mathcal{N}(t, B_{\mathcal{X}}, \|\cdot\|) \leq (3/4)^k;
\]

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

\[
\mathcal{C}_Z := \{E = F \times G \subset \mathcal{Z} : F \in \mathcal{C}_X, G \in \mathcal{C}_Y\}
\]

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

\[
\mathcal{N}(t, B_{\mathcal{Z}}, \|\cdot\|_\mathcal{Z}) \leq (6/4)^{2k}.
\]

Finally, we endow $F_L$ with the $\ell_\infty$ norm, and use a Kolmogorov-Tihomirov type covering estimate (see, e.g., Gottlieb et al. 2016, Lemma 5.2):

\[
\log \mathcal{N}(t, F_L, \|\cdot\|_\infty) \leq (96(L + 1)/t)^{2k} \log(8/t).
\]

We can now use [Gottlieb et al. 2016, Theorem 4.3]):

**Theorem C.1.** Let $\mathcal{F}_L$ be the collection of $L$-Lipschitz $[0, 2]$-valued functions defined on a metric space $(\mathcal{Z}, \|\cdot\|_\mathcal{Z})$ with diameter 1 and doubling dimension $d$. Then $\hat{R}_n(F_L; \mathcal{Z}) = O\left(\frac{L^k}{n^{1/d + 1}}\right)$.

Putting $d = k + \ell$ yields our generalization bound:

\[
Q_n(k, \ell, L) = R(h(z)) - \hat{R}_n(h(z)) = O\left(\frac{L}{n^{1/d + 1}}\right). \tag{7}
\]

**D Additional experiments.**

For completeness we add here the comparison of the results from the experiment for $f(x) = \sin(x)$ for $x \in [-2\pi, 2\pi]$. 

18
Table 7: ERM of the smoothing process

| training points | 100 | 200 | 500 | 1000 |
|-----------------|-----|-----|-----|------|
| Algorithm       | Avg. loss |
| MWU             | 5.5e-09 | 1.6639e-08 | 3.7683e-08 | 6.8339e-08 |
| IntPt           | 5.3e-16 | 5913495.3623 |     |      |

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

| training points | 100 | 200 | 500 | 1000 |
|-----------------|-----|-----|-----|------|
| Algorithm       | Avg. loss |
| MWU             | 5.1933 | 11.3827 | 66.5677 | 335.0659 |
| IntPt           | 3508.4048 | 3936.6494 |     |      |

Table 9: Single* smoothing process running time in seconds

| training points | 100 | 200 | 500 | 1000 |
|-----------------|-----|-----|-----|------|
| Algorithm       | Avg. loss |
| MWU             | 0.78109 | 1.8081 | 8.1855 | 48.5127 |
| IntPt           | 114.5577 | 778.1016 |     |      |

Table 10: Extension avg loss

| training points | 100 | 200 | 500 | 1000 |
|-----------------|-----|-----|-----|------|
| Algorithm       | Avg. loss |
| MWU             | 5.5e-09 | 1.09e-08 | 3.3e-08 | 7.3e-08 |
| IntPt           | 0.2877 | 0.83248 |     |      |

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

| training points | 100 | 200 | 500 | 1000 |
|-----------------|-----|-----|-----|------|
| Algorithm       | Avg. loss |
| MWU             | 0.0026 | 0.0038 | 0.0061 | 0.0095 |
| IntPt           | 4.3164 | 1.824 |     |      |
Figure 1: Figures (a)-(d) demonstrate the comparison, where training size is 200 random points, and \( f = x^3 \). Figures (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.