Proceedings of Machine Learning Research vol 195:1–32, 2023 36th Annual Conference on Learning Theory

Fast Algorithms for a New Relaxation of Optimal Transport

Moses Charikar MOSES@CS.STANFORD.EDU
Stanford University

Beidi Chen BEIDIC@ANDREW.CMU.EDU
Carnegie Mellow University

Christopher Ré CHRISMRE@CS.STANFORD.EDU
Stanford University

Erik Waingarten EWAINGAR@CIS.UPENN.EDU
University of Pennsylvania

Editors: Gergely Neu and Lorenzo Rosasco

Abstract

We introduce a new class of objectives for optimal transport computations of datasets in high-dimensional Euclidean spaces. The new objectives are parametrized by $\rho \geq 1$, and provide a metric space $\mathcal{R}_\rho(\cdot,\cdot)$ for discrete probability distributions in $\mathbb{R}^d$. As ρ approaches 1, the metric approaches the Earth Mover's distance, but for ρ larger than (but close to) 1, admits significantly faster algorithms. Namely, for distributions μ and ν supported on n and m vectors in $\mathbb{R}^d$ of norm at most r and any $\varepsilon > 0$, we give an algorithm which outputs an additive εr-approximation to $\mathcal{R}_\rho(\mu,\nu)$ in time $(n+m)\cdot \mathrm{poly}((nm)^{(\rho-1)/\rho}\cdot 2^{\rho/(\rho-1)}/\varepsilon)$.

Keywords: Optimal transport, Earth Mover's distance, Sinkhorn distance

1. Introduction

This paper is about algorithms for optimal transport problems in high dimensional Euclidean spaces. At a very high level, optimal transport problems provide a convenient metric space between probability distributions supported on vectors in geometric spaces. The most classical such problem is the Earth Mover's Distance (EMD). Let μ and ν be two distributions supported on vectors in $\mathbb{R}^d$. The Earth Mover's Distance between μ and ν, also known as the Wasserstein-1 distance, is given by minimizing the average distance between pairs of points sampled from a coupling γ of μ and ν:

$$\mathsf{EMD}(\mu,\nu) = \min\left\{ \mathop{\mathbf{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma}\left[\|\boldsymbol{x}-\boldsymbol{y}\|_2\right] : \gamma \text{ is a coupling of } \mu \text{ and } \nu^{1} \right\}. \tag{1}$$

Importantly, the Earth Mover's distance is a metric on the space of probability distributions supported on $\mathbb{R}^d$, which takes a "ground metric" (in this case, the Euclidean distances) and defines a metric over the space of distributions supported on the ground metric. The resulting notion of similarity or dissimilarity is then used to formulate problems on approximating or learning a distribution supported in $\mathbb{R}^d$.

1. If the distributions μ and ν are supported on the set of points $x_1,\dots,x_n$ and $y_1,\dots,y_m$, respectively, then we can think of γ as being specified by an $n\times m$ matrix of non-negative real numbers. The constraint "γ is a coupling of μ and ν" means that the i-th row of the $n\times m$ matrix γ sums to μ_i and the j-th column of the $n\times m$ matrix sums to ν_j.

It is no surprise that the optimal transport has become ubiquitous in machine learning. We refer the reader to the monograph Peyré and Cuturi (2019) for a comprehensive overview, but a few notable examples include Kusner et al. (2015); Courty et al. (2016); Arjovsky et al. (2017). As argued in Peyré and Cuturi (2019), the most recent progress on optimal transport for machine learning has been due to new formulations and approximation algorithms which can scale to larger problem instances. Specifically, there has been a focus on the so-called entropy-regularized optimal transport, also known as "Sinkhorn distances," and (accurate) approximation algorithms which run in quadratic time (in the original representation for Euclidean inputs) Cuturi (2013). The goal of this work is to further explore such optimal transport questions from the computational perspective, where we will seek much faster *sub-quadratic* algorithms for computing optimal transport distances.

As we explain next, the algorithmic landscape for optimal transport remains very much unknown. On the one hand, the algorithms community has devoted a significant effort (Charikar (2002); Indyk and Thaper (2003); Indyk (2004); Andoni et al. (2008, 2009); Sharathkumar and Agarwal (2012); Agarwal and Sharathkumar (2014); Andoni et al. (2014); Bačkurs and Indyk (2014); Andoni et al. (2015); Khesin et al. (2019); Backurs et al. (2020); Chen et al. (2022b); Agarwal et al. (2022)) to developing fast algorithms for approximating EMD. We expand on these shortly, but, at a high level, all approaches rely on efficient spanner constructions or approximate nearest neighbor data structures. These algorithm are fast (approaching linear time), but they run into a serious approximation bottleneck. For high-dimensional Euclidean spaces, almost-linear time algorithms incur large constant-factor approximations, making these approaches undesirable.[2] Instantiating these techniques for accurate, $(1 \pm \varepsilon)$-approximations degrades the algorithmic performance to essentially quadratic time.[3]

On the other hand, algorithms for the entropy-regularized optimal transport do achieve accurate additive $\pm\varepsilon r$ approximations for datasets of diameter r, but have running times which are quadratic in the original representation of the input. In particular, the input distributions are specified by the vectors in their support and the probabilities with which they are sampled. However, the first step of the algorithm involves explicitly materializing the distance matrix encoding all pairwise distances between the vectors. As the support of these distributions grows, this first step is already a major hurdle. While there have been approaches to avoid materializing the entire matrix Bonneel et al. (2015); Altschuler et al. (2019); Paty and Cuturi (2019), these methods consider a projection of the points onto a low-dimensional space and the resulting optimization costs (of the low-dimensional EMD or Sinkhorn distances) cannot be related back to the original distribution without a significant loss in approximation.

This work seeks to explore the best of both worlds from the algorithmic perspective. We will give a new class of objectives for optimal transport problems which also provide metric spaces for probability distributions of high-dimensional Euclidean spaces (like the Earth Mover's distance and Sinkhorn distances). The main benefit is that (i) these metrics smoothly perturb the Earth Mover's distance, (ii) admit efficient algorithms with running times which are significantly sub-quadratic (like the Earth Mover's distance), and (iii) give accurate $\pm\varepsilon r$-approximations for distri-

2. For example, a 2-approximation which is already oftentimes too big, incur a polynomial overhead of $(n+m)^{1/7}$ Andoni and Razenshteyn (2015).

3. An algorithm for $\mathsf{EMD}(\mu, \nu)$ (or any problem whose output is a positive real number) which achieves approximation factor $c > 1$ is an algorithm which outputs a number which is larger than $\mathsf{EMD}(\mu, \nu)$ and is at most $c \cdot \mathsf{EMD}(\mu, \nu)$ with high probability. These are *multiplicative* approximations, and we will also refer to *additive* εr-approximations which outputs a quantity which is up to $\pm\varepsilon r$ from a desired quantity.

butions whose supports have diameter at most r (like the Sinkhorn distances). The key will be to never explicitly compute the quadratic-size distance matrix. Instead, we show how one may implement a Sinkhorn-like update procedure using recent algorithms for the problem of kernel density estimation.

1.1. Related Work: The Spanner Approach for EMD

The Earth Mover's Distance can be naturally cast as an uncapacitated minimum cost flow problem. The reduction is straight-forward. One may consider the (weighted) complete bipartite graph $G = (U, V, E = U \times V, w)$ where each vertex of U is a vector in the support of μ and each vertex in V is a vector in the support of ν and the weights (or cost) w of an edge $e = (i, j)$ is $w(e) = \|x_i - y_j\|_2$. The distributions may then be written as vectors $\mu \in \mathbb{R}^n$ and $\nu \in \mathbb{R}^m$ which encode the "supply" and "demand", and the Earth Mover's Distance is the minimum cost flow on G according to the supply/demands μ and ν with costs w (there is no need for capacities in this reduction). Over the years, graph algorithms have become incredibly efficient, so applying graph-based min-cost flow solvers with the reduction above gives exact algorithms for EMD running in time $(nm)^{1+o(1)}$.[4]

The above approach paves the way for faster approximation algorithms by using graph spanners. For any $c > 1$, one seeks a graph H with substantially fewer edges on the vertex set U and V. The desired property is that for any $i \in U$ and $j \in V$, the total length of the shortest path between $i \in U$ and $j \in V$ along edges of H should be a factor of c-approximation to the distance between the underlying vectors x_i and y_j. Running the min-cost flow algorithms on H is faster (since there are fewer edges), and give a c-approximation for EMD. While sparse spanners for Euclidean distances do exist, as the approximation c approaches $1 + \varepsilon$, the size of these spanners become mn.

Instead, the focus has been on obtaining sparse spanners for (large) constant factor approximations. For example, for any $c > 1$, Har-Peled et al. (2013) gives c-spanners of size $(n + m)^{1+1/c^2}$ for Euclidean spaces (ℓ_2) in time $\tilde{O}((n + m)^{1+1/c^2})$ (which is fast when we allow a large c). The other approach, taken in Agarwal and Sharathkumar (2014), does not explicitly use a spanner, but uses an approximate nearest neighbor search data structure. The resulting time and approximation depends on the time and approximation for nearest neighbor search, but similarly to before, the approximation is large when the algorithms are fast.

1.2. Related Work: Sinkhorn Distances

The algorithm which is widely used for computing an optimal transport is the Sinkhorn algorithm for entropy-regularized optimal transport Cuturi (2013); Altschuler et al. (2017) (see also, the recent work Kiem et al. (2020); Le et al. (2021)). Given two distributions μ and ν supported on vectors in $\mathbb{R}^d$, the entropy-regularized optimal transport introduces an entropic regularization term to the the Earth Mover's distance. Specifically, for any $\eta \geq 0$, it optimizes

$$\mathsf{SNK}_\eta(\mu, \nu) = \min \left\{ \mathop{\mathbf{E}}_{(\boldsymbol{x},\boldsymbol{y})\sim\gamma} [\|\boldsymbol{x} - \boldsymbol{y}\|_2] - \eta H(\gamma) : \gamma \text{ is a coupling of } \mu \text{ and } \nu \right\}.$$

4. The relevant citation for a fast min-cost flow algorithm is the recent breakthrough of Chen et al. (2022a). These give exact algorithms for graphs whose time in almost-linear in the number of edges, nm of the graph. The other relevant citation is Sherman (2017), giving algorithms for $1 + \varepsilon$-approximation to uncapacitated min-cost flow in the same amount of time, which suffices for EMD.

The main benefit is that the algorithm for optimizing $\mathsf{SNK}_\eta(\mu, \nu)$ performs extremely well. The algorithm used is iterative, and uses $\mathrm{poly}(1/(\eta\varepsilon))$ iterations to output a solution which is an additive $\pm\varepsilon r$-approximation (where r is the maximum distance between any pair of points in the support of μ and ν). Oftentimes, the maximum distance r is not too large (for example, it is at most 2 on the unit sphere), making the algorithm very desirable in practice. However, the main downside is that the algorithm explicitly computes the nm-distance matrix of pairwise distances of vectors in the support of μ and ν. Indeed, the algorithm does not use the fact that distances are Euclidean and generalizes to non-Euclidean metrics. The main downside is that, for distributions on Euclidean spaces, the description of the input (of size $O(d(n+m))$) is blown up to a quadratic nm-size distance matrix, which can be a major bottleneck in the computation if n and m are very large. Finally, it is important to note that, we currently do not know whether the original Earth Mover's distance admits a similar εr-approximation for bounded datasets in time substantially smaller than nm.

1.3. Our Contributions

This paper addresses the following questions:

1. Do there exists optimal transport metrics which do admit good approximations in significantly sub-quadratic time? In particular, can we match the approximation guarantees from Sinkhorn distances with the algorithmic techniques from the Earth Mover's distance?

2. Can one combine techniques, like locality-sensitive hashing (LSH) and embeddings, with the alternating updates procedure in Sinkhorn's algorithm even though approximations incurred from using LSH and embeddings tend to incur large constant factors?

Our main contribution is introducing a class of objective functions for optimal transport computations. The new objectives $\mathcal{R}_\rho(\mu, \nu)$ are parametrized by $\rho \geq 1$ and provide metric spaces over discrete distributions in $\mathbb{R}^d$. As ρ approaches 1, $\mathcal{R}_\rho(\mu, \nu)$ approaches $\mathsf{EMD}(\mu, \nu)$, but enjoys favorable computational properties. In particular, we will show that $\mathcal{R}_\rho(\mu, \nu)$ may be approximated up to additive εr-error for datasets of diameter at most r in time which is near-linear (for small ρ close to 1). We view ρ as introducing a new "knob" for the Earth Mover's distance: as $\rho \to 1$, the metrics $\mathcal{R}_\rho(\cdot, \cdot)$ approach $\mathsf{EMD}(\cdot, \cdot)$; however, for ρ close to (but not too close to) 1, very fast algorithms with accurate approximations are possible. Thus, our new algorithm gives a positive answer to Question 1. Namely, if one is willing to change the problem slightly, one can achieve the approximation guarantees of Sinkhorn distances with the running times like the Earth Mover's distance.

While Question 2 is inherently vague, such techniques are known in a related algorithmic context. One of our main conceptual contributions is drawing a connection to *kernel density estimation* Charikar and Siminelakis (2017); Backurs et al. (2018); Siminelakis et al. (2019); Charikar et al. (2020); Backurs et al. (2021); Bakshi et al. (2022). The algorithms developed in that context use locality-sensitive hashing and embeddings, but are still able to output $(1 \pm \varepsilon)$-approximations. In particular, a key feature of those works is that the distortion incurred by locality-sensitive hashing and embeddings factors into the running time of the algorithm and not the final approximation. In summary, our main conceptual contributions may be summarized as follows:

- There exists a class of optimal transport metrics parametrized by ρ which smoothly perturb the Earth Mover's distance (approaching EMD as $\rho \to 1$).

- For a small setting of $\rho > 1$, these problems can be optimized in significantly sub-quadratic time to arbitrarily accurate additive approximations for bounded datasets.

We believe the new problem formulation and the ideas behind the algorithm will lead to improvements in practical algorithms for optimal transport metrics. We emphasize that there are no algorithmic approaches that achieve $(1 \pm \varepsilon)$-approximations or εr-additive approximations for either EMD nor SNK in time $n^{1.99}$. In addition, there is some reason to believe that this may be impossible for EMD Rohatgi (2019). By changing the problem and allowing a small additive error, we avoid the large constant factors. We also suggest looking at Section 4 of Backurs et al. (2020), who group algorithms by their running times; the new techniques achieve the accurate approximations of the "quadratic time" algorithms, even though they run much faster (at least in theory).

Outline. The next section gives the new objective $\mathcal{R}_\rho(\mu, \nu)$ and states our main Theorem 1. We will overview the components of the proof in the next section. Then, we give a description of the main algorithm while assuming algorithms for estimating the gradients and the penalty term.

2. The Definition of ℓ_ρ-Optimal Transports

For any dimension $d \in \mathbb{N}$, let μ and ν denote two discrete distributions supported on n and m point masses in $\mathbb{R}^d$, respectively. More specifically, μ is specified by n points $x_1, \dots, x_n \in \mathbb{R}^d$ and corresponding weights $\mu_1, \dots, \mu_n \in \mathbb{R}_{>0}$ where $\sum_{i=1}^n \mu_i = 1$, and ν is specified by m points $y_1, \dots, y_m \in \mathbb{R}^d$ with the corresponding weights $\nu_1, \dots, \nu_m \in \mathbb{R}_{>0}$ with $\sum_{i=1}^m \nu_i = 1$ (note that we can always assume that μ_i and ν_j are strictly positive by a linear-time scan which can remove points of weight-0). One ought to think of $d = \omega(\log n)$, so we seek algorithms which overcome the "curse of dimensionality" and do not have running times which scale exponentially in d.

For any parameter $\rho > 1$, we seek to optimize the following objective, which will specify a metric space over probability distributions which relax the optimal transport problem (Lemma 5 in Appendix A):

$$\mathcal{R}_\rho(\mu, \nu) = \min \left\{ \left(\mathop{\mathbf{E}}_{\substack{\boldsymbol{i} \sim \mu \\ \boldsymbol{j} \sim \nu}} \left[\left(\frac{\gamma_{\boldsymbol{ij}}}{\mu_{\boldsymbol{i}} \nu_{\boldsymbol{j}}} \cdot \|x_{\boldsymbol{i}} - y_{\boldsymbol{j}}\|_2 \right)^\rho \right] \right)^{1/\rho} : \gamma \text{ is a coupling of } \mu \text{ and } \nu \right\}. \tag{2}$$

In words, for any coupling γ between the distributions μ and ν, one may associate an nm-dimensional vector encoding the costs associated with each point-mass. Each point x_i from μ and y_j from ν, the coupling γ transports γ_{ij} "mass" from x_i to y_j and pays a function of the distance between x_i and y_i times $\gamma_{ij}/(\mu_i \nu_j)$. In $\mathcal{R}_\rho(\mu, \nu)$, we optimize the normalized ℓ_ρ-norm of the cost vector. Notice that, when $\rho = 1$, $\mathcal{R}_\rho(\mu, \nu)$ is the Earth Mover's distance distance between μ and ν. As we vary $\rho \geq 1$, one may relate the ℓ_ρ- and ℓ_1-norm, implying

$$\mathsf{EMD}(\mu, \nu) \leq \mathcal{R}_\rho(\mu, \nu) \leq \sup_{i,j} \left| \frac{1}{\mu_i \nu_j} \right|^{(\rho-1)/\rho} \mathsf{EMD}(\mu, \nu).$$

When $\rho > 1$, we will obtain a sequence of (as we will see) computationally easier metric spaces which approach $\mathsf{EMD}(\mu, \nu)$. The key is that performing this modification will allow for significantly faster algorithms in terms of n and m (the number of points), while having a dependence on ρ which will be $2^{O(\rho/(\rho-1))}$.

We view $\rho > 1$ as a desired computational "knob," which allows one to tradeoff the running time of an algorithm and the metric's relation to EMD. Note that, in a c-approximation algorithm for EMD, c also trades-off faster/slower running times for looser/tighter relations to EMD. The difference, however, is that for any $\rho > 1$, $\mathcal{R}_\rho(\cdot, \cdot)$ is still a metric space over probability distributions (and the same cannot be said of a 3-approximation to EMD). The specific choice of metric space (EMD, Wasserstein-p, or SNK_η) is oftentimes flexible, so long as it captures the desired notion of similarity/dissimilarity of distributions. The hope is that for moderate values of ρ, $\mathcal{R}_\rho(\mu, \nu)$ suffices for downstream applications, and captures the desired properties of an optimal-transport γ.

From a more technical perspective, (2) encourages couplings γ whose contribution to the cost vector is "spread", so that the ℓ_ρ-norm will be small. The main advantage is that, using a connection to recent work on kernel density estimation in high-dimensions Backurs et al. (2018) and scaling approaches to entropy regularized optimal transport Cuturi (2013); Altschuler et al. (2017), we give very efficient (and simple) algorithms for approximating $\mathcal{R}_\rho(\mu, \nu)$.

Notation for Running Time Bounds. We will use the following notation in order to describe the running time bounds. The focus is on improving on the dependence on n and m when estimating optimal transports, so we use the notation $\mathrm{poly}^*(f)$ to denote a fixed polynomial function of f, and which hides poly-logarithmic factors n, m, δ (the failure probability), ε (the accuracy) and r (the radius of the dataset). In addition, since we will incur a polynomial dependence on ε, we will automatically apply the Johnson-Lindenstrauss lemma and assume that $d = O(\log(nm)/\varepsilon^2)$.

Theorem 1 *There exists a randomized algorithm with the following guarantees. The algorithm receives as input*

- *Two sets of points $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ in $\mathbb{R}^d$ where the maximum pairwise distance between points $\sup_{i,j} \|x_i - y_j\|_2 \le r$.*
- *Two vectors $\mu \in \mathbb{R}^n_{\ge 0}$ and $\nu \in \mathbb{R}^m_{\ge 0}$ whose coordinates sum to 1 and encode the distributions over $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_n\}$, respectively.*
- *An accuracy parameter $\varepsilon > 0$, a failure probability $\delta > 0$, and a parameter $\rho \in [1, 2]$.*

The algorithm runs in time $(n + m) \cdot \mathrm{poly}^*((nm)^{(\rho-1)/\rho} \cdot 2^{\rho/(\rho-1)}/\varepsilon)$, and outputs an estimate $\widehat{\boldsymbol{\eta}} > 0$ which satisfies

$$|\widehat{\boldsymbol{\eta}} - \mathcal{R}_\rho(\mu, \nu)| \le \varepsilon \cdot r$$

with probability at least $1 - \delta$.

The main advantage of Theorem 1 is that it does not pay the quadratic nm-factor in the running time and at the same time obtains accurate approximations. In particular, suppose we consider a setting of ρ which is $\rho = 1 + 1/\sqrt{\log(nm)}$, then the corresponding running time of Theorem 1 to approximate $\mathcal{R}_\rho(\mu, \nu)$ up to an additive $\pm\varepsilon r$ becomes

$$(n + m)^{1+o(1)} \cdot \mathrm{poly}(1/\varepsilon).$$

Generally, as ρ becomes close to 1, the metric $\mathcal{R}_\rho(\cdot, \cdot)$ approaches $\mathsf{EMD}(\cdot, \cdot)$ and the dependence on n and m becomes better, since $(n + m) \cdot (nm)^{O((\rho-1)/\rho)}$. However, one does not want to set ρ to be too close to 1, since the factor of $2^{O(\rho/(\rho-1))}$ may begin to dominate.

Remark 2 (Challenges when $\rho \to 1$) *In order to use $\mathcal{R}_\rho(\cdot,\cdot)$ to approximate $\mathsf{EMD}(\cdot,\cdot)$ up to $(1+\varepsilon)$-factor, one would need to set ρ to roughly $1 + O(\varepsilon/\log(nm))$; however, this approach runs into a technical challenge. There is a concrete sense in which the parameter $\rho \geq 1$ adds a certain "smoothness" which is not present in EMD. At a very high level, we show that an additive approximation of $\mathcal{R}_\rho$ reduces to queries for "smooth" kernel density evaluation Backurs et al. (2018) which suffer an exponential dependence on $\rho/(\rho-1)$. With $\rho = 1 + O(\varepsilon/\log(nm))$, this dependence would become $(nm)^{O(1/\varepsilon)}$—worse than the $(nm)^{1+o(1)}$ time required from prior work.*

2.1. Proof of Theorem 1 Overview

We overview the major components of the proof of Theorem 1. While (relatively minor) technical challenges arise when fleshing out the details, the structure and algorithm proceed with the following plan.

The Duals of $\mathsf{EMD}(\mu,\nu)$ and $\mathcal{R}_\rho(\mu,\nu)^\rho$. The challenge in optimizing $\mathcal{R}_\rho(\mu,\nu)^\rho$ (which also appears in $\mathsf{EMD}(\mu,\nu)$) is that an algorithm cannot even write down the explicit description of the optimization, nor can it explicitly maintain a coupling γ, since this requires $\Omega(nm)$ values. On the other hand, both $\mathsf{EMD}(\mu,\nu)$ and $\mathcal{R}_\rho(\mu,\nu)^\rho$ only have $n+m$ equality constraints, so the duals are maximization problems over $n+m$ variables (one for each constraint). The approach will be to show that, using data structures for kernel density estimation, we can implicitly maximize the dual of $\mathcal{R}_\rho(\mu,\nu)^\rho$ while only maintaining the $n+m$ dual variables.

To see the connection, we first write down the dual for $\mathsf{EMD}(\mu,\nu)$, which has $n+m$ variables $\alpha_1,\dots,\alpha_n$ and $\beta_1,\dots,\beta_m$ and asks to maximize

$$\mathsf{EMD}(\mu,\nu) = \max_{\substack{\alpha\in\mathbb{R}^n\\ \beta\in\mathbb{R}^m}} \left\{ \sum_{i=1}^n \mu_i\alpha_i - \sum_{j=1}^m \nu_j\beta_j : \forall (i,j)\in[n]\times[m], \alpha_i - \beta_j \leq \|x_i - y_j\|_2 \right\}. \tag{3}$$

For $\rho > 1$, the Hölder conjugate $s > 1$, is the number satisfying $1/\rho + 1/s = 1$. The dual for $\mathcal{R}_\rho(\mu,\nu)^\rho$ is the following unconstrained maximization problem on $n+m$ variables $\alpha_1,\dots,\alpha_n$ and $\beta_1,\dots,\beta_m$,

$$\mathcal{R}_\rho(\mu,\nu)^\rho = \max_{\substack{\alpha\in\mathbb{R}^n\\ \beta\in\mathbb{R}^m}} \left\{ \sum_{i=1}^n \mu_i\alpha_i - \sum_{j=1}^m \nu_j\beta_j - \frac{1}{s}\left(1-\frac{1}{s}\right)^{s-1} \sum_{i=1}^n\sum_{j=1}^m \mu_i\nu_j \left(\frac{(\alpha_i-\beta_j)^+}{\|x_i-y_j\|_2}\right)^s \right\}, \tag{4}$$

where we consider $\frac{0}{0}=0$, and $(\alpha_i - \beta_j)^+$ is $\alpha_i - \beta_j$ if positive and 0 otherwise. Note the difference: in (3), there are nm hard constraints which enforce $(\alpha_i-\beta_j)^+/\|x_i-y_j\|_2 \leq 1$ for every $i \neq j$. In (4), the nm constraints are relaxed. The optimization is allowed to set $\alpha_i - \beta_j$ larger than $\|x_i - y_j\|_2$, but pays a penalty in the objective proportional to $((\alpha_i-\beta_j)^+/\|x_i-y_j\|_2)^s$. As ρ gets closer to 1, the Hölder conjugate s becomes larger, and the penalty becomes more pronounced. For simplicity in the notation, we will write

$$g(\alpha,\beta) \stackrel{\text{def}}{=} \sum_{i=1}^n \mu_i\alpha_i - \sum_{j=1}^m \nu_j\beta_j - \frac{1}{s}\left(1-\frac{1}{s}\right)^{s-1} \sum_{i=1}^n\sum_{j=1}^m \mu_i\nu_j \left(\frac{(\alpha_i-\beta_j)^+}{\|x_i-y_j\|_2}\right)^s.$$

Partial Derivatives via Kernel Density Estimation Since (4) is a concave maximization problem, a simple approach is to simulate a gradient ascent algorithm on the dual variables $\alpha_1, \dots, \alpha_n$ and $\beta_1, \dots, \beta_m$, where we update in the direction of the partial derivatives. The partial derivatives with respect to α_i and β_j are given by

$$\frac{\partial g}{\partial \alpha_i} = \mu_i \left(1 - \left(1 - \frac{1}{s}\right)^{s-1} \sum_{j=1}^{m} \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s}\right) \tag{5}$$

$$\frac{\partial g}{\partial \beta_j} = -\nu_j \left(1 - \left(1 - \frac{1}{s}\right)^{s-1} \sum_{i=1}^{n} \mu_i \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s}\right). \tag{6}$$

Importantly, the partial derivatives depend on μ_i and ν_j and a weighted sum of $1/\|x_i - y_j\|_2^s$. First, note that we receive μ and ν as input, so μ_i and ν_j are $n+m$ constants throughout the execution. The weighted sums are the more challenging parts, and for these we use the kernel density estimation data structures. We interpret $\mathsf{K}(x_i, y_j) = 1/\|x_i - y_j\|_2^s$ as a "smooth" kernel, similar to the Student-t Kernel studied in Backurs et al. (2018). These smooth kernels decay polynomially as a function of the distance $\|\cdot\|_2$ and admit very efficient data structures. Specializing the results of Backurs et al. (2018) for K, they give data structures which preprocess a set of points P and can support $(1 \pm \varepsilon)$-approximate kernel evaluation queries of the form $\sum_{x \in P} \mathsf{K}(x, y)$ for any $y \in \mathbb{R}^d$. The query complexity is $\mathrm{poly}^*(2^s/\varepsilon)$ and s becomes $\rho/(\rho - 1)$. In order to use these for Theorem 1, we incorporate the weights $((\alpha_i - \beta_j)^+)^{s-1}$ by augmenting those data structures in Section C (we overview the augmentations shortly). Once this is done, the algorithm can initialize $\alpha \in \mathbb{R}^n$ and $\beta \in \mathbb{R}^m$ to 0 and effectively update α and β in the directions of the partial derivatives in order to increase the objective function.

The only remaining challenge is setting the step size of the update, and ensuring that the function is smooth enough. Note that because of the non-linear penalty term, there is no global Lipschitz constant, but we will argue that our optimization always remains within a smooth enough region if the step size is set appropriately. We do this final argument by applying a simple preprocessing step. The preprocessing will guarantee that the distance between any x_i and y_j is always between εr and r (which changes $\mathcal{R}_\rho(\mu, \nu)$ by at most εr), and that every non-zero element of the support of μ and ν is sampled with at least some probability. This means that an update which changes some α or β does not change the penalty term significantly (because the fact that the distance $\|x_i - y_j\|_2$ in the denominator is at least εr ensures the penalty does not blow up).

Augmenting Kernel Density Estimates to Incorporate Weights For $s > 1$, we want to maintain a set of points $P = \{x_1, \dots, x_n\}$ in $\mathbb{R}^d$, where each point is associated with a weight $\alpha_1, \dots, \alpha_n \in \mathbb{R}$ and a parameter $\mu_1, \dots, \mu_i$ which are between $1/\mathrm{poly}(n)$ and 1. A query is specified by another vector $y \in \mathbb{R}^d$ and its weight β, and the task is to output

$$\sum_{i=1}^{n} \mu_i \cdot ((\alpha_i - \beta)^+)^{s-1} \cdot \mathsf{K}(x_i, y), \tag{7}$$

where $\mathsf{K}(x_i, y) = 1/\|x_i - y\|_2^s$. We will augment the data structures from Backurs et al. (2018) as follows. First, partition P into $O(\log n/\varepsilon)$ ranges which partition $[1/\mathrm{poly}(n), 1]$ according to powers of $1 + \varepsilon$ so as to assume that μ_j is the same within each range. Note that we know the weights $\mu_1, \dots, \mu_n$ during the preprocessing, so that we may perform this partition; however, since we do not know β during the preprocessing, we cannot similarly partition according to the value of $(\alpha_i - \beta)^s$.

Instead, we will proceed with the following. For each range j, the resulting set P_j is stored sorted in a binary tree according to the weights α, and let $\alpha_{\max}$ be the largest weight. Each internal node holds a data structure of Backurs et al. (2018) maintaining points in its subtree. When a query $(y, \beta) \in \mathbb{R}^d \times \mathbb{R}$ comes, one may perform the following:

1. Let $\boldsymbol{\xi}$ be uniformly drawn from the interval $[0, (\alpha_{\max} - \beta)^{s-1}]$.
2. Find the value $\beta + \boldsymbol{\xi}^{1/(s-1)}$ in the binary tree, and we consider the $k = O(\log n)$ nodes which partition the interval $[\beta + \boldsymbol{\xi}^{1/(s-1)}, \alpha_{\max}]$.
3. Query all k kernel evaluation data structures stored at those nodes. If $\widehat{\boldsymbol{\eta}}_1, \dots, \widehat{\boldsymbol{\eta}}_k$ are the estimates output by the k data structures with y, output $(\alpha_{\max} - \beta)^{s-1} \sum_{\ell=1}^{k} \widehat{\boldsymbol{\eta}}_\ell$.

The main observation is that the sampling automatically incorporates weights. For example, suppose the data structures of Backurs et al. (2018) were exact, then our estimate is an unbiased estimator of (7):

$$\mathop{\mathbf{E}}_{\boldsymbol{\xi}}\left[(\alpha_{\max} - \beta)^{s-1} \sum_{\ell=1}^{k} \widehat{\boldsymbol{\eta}}_\ell\right] = \sum_{i=1}^{m} (\alpha_{\max} - \beta)^{s-1} \cdot \mathop{\mathbf{Pr}}_{\boldsymbol{\xi}}\left[\alpha_i \geq \beta + \boldsymbol{\xi}^{1/(s-1)}\right] \cdot \mathsf{K}(x_i, y),$$

and the probability that $\alpha_i \geq \beta + \boldsymbol{\xi}^{1/(s-1)}$ is exactly $((\alpha_i - \beta)^+)^{s-1}/(\alpha_{\max} - \beta)^{s-1}$. The variance of the above estimation is too large (which occurs because $\alpha_{\max} \gg \beta$), so we make the following minor modification. We partition the interval $[\beta, \alpha_{\max}]$ into poly-logarithmic, geometrically increasing groups, and perform the above process for each group. This is then enough to bound the variance.

3. A Gradient Ascent Algorithm

3.1. A Simple Preprocessing

Before we give the description of the algorithm, we will run a simple preprocessing step which simplifies our input. We will think of μ and ν as the distribution over $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$, respectively. For small parameters $\sigma, \sigma_\mu, \sigma_\nu > 0$, we will define the distributions μ' and ν' in the following way:

- First, we consider the points $x'_1, \dots, x'_n$ and $y'_1, \dots, y'_m$ in $\mathbb{R}^{d+1}$ where we append a coordinate and we let $x'_i = (x_i, \sigma r)$ and $y'_j = (y_j, 0)$. This way, we guarantee that for every $i \in [n]$ and $j \in [m]$, we satisfy $\sigma r \leq \|x'_i - y'_j\|_2 \leq r\sqrt{1 + \sigma^2}$ (where the upper bound follows from the fact $\|x_i - y_j\|_2 \leq r$).
- We define the sets $L_\mu \subset [n]$ and $L_\nu \subset [m]$ for the indices of μ and ν which have low probability, i.e., $L_\mu = \{i \in [n] : \mu_i < \sigma_\mu/n\}$ and $L_\nu = \{j \in [m] : \nu_j < \sigma_\nu/m\}$. We denote $\zeta_\mu = \sum_{i \in L_\mu} \mu_i \leq \sigma_\mu$ and $\zeta_\nu = \sum_{j \in L_\nu} \nu_j \leq \sigma_\nu$. The distribution μ' is supported on the points $x'_1, \dots, x'_n$, and ν' is supported on the points $y'_1, \dots, y'_n$ given by

$$\mu'_i = \begin{cases} 0 & i \in L_\mu \\ \mu_i/(1 - \zeta_\mu) & i \in [n] \setminus L_\mu \end{cases} \qquad \text{and} \qquad \nu'_j = \begin{cases} 0 & j \in L_\nu \\ \nu_j/(1 - \zeta_\nu) & j \in [m] \setminus L_\nu \end{cases}.$$

The above transformations has the benefit that we now have a lower bound on the minimum distance between any point from the support of μ' and any point from the support of ν', while only increasing the maximum distance by at most a factor of $\sqrt{1+\sigma^2}$. Furthermore, the distributions μ' and ν' have all elements of their support with probability at least σ_μ/n and σ_ν/m, respectively, since we have removed the low-probability items. Thus, the algorithm below will apply the above perturbation, and we may assume throughout the execution the corresponding properties of μ and ν. Note that as long as we ensure the parameter

$$\left(n^{\rho-1}\cdot\frac{\sigma}{\sigma_\mu^{\rho-1}}+\sigma_\mu\right)^{1/\rho}\leq\varepsilon \qquad \text{and} \qquad \sigma_\nu^{1/\rho}\leq\varepsilon,$$

then by the triangle inequality, we will have $\mathcal{R}_\rho(\mu',\nu')$ is up to an additive $2\varepsilon r$, the same as $\mathcal{R}_\rho(\mu,\nu)$. In particular, we can let σ_ν be ε^ρ and $\sigma_\mu=\varepsilon^\rho/n$ and $\sigma=\varepsilon^\rho$.

3.2. Description of the Algorithm

We will assume hence-forth that our input distributions μ and ν, whose support is $\{x_1,\dots,x_n\}$ and $\{y_1,\dots,y_n\}$ satisfy:

- Every $i\in[n]$ and $j\in[m]$, the distance $\|x_i-y_j\|_2$ is always between σr and r (for a small parameter $\sigma>0$, we have $r\sqrt{1+\sigma^2}\leq 2r$ so, in order to simplify the notation, one may think of σ as being decreased by a factor of 2).
- The distributions μ and ν have a "granularity" property, so that every $i\in[n]$ for which μ_i is non-zero is at least σ_μ/n, and every $j\in[m]$ for which ν_j is non-zero is at least σ_ν/m. This will allow us to upper bound $1/(\mu_i\nu_j)\leq mn/(\sigma_\mu\sigma_\nu)$.

The algorithm will maintain a setting of the dual variables $(\alpha_t,\beta_t)\in\mathbb{R}^{n+m}$ which it will update in each iteration, and it will seek to maximize

$$g(\alpha,\beta)=\sum_{i=1}^n\mu_i\alpha_i-\sum_{j=1}^m\nu_j\beta_j-C_s\sum_{i=1}^n\sum_{j=1}^m\mu_i\nu_j\left(\frac{(\alpha_i-\beta_j)^+}{\|x_i-y_j\|_2}\right)^s.$$

In the description of the algorithm below, we will assume access to three sub-routines `Est-Alpha`, `Est-Beta`, and `Est-Penalty` which we specify later (see Subsection B.1 for a description of the guarantees). At a high level, the sub-routines `Est-Alpha` will help us get an approximation of the gradient $\nabla g(\alpha_t,\beta_t)$ along directions in α, and `Est-Beta` will help us get an approximation of the gradient $\nabla g(\alpha_t,\beta_t)$ along directions in β. The sub-routine `Est-Penalty` will come in at the end, since we will need to estimate the "penalty" term in order to output an approximation to $g(\alpha_t,\beta_t)$. We will instantiate the algorithm with the following parameters:

- **Accuracy of Terminating Condition**: we denote this parameter $\varepsilon_2>0$, which will be set to $c_0\cdot\varepsilon\cdot(\sigma_\mu\sigma_\nu/(mn))^{(\rho-1)/\rho}$ for a small enough constant $c_0>0$. This parameter will dictate when our algorithm has found a dual solution which is close enough to the optimal one.
- **Accuracy for Estimation**: There are two parameters which specify the accuracy needed in the estimations `Est-Alpha` and `Est-Beta`. We let $\varepsilon_1>0$ denote the multiplicative error bound which we will tolerate, set to $c_1\varepsilon_2/s$ for a small enough constant c_1, and τ which will

be an additive error bound will may be interpreted as a granularity condition on the weights α, β. It will suffice to set $\tau = c_2\varepsilon_2$, but the final dependence on τ will be poly-logarithmic in $1/\tau$, the notation $\mathrm{poly}^*(\cdot)$ will suppress it.

- **Step Size of Gradient Ascent**: The parameter $\lambda \geq 0$ will denote the step size of our gradient ascent algorithm. We set $\lambda = c_3\varepsilon_2 \cdot (\sigma/s)^2 \cdot r^\rho$, for a small constant $c_3 > 0$.

We will also consider a small enough parameter $\delta > 0$ which will denote the failure probabilities of our estimation algorithms. The final dependence on δ is only poly-logarithmic, so it will suffice to set δ to be a small enough polynomial factor of all parameters of the algorithm (i.e., $n, m, 1/\varepsilon, 1/\sigma, s$) such that all executions of `Est-Alpha`, `Est-Beta`, and `Est-Penalty` succeed with high probability. For simplicity in the notation, we will drop δ from the notation, and assume that all executions of our (randomized) sub-routines succeed.

3.3. Analysis of the Algorithm

We now show that the algorithm presented at the top of Subsection 3.2 finds an approximately optimal maximizer of g, assuming the lemmas on the guarantees of the subroutines of Subsection B.1. In particular, this section shows two lemmas. The first lemma shows that if the algorithm does not perform an update, then the value (α_t, β_t) that the algorithm holds is an approximate maximizer of g, this will then imply, from Lemma 12 that we can output an estimate of $g(\alpha_t, \beta_t)$. The second lemma says that if the algorithm performs an update, then the value of the objective function g increases by $\Omega(\varepsilon_2 \cdot \lambda)$. In particular, since the objective function is a maximization problem which is always at most r^ρ, this implies that the algorithm performs at most $O(r^\rho/(\varepsilon_2 \cdot \lambda))$ updates before it must terminate. In addition, when it terminates, Lemma 7 implies that the quantity $\boldsymbol{\omega}$ output by `Est-Penalty` is at most $O(r^\rho)$. This means that in the final estimate, for $\boldsymbol{\omega}$, it suffices to set τ to εr^ρ.

Lemma 3 (Termination Condition) *Suppose $(\alpha_t, \beta_t) \in \mathbb{R}^{n+m}$ satisfies $g(\alpha_t, \beta_t) \geq 0$ and the algorithms* `Est-Alpha` *and* `Est-Beta` *produce a sequence of quantities $\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_n$ and $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_m$ which satisfy the guarantees of Lemma 10 and Lemma 11, and*

$$\sum_{i=1}^{n} \mu_i \left|\boldsymbol{\eta}_i - 1\right| \leq \varepsilon_2 \qquad \textit{and} \qquad \sum_{j=1}^{m} \nu_j \left|\boldsymbol{\xi}_j - 1\right| \leq \varepsilon_2.$$

Then, letting (α^*, β^*) be the maximizer of $g(\alpha^*, \beta^*)$, we have

$$g(\alpha^*, \beta^*) - g(\alpha_t, \beta_t) \leq O(1) \cdot \left(\frac{nm}{\sigma_\mu \sigma_\nu}\right)^{(\rho-1)/\rho} \cdot r^\rho \cdot \left(\varepsilon_2 + \tau + \frac{\varepsilon_1 s}{\sigma}\right)$$

Lemma 4 (Updates Increase Objective) *Suppose $(\alpha_t, \beta_t) \in \mathbb{R}^{n+m}$ satisfies $g(\alpha_t, \beta_t) \geq 0$ and $(\alpha_{t+1}, \beta_t) \in \mathbb{R}^{n+m}$ is a vector, for which the algorithms* `Est-Alpha` *produce the sequence of outputs $\boldsymbol{\eta}_1, \dots, \boldsymbol{\eta}_n$ which satisfy the guarantees of Lemma 10 and*

$$\sum_{i=1}^{n} \mu_i \left|\boldsymbol{\eta}_i - 1\right| \geq \varepsilon_2.$$

Then, $g(\alpha_{t+1}, \beta_t) - g(\alpha_t, \beta_t) \geq \Omega(\lambda \cdot \varepsilon_2)$.

Main Algorithm for Computing $\mathcal{R}_\rho(\mu,\nu)^\rho$ (after preprocessing from Subsection 3.1).

Input: Two vectors $\mu \in \mathbb{R}^n$ and $\nu \in \mathbb{R}^m$ which encode two distributions supported on the points $\{x_1,\dots,x_n\}$ and $\{y_1,\dots,y_m\}$ in $\mathbb{R}^d$, and an accuracy parameter $\varepsilon > 0$.
Assumptions: Every $i \in [n]$ and $j \in [m]$ satisfies $\sigma r \le \|x_i - y_j\|_2 \le r$. Every $i \in [n]$ has $\mu_i \ge \sigma_\mu/n$ and every $j \in [m]$ has $\nu_j \ge \sigma_\nu/m$. We refer to parameters $\varepsilon_1, \varepsilon_2, \tau$ and λ specified above (as a function of ε), and access to sub-routines `Est-Alpha`, `Est-Beta`, and `Est-Penalty`.

We initialize $(\alpha_0,\beta_0) = (0,0) \in \mathbb{R}^{n+m}$ and iteratively perform the following updates for $t \ge 1$:

- **Run Estimates**: We execute `Est-Alpha`$(\alpha_t,\beta_t,\varepsilon_1,\tau)$ which produces as output a sequence of n numbers $\boldsymbol{\eta}_1,\dots,\boldsymbol{\eta}_n$, and we execute `Est-Beta`$(\alpha_t,\beta_t,\varepsilon_1,\tau)$ which returns a sequence of m numbers $\boldsymbol{\xi}_1,\dots,\boldsymbol{\xi}_m$.

- **Update α's**: If $\sum_{i=1}^n \mu_i|1-\boldsymbol{\eta}_i| \ge \varepsilon_2$, we will update the α's by letting

$$\alpha_{t+1} = \alpha_t + \lambda \cdot \mathrm{sign}(\mathbf{1} - \boldsymbol{\eta}),$$

  where $\mathrm{sign}(\mathbf{1}-\boldsymbol{\eta})$ is the vector in $\{-1,1\}^n$ where the i-th entry is $\mathrm{sign}(1-\boldsymbol{\eta}_i)$. We also update $\beta_{t+1} = \beta_t$ and increment t, beginning a new iteration.

- **Update β's**: If $\sum_{j=1}^m \nu_j|\boldsymbol{\xi}_i - 1| \ge \varepsilon_2$, then we will update the β's by letting

$$\beta_{t+1} = \beta_t - \lambda \cdot \mathrm{sign}(\boldsymbol{\xi} - \mathbf{1}).$$

  We update $\alpha_{t+1} = \alpha_t$ and increment t, beginning a new iteration.

- **Termination**: Otherwise, if no updates where performed, then (α_t,β_t) satisfies both $\sum_{i=1}^n \mu_i|\boldsymbol{\eta}_i - 1| \le \varepsilon_2$ and $\sum_{j=1}^m \nu_j|\boldsymbol{\xi}_j - 1| \le \varepsilon_2$. In this case, we execute `Est-Penalty`$(\alpha_t,\beta_t,\varepsilon_1,\varepsilon r^\rho,\delta)$ which outputs a number $\boldsymbol{\omega} \in \mathbb{R}_{\ge 0}$, and we output

$$\sum_{i=1}^n \mu_i(\alpha_t)_i - \sum_{j=1}^m \nu_j(\beta_t)_j - \boldsymbol{\omega}.$$

Figure 1: Main Algorithm for Estimating $\mathcal{R}_\rho(\mu,\nu)^\rho$.

3.4. Proof of Theorem 1

Consider the algorithm which first runs the preprocessing step of Subsection 3.1 and then executes the main iterative sub-routine of Figure 1 in order to estimate $\mathcal{R}_\rho(\mu,\nu)^\rho$. When the algorithm from Figure 1 outputs, we output

$$\left(\sum_{i=1}^n \mu_i(\alpha_t)_i - \sum_{j=1}^m \nu_j(\beta_t)_j - \boldsymbol{\omega}\right)^{1/\rho}.$$

First, we note the running time of the algorithm is as specified. In particular, the preprocessing step takes $O(n+m)$ time. Notice that each iteration of Figure 1 takes $O(n+m)\cdot \mathrm{poly}^*(2^s/\varepsilon_1)$ time, which is $O(n+m)\cdot\mathrm{poly}^*(2^{\rho/(\rho-1)}(mn)^{(\rho-1)/\rho}/\varepsilon)$ by setting of ε_1, s, and σ_μ,σ_ν and σ. Furthermore, since $g(\alpha_0,\beta_0)=0$ and $g(\alpha_t,\beta_t)\le r^\rho$, Lemma 4 will imply that the number of iterations is at most $O(r^\rho/(\lambda\varepsilon_2))$, and by the setting of ε_2 and λ, this is at most $\mathrm{poly}^*((mn)^{(\rho-1)/\rho}/\varepsilon)$. The total running time then follows.

In order to show correctness, note that the setting of ε_2, ε_1, when the algorithm terminates, we have

$$\mathcal{R}_\rho(\mu,\nu)^\rho - \sum_{i=1}^n \mu_i(\alpha_t)_i - \sum_{j=1}^m \nu_j(\beta_t)_j - C_s\sum_{i=1}^n\sum_{j=1}^m \mu_i\nu_j\left(\frac{((\alpha_t)_i-(\beta_t)_j)^+}{\|x_i-y_j\|_2}\right)^s \le \varepsilon\cdot r^\rho,$$

and we are guaranteed by Lemma 12 and Lemma 7 and the setting of τ for `Est-Penalty` that

$$\left|\boldsymbol{\omega} - C_s\sum_{i=1}^n\sum_{j=1}^m \mu_i\nu_j\left(\frac{((\alpha_t)_i-(\beta_t)_j)^+}{\|x_i-y_j\|_2}\right)^s\right| \le O(\varepsilon\cdot r^\rho).$$

Therefore, our output (using the fact $\mathcal{R}_\rho(\mu,\nu)^\rho\le r^\rho$), will satisfy

$$\left|\left(\sum_{i=1}^n \mu_i(\alpha_t)_i - \sum_{j=1}^m \nu_j(\beta_t)_j - \boldsymbol{\omega}\right)^{1/\rho} - \mathcal{R}_\rho(\mu,\nu)\right| \le O(\varepsilon\cdot r).$$

4. Open Problems

We hope that our approach, of slightly changing the problem, will prove useful for other Euclidean problems for which we do not have fast algorithms with $(1+\varepsilon)$-approximations. We mention two immediate open problems:

- Multiplicative $(1+\varepsilon)$-approximations for $\mathcal{R}_\rho(\mu,\nu)$. Our algorithms achieved additive εr-approximations for datasets bounded within distance r, but a more accurate multiplicative $(1+\varepsilon)$-approximation would be desired when the dataset may not necessarily be bounded. Does there exists an algorithm which is just as fast as Theorem 1 and outputs a number $\widehat{\boldsymbol{\eta}}$ which is between $\mathcal{R}_\rho(\mu,\nu)$ and $(1+\varepsilon)\mathcal{R}_\rho(\mu,\nu)$ with high probability?

- Accurate Approximations for EMD. It is still possible that for any $\varepsilon>0$, there exists an algorithm which can estimate the cost of $\mathsf{EMD}(\mu,\nu)$ up to a multiplicative $(1+\varepsilon)$-factor in time

$n \cdot \mathrm{poly}(d \log n/\varepsilon)$. Does there exist such an algorithm, or is there compelling complexity-theoretic reasons why this may not be possible? We note that Rohatgi (2019) shows that, in the case μ and ν are uniform on a support of size n, such an algorithm should not be able to output a $(1+\varepsilon)$-approximate matching between points of μ and ν (assuming the Hitting Set conjecture). However, no such evidence against near-linear time algorithms for the *cost* of EMD exists.

Acknowledgments

Part of this work was done while Erik Waingarten was a postdoc at Stanford University, supported by an NSF postdoctoral fellowship and by Moses Charikar's Simons Investigator Award.

References

Pankaj K. Agarwal and R. Sharathkumar. Approximation algorithms for bipartite matching with metric and geometric costs. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC '2014)*, pages 555–564, 2014.

Pankaj K. Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. Deterministic, near-linear ε-approximation algorithm for geometric bipartite matching. In *Proceedings of the 54th ACM Symposium on the Theory of Computing (STOC '2022)*, 2022.

Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS '2017)*, 2017.

Jason Altschuler, Francis Bach, Alessandro Rudi, and Jonathan Niles-Weed. Massively scalable sinkhorn distances via the nyström method. In *Proceedings of Advances in Neural Information Processing Systems 32 (NeurIPS '2019)*, 2019.

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 793–801, 2015. Available as arXiv:1501.01062.

Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA '2008)*, pages 343–352, 2008.

Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for earth-mover distance, with applications. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*, 2009.

Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC '2014)*, 2014.

Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and embedding are equivalent for norms. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 479–488, 2015. Available as arXiv:1411.2577.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML '2017)*, 2017.

Arturs Backurs, Moses Charikar, Piotr Indyk, and Paris Siminelakis. Efficient density evaluation for smooth kernels. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2018)*, 2018.

Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Scalable nearest neighbor search for optimal transport. In *Proceedings of the 37th International Conference on Machine Learning (ICML '2020)*, 2020.

Arturs Backurs, Piotr Indyk, Cameron Musco, and Tal Wagner. Faster kernel matrix algebra via density estimation. In *Proceedings of the 38th International Conference on Machine Learning (ICML '2021)*, 2021.

Ainesh Bakshi, Piotr Indyk, Praneeth Kacham, Sandeep Silwal, and Samson Zhou. Sub-quadratic algorithms for kernel matrices via kernel density estimation. In *arXiv preprint arXiv:2212.00642*, 2022.

Arturs Bačkurs and Piotr Indyk. Better embeddings for planar earth-mover distance over sparse sets. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP '2014)*, 2014.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51, 2015.

Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*, pages 380–388, 2002.

Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2017)*, 2017.

Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2020)*, 2020.

Li Chen, Rasmus Kyng, Yang P. Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *Proceedings of the 63rd Annual IEEE Symposium on Foudnations of Computer Science (FOCS '2022)*, 2022a.

Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. New streaming algorithms for high dimensional emd and mst. In *Proceedings of the 54th ACM Symposium on the Theory of Computing (STOC '2022)*, 2022b.

Nicolas Courty, Rémy Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9): 1853 – 1865, 2016.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Proceedings of Advances in Neural Information Processing Systems (NIPS '2013)*, 2013.

Sariel Har-Peled, Piotr Indyk, and Anastasios Sidiropoulos. Euclidean spanners in high dimensions. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA '2013)*, 2013.

Piotr Indyk. Approximate nearest neighbor under edit distance via product metrics. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '2004)*, pages 646–650, 2004.

Piotr Indyk and Nitin Thaper. Fast color image retrieval via embeddings. In *Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003.

Andrey Boris Khesin, Aleksandar Nikolov, and Dmitry Paramonov. Preconditioning for the geometric transportation problem. In *Proceedings of the 35th International Symposium on Computational Geometry (SoCG '2019)*, 2019.

Pham Kiem, Khang Le, Nhat Ho, Tung Pham, and Hung Bui. On unbalanced optimal transport: An analysis of sinkhorn algorithm. In *Proceedings of the International Conference on Machine Learning (ICML '2020)*, 2020.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML '2015)*, 2015.

Khang Le, Huy Nguyen, Quang M Nguyen, Tung Pham, Hung Bui, and Nhat Ho. On robust optimal transport: computational complexity and barycenter computation. In *Proceedings of Advances in Neural Information Processing Systems 34 (NeurIPS '2021)*, 2021.

François-Pierre Paty and Marco Cuturi. Subspace robust wasserstein distances. In *Proceedings of the 36th International Conference on Machine Learning (ICML '2019)*, 2019.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5–6):355–607, 2019.

Dhruv Rohatgi. Conditional hardness of earth movers distance. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, 2019.

R. Sharathkumar and Pankaj K. Agarwal. A near-linear time ε-approximation algorithm for bipartite geometric matching. In *Proceedings of the 44th ACM Symposium on the Theory of Computing (STOC '2012)*, 2012.

Jonah Sherman. Generalized preconditioning and undirected minimum cost flow. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA '2017)*, 2017.

Paris Siminelakis, Kexin Rong, Peter Bailis, Moses Charikar, and Philip Levis. Rehashing kernel evaluation in high dimensions. In *Proceedings of the 36th International Conference on Machine Learning (ICML '2019)*, 2019.

Appendix A. Basic Properties of $\mathcal{R}_\rho(\mu, \nu)$

The main property of $\mathcal{R}_\rho(\mu, \nu)$ aligning with the Earth Mover's Distance is that $\mathcal{R}_\rho(\mu, \nu)$ defines a metric space over the space of distributions. As is the case with the Earth Mover's distance, the metric takes into account the underlying geometry of the space. The advantage to these formulations, as opposed to, say the total variation distance, is that the total variation distance between a distribution μ and that same distribution μ' after a tiny perturbation of the points is 1; whereas the Earth Mover's Distance and $\mathcal{R}_\rho(\mu, \mu')$ will remain small.

A.1. $\mathcal{R}_\rho(\mu, \nu)$ Defines a Metric Space

The one non-trivial aspect of showing $\mathcal{R}_\rho(\cdot, \cdot)$ is a metric over distributions in $\mathbb{R}^d$ is establishing the triangle inequality. Indeed, the definition $\mathcal{R}_\rho(\mu, \nu) = \mathcal{R}_\rho(\nu, \mu)$ by symmetry in the definition, and it is clear that $\mathcal{R}_\rho(\mu, \nu) \geq 0$ and equal to zero whenever μ and ν are equal.

Lemma 5 (Triangle Inequality) *Suppose μ, ν and ξ are discrete distributions supported on vectors in $\mathbb{R}^d$. Then, for any $\rho \geq 1$,*

$$\mathcal{R}_\rho(\mu, \xi) \leq \mathcal{R}_\rho(\mu, \nu) + \mathcal{R}_\rho(\nu, \xi).$$

Proof Let $\{x_1, \dots, x_n\}$ denote the set of points in the supports on μ, ν and ξ. We will then use μ, ν, ξ as vectors in $\mathbb{R}^n_{\geq 0}$ whose coordinates sum to 1 and specify with which probability to sample the point x_i. Suppose that $\gamma^{(1)}, \gamma^{(2)} \in \mathbb{R}^{n\times n}_{\geq 0}$ are the couplings which minimize $\mathcal{R}_\rho(\mu, \nu)$ and $\mathcal{R}_\rho(\nu, \xi)$, respectively. Then, consider setting

$$\gamma_{ij} = \sum_{\ell=1}^{n} \frac{\gamma^{(1)}_{i\ell} \cdot \gamma^{(2)}_{\ell j}}{\nu_\ell}.$$

First, one can easily verify that γ satisfies the constraints of $\mathcal{R}_\rho(\mu, \xi)$, so that it is a coupling of μ and ξ. The triangle inequality then comes from using the triangle inequality in the ground metric (ℓ_2), and then the triangle inequality for the ℓ_ρ-norm encoding the cost. Specifically,

$$\begin{aligned}
\mathcal{R}_\rho(\mu, \xi) &\leq \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \xi_j \left(\frac{\gamma_{ij}}{\mu_i \xi_j} \|x_i - x_j\|_2 \right)^\rho \right)^{1/\rho} \\
&\leq \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \xi_j \left(\frac{1}{\mu_i \xi_j \nu_\ell} \sum_{\ell=1}^{n} \gamma^{(1)}_{i\ell} \gamma^{(2)}_{\ell j} \|x_i - x_\ell\|_2 \right)^\rho \right)^{1/\rho} \qquad (8) \\
&\qquad + \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \xi_j \left(\frac{1}{\mu_i \xi_j \nu_\ell} \sum_{\ell=1}^{n} \gamma^{(1)}_{i\ell} \gamma^{(2)}_{\ell j} \|x_\ell - x_j\|_2 \right)^\rho \right)^{1/\rho}. \qquad (9)
\end{aligned}$$

We work on each term individually. In particular, for each $j \in [n]$, we use Jensen's inequality, applied to the distribution $\tilde{\gamma}^{(2)}_j$ which is supported on $[n]$ and samples $\boldsymbol{\ell} = \ell$ with probability $\gamma^{(2)}_{\ell j} / \xi_j$

can write

$$\sum_{j=1}^{n} \xi_j \left(\frac{1}{\mu_i \xi_j \nu_\ell} \sum_{\ell=1}^{n} \gamma_{i\ell}^{(1)} \gamma_{\ell j}^{(2)} \cdot \|x_i - x_\ell\|_2 \right)^{\rho} = \sum_{j=1}^{n} \xi_j \left(\mathop{\mathbf{E}}_{\boldsymbol{\ell} \sim \tilde{\gamma}_j^{(2)}} \left[\frac{\gamma_{i\boldsymbol{\ell}}^{(1)}}{\mu_i \nu_{\boldsymbol{\ell}}} \cdot \|x_i - x_{\boldsymbol{\ell}}\|_2 \right] \right)^{\rho}$$
$$\leq \sum_{j=1}^{n} \xi_j \sum_{\ell=1}^{n} \frac{\gamma_{\ell j}^{(2)}}{\xi_j} \cdot \left(\frac{\gamma_{i\ell}^{(1)}}{\mu_i \nu_\ell} \cdot \|x_i - x_\ell\|_2 \right)^{\rho}$$
$$= \sum_{\ell=1}^{n} \nu_\ell \left(\frac{\gamma_{i\ell}^{(1)}}{\mu_i \nu_\ell} \cdot \|x_i - x_\ell\|_2 \right)^{\rho}.$$

Applied to each $i \in [n]$, we have that (8) is at most $\mathcal{R}_\rho(\mu, \nu)$. Similarly, we have (9) is at most $\mathcal{R}_\rho(\nu, \xi)$. ■

A.2. Dual of $\mathcal{R}_\rho(\mu, \nu)^\rho$

As discussed, the advantage of $\mathcal{R}_\rho(\mu, \nu)$ is its computational properties, since we will give an algorithm to additively approximate $\mathcal{R}_\rho(\mu, \nu)$ efficiently (for small ρ). The algorithm will proceed by optimizing over the dual formulation of $\mathcal{R}_\rho(\mu, \nu)^\rho$. In this subsection, we specify what the dual of $\mathcal{R}_\rho(\mu, \nu)^\rho$ looks like, as well as a few important properties of the dual which we will use throughout the execution of the algorithm. In the next lemma, it will be important that the support of the distributions μ and ν are disjoint (so that we never divide by zero); in our algorithm, we will apply a preprocessing step so as to assume without loss of generality, that the supports of μ and ν are disjoint.

Lemma 6 *Let $\mu \in \mathbb{R}_{>0}^n$ encode a distribution supported on $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$ and $\nu \in \mathbb{R}_{>0}^m$ encode a distribution supported on $\{y_1, \dots, y_m\} \subset \mathbb{R}^d$. Whenever $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ are disjoint, the following is true for any $\rho > 1$. Let $s \geq 1$ denote its Hölder conjugate, $1/\rho + 1/s = 1$, and $C_s = (1/s)(1 - 1/s)^{s-1}$. Then,*

$$\mathcal{R}_\rho(\mu, \nu)^\rho = \max_{\substack{\alpha \in \mathbb{R}^n \\ \beta \in \mathbb{R}^m}} \sum_{i=1}^{n} \mu_i \alpha_i - \sum_{j=1}^{m} \nu_j \beta_j - C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s. \tag{10}$$

Proof We compute the dual of $\mathcal{R}_\rho(\mu, \nu)^\rho$ by introducing Lagrangian multipliers.

$$\mathcal{R}_\rho(\mu, \nu)^\rho = \max_{\substack{\alpha \in \mathbb{R}^n \\ \beta \in \mathbb{R}^m}} \min_{\gamma \in \mathbb{R}_{\geq 0}^{n \times m}} \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \left(\frac{\gamma_{ij}}{\mu_i \nu_j} \right)^{\rho} \|x_i - y_j\|_2^{\rho}$$
$$+ \sum_{i=1}^{n} \alpha_i \left(\mu_i - \sum_{j=1}^{m} \gamma_{ij} \right) - \sum_{j=1}^{m} \beta_j \left(\nu_j - \sum_{i=1}^{n} \gamma_{ij} \right),$$

and we re-write this, by minimax duality

$$\min_{\gamma} \max_{\alpha, \beta} \sum_{i=1}^{n} \mu_i \alpha_i + \sum_{j=1}^{m} \nu_j \beta_j + \sum_{i=1}^{n} \sum_{j=1}^{m} \left((\mu_i \nu_j)^{1-\rho} \gamma_{ij}^{\rho} \|x_i - y_j\|_2^{\rho} - (\alpha_i - \beta_j) \gamma_{ij} \right).$$

Since the right-most term is the only one to depend on γ, once we fix α, β, we can compute the minimizing value $\gamma_{ij} \geq 0$ for each term i, j. The expression for a (generic) term i, j considers a positive value $a > 0$ (the fact the term is positive is by disjointness of $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$) and some term $b \in \mathbb{R}$, and we want to determine

$$\min_{\gamma_{ij} \geq 0} \gamma_{ij}^{\rho} a - \gamma_{ij} b.$$

Note that if b is negative, then the minimizing value occurs at 0 with $\gamma_{ij} = 0$. On the other hand, if b is positive, then we want to set γ_{ij} such that the derivative with respect to γ_{ij} is zero, since this corresponds to the (positive) value of γ_{ij} where we get a negative value of the optimum above:

$$\rho \gamma_{ij}^{\rho-1} a - b = 0 \Longrightarrow \gamma_{ij} = \left(\frac{b}{\rho a}\right)^{1/(\rho-1)} \Longrightarrow \gamma_{ij}^{\rho} a - \gamma_{ij} b = \frac{b^{\rho/(\rho-1)}}{a^{1/(\rho-1)}} \cdot \left(\frac{1}{\rho^{\rho/(\rho-1)}} - \frac{1}{\rho^{1/(\rho-1)}}\right)$$

In order to simplify the notation, let $s \geq 1$ denote the Hölder conjugate of ρ. So that

$$\frac{1}{\rho} + \frac{1}{s} = 1 \qquad \frac{\rho}{\rho - 1} = s \qquad \frac{1}{\rho - 1} = s - 1,$$

and the above expression becomes:

$$-\left(\frac{b}{a}\right)^{s} \cdot a \cdot \frac{1}{s}\left(1 - \frac{1}{s}\right)^{s-1}.$$

Plugging this in to each term, and letting $C_s = (1/s)(1 - 1/s)^{s-1}$, each term becomes

$$C_s \left(\left(\frac{1}{\mu_i \nu_j}\right)^{(1-\rho)} \frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2^{\rho}} \right)^{s} (\mu_i \nu_j)^{1-\rho} \|x_i - y_j\|_2^{\rho} = C_s \cdot \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2}\right)^{s}.$$

which means the dual becomes

$$\max_{\alpha, \beta \in \mathbb{R}^n} \sum_{i=1}^{n} \mu_i \alpha_i - \sum_{j=1}^{n} \nu_j \beta_j - C_s \sum_{i=1}^{n} \sum_{j=1}^{n} \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - x_j\|_2}\right)^{s},$$

and for any $\alpha, \beta \in \mathbb{R}^n$, the minimizing value of γ is

$$\gamma_{ij} = s C_s \cdot \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - x_j\|_2^{s}}.$$

■

We intuitively think of the dual of $\mathcal{R}_\rho(\mu, \nu)^\rho$ in (10) as consisting of two parts. The linear term, the first two summands in (10), and a non-linear "penalty" term given by $C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j ((\alpha_i - \beta_j)^+ / \|x_i - y_j\|_2)^s$. It is useful to compare the dual of $\mathcal{R}_\rho(\mu, \nu)^\rho$ to the dual of the Earth Mover's distance, which would essentially only consider the two linear terms and always enforce that $\alpha_i - \beta_j \leq \|x_i - y_j\|_2$. In (10), the nm constraints of the Earth Mover's distance instead become a single "penalty" which encode how much larger $\alpha_i - \beta_j$ is, in comparison to $\|x_i - y_j\|_2$. The following two lemmas will become important in the analysis of the algorithm, since they will say that the non-linear "penalty" term does not become too large throughout the optimization. Intuitively, this will mean that as we perform a gradient ascent algorithm, the function which we optimize remains smooth.

Lemma 7 *Let $s \geq 2$ and $g\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ denote the objective function which we seek to optimize, i.e., that which is the dual of $\mathcal{R}_\rho(\mu,\nu)^\rho$,*

$$g(\alpha,\beta) = \sum_{i=1}^n \mu_i \alpha_i - \sum_{j=1}^m \nu_j \beta_j - C_s \sum_{i=1}^n \sum_{j=1}^m \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s.$$

Then, if $(\alpha,\beta) \in \mathbb{R}^{n+m}$ is any point where $g(\alpha,\beta) \geq 0$, then

$$C_s \sum_{i=1}^n \sum_{j=1}^m \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s \leq 4 \cdot \mathcal{R}_\rho(\mu,\nu)^\rho.$$

In addition, there exists an input $(\alpha',\beta') \in \mathbb{R}^{n+m}$ with $g(\alpha',\beta') \geq g(\alpha,\beta)$ which satisfies

$$\|(\alpha',\beta')\|_\infty \leq \frac{4^{1/s}}{2} \cdot \sup_{i,j} \left(\frac{1}{\mu_i \nu_j} \right)^{(\rho-1)/\rho} \cdot r^\rho.$$

Proof Let ψ denote the quantity

$$\psi = C_s \sum_{i=1}^n \sum_{j=1}^m \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s$$

which we wish to upper bound. Then, by the fact $g(\alpha,\beta) \geq 0$, we must have that

$$\zeta = \sum_{i=1}^n \mu_i \alpha_i - \sum_{j=1}^m \nu_j \beta_j \geq \psi.$$

If we consider the point $(\alpha',\beta') \in \mathbb{R}^{n+m}$ given by $(\alpha,\beta)/2$, we have $g(\alpha',\beta') = \zeta/2 - \psi/2^s$, and since the dual of $\mathcal{R}_\rho(\mu,\nu)^\rho$ is a maximization problem, we must have

$$\psi\left(\frac{1}{2} - \frac{1}{2^s}\right) \leq \frac{\zeta}{2} - \frac{\psi}{2^s} = g(\alpha',\beta') \leq \mathcal{R}_\rho(\mu,\nu)^\rho.$$

When $s \geq 2$, we may then obtain the desired upper bound. For the second part of the claim, we note that given (α,β), we can let (α',β') be the vector where $\alpha'_i = \max\{\alpha_i, \min_j \beta_j\}$ and $\beta'_j = \min\{\beta_j, \max_i \alpha'_i\}$. This means that $g(\alpha',\beta')$ cannot be smaller than $g(\alpha,\beta)$, since we have not increased the penalty ψ, while potentially increasing ζ. This means that every entry of α' and β' lies between the smallest entry of β' and the largest entry of α'. However, we note that $\max_i \alpha'_i - \min_j \beta'_j$ can be at most

$$4^{1/s} \cdot r^\rho \cdot \sup_{i\in[n], j\in[m]} \left(\frac{1}{\mu_i \nu_j} \right)^{1/s},$$

before the penalty term ψ exceeds $4 \cdot \mathcal{R}_\rho(\mu,\nu)^\rho \leq 4r^\rho$. Thus, we can translate every entry of α' by subtracting c and every entry of β' by adding c without changing the objective function, and ensuring the bound on the ℓ_∞-norm of α' and β'. ■

Lemma 8 *Let $\rho \in (1, 2)$ and $s > 2$ be the Hölder conjugate, and g as in Lemma 7. Then, if $(\alpha, \beta) \in \mathbb{R}^{n+m}$ where $g(\alpha, \beta) \geq 0$, then if* $r_{\min} = \min_{i,j} \|x_i - y_j\|_2$ *and* $\max_{i,j} \|x_i - y_j\|_2 \leq r$, *we have*

$$C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \leq \frac{4 \cdot r}{r_{\min}} \qquad \text{and} \qquad C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-2}}{\|x_i - y_j\|_2^s} \leq \frac{4 \cdot r}{r_{\min}}$$

Proof Both upper bounds follow from Jensen's inequality and Lemma 7. Namely, we use the fact that the function $(\cdot)^{s/(s-1)}$ is convex, and that $\rho = s/(s-1)$,

$$\begin{aligned} C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} &\leq \frac{C_s}{r_{\min}} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s \right)^{1/\rho} \\ &\leq \frac{C_s}{r_{\min}} \left(\frac{4 \cdot r^\rho}{C_s} \right)^{1/\rho} \leq \frac{4 \cdot r}{r_{\min}}. \end{aligned}$$

Similarly, $(\cdot)^{s/(s-2)}$ is convex, and $\rho(s-2)/s = 2 - \rho$,

$$\begin{aligned} C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-2}}{\|x_i - y_j\|_2^s} &\leq \frac{C_s}{r_{\min}^2} \left(\sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s \right)^{(s-2)/s} \\ &\leq \frac{C_s}{r_{\min}^2} \left(\frac{4 \cdot r^\rho}{C_s} \right)^{(s-2)/s} \leq \frac{4 \cdot r^{2-\rho}}{r_{\min}^2}. \end{aligned}$$

■

Appendix B. Lemmas and Proofs of Section 3

Lemma 9 *For any $\rho \geq 1$ as well as parameters $\sigma, \sigma_\mu, \sigma_\nu \geq 0$, the distributions μ' and ν' satisfy:*

- *Every point in the support of μ' and every point in the support of ν' has distance between σr and $r\sqrt{1 + \sigma^2}$.*
- *Every element of the support of μ' is sampled with probability at least σ_μ / n, and every element of ν' is sampled with probability at least σ_ν / m.*
- *Both μ' and ν' are minor perturbations of μ and ν, i.e.,*

$$\mathcal{R}_\rho(\mu, \mu') \leq \left(n^{\rho-1} \cdot \frac{\sigma}{\sigma_\mu^{\rho-1}} + \sigma_\mu \right)^{1/\rho} \cdot r \qquad \text{and} \qquad \mathcal{R}_\rho(\nu, \nu') \leq \sigma_\nu^{1/\rho} \cdot r.$$

Proof Consider the coupling $\gamma^{(1)} \in \mathbb{R}^{n \times n}$ between μ and μ' given by

$$\gamma_{ij}^{(1)} = \begin{cases} \mu_i & i = j \text{ and } i \in [n] \setminus L_\mu \\ \mu_i \mu_j / (1 - \zeta_\mu) & i \in L_\mu \text{ and } j \in [n] \setminus L_\mu \\ 0 & j \in L_\mu \end{cases}.$$

It is simple to verify, using the fact $\sum_{i\in L_\mu} \mu_i = \zeta_\mu$ that the above is indeed a coupling, and we now upper bound the cost:

$$\mathcal{R}_\rho(\mu,\mu')^\rho \leq \sum_{i\in[n]\setminus L_\mu} \frac{\mu_i^2}{1-\zeta_\mu}\cdot\left(\frac{\mu_i(1-\zeta_\mu)}{\mu_i^2}\cdot\sigma r\right)^\rho + \sum_{i\in L_\mu}\sum_{j\in[n]\setminus L_\mu}\frac{\mu_i\mu_j}{1-\zeta_\mu}\cdot r^\rho$$
$$\leq \sum_{i\in[n]\setminus L_\mu} \mu_i\cdot(1/\mu_i)^{\rho-1}\cdot(\sigma r)^\rho + \zeta_\mu\cdot r^\rho \leq \left(n^{\rho-1}\cdot\frac{\sigma}{\sigma_\mu^{\rho-1}}+\sigma_\mu\right)\cdot r^\rho.$$

Similarly, we may write the coupling $\gamma^{(2)}\in\mathbb{R}^{m\times m}$ between ν and ν' given by

$$\gamma^{(2)}_{ij} = \begin{cases} \nu_i & i=j \text{ and } i\in[m]\setminus L_\nu \\ \nu_i\nu_j/(1-\zeta_\nu) & i\in L_\nu \text{ and } j\in[m]\setminus L_\nu \\ 0 & j\in L_\nu \end{cases},$$

which allows us to upper bound $\mathcal{R}_\rho(\nu,\nu')$ by

$$\mathcal{R}_\rho(\nu,\nu')^\rho \leq \sum_{i\in L_\nu}\sum_{j\in[m]\setminus L_\nu}\frac{\nu_i\nu_j}{1-\zeta_\nu}\cdot r^\rho \leq \sigma_\nu\cdot r^\rho$$

■

B.1. Additional Details from Section 3.2

Three Sub-routines Before stating the main lemma which we will prove for the guarantees on the above algorithm, we give the three lemmas which encapsulate the performance guarantees on `Est-Alpha`, `Est-Beta`, and `Est-Penalty`. Assuming these lemmas, we will then prove the main lemma, and show how that implies Theorem 1. We will defer the proof the three lemmas until after the analysis of the algorithm, as they rely on the data structures from Appendix C. Thus, the proofs of Lemma 10 and Lemma 11 appear in Appendix C.2, and the proof of Lemma 12 appears in Appendix C.3.

Lemma 10 (Guarantees on `Est-Alpha`) *Fix a parameter $s\geq 1$ and a small parameter $\tau>0$, we also fix two distributions μ,ν supported on points $\{x_1,\dots,x_n\}$ and $\{y_1,\dots,y_m\}$ whose pairwise distance is between σr and r. For any $(\alpha,\beta)\in\mathbb{R}^{n+m}$ and any $\varepsilon>0$, there is a randomized algorithm* `Est-Alpha` *with the following guarantees:*

- *The algorithm receives as input the vector $(\alpha,\beta)\in\mathbb{R}^{n+m}$, the accuracy parameters $\varepsilon>0$ and $\tau>0$, and failure probability $\delta\in(0,1)$. The algorithm produces as output a sequence of n numbers $\boldsymbol{\eta}_1,\dots,\boldsymbol{\eta}_n\in\mathbb{R}_{\geq 0}$.*
- *The algorithm runs in time $(n+m)\cdot\mathrm{poly}^*(2^s/\varepsilon)$ and with probability at least $1-\delta$, every $i\in[n]$ satisfies*

$$\left(sC_s\sum_{j=1}^m \nu_j\cdot\frac{((\alpha_i-\beta_j)^+)^{s-1}}{\|x_i-y_j\|_2^s}\right)-\tau\leq\boldsymbol{\eta}_i\leq(1+\varepsilon)\left(sC_s\sum_{j=1}^m \nu_j\cdot\frac{((\alpha_i-\beta_j)^+)^{s-1}}{\|x_i-y_j\|_2^s}\right).$$

Lemma 11 (Guarantees on `Est-Beta`) *Fix a parameter $s \geq 1$ and a small parameter $\tau > 0$, we also fix two distributions μ, ν supported on points $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ whose pairwise distance is between σr and r. For any $(\alpha, \beta) \in \mathbb{R}^{n+m}$ and any $\varepsilon > 0$, there is a randomized algorithm* `Est-Beta` *with the following guarantees:*

- *The algorithm receives as input the vector $(\alpha, \beta) \in \mathbb{R}^{n+m}$, the accuracy parameters $\varepsilon > 0$ and $\tau > 0$, and failure probability $\delta \in (0, 1)$. The algorithm produces as output a sequence of n numbers $\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_m \in \mathbb{R}_{\geq 0}$.*
- *The algorithm runs in time $(n + m) \cdot \text{poly}^*(2^s/\varepsilon)$ and with probability at least $1 - \delta$, every $j \in [m]$ satisfies*

$$\left(sC_s \sum_{i=1}^{n} \mu_i \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \right) - \tau \leq \boldsymbol{\xi}_j \leq (1 + \varepsilon) \left(sC_s \sum_{i=1}^{n} \mu_i \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \right).$$

Lemma 12 (Guarantees on `Est-Penalty`) *Fix a parameter $s \geq 1$ and a small parameter $\tau > 0$, we also fix two distributions μ, ν supported on points $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$ whose pairwise distance is between σr and r. For any $(\alpha, \beta) \in \mathbb{R}^{n+m}$ and any $\varepsilon > 0$, there is a randomized algorithm* `Est-Penalty` *with the following guarantees:*

- *The algorithm receives as input the vector $(\alpha, \beta) \in \mathbb{R}^{n+m}$, the accuracy parameters $\varepsilon > 0$ and $\tau > 0$, and failure probability $\delta \in (0, 1)$. The algorithm produces as output a number $\boldsymbol{\omega} \in \mathbb{R}_{\geq 0}$.*
- *The algorithm runs in time $(n + m) \cdot \text{poly}^*(2^s/\varepsilon)$ and satisfies that with high probability,*

$$\left(C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s \right) - \tau \leq \boldsymbol{\omega} \leq (1 + \varepsilon) \left(C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s \right).$$

B.2. Proofs from the Analysis of the Algorithm

Proof [Proof of the Termination Condition] First, we note that we can apply the translation so as to assume that (α_t, β_t) and (α^*, β^*) satisfies

$$\|\alpha_t\|_\infty, \|\beta_t\|_\infty, \|\alpha^*\|_\infty, \|\beta^*\|_\infty \leq O(1) \cdot \left(\frac{nm}{\sigma_\mu \sigma_\nu} \right)^{(\rho-1)/\rho} \cdot r^\rho.$$

Let $\eta_1, \dots, \eta_n$ and $\xi_1, \dots, \xi_m$ denote the quantities

$$\eta_i = sC_s \sum_{j=1}^{m} \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \qquad \text{and} \qquad \xi_j = sC_s \sum_{i=1}^{n} \mu_i \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s},$$

and consider the function

$$\tilde{g}(\alpha, \beta) = \sum_{i=1}^{n} \mu_i \eta_i \alpha_i - \sum_{j=1}^{m} \nu_j \xi_j \beta_j - C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \left(\frac{(\alpha_i - \beta_j)^+}{\|x_i - y_j\|_2} \right)^s.$$

Notice that $\tilde{g}$ is a concave function, and the partial derivatives at (α_t, β_t) are all zero, so that $\tilde{g}$ is maximized at (α_t, β_t). Thus,

$$\begin{aligned} g(\alpha_t, \beta_t) &= \tilde{g}(\alpha_t, \beta_t) + \sum_{i=1}^{n} \mu_i (1 - \eta_i)\alpha_i - \sum_{j=1}^{m} \nu_j (1 - \xi_j)\beta_j \\ &\geq \tilde{g}(\alpha^*, \beta^*) + \sum_{i=1}^{n} \mu_i (1 - \eta_i)\alpha_i - \sum_{j=1}^{m} \nu_j (1 - \xi_j)\beta_j \\ &= g(\alpha^*, \beta^*) + \sum_{i=1}^{n} \mu_i (1 - \eta_i)(\alpha_i - \alpha_i^*) - \sum_{j=1}^{m} \nu_j (1 - \xi_j)(\beta_j - \beta_j^*). \end{aligned}$$

This implies that

$$g(\alpha^*, \beta^*) - g(\alpha_t, \beta_t) \leq (\|\alpha_t\|_\infty + \|\alpha^*\|_\infty) \sum_{i=1}^{n} \mu_i |1 - \eta_i| + (\|\beta_t\|_\infty + \|\beta^*\|_\infty) \sum_{j=1}^{m} \nu_j |1 - \xi_j|.$$

By correctness of the algorithms `Est-Alpha` and `Est-Beta`, $\boldsymbol{\eta}_i$ is a good approximation to η_i and $\boldsymbol{\xi}_j$ is a good approximation to ξ_j, which allows us to upper bound the above expression by

$$(\|\alpha_t\|_\infty + \|\alpha^*\|_\infty + \|\beta_t\|_\infty + \|\beta^*\|_\infty) \left(\varepsilon_2 + \tau + \varepsilon_1 \cdot sC_s \sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \right),$$

which by Lemma 8 is at most

$$O(1) \cdot \left(\frac{nm}{\sigma_\mu \sigma_\nu} \right)^{(\rho-1)/\rho} \cdot r^\rho \cdot \left(\varepsilon_2 + \tau + \frac{\varepsilon_1 s}{\sigma} \right),$$

since $r/r_{\min} \leq \sigma$. ■

Proof [Proof that Updates Increase Objective (Lemma 4)] In order to simplify the notation, we will let $\beta = \beta_t = \beta_{t+1}$, and also denote $\tilde{\alpha} = \alpha_{t+1}$ and $\alpha = \alpha_t$. Then, we have

$$g(\tilde{\alpha}, \beta) - g(\alpha, \beta) = \sum_{i=1}^{n} \mu_i (\tilde{\alpha}_i - \alpha_i) + C_s \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{\mu_i \nu_j}{\|x_i - y_j\|_2^s} \cdot \left(((\alpha_i - \beta_j)^+)^s - ((\tilde{\alpha}_i - \beta_j)^+)^s \right).$$

We now use the following simple consequence of convexity of $((t)^+)^s$ and $((t)^+)^{s-1}$:

$$\begin{aligned} &((\alpha_i - \beta_j)^+)^s - ((\tilde{\alpha}_i - \beta_j)^+)^s \\ &\qquad \geq s(\alpha_i - \tilde{\alpha}_i) \cdot ((\alpha_i - \beta_j)^+)^{s-1} - s(s-1)(\alpha_i - \tilde{\alpha}_i)^2 ((\alpha_i - \beta_j)^+)^{s-2}. \end{aligned}$$

In particular, we may lower bound $g(\tilde{\alpha}, \beta) - g(\alpha, \beta)$ by

$$\sum_{i=1}^{n} (\tilde{\alpha}_i - \alpha_i) \cdot \mu_i \cdot \left(1 - sC_s \sum_{j=1}^{m} \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \right) \tag{11}$$

$$- s(s-1)C_s \sum_{i=1}^{n} (\tilde{\alpha}_i - \alpha_i)^2 \sum_{j=1}^{m} \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-2}}{\|x_i - y_j\|_2^s}. \tag{12}$$

We will lower bound the first term (11) and upper bound the second term (12). In particular, recall that due to our approximation guarantee on $\boldsymbol{\eta}_i$, every $i \in [n]$ satisfies

$$\left| \boldsymbol{\eta}_i - sC_s \sum_{j=1}^m \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \right| \leq \tau + \varepsilon_1 sC_s \sum_{j=1}^m \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s},$$

and by our definition of the update, we may lower bound (11), using Lemma 8, by

$$\lambda \sum_{i=1}^n \mu_i |1 - \boldsymbol{\eta}_i| - \lambda\tau - \lambda\varepsilon_1 \cdot sC_s \sum_{i=1}^n \sum_{j=1}^m \mu_i \nu_j \cdot \frac{((\alpha_i - \beta_j)^+)^{s-1}}{\|x_i - y_j\|_2^s} \geq \lambda \left(\varepsilon_2 - \tau - \frac{4\varepsilon_1 \cdot s}{\sigma} \right).$$

Then, we may upper bound (12) by $(\tilde{\alpha}_i - \alpha_i)^2 \leq \lambda^2$ for every $i \in [n]$, which implies, also by Lemma 8 that (12) is at most

$$O(1) \cdot \frac{s(s-1) \cdot \lambda^2}{\sigma^2 \cdot r^\rho}.$$

In particular, since λ is a small constant factor of $\varepsilon_2 \cdot r^\rho \cdot \sigma^2 / s^2$, and both τ and $\varepsilon_1 s/\sigma$ are substantially smaller than ε_2, we may lower bound

$$g(\tilde{\alpha}, \beta) - g(\alpha, \beta) \geq \Omega(\lambda \cdot \varepsilon_2).$$

$\blacksquare$

Appendix C. Augmenting Kernel Density Estimation Data Structures

This section gives the algorithms for `Est-Alpha`, `Est-Beta`, and `Est-Penalty` giving the proofs of Lemma 10, Lemma 11, and Lemma 12. We first draw the connection to kernel density estimation and define the modified data structure problem that we will need. Then, Lemma 10, Lemma 11 and Lemma 12 will follow from different instantiations of one data structure.

Definition 13 (Kernel Density Estimation) *Let $\mathsf{K} \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ be a function, $\Phi > 1$ an aspect ratio bound, $\varepsilon > 0$ a multiplicative error parameter, and $\delta > 0$ a failure probability.* $\texttt{KDE}(\mathsf{K}, \Phi, \varepsilon, \delta)$ *is the following data structure problem:*

- **Preprocessing***: The data structure receives a set of points $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$.*
- **Query***: A query is specified by a point $y \in \mathbb{R}^d$, and we will have the promise that* $\max_i \|x_i - y\|_2 / \min_i \|x_i - y\|_2$ *is at most Φ. The data structure should output an estimate $\widehat{\boldsymbol{\xi}} \in \mathbb{R}_{\geq 0}$.*

The guarantee is that for any dataset and any query $y \in \mathbb{R}^d$, with probability at least $1 - \delta$ over the randomness in constructing the data structure,

$$(1 - \varepsilon) \sum_{i=1}^n \mathsf{K}(x_i, y) \leq \widehat{\boldsymbol{\xi}} \leq (1 + \varepsilon) \sum_{i=1}^n \mathsf{K}(x_i, y).$$

In using data structures for kernel density estimation, we will instantiate the data structure for sets of vectors which will be subsets of the support of the distributions μ and ν. In addition, the aspect ratio bound will be $\Phi = 1/\sigma$ (since we consider inputs whose distance is at most r and the minimum distance is at least σr). For a small parameter $\varepsilon_0 > 0$, we will be interested in kernel functions $\mathsf{K} \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ of the form:

$$\mathsf{K}(x_i, y) = \frac{1}{\varepsilon_0 \cdot (\sigma r)^s + \|x_i - y\|_2^s}. \tag{13}$$

The kernel (13) is a scaled Student-t Kernel, and falls within the kernels explored in Backurs et al. (2018). The results of Backurs et al. (2018) hold more generally for classes of "smooth" kernels, where they formally define (L, t)-smooth kernels (see Definition 1 in Backurs et al. (2018)). We note that K in (13) is a $(1, s)$-smooth kernel, so that their results will apply with $L = 1$ and $t = s$. For this setting, we have every $i \in [n]$ and $y \in \mathbb{R}^d$ within distance between σr and r from x_i,

$$\frac{1 - \varepsilon_0}{\|x_i - y\|_2^s} \leq \mathsf{K}(x_i, y) \leq \frac{1}{\|x_i - y\|_2^s}.$$

Theorem 14 (Main Theorem of Backurs et al. (2018), instantiated to K in (13)) *For any $\Phi > 1$, and $\varepsilon, \delta > 0$, there exists two randomized algorithms* `Preprocess`, *and* `Query` *for solving* $\texttt{KDE}(\mathsf{K}, \Phi, \varepsilon, \delta)$, *with the following guarantees:*

- $\texttt{Preprocess}(X)$ *receives as a dataset $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$, and outputs a pointer v to a data structure for* $\texttt{KDE}(\mathsf{K}, \Phi, \varepsilon, \delta)$.
- $\texttt{Query}(v, y)$ *receives as input a pointer to a data structure v for* $\texttt{KDE}(\mathsf{K}, \Phi, \varepsilon, \delta)$ *and returns the query at y for* $\texttt{KDE}(\mathsf{K}, \Phi, \varepsilon, \delta)$.

We are guaranteed that `Query` *takes time* $\mathrm{poly}^*(2^s/\varepsilon)$, *and the algorithm* `Preprocess` *takes time* $O(n) \cdot \mathrm{poly}^*(2^s/\varepsilon)$.

We now introduce the augmented data structure problem which we need in order to solve `Est-Alpha`, `Est-Beta`, and `Est-Penalty`. The goal is to incorporate the fact that points have some associated real values α, β.

Definition 15 (Augmented Kernel Density Estimation) *Let $s_2 \geq 1$ be a parameter, $\Phi > 1$ is an aspect ratio bound, $\varepsilon > 0$ be a multiplicative error parameter, and $\delta > 0$ be a desired failure probability.* $\texttt{Augmented-KDE}(\mathsf{K}, s_2, \Phi, \varepsilon, \delta)$ *is the following data structure problem.*

- **Preprocessing**: *We receive a set of points $X = \{x_1, \dots, x_n\} \in \mathbb{R}^d$. In addition, each point has an associated weight $\alpha_i \in \mathbb{R}$ with $|\alpha_i| \leq r \cdot \mathrm{poly}(dn\Phi 2^s/\varepsilon)$ and a parameter $\mu_i \in [1/\mathrm{poly}(n), 1]$, for a parameter $r \geq 0$ which will be the maximum distance considered.*
- **Query**: *A query is specified by a point $y \in \mathbb{R}^d$ and weight $\beta \in \mathbb{R}$. We are promised that:*
  - *The point $y \in \mathbb{R}^d$ satisfies $\max_{i \in [n]} \|x_i - y\|_2 \leq r$ and that $\min_i \|x_i - y\|_2$ is at least σr.*
  - *In addition, for every i, $|\alpha_i - \beta| \in \{0\} \cup [\sigma r/\mathrm{poly}(dn\Phi 2^s/\varepsilon), r \cdot \mathrm{poly}(dn\Phi 2^s/\varepsilon)]$, and the data structure outputs a quantity $\widehat{\boldsymbol{\eta}} \in \mathbb{R}_{\geq 0}$.*

The guarantee is that for any fixed query, with probability at least $1-\delta$ over the randomness in the construction of the data structure,

$$(1-\varepsilon)\sum_{i=1}^{n}\mu_i\cdot((\alpha_i-\beta)^+)^{s_2}\cdot \mathsf{K}(x_i,y)\le \widehat{\boldsymbol{\xi}}\le (1+\varepsilon)\sum_{i=1}^{n}\mu_i\cdot((\alpha_i-\beta)^+)^{s_2}\cdot \mathsf{K}(x_i,y).$$

Theorem 16 *For any $s_2\ge 1$, $\Phi>1$, and $\varepsilon,\delta>0$, there exists three randomized algorithms* `PreprocessA` *and* `QueryA` *for solving* `Augmented-KDE`$(\mathsf{K},s_2,\Phi,\varepsilon,\delta)$*, with the following guarantees:*

- `PreprocessA`(X,α) *receives as input a dataset X of at most n points, and a vector α indicating a weight for each point and the vector μ. The algorithm outputs a pointer v to a data structure for* `Augmented-KDE`$(\mathsf{K},s_2,\Phi,\varepsilon,\delta)$.
- `QueryA`(v,y,β) *receives as input a pointer to a data structure for* `Augmented-KDE`$(\mathsf{K},s_2,\Phi,\varepsilon,\delta)$*, a point $y\in\mathbb{R}^d$, and a weight $\beta\in\mathbb{R}$ such that $|\alpha_i-\beta|\in\{0\}\cup\{\sigma r/\mathrm{poly}(dn\Phi 2^s/\varepsilon), r\cdot \mathrm{poly}(dn\Phi 2^s/\varepsilon)]$. The algorithm outputs query at y with weight β for* `Augmented-KDE`$(\mathsf{K},s_2,\Phi,\varepsilon,\delta)$.

We are guaranteed that `QueryA` *takes time* $\mathrm{poly}^*(2^{s+s_2}/\varepsilon)$*, and* `PreprocessA` *takes time* $O(n)\cdot \mathrm{poly}^*(2^{s+s_2}/\varepsilon)$.

C.1. Proof of Theorem 16

Since we are promised that every index $\mu_1,\dots,\mu_n$ is a number between $\mu_{\min}=1/\mathrm{poly}(n)$ and 1, and we can output a multiplicative $1\pm\varepsilon$-approximation to the final sum, we will partition the set of points into $O(\log n/\varepsilon)$ many parts, according to the range for which $\mu_i\in[\mu_{\min}(1+\varepsilon)^j,\mu_{\min}(1+\varepsilon)^{j+1}]$. Then, it suffices to output, for each of the $O(\log n/\varepsilon)$ many ranges j, a $1\pm\varepsilon$-approximation to the quantity

$$\sum_{i\in P_j}((\alpha_i-\beta)^+)^{s_2}\cdot \mathsf{K}(x_i,y).$$

For the remainder of the discussion we will assume that we have performed this partition (to drop μ from the notation), and assume henceforth that all of the weights specified by μ are equal.

We refer to Figure 2 for the description of the data structure, which maintains a binary tree over the points in X sorted according to α, where each internal node of the tree additionally holds a pointer to a `KDE`$(\mathsf{K},\Phi,\varepsilon,\delta)$ data structure. From the description of Figure 2, the algorithms `PreprocessA` is straight-forward, and we will mostly give and analyze `QueryA`.

Lemma 17 *An execution of* `QueryA`(v,y,β) *takes time* $\mathrm{poly}^*(2^{s+s_2}/\varepsilon)$.

Proof The above claim on the running time of `QueryA`(v,y,β) follows from inspection of Figure 3. Indeed, Step 1 takes $O(1)$ time and in Steps 2 to 3, there are $\mathrm{poly}^*(2^{s_2}/\varepsilon)$ many calls to `Query`, where each takes time $\mathrm{poly}^*(2^s/\varepsilon)$, by Theorem 14. ■

Definition 18 *Consider a construction of the data structure for* `Augmented-KDE`$(\mathsf{K},s_2,\Phi,\varepsilon,\delta)$*, and suppose we fix the randomness and let $\mathcal{V}$ be the set of all nodes in the tree rooted at v. For $y\in\mathbb{R}^d$, we consider the collection $\{\widehat{\boldsymbol{\zeta}}_{u,t}(y):u\in\mathcal{V}\}$, where $\widehat{\boldsymbol{\zeta}}_u(y)$ is the output of* `Query`$(u.\mathrm{ds},y)$ *(since we assumed* `Query`$(u.\mathrm{ds},y)$*, is deterministic, we don't require adding the additional parameters $t\in[T]$ in case it is called multiple times).*

Data Structure for `Augmented-KDE`$(\mathsf{K}, s_2, \Phi, \varepsilon, \delta)$

Preprocessing: The data structure preprocesses a set $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$, where each point has its associated weight $\alpha_1, \dots, \alpha_n \in \mathbb{R}$ and a weight $\mu_1, \dots, \mu_n$ (which after a partitioning step, we will assume are equal – see Subsection C.1).
Pointer: v will be the pointer to the root of a binary tree.

- The data structure is organized into a balanced binary tree of depth $O(\log n)$, where the n leaves correspond to the points of X stored in sorted order according to their weights $\alpha_1, \dots, \alpha_n$.
- Each node v of the binary tree maintains the following information:
  - A set $v.S \subset X$ in the subtree of v.
  - A pointer $v.\mathrm{ds}$ to a data structure for `KDE`$(\mathsf{K}, \Phi, \varepsilon, \delta\varepsilon^2/(O(n) \cdot 2^{O(s_2)})$ storing $v.S$,
  - Three numbers $v.\min, v.\max \in \mathbb{R}$ such that

$$
\begin{aligned}
v.\min &= \min \{\alpha_i : x_i \in v.S\}, \\
v.\max &= \max \{\alpha_i : x_i \in v.S\}, \\
v.\mathrm{med} &= \mathrm{median} \{\alpha_i : x_i \in v.S\}
\end{aligned}
$$

  - If $v.S$ contains more than one point, it has two children $v.\mathrm{LeftChild}$ and $v.\mathrm{RightChild}$. The left child $v.\mathrm{LeftChild}$ stores the points $x_i \in v.S$ where $\alpha_i \leq v.\mathrm{med}$ and the right child $v.\mathrm{RightChild}$ stores the points $x_i \in v.S$ where $\alpha_i > v.\mathrm{med}$.
- The algorithms `PreprocessA`(X, α) works by first building the balanced tree, and in the sorted order of α. Furthermore, for every internal node v we consider the dataset $v.S$ and execute `Preprocess`$(v.S)$ and store the data structure in $v.\mathrm{ds}$.

Figure 2: Data Structure for `Augmented-KDE`.

Algorithm `QueryA`(v, y, β)

Input: A pointer to a data structure v for `Augmented-KDE`$(\mathsf{K}, s_2, \Phi, \varepsilon, \delta)$, a point $y \in \mathbb{R}^d$, and a weight $\beta \in \mathbb{R}$.
Output: An estimate $\widehat{\boldsymbol{\xi}} \in \mathbb{R}_{\geq 0}$.

1. We first check whether $v.\max \leq \beta$. If so, then every weight $\alpha_i - \beta \leq 0$ and hence $(\alpha_i - \beta)^+ = 0$, so output $\widehat{\boldsymbol{\xi}} = 0$.

2. Otherwise, let $k = \left\lceil \log_2 \left(\frac{(v.\max - \beta)}{\varepsilon_0 \sigma r} \cdot \mathrm{poly}(nd\Phi 2^s/\varepsilon) \right) \right\rceil$ (which will become "hidden" in the notation $\mathrm{poly}^*(\cdot)$), and consider the $k+2$ indices $\sigma_0, \dots, \sigma_k \in [0, v.\max - \beta]$ where
$$\sigma_\ell = \begin{cases} 0 & \ell = 0 \\ \left(\frac{\varepsilon_0 \sigma r}{\mathrm{poly}(nd\Phi 2^s/\varepsilon)}\right) \cdot 2^{\ell-1} & \ell > 0 \\ v.\max - \beta & \ell = k+1 \end{cases},$$
and let $I_0, \dots, I_k$ be the disjoint and consecutive intervals $I_\ell = (\beta + \sigma_\ell, \beta + \sigma_{\ell+1}]$ which partition $(\beta, v.\max]$.

3. For each $\ell \in \{1, \dots, k\}$, and $t \in [T]$, for $T = 2^{O(s_2)}/\varepsilon^2$, we perform the following:
   - Sample $\boldsymbol{w}_{\ell,t} \sim [\sigma_\ell^{s_2}, \sigma_{\ell+1}^{s_2}]$ uniformly at random.
   - Let $\mathcal{V}_\ell$ be the set of all nodes u where $[u.\min, u.\max] \subset I_\ell$, and let $\boldsymbol{\mathcal{V}}_\ell(\boldsymbol{w}_{\ell,t}) = \{\boldsymbol{v}^{(1)}, \dots, \boldsymbol{v}^{(h)}\}$ be the minimal subset of $\mathcal{V}_\ell$ which satisfies
   $$(\boldsymbol{v}^{(1)}.S, \boldsymbol{v}^{(2)}.S, \dots, \boldsymbol{v}^{(h)}.S) \text{ partition } \left\{ x_i \in \Omega : \alpha_i \in I_\ell \text{ and } \alpha_i \geq \beta + \boldsymbol{w}_{\ell,t}^{1/s_2} \right\},$$
   and note that $h = O(\log n)$, and we may identify these nodes in $O(\log n)$ time.
   - For each $l \in [h]$, we execute `Query`$(\boldsymbol{v}^{(l)}.\mathrm{ds}, y)$ and let $\widehat{\boldsymbol{\zeta}}_{\ell,t,l}$ be its output, and let
   $$\widehat{\boldsymbol{\xi}}_{\ell,t} = \sum_{l=1}^{h} \widehat{\boldsymbol{\zeta}}_{\ell,t,l}.$$

4. We output
$$\widehat{\boldsymbol{\xi}} = \frac{1}{T} \sum_{\ell=1}^{k} \sum_{t=1}^{T} (\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}) \cdot \widehat{\boldsymbol{\xi}}_{\ell,t}.$$

Figure 3: Description for `QueryA` Algorithm.

Lemma 19 *With probability at least $1 - \delta/2$ over the construction of the data structures for* $\texttt{KDE}(\mathsf{K}, \Phi, \varepsilon, \delta)$*, every node $u \in \mathcal{V}$ satisfies*

$$(1-\varepsilon) \sum_{x \in u.S} \mathsf{K}(x,y) \le \widehat{\boldsymbol{\zeta}}_u(y) \le (1+\varepsilon) \sum_{x \in u.S} \mathsf{K}(x,y).$$

Proof We apply Theorem 14, and the fact that, in Figure 2, we've instantiated the data structures with failure probability $\delta\varepsilon^2/(O(n) \cdot 2^{O(s_2)})$, such that we can union bound over all nodes in $\mathcal{V}$. ■

The remainder of the argument proceeds by computing the expectation $\widehat{\boldsymbol{\xi}}$ over the randomness in $\{\boldsymbol{\xi}_{\ell,t} : \ell \in [h], t \in [T]\}$ as well as the variance. Specifically, we show that $\mathbf{E}[\widehat{\boldsymbol{\xi}}]$ satisfies the output guarantees, and that $\mathbf{Var}[\widehat{\boldsymbol{\xi}}] \le \varepsilon\, \mathbf{E}[\widehat{\boldsymbol{\xi}}]^2$, such that we can apply Chebyshev's inequality. Establishing the correctness of the estimate with high probability follows from a standard repetition argument.

Lemma 20 *Consider a fixed construction of the data structure, and suppose that the conclusion of Claim 19 holds (which occurs with probability at least $1 - \delta/2$). Then, for any fixed query $y \in \mathbb{R}^d$ with weight β, the expectation of $\widehat{\boldsymbol{\xi}}$ over the randomness in* $\texttt{QueryA}(v, y, \beta)$*,*

$$(1-\varepsilon)\sum_{i=1}^{n} ((\alpha_i - \beta)^+)^{s_2} \cdot \mathsf{K}(x_i, y) \le \mathbf{E}\left[\widehat{\boldsymbol{\xi}}\right] \le (1+\varepsilon)\sum_{i=1}^{n} ((\alpha_i - \beta)^+)^{s_2} \cdot \mathsf{K}(x_i, y).$$

Proof We have that for any $\ell \in \{1, \dots, k\}$ and $t \in [T]$,

$$\begin{aligned} \widehat{\boldsymbol{\xi}}_{\ell,t} = \sum_{u \in \mathcal{V}_\ell} \mathbf{1}\{u \in \mathcal{V}_\ell(\boldsymbol{w}_{\ell,t})\} \cdot \widehat{\boldsymbol{\zeta}}_u(y) &\le (1+\varepsilon) \sum_{u \in \mathcal{V}_\ell} \sum_{x \in u.S} \mathbf{1}\{u \in \boldsymbol{\mathcal{V}}_\ell(\boldsymbol{w}_{\ell,t})\} \cdot \mathsf{K}(x,y) \\ &= (1+\varepsilon) \sum_{x \in X} \mathbf{1}\{\exists u \in \boldsymbol{\mathcal{V}}_\ell(\boldsymbol{w}_{\ell,t}), x \in u.S\} \cdot \mathsf{K}(x,y), \end{aligned} \tag{14}$$

where in the second line, we used that every $x \in X$ which appears in some $u \in \boldsymbol{\mathcal{V}}_\ell(\boldsymbol{w}_{\ell,t})$ appears at most once. Similarly,

$$\widehat{\boldsymbol{\xi}}_{\ell,t} \ge (1-\varepsilon) \sum_{x \in X} \mathbf{1}\{\exists u \in \boldsymbol{\mathcal{V}}_\ell(\boldsymbol{w}_{\ell,t}), x \in u.S\} \cdot \mathsf{K}(x,y). \tag{15}$$

Then, for every $x_i \in X$ with weight α_i,

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\boldsymbol{w}_{\ell,t} \sim [\sigma_\ell^{s_2}, \sigma_{\ell+1}^{s_2}]} [\exists u \in \boldsymbol{\mathcal{V}}_\ell(\boldsymbol{w}_{\ell,t}), x_i \in u.S] &= \mathop{\mathbf{Pr}}_{\boldsymbol{w}_{\ell,t}} \left[\beta + \boldsymbol{w}_{\ell,t}^{1/s_2} \le \alpha_i \le \beta + \sigma_{\ell+1}\right] \\ &= \mathbf{1}\{\alpha_i \in I_\ell\} \cdot \frac{((\alpha_i - \beta)^+)^{s_2}}{\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}}. \end{aligned} \tag{16}$$

In particular, we may upper and lower bound the expectation of $\widehat{\boldsymbol{\xi}}$ over the randomness in drawing $\boldsymbol{w}_{\ell,t}$ by plugging (16) into (14) and (15). Namely, we first note that for a fixed $\ell \in \{1, \dots, k\}$, all the draws from $t \in [T]$ of $\boldsymbol{w}_{\ell,t}$ are identically distributed, so we may simplify

$$\mathbf{E}\left[\widehat{\boldsymbol{\xi}}\right] = \sum_{\ell=1}^{k} \left(\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}\right) \mathop{\mathbf{E}}_{\boldsymbol{w}_{\ell,t}} \left[\widehat{\boldsymbol{\xi}}_{\ell,t}\right].$$

Then, we have that (16) and (14) implies

$$\left(\sigma_{\ell+1}^{s_2} - \sigma_{\ell}^{s_2}\right) \mathop{\mathbf{E}}_{\boldsymbol{w}_{\ell,t}} \left[\widehat{\boldsymbol{\xi}}_{\ell,t}\right] \leq (1+\varepsilon) \sum_{i=1}^{n} \mathbf{1}\{\alpha_i \in I_\ell\} \cdot ((\alpha_i - \beta)^+)^{s_2} \cdot \mathsf{K}(x_i, y).$$

The lower bound proceeds similarly, expect we plug (16) into (15). In particular, since there is no $i \in [n]$ where $\alpha_i \in I_0$, once $\ell \in \{1, \dots, k\}$, we cover the entire interval $(\beta, v.\max]$. ■

Lemma 21 *Consider a fixed construction of the data structure, and suppose that the conclusion of Claim 19 holds (which occurs with probability at least $1 - \delta/2$). Then, for any fixed query $y \in \mathbb{R}^d$ with weight β, the variance of $\widehat{\boldsymbol{x}}$ over the randomness in* `QueryA`(v, y, β) *satisfies*

$$\mathbf{Var}\left[\widehat{\boldsymbol{\xi}}\right] \leq \varepsilon \left(\mathbf{E}\left[\widehat{\boldsymbol{\xi}}\right]\right)^2.$$

Proof For various settings of $\ell \in \{1, \dots, k\}$ and $t \in [T]$, the draws of $\boldsymbol{w}_{\ell,t}$ are independent, so that we may write

$$\mathbf{Var}\left[\widehat{\boldsymbol{\xi}}\right] = \sum_{\ell=1}^{k} \frac{1}{T} \cdot \mathbf{Var}\left[\widehat{\boldsymbol{\xi}}_{\ell,t}\right],$$

and it suffices to upper bound that. We note that using the same upper bound in (14),

$$\begin{aligned} &\mathbf{Var}[\widehat{\boldsymbol{\xi}}_{\ell,t}] \leq \mathbf{E}[\widehat{\boldsymbol{\xi}}_{\ell,t}^2] \\ &\quad \leq (1+\varepsilon)^2 \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}\{\alpha_i, \alpha_j \in I_\ell\} \cdot \mathsf{K}(x_i, y) \cdot \mathsf{K}(x_j, y) \cdot \mathop{\mathbf{Pr}}_{\boldsymbol{w}_{\ell,t}} \left[\beta + \boldsymbol{w}_{\ell,t}^{1/s_2} \leq \min\{\alpha_i, \alpha_j\}\right]. \end{aligned} \tag{17}$$

Suppose first that $\ell > 0$. Then if $\alpha_i, \alpha_j \in I_\ell$, then $(\max\{\alpha_i, \alpha_j\} - \beta)^{s_2} \geq \sigma_\ell^{s_2}$,

$$\begin{aligned} \mathop{\mathbf{Pr}}_{\boldsymbol{w}_{\ell,t}} \left[\beta + \boldsymbol{w}_{\ell,t}^{1/s_2} \leq \min\{\alpha_i, \alpha_j\}\right] &\leq \frac{((\min\{\alpha_i, \alpha_j\} - \beta_j)^+)^{s_2}}{\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}} \cdot \frac{((\max\{\alpha_i, \alpha_j\} - \beta)^+)^{s_2}}{\sigma_\ell^{s_2}} \\ &\leq \left(\frac{((\alpha_i - \beta)^+)^{s_2}}{\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}} \cdot \frac{((\alpha_i - \beta)^+)^{s_2}}{\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}}\right) \cdot \left(\frac{\sigma_{\ell+1}^{s_2} - \sigma_\ell^{s_2}}{\sigma_\ell^{s_2}}\right) \\ &\leq 2^{s_2} \cdot \mathop{\mathbf{Pr}}_{\boldsymbol{w}_{\ell,t}} \left[\beta + \boldsymbol{w}_{\ell,t}^{1/s_2} \leq \alpha_i\right] \mathop{\mathbf{Pr}}_{\boldsymbol{w}_{\ell,t}} \left[\beta + \boldsymbol{w}_{\ell,t}^{1/s_2} \leq \alpha_j\right]. \end{aligned} \tag{18}$$

Plugging (18) into (17), we have that every $\ell \in \{1, \dots, k\}$ satisfies

$$\mathbf{Var}\left[\widehat{\boldsymbol{\xi}}_{\ell,t}\right] \leq (1+\varepsilon)^2 \cdot 2^{s_2} \left(\mathbf{E}[\widehat{\boldsymbol{\xi}}_{\ell,t}]\right)^2.$$

By the setting of T, we obtain the desired bound. ■

C.2. Proof of Lemma 10 and Lemma 11

We briefly describe `Est-Beta` (the case of `Est-Alpha` is a symmetric argument, by replacing α's and β's, as well as changing the signs). The intuition is the following: we initialize a data structure for `Augmented-KDE` with the kernel K in (13) and the parameter $s_2 = s - 1$. This is done so that for every $i \in [n]$ and $j \in [m]$,

$$f_{ij}(\alpha_i, \beta_j) = \left(1 - \frac{1}{s}\right)^{s-1} \cdot \mu_i \left((\alpha_i - \beta_j)^+\right)^{s_2} \cdot \mathsf{K}(x_i, y_j).$$

Thus, we preprocess the data structure with the dataset $\{x_1, \dots, x_n\}$ and weights $\alpha \in \mathbb{R}^n$ and $\mu \in \mathbb{R}^n$. Then, we will iterate through each $j \in [m]$, and we query the data structure with y_j and β_j to obtain $\boldsymbol{\xi}_j$.

C.3. Proof of Lemma 12

The algorithm `Est-Penalty` also uses Theorem 16. We initialize the kernel function $\mathsf{K} \colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ as in (13), but with $s_2 = s$. Since $\rho = s/(s-1)$, we've set things up so

$$\begin{aligned}(1 - \varepsilon_0)\left(f_{ij}(\alpha_i, \beta_j) \cdot \|x_i - y_j\|_2\right)^{\rho} &\leq \left(1 - \frac{1}{s}\right)^{s_2} \cdot ((\alpha_i - \beta_j)^+)^s \cdot \mathsf{K}(x_i, y_j) \\ &\leq \left(f_{ij}(\alpha_i, \beta_j) \cdot \|x_i - y_j\|_2\right)^{\rho},\end{aligned}$$

and thus `Est-Penalty`$(\alpha, \beta, \varepsilon, \delta)$ needs to approximate

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \mu_i \nu_j \cdot ((\alpha_i - \beta_j)^+)^{s_2} \cdot \mathsf{K}(x_i, y_j).$$

This is a simple application of Theorem 16. We preprocess a data structure v for `Augmented-KDE`$(\mathsf{K}, s, \Phi, \varepsilon/2, \delta)$ with the dataset $\{x_1, \dots, x_n\}$ and weights α, and we query it for each $y_1, \dots, y_m$ with the weights $\beta \in \mathbb{R}^m$. If each estimate is $\widehat{\boldsymbol{\xi}}_j$, the desired output estimate is given by $\sum_{j=1}^{m} \widehat{\boldsymbol{\xi}}_j$.