SUCCINCT DATA STRUCTURES FOR APPROXIMATING CONVEX FUNCTIONS, WITH APPLICATIONS[†]

Prosenjit Bose[‡] Luc Devroye[§] Pat Morin[‡]

ABSTRACT. We study data structures for providing ε-approximations of convex functions whose slopes are bounded from above and below by n and $-n$, respectively. The structures we describe have size $O((1/\varepsilon)\log n)$ and can answer queries in $O(\log(1/\varepsilon)+\log\log n)$ time. We also give an information-theoretic lower-bound, that shows it is impossible to obtain structures of size $O(1/\varepsilon)$ for approximating this class of convex functions. Finally, we show that our structures have applications to efficiently solving problems in clustering and facility location.

1 Introduction

We consider the problem of approximating convex functions of one variable whose slopes are bounded. We say that a non-negative number y is an ε-approximation to a non-negative number x if $(1-\varepsilon)x \le y \le x$.[1] We say that a function g is an ε-approximation to a function f if $g(x)$ is an ε-approximation to $f(x)$ for all x in the domain of f.

Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function that is non-negative everywhere. In this paper we show that, if the absolute value of the slope of f is bounded above by n, then there exists a piecewise-linear function g that ε approximates f at all points x except where the slope of f is small (less than 1) and that consists of $O(\log_E n)$ pieces, where $E = 1/(1-\varepsilon)$. The function g can be computed in $O(K\log_E n)$ time, where K is the time it takes to evaluate expressions of the form $\sup\{x : f'(x) \le t\}$ and f' is the first derivative of f. Once we have computed the function g, we can store the pieces of g in an array sorted by x values so that we can evaluate $g(x)$ for any query value x in $O(\log_2\log_E n)$ time. Since we are interested in the joint complexity as a function of ε and n, it is worth noting that $\log_E x = \Theta((1/\varepsilon)\log x)$ (as can be seen by taking the limit $\lim_{\varepsilon\to 0+}(\varepsilon/\log E)$ using one application of L'Hôpital's Rule). Thus, $\log_E n = \Theta((1/\varepsilon)\log n)$ and $\log\log_E n = \Theta(\log(1/\varepsilon)+\log\log n)$.

As an application of these results, we consider functions defined by sums of Euclidean distances in d dimensions and show that they can be approximated using the above results. To achieve this, we use the technique of random projections [6, 7]. We show that the sum of Euclidean distances from a point to a set of n points can be closely approximated by many sums of 1-dimensional distances from the point to the set, both in a probabilistic and a worst-case sense. This technique is very simple and of independent interest.

[†]This research was partly supported by NSERC.

[‡]School of Computer Science, Carleton University, {jit,morin}@scs.carleton.ca

[§]School of Computer Science, McGill University, luc@cgm.cs.mcgill.ca

[1]This definition is a bit more one-sided than the usual definition, which allows any y such that $|x-y| \le \varepsilon x$. We use this definition because it leads to simpler calculations later on, and the results extend immediately to the (less restrictive) standard definition.

The remainder of the paper is organized as follows. Section 2 presents our result on approximating convex functions using few linear pieces. Section 3 discusses how these results can be interpreted in terms of data structures for approximating convex functions. Section 4 gives lower bounds on the space complexity of approximating convex functions. Section 5 describes applications of this work to facility location and clustering problems.

2 Approximating Convex Functions

Let $h(x) = c+|nx|$, for some $c, n \geq 0$. Then, it is clear that the function g such that $g(x) = c+(1-\varepsilon)|nx|$ is an ε-approximation of h. Furthermore, g is an ε-approximation for any function h_2 such that $g(x) \leq h_2(x) \leq h(x)$ for all $x \in \mathbb{R}$. (see Fig. 1). This trivial observation is the basis of our data structure for approximating convex functions.

Let f be a non-negative convex function and let f' be the first derivative of f. Assume that $f'(x)$ is defined for all but a finite number of values of x and that $|f'(x)| \leq n$ for all x in the domain of f'. For convenience, we define the *right derivative* $f^*(x)$ as follows: If $f'(x)$ is defined, then $f^*(x) = f'(x)$. Otherwise, $f^*(x) = \lim_{\delta \to 0+} f'(x+\delta)$. To avoid overly long qualifications we will abuse standard terminology slightly and call $f^*(x)$ the *slope* of f at x.

Let a be the largest value at which the slope of f is at most $-(1-\varepsilon)n$, i.e.,

$$a = \max\{x : f^*(x) \leq -(1-\varepsilon)n\} \ .$$

(Here, and throughout, we use the convention that $\max\emptyset = -\infty$ and $\min\emptyset = \infty$.) Similarly, let $b = \min\{x : f^*(x) \geq (1-\varepsilon)n\}$. Then, from the above discussion, it is clear that the function

$$g(x) = \begin{cases} f(a) + (1-\varepsilon)(a-x)n & \text{if } x \leq a \\ f(b) + (1-\varepsilon)(x-b)n & \text{if } x \geq b \\ f(x) & \text{otherwise} \end{cases} \tag{1}$$

is an ε-approximation of f (see Fig. 2).

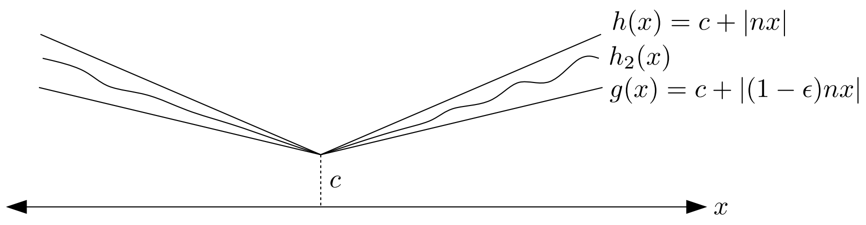
Equation (1) tells us that we can approximate f by using two linear pieces and then recursively approximating f in the range (a, b). However, in the range (a, b), f^* is in the range $(-(1-\varepsilon)n, (1-\varepsilon)n)$. Therefore, if we recurse $\lceil \log_E n \rceil$ times, we obtain a function g with $O(\log_E n) = O((1/\varepsilon)\log n)$ linear pieces that approximates f at all points except possibly where f^* is less than 1.

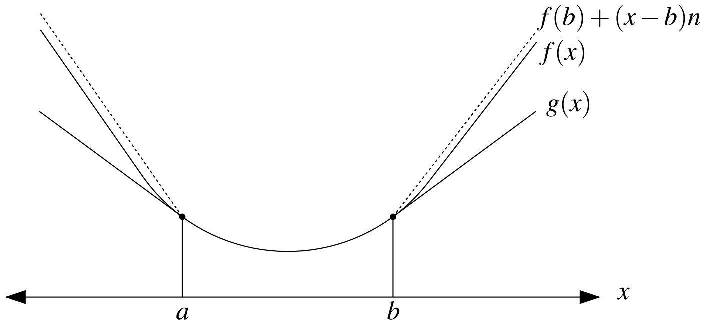
Theorem 1. *Let f and f^* be defined as above. Then there exists a piecewise-linear function $g(x)$ with $O((1/\varepsilon)\log n)$ pieces that is an ε-approximation to $f(x)$ at all values where $|f^*(x)| \geq 1$.*

For some special classes of convex functions (for example, the *extent* of a line arrangement [1]) it is possible to find ε-approximations whose size is only a function ε, and is independent of n. As we will see in Section 4, this is not the case for the general class of convex functions we consider here.

3 Data Structures

In this section, we consider the consequences of Theorem 1 in terms of data structures for approximating convex functions. By storing the pieces of g in an array sorted by x values, we obtain the following.


$h(x) = c + |nx|$
$h_2(x)$
$g(x) = c + |(1-\epsilon)nx|$
c
x


$f(b)+(x-b)n$
$f(x)$
$g(x)$
x
a
b

Theorem 2. *Let f and f^* be defined as in Section 2. Then there exists a data structure of size $O((1/\varepsilon)\log n)$ that can compute an ε-approximation to $f(x)$ in $O(\log(1/\varepsilon) + \log\log n)$ time for any query value x where $|f^*(x)| \geq 1$.*

Next, we consider a more dynamic model, in which the function f is updated over time. In particular, we consider the following operations that are applied to the initial function $f(x) = 0$, for all $x \in \mathbb{R}$.

1. Query(x): Return an ε-approximation to $f(x)$.
2. Insert(a): Increase the slope of f by 1 in the range (a, ∞), i.e., set $f(x) \leftarrow f(x) + x - a$ for all $x \in [a, \infty)$.
3. Delete(x): Decrease the slope of f by 1 in the range (x, ∞). In order to maintain convexity, the number of calls to Delete(x) may not exceed the number of calls to Insert(x) for any value of x.

Note that a sequence of Insert and Delete operations can only produce a monotonically increasing function f whose slopes are all integers. This is done to simplify the exposition of the data structure. If an application requires that f be allowed to decrease and increase then two data structures can be used and their results summed.

The function f has some number $m \leq n$ of breakpoints, where the slope of f changes. We store these breakpoints in a balanced search tree T, sorted by x-coordinate. With each breakpoint x, we also maintain the value $\Delta(x)$ by which the slope of f increases at x. In addition, we link the nodes of T in a doubly-linked list, so that the immediate successor and predecessor of a node can be found in constant time. It is clear that T can be maintained in $O(\log n)$ time per operation using any balanced search tree data structure.

In addition to the search tree T, we also maintain an array A of size $O((1/\varepsilon)\log n)$ that contains the piecewise linear approximation of f. The ith element in this array contains the value x_i such that $x_i = \min\{x : f^*(x) \geq E^i\}$, a pointer to the node in T that contains x_i, and the values of $f(x_i)$ and $f^*(x_i)$, i.e., the value of f at x_i and slope of f at x_i. To update this array during an Insert or Delete operation, we first update the values of $f(x_i)$ and $f^*(x_i)$ for each i. Since there are only $O((1/\varepsilon)\log n)$ array entries, this can be done in $O((1/\varepsilon)\log n)$ time.

Next, we go through the array again and check which values of x_i need to be changed (recall that $x_i = \min\{x : f^*(x) \geq E^i\}$). Note that, since Insert or Delete can only change the value of $f^*(x)$ by 1, if the value of x_i changes then it changes only to its successor or predecessor in T. Since the nodes of T are linked in a doubly-linked list, and we store the values of $f(x_i)$ and $f^*(x_i)$ we can detect this and update the value of x_i, $f(x_i)$ and $f^*(x_i)$ in constant time. Therefore, over all array entries, this takes $O((1/\varepsilon)\log n)$ time.

To evaluate an approximation to $f(x)$, we do a binary search on A to find the index i such that $[x_i, x_{i+1})$ contains x and then output $f(x_i) + (x - x_i)f^*(x_i)$. By the choice of x_i, this is a ε-approximation to $f(x)$.

Thus far, we have described a data structure whose size is $O(n + (1/\varepsilon)\log n)$. However, if $(1/\varepsilon)\log n > n$ then a simpler data structure, namely the one that stores the function f explicitly in an array achieves the same update and search time bounds and requires only $O(n)$ space. This completes the proof of:

Theorem 3. *There exists a data structure of size $O(n)$ that supports the operations* Insert*,* Delete *in $O((1/\varepsilon)\log n)$ time and* Query *in $O(\log(1/\varepsilon)+\log\log n)$ time, where n is the maximum slope of the function f being maintained.*

Remark: In some cases, it may be desirable to maintain functions that have arbitrarily small or large changes in their slopes. That is, the operations Insert and Delete come with an extra parameter δ that specifies by how much to increase or decrease the slope. In this case, we can still use a similar approach, but the update time bound becomes $O((1/\varepsilon)\log n\log m)$, where m is the number of locations at which the slope changes. This running time comes from the fact that, when a relatively large value of δ is used, the locations of the x_i's may change by a lot, and each of these $O((1/\varepsilon)\log n)$ values requires a search in a tree of size m.

4 A Lower Bound on Storage

Recently, several approximation schemes have been presented in which the size of the data structure is a function only of ε and not of any of the other input parameters [1, 6]. In this section, we show that such a result is not possible in our setting. Indeed, the amount of space required is a function of the maximum slope n, and the maximum value X at which the slope is allowed to increase.

For $i=1,\dots,M$ let $x_i=((1-\varepsilon)/\varepsilon)^i$, let $D=1/(1-2\varepsilon)$ and let $m=\lfloor\log_D n\rfloor$. Let $\{j_1,\dots,j_m\}$ be a subset of $\{1,\dots,M\}$. We will show that $j_1,\dots,j_m$ can be encoded as a piecewise linear non-decreasing convex function f whose maximum slope is bounded by n and whose pieces begin and end only at the elements of $\{x_1,\dots,x_M\}$. Furthermore, this encoding is robust in the sense that, given only an ε-approximation g of f we can use to g to recover the subset of $\{1,\dots,M\}$ used to define f. Thus, we conclude that any data structure must use $\log\binom{M}{m}\approx(1/\varepsilon)\log n\log M$ bits of storage.

We define the convex function f as follows:

1. For $x\in[-\infty,0)$, $f(x)=0$.
2. For $x\in(x_{j_i},x_{j_{i+1}})$, $f(x)$ has slope D^j.
3. For $x>j_m$, $f(x)$ has slope n.

Next we show how, given only an approximation g of f, we can completely reconstruct the function f. Suppose we have already reconstructed the values $f(x_1),f(x_2),\dots,f(x_i)$ and we now want to reconstruct the value $f(x_{i+1})$. More precisely, we have already determined the slope D^j of $f(x)$ for $x\in(x_{i-1},x_i)$ and we need to decide whether the slope of $f(x)$ for $x\in(x_i,x_{i+1})$ is D^j or D^{j+1}. If the new slope is D^j then, because f and f^* are non-decreasing and $f(0)=0$, we have

$$\begin{aligned} g(x_{i+1}) &\leq f(x_{i+1}) \\ &< x_{i+1}D^j \ . \end{aligned}$$

On the other hand, if the new slope is D^{j+1} then

$$\begin{aligned}
g(x_{i+1}) &\geq (1-\varepsilon)f(x_{i+1}) \\
&= (1-\varepsilon)(f(x_i) + (x_{i+1}-x_i)D^{j+1}) \\
&\geq (1-\varepsilon)(x_{i+1}-x_i)D^{j+1} \\
&= \left(\frac{1-\varepsilon}{1-2\varepsilon}\right)(x_{i+1}-x_i)D^j \\
&= \left(\frac{1-\varepsilon}{1-2\varepsilon}\right)\left(\left(\frac{1-\varepsilon}{\varepsilon}\right)x_i - x_i\right)D^j \\
&= \left(\frac{1-\varepsilon}{\varepsilon}\right)x_i D^j \\
&= x_{i+1}D^j \ .
\end{aligned}$$

Thus, we can determine if the slope (and hence the value) of $f(x)$ for $x \in (x_i, x_{i+1})$ by testing if $g(x_{i+1}) < x_{i+1}D^j$. We conclude that we can reconstruct the entire function f, and therefore the values $j_1, \ldots, j_m$, given only a function g that is an ε-approximation to f.

Theorem 4. *Let $m = \lfloor \log_E n \rfloor$. Any data structure that can represent an ε-approximation to an increasing, piecewise-linear convex function whose slopes are integers in the range $[0, n]$ and whose slope changes only at integers in the range $[0, ((1-\varepsilon)/\varepsilon)^M]$ requires $\log \binom{M}{m}$ bits of storage, in the worst case.*

Remark 1. Some readers may complain that the function used in our lower bound construction uses linear pieces whose lengths are exponential in M, so representing a single such value requires $\Omega(M)$ bits. However, all these values are powers of $(1-\varepsilon)/\varepsilon$ and can therefore be encoded using $O(\log M)$ bits each using, e.g., a floating point representation.

5 Applications

Next, we consider applications of our approximation technique for convex functions to the problem of approximating sums of distances in d dimensions. Let S be a set of n points in d dimensions. The *Fermat-Weber weight* of a point $q \in \mathbb{R}^d$ is

$$\mathrm{FW}(p) = \sum_{q \in S} \|pq\| \ ,$$

where $\|pq\|$ denotes the distance between points p and q. Of course, different definitions of distance (e.g., Euclidean distance, Manhattan distance) yield different Fermat-Weber weights.

5.1 The 1-dimensional Case

One setting in which distance is certainly well defined is in one dimension. In this case,

$$\|pq\| = |p - q| \ ,$$

so the Fermat-Weber weight of x is given by

$$\mathrm{FW}(x) = f(x) = \sum_{y \in S} |x - y| \ .$$

Note that the function f is convex (it is the sum of n convex functions) and has slopes bounded below by $-n$ and above by n, so it can be approximated using the techniques Section 3. Furthermore, adding or removing a point p to/from S decreases the slope of f by 1 in the range $(-\infty, p)$ and increases the slope of f by 1 in the range (p, ∞), so the dynamic data structure of the previous section can be used to maintain an ε-approximation of f in $O(\log_E n) = O((1/\varepsilon)\log n)$ time per update.

Given the set S, constructing the ε-approximation for f can be done in $O(n/\varepsilon)$ time by a fairly straightforward algorithm: Using a linear-time selection algorithm, one finds the elements of S with ranks $\lfloor \varepsilon n/2 \rfloor$ and $\lceil (1-\varepsilon/2)n \rceil$. These are the values a and b in (1). Once this is done, the remaining problem has size $(1-\varepsilon)n$ and is solved recursively. Although some care is required to compute the values $f(a)$ and $f(b)$ at each stage, the details are not difficult and are left to the interested reader.

Remark 2. A general (and surprising) result of Agarwal and Har-Peled [1] implies that the Fermat-Weber weight of points in one dimension can actually be ε-approximated by a piecewise-linear function with $O(1/\varepsilon)$ pieces, independent of n. However, it is not clear how easily this approach can be made dynamic to handle insertion and deletions of points. The data structure of the previous section is also (arguably) considerably simpler.

5.2 The Manhattan Case

The Manhattan distance between two points p and q in $\mathbb{R}^d$ is

$$\|pq\|_1 = \sum_{i=1}^{d} |p_i - q_i| \ ,$$

where p_i denotes the ith coordinate of point p. We simply observe that Manhattan distance is the sum of d 1-dimensional distances, so the Fermat-Weber weight under the Manhattan distance can be approximated using d one-dimensional data structures.

5.3 The Euclidean Case

The Euclidean distance between two points p and q in $\mathbb{R}^d$ is

$$\|pq\|_2 = \left(\sum_{i=1}^{d} (p_i - q_i)^2 \right)^{1/2} \ .$$

A general technique used to approximate Euclidean distance is to use a polyhedral distance function, in which the unit sphere is replaced with a polyhedron that closely resembles a sphere. For example, the Manhattan distance function is a polyhedral distance function in which the unit sphere is replaced with a unit hypercube. Although this technique works well when d is small, such metrics generally require a polyhedron with a number of vertices that is exponential in d, making them less well-suited for high dimensional applications.

Another technique, that works well when d is very large (greater than $\log n$), and for many distance functions, is that of random projections [6]. Here, a random $O(\log n)$-flat is chosen and the points of S are projected orthogonally onto this flat. With high probability, all interpoint distances are faithfully preserved after the projection, so the problem is reduced to one in which the dimension of the point set is $O(\log n)$ (see Ref. [6, Lemma 3] for details). A related technique is to project the points onto a set of randomly chosen lines and use the sum of distances in these projections as an estimate of the interpoint distance [7]. The drawback of these techniques for Fermat-Weber type problems is that they only guarantee that each individual point in $\mathbb{R}^d$ has its Fermat-Weber weight ε-approximated with high probability. This makes them less well-suited for use in search algorithms whose correctness relies on access to an ε-approximation for the Fermat-Weber weight of every point (or many points) in $\mathbb{R}^d$.

In this section we show that both of the above strategies can be used in conjunction with our approximation scheme for convex functions.

5.3.1 A Randomized Approximation

Here we describe a technique based on Kleinberg's method of projecting onto random lines [7]. Consider the following experiment: Pick a random point v on the surface of the unit sphere $\mathbb{S}^{d-1}$ in $\mathbb{R}^d$ and consider the line ℓ through the origin and v. For two points $p, q \in \mathbb{R}^d$, project p and q orthogonally onto ℓ to obtain points p_ℓ and q_ℓ, respectively. Consider the random variable $|p_\ell - q_\ell|$ that measures the distance between the two projected points. Then $\mathbf{E}[|p_\ell - q_\ell|] = c_d\|pq\|_2$, where $c_d = \Theta(1/\sqrt{d})$ is a constant whose value is discussed in Appendix A.

Let $f(p)$ denote the Fermat-Weber weight of p under the Euclidean distance function. Choose k random lines $\ell_1, \ldots, \ell_k$ as described above and let $f_i(p)$ denote the Fermat-Weber weight of the 1-dimensional point p_{ℓ_i} in the 1-dimensional set S_{ℓ_i}. That is,

$$f_i(p) = \sum_{q \in S} |p_{\ell_i} - q_{\ell_i}| \ .$$

Then $f_i(p)$ is a random variable that may take on any value in the range $[0, c_d f(p)]$. In particular, $f_i(p)$ has an expected value

$$\mathbf{E}[f_i(p)] = c_d f(p) \ .$$

Next consider the normalized average function

$$g(p) = \frac{1}{kc_d} \times \sum_{i=1}^{k} f_i(p)$$

that approximates the Fermat-Weber weight under Euclidean distance.

Lemma 1. $\Pr\{|g(p) - f(p)| \geq \delta f(p)\} = \exp(-\Omega(\delta^2 k))$

Proof. The value of $g(p)$ is a random variable whose expected value is $f(p)$ and it is the sum of k independent random variables, all of which are in the range $[0, f(p)/c_d]$. Applying Hoeffding's inequality[2] and simplifying yields the desired result. □

[2]Hoeffding's Inequality [4]: Let $S_n = \sum_{i=1}^n X_i$ where $X_1, \ldots, X_n$ are bounded independent bounded random variables such that $X_i \in [a_i, b_i]$ with probability 1. Then, for any $t > 0$, $\Pr\{S_n - \mathbf{E}[S_n] \geq t\} \leq e^{-2t^2/\sum_{i=1}^n (b_i - a_i)^2}$ and $\Pr\{S_n - \mathbf{E}[S_n] \leq -t\} \leq e^{-2t^2/\sum_{i=1}^n (b_i - a_i)^2}$.

In summary, $g(p)$ is a δ-approximation of $f(p)$ with probability $1 - e^{-\Omega(\delta^2 k)}$. Furthermore, $g(p)$ is the sum of k 1-dimensional Fermat-Weber weights which can be ε-approximated using the results of Section 3. This gives a data structure of size $O(k(1/\varepsilon)\log n)$ that can be constructed in time $O(kn)$. Queries in the data structure take $O(k(\log(1/\varepsilon) + \log\log n))$ time and return an $(\varepsilon + \delta)$-approximation to the Fermat-Weber weight of any point $p \in \mathbb{R}^d$ with probability at least $1 - e^{-\Omega(\delta^2 k)}$.

5.3.2 A Deterministic Approximation

Next we show how to derandomize the data structure of the previous section to obtain a data structure that guarantees an ε-approximation to $f(p)$ for *every* point $p \in \mathbb{R}^d$. We do this by replacing the random points used in the previous construction with k points evenly distributed on $\mathbb{S}^{d-1}$. In particular, in Appendix B we show how to find $k = O((d^3/\delta^4)\log(d/\delta))$ points on $\mathbb{S}^{d-1}$ to obtain a set of lines $\ell_1, \ldots, \ell_k$ such that

$$(1-\delta)|pq| \leq \frac{1}{kc_d}\sum_{i=1}^{k} |p_{\ell_i} - q_{\ell_i}| \leq (1+\delta)|pq|$$

for every $p, q \in \mathbb{R}^d$.

Thus, we can maintain a δ-approximation to the Fermat-Weber function f by maintaining k 1-dimensional data structures. To summarize, this gives us a data structure of size $O(((d^3/\delta^4)\log(d/\delta))(1/\varepsilon)\log n)$ that can be constructed in $O(((d^3/\delta^4)\log(d/\delta)n)$ time and that can return an $(\varepsilon + \delta)$-approximation to $f(p)$ for any $p \in \mathbb{R}^d$ in $O(((d^3/\delta^4)\log(d/\delta))(\log(1/\varepsilon) + \log\log n))$ time.

5.4 Clustering and Facility Location

Bose *et al.* [2] describe data structures for approximating sums of distances. They show how, for fixed d and ε, to build a data structure in $O(n\log n)$ time that can $(1-\varepsilon)$-approximate the Fermat-Weber weight of any point in $O(\log n)$ time. These results have a leading constant of the form $\Omega((1/\varepsilon)^d)$.

The same authors give applications of their data structures to a number of facility-location and clustering problems, including evaluation of the Medoid and AverageDistance clustering measures, the Fermat-Weber problem, the constrained Fermat-Weber problem, and the constrained obnoxious facility-location problem. All of these applications also work with the data structure of Section 3, many with improved running times.

A summary of these results is given in Table 1, which shows running times assuming ε and d are fixed constants, independent of n. The results in the right-hand column are obtained simply by using the methods described by Bose *et al.* in combination with the deterministic data structure of Section 5.3.2. It is worth noting that the leading constant in the algorithms obtained this way is only $O((d^3/\varepsilon^4)\log(d/\varepsilon))$, while the leading constant in the algorithm by Bose *et al.* is $\Omega((1/\varepsilon)^d)$.

References

[1] P. K. Agarwal and S. Har-Peled. Maintaining the approximate extent measures of moving points. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 148–157,

Problem	Previous ε-approx.	Ref.	New ε-approx.
Average distance	[a] $O(n)$ $O(n \log n)$	[2, 5]	$O(n \log\log n)$
Medoid (1-Median)	[a] $O(n)$ $O(n \log n)$	[2, 5]	$O(n \log\log n)$
Discrete Fermat-Weber	[a] $O(n)$ $O(n \log n)$	[2, 5]	$O(n \log\log n)$
Fermat-Weber	[b] $O(n)$ $O(n \log n)$	[2, 5]	$O(n)$
Constrained Fermat-Weber	[b] $O(n)$ $O(n \log n)$	[2, 5]	$O(n)$
Constrained OFL	[a] $O(n)$ $O(n \log n)$	[2, 5, 7]	$O(n \log\log n)$

[a]The first running time refers to a randomized algorithm that outputs a $(1-\varepsilon)$-approximation with constant probability.

[b]The first running time refers to a randomized algorithm that outputs a $(1-\varepsilon)$-approximation with high probability, i.e., with probability $1 - n^{-c}$, for some $c > 0$.

Table 1: Applications of the data structure for evaluating the Fermat-Weber weights of points under the Euclidean distance function.

2001.

[2] P. Bose, A. Maheshwari, and P. Morin. Fast approximations for sums of distances, clustering and the Fermat-Weber problem. *Computational Geometry: Theory and Apllications*, 24:135–146, 2002.

[3] B. Chazelle. *The Discrepancy Method: Randomness and Complexity.* Cambridge University Press, New York, 2000.

[4] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[5] P. Indyk. Sublinear time algorithms for metric space problems. In *Proceedings of the 31st ACM Symposium on Theory of Computing (STOC'99)*, 1999.

[6] P. Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 10–33, 2001.

[7] J. Kleinberg. Two algorithms for nearest neighbour search in high dimensions. In *Proceedings of the 29th ACM Symposium on Theory of Computing (STOC'97)*, pages 599–608, 1997.

[8] D. S. Mitrinovic. *Analytic Inequalities.* Springer, New York, 1970.

A The value of $\mathbf{c_d}$

The value of c_d that appears in Section 5 is given by

$$c_d = \mathbf{E}\left[|x_1|\right] \ ,$$

where $(x_1, \ldots, x_d)$ is a point taken from the uniform distribution on the unit sphere $\mathbb{S}^{d-1}$ in $\mathbb{R}^d$. We observe that $(x_1^2, \ldots, x_d^2)$ is distributed as

$$\left(\frac{N_1^2}{N^2}, \ldots, \frac{N_d^2}{N^2}\right) \ ,$$

where $N^2 = \sum_{i=1}^d N_i^2$ and $(N_1, \ldots, N_d)$ are i.i.d. normal$(0,1)$. Clearly,

$$\frac{N_1^2}{N^2} = \frac{N_1^2}{N_1^2 + \sum_{i=2}^d N_i^2} \overset{\mathcal{L}}{=} \frac{G(\frac{1}{2})}{G(\frac{1}{2}) + G(\frac{d-1}{2})}$$

where $G(\frac{1}{2})$, and $G(\frac{d-1}{2})$ are independent gamma($\frac{1}{2}$) and gamma($\frac{d-1}{2}$) random variables, respectively. Thus, N_1^2/N is distributed as a beta($\frac{1}{2}, \frac{d-1}{2}$) random variable, $\beta(\frac{1}{2}, \frac{d-1}{2})$. We have:

$$\begin{aligned}
\mathbf{E}\left[|x_1|\right] &= \mathbf{E}\left[\sqrt{\beta\left(\frac{1}{2}, \frac{d-1}{2}\right)}\right] \\
&= \int_0^1 \frac{x^{\frac{1}{2}-1}(1-x)^{\frac{d-1}{2}-1}}{B(\frac{1}{2}, \frac{d-1}{2})} \cdot \sqrt{x} \quad dx \\
&= \frac{B(1, \frac{d-1}{2})}{B(\frac{1}{2}, \frac{d-1}{2})} \\
&= \frac{2}{dB(\frac{1}{2}, \frac{d+1}{2})} \ ,
\end{aligned}$$

where $B(a, b)$ is the beta function.

From Mitrinovic [8, p. 286], we note:

$$\begin{aligned}
\frac{2}{dB(\frac{1}{2}, \frac{d+1}{2})} &\geq \frac{1}{d}\sqrt{\frac{d}{2} + \frac{1}{4} + \frac{1}{16d+32}} \cdot \frac{1}{\Gamma(\frac{1}{2})} && (2) \\
&= \frac{2}{d}\sqrt{\frac{d}{2} + \frac{1}{4} + \frac{1}{16d+32}} \cdot \frac{1}{\sqrt{\pi}} && (3) \\
&\geq \frac{1}{d}\sqrt{\frac{2d+1}{\pi}} \ . && (4)
\end{aligned}$$

Furthermore,

$$\begin{aligned}
\mathbf{E}\left[|x_1|\right] &\leq \frac{1}{d}\frac{d+1}{\sqrt{\pi} \cdot \sqrt{\frac{d}{2} + \frac{3}{4} + \frac{1}{16d+48}}} \\
&\leq \frac{1}{d}\frac{2(d+1)}{\sqrt{\pi} \cdot \sqrt{2d+3}} \\
&\leq \frac{1}{d}\sqrt{\frac{2(d+1)}{\pi}} \ .
\end{aligned}$$

In summary,

$$\frac{1}{d}\sqrt{\frac{2d+1}{\pi}} \leq c_d = \frac{1}{d}\frac{2\Gamma(\frac{d}{2}+1)}{\sqrt{\pi} \cdot \Gamma(\frac{d+1}{2})} \leq \frac{1}{d}\sqrt{\frac{2(d+1)}{\pi}} \ .$$

B Choosing the Lines and Weights

From the previous appendix, we know that c_d is the expected absolute value of the x-coordinate of a point v chosen uniformly at random from $\mathbb{S}^{d-1}$. Let cap(θ) denote the spherical cap of angular radius θ obtained by intersecting $\mathbb{S}^{d-1}$ with the halfspace $x_1 \geq \cos(\theta)$ and let vol(C) denote the surface area (volume) of the spherical cap C. Then, we have the following inequalities for c_d,

$$\begin{aligned}
c_d &= \mathbf{E}[|x_1|] \\
&= 2\sum_{i=1}^{m} \Pr\left\{x \in \operatorname{cap}\left(\frac{i\pi}{m}\right) \setminus \operatorname{cap}\left(\frac{(i-1)\pi}{m}\right)\right\} \cdot \mathbf{E}\left[x_1 \mid x \in \operatorname{cap}\left(\frac{i\pi}{m}\right) \setminus \operatorname{cap}\left(\frac{(i-1)\pi}{m}\right)\right] \\
&\le \frac{2}{\operatorname{vol}(\mathbb{S}^{d-1})} \cdot \sum_{i=1}^{m} \cos\left(\frac{(i-1)\pi}{2m}\right)\left(\operatorname{vol}\left(\operatorname{cap}\left(\frac{i\pi}{2m}\right)\right) - \operatorname{vol}\left(\operatorname{cap}\left(\frac{(i-1)\pi}{2m}\right)\right)\right) \\
&\le \frac{2}{\operatorname{vol}(\mathbb{S}^{d-1})} \cdot \sum_{i=1}^{m} \left(\cos\left(\frac{i\pi}{2m}\right) + \frac{2}{m}\right)\left(\operatorname{vol}\left(\operatorname{cap}\left(\frac{i\pi}{2m}\right)\right) - \operatorname{vol}\left(\operatorname{cap}\left(\frac{(i-1)\pi}{2m}\right)\right)\right) \\
&= \frac{2}{\operatorname{vol}(\mathbb{S}^{d-1})} \cdot \sum_{i=1}^{m} \cos\left(\frac{i\pi}{2m}\right)\left(\operatorname{vol}\left(\operatorname{cap}\left(\frac{i\pi}{2m}\right)\right) - \operatorname{vol}\left(\operatorname{cap}\left(\frac{(i-1)\pi}{2m}\right)\right)\right) + \frac{2}{m} \\
&\le c_d + \frac{2}{m} \ .
\end{aligned}$$

Using the theory of ε-approximations [3, Chapter 4], it is possible, for any $r \ge 1$ to find a set S of $k = O(dr^2 \log dr)$ points on $\mathbb{S}^{d-1}$ such that, for *any* spherical cap C

$$\left|\frac{\operatorname{vol}(C)}{\operatorname{vol}(\mathbb{S}^{d-1})} - \frac{|C \cap S|}{|S|}\right| \le \frac{1}{r}$$

and by increasing the size of S by a constant factor we may assume that S is symmetric ($x \in S$ iff $-x \in S$). Let S be such a set of points and consider the average x-coordinate of an element in S.[3]

$$\begin{aligned}
\frac{1}{|S|} \cdot \sum_{x \in S} |x_1| &\le \frac{2}{|S|} \cdot \sum_{i=1}^{m} \cos\left(\frac{(i-1)\pi}{m}\right)\left(\left|\operatorname{cap}\left(\frac{i\pi}{m}\right) \cap S\right| - \left|\operatorname{cap}\left(\frac{(i-1)\pi}{m}\right) \cap S\right|\right) \\
&\le \frac{2}{\operatorname{vol}(\mathbb{S}^{d-1})} \cdot \sum_{i=1}^{m} \cos\left(\frac{(i-1)\pi}{m}\right)\left(\operatorname{vol}\left(\operatorname{cap}\left(\frac{i\pi}{m}\right)\right) - \operatorname{vol}\left(\operatorname{cap}\left(\frac{(i-1)\pi}{m}\right)\right) + \frac{2\operatorname{vol}(\mathbb{S}^{d-1})}{r}\right) \\
&\le \frac{2}{\operatorname{vol}(\mathbb{S}^{d-1})} \cdot \sum_{i=1}^{m} \cos\left(\frac{(i-1)\pi}{m}\right)\left(\operatorname{vol}\left(\operatorname{cap}\left(\frac{i\pi}{m}\right)\right) - \operatorname{vol}\left(\operatorname{cap}\left(\frac{(i-1)\pi}{m}\right)\right)\right) + \frac{4m}{r} \\
&\le c_d + \frac{2}{m} + \frac{4m}{r} \\
&= c_d + \frac{3}{m}
\end{aligned}$$

where the last inequality follows by taking $r = 4m^2$, in which case $|S| = O(dm^4 \log dm)$. Taking $m = 3/\delta c_d$ and using a symmetric argument to the one above to obtain a lower bound, we obtain

$$1 - \delta \le \frac{1}{c_d|S|} \cdot \sum_{x \in S} |x_1| \le 1 + \delta$$

with $|S| = O((d^3/\delta^4)\log(d/\delta))$.

[3]The x-coordinate x_1 in this argument is simply a placeholder for the projection of $p - q$ onto x. That is, we are assuming without loss of generality that $p - q = (1, 0, 0, \dots, 0)$, so that $p \cdot x - q \cdot x = (p - q) \cdot x = x_1$.