Linear Sketches for Geometric LP-Type Problems

Nabi Efe Çekirge

CMU-CS-24-162

December 2024

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science.

Keywords: Sketching, streaming, lp-type problems

For my parents and my partner.

Abstract

LP-type problems such as the Minimum Enclosing Ball (MEB), Linear Support Vector Machine (SVM), Linear Programming (LP), and Semidefinite Programming (SDP) are fundamental combinatorial optimization problems, with many important applications in machine learning applications such as classification, bioinformatics, and noisy learning. We study LP-type problems in several streaming and distributed big data models, giving ε-approximation linear sketching algorithms with a focus on the high accuracy regime with low dimensionality d, that is, when $d < (1/\varepsilon)^{0.999}$. Our main result is an $O(ds)$ pass algorithm with $O\left(s\left(\sqrt{d}/\varepsilon\right)^{3d/s}\right) \cdot \mathrm{poly}\left(d, \log\left(1/\varepsilon\right)\right)$ space complexity, for any parameter $s \in [1, d\log\left(\sqrt{d}/\varepsilon\right)]$, to solve ε-approximate LP-type problems of $O(d)$ combinatorial and VC dimension. Notably, by taking $s = d\log\left(\sqrt{d}/\varepsilon\right)$, we achieve space complexity polynomial in d and polylogarithmic in $1/\varepsilon$, presenting exponential improvements in $1/\varepsilon$ over current algorithms. We complement our results by showing lower bounds of $(1/\varepsilon)^{\Omega(d)}$ for any 1-pass algorithm solving the $(1+\varepsilon)$-approximation MEB and linear SVM problems, further motivating our multi-pass approach.

Acknowledgments

I would like to thank my research and thesis advisor Dr. David Woodruff for his guidance throughout this project. His help has been instrumental to the creation of this work, and for setting me on the path of research I am on. I would also like to thank William Gay, who has been a coauthor, and importantly a friend, throughout our work on this project.

I would like to thank Dr. Richard Peng for serving on my thesis committee and for his support of my research and academic searching. I would also like to thank Dr. Daniel Anderson, whose whiteboard has seen much of our proof scratch work and whose office door has always been open to us. I would like to extend my thanks to Dr. Dave Eckhardt, Dr. Ruben Martins, and Angy Malloy, for their help throughout my Master's degree and this thesis.

Finally, on a personal note, I would like to thank my parents and my partner Liz for their immeasurable help and support throughout this project. I would not be where I am without them.

Contents

List of Tables

Chapter 1

Introduction

LP-type problems are a class of problems fundamental to the field of combinatorial optimization. Introduced originally by Shahir & Welzl, they are defined by a finite set of elements S and a solution function f which maps subsets of S to a totally ordered domain and also satisfies the properties of locality and monotonicity [41]. Examples of LP-type problems include the Minimum Enclosing Ball (MEB) problem, the Linear Support Vector Machine(SVM) problem, and linear programming (LP) problems for which the class is named. These problems have many applications in different fields such as machine learning. For example, the MEB problem has applications in classifiers [14], and support vector machines [15, 44]. The linear SVM problem is a fundamental problem in machine learning, used in many classification problems such as text classification, character recognition, and bioinformatics [19]. Linear programs have many uses in data science problems, with [35] showing that most common machine learning algorithms can be expressed as LPs. One such example is the linear classification problem, which has applications in many noisy learning and optimization problems [13, 16, 27]. A related class of problems are semidefinite programming (SDP) problems, which are useful for many machine learning tasks such as distance metric learning [48] and matrix completion [17, 18, 40].

For these applications, the size of the input problem can be very large, and often too expensive to store in local memory. Further, the data set can be distributed between multiple machines which require a joint computation. In these applications, the large size of data generally means that finding exact solutions to problems is prohibitively expensive in space or communication complexity, and such many algorithms instead keep and perform computations on a small representation of the data which allows them to give an answer within a small approximation factor of the original solution.

One such way to represent data is with linear sketches. For a large dataset P of n elements from a universe of size U, which can be thought of as a vector $\boldsymbol{U} \in \{0,1\}^U$ where $\boldsymbol{U_p}$ is 1 for each element $\boldsymbol{p} \in P$ and 0 otherwise, a linear sketch generally is in the form of an $N \times U$ matrix A, for $N \ll n$. Thus, one only has to store $A \cdot \boldsymbol{U}$, a vector of size N, which is much cheaper than storing the full n element data set. Linear sketches are very useful in many big data applications, such as data streams where data points are read one by one, or parallel computation where the data set is split among different machines. This is because updating the data vector $\boldsymbol{U} \leftarrow \boldsymbol{U} + \boldsymbol{e_p}$ for a newly read data point $\boldsymbol{p}$, where $\boldsymbol{e_p}$ is the $\boldsymbol{p}^{\text{th}}$ standard unit vector, simply requires updating the sketch as $A \cdot \boldsymbol{U} + A_{\cdot \boldsymbol{p}}$ for the $\boldsymbol{p}^{\text{th}}$ column of A, $A_{\cdot \boldsymbol{p}}$. Similarly, for data distributed among k machines as $P = \sum_{i \in [k]} P_k$, the sketch can be maintained separately by each machine, since $A \cdot P = \sum_{i \in [k]} A \cdot P_i$. As linear sketches consist of matrix-vector products, they are useful in GPU-based applications, which are particularly suited for these operations. Linear sketching algorithms have extensive study in big data applications, see [5, 34, 47] for some examples and surveys. An extension which we build our algorithm in is the multiple linear sketch model, which corresponds to making adaptive matrix-vector products in each of a small number of rounds [43].

1.1 Our Contributions

In this work, we explore using linear sketches in order to solve approximate LP-type problems in high accuracy regimes, in which $d < (1/\varepsilon)^{0.999}$, where d is the dimensionality of the problem and ε is the (multiplicative or additive) approximation accuracy. This regime is applicable to many real world big data applications, where $1/\varepsilon$ might be several orders of magnitude larger than d. Take for example [38], where $d = 2$, $1/\varepsilon \in [10^2, 10^5]$, and $n = 10^5$, or [44] where for some data sets $1/\varepsilon$ is taken to be much larger than d yet small compared to n.

We present an algorithm in the multiple linear sketch model for solving certain classes of LP-type problems, which can be utilized in several prominent big data models. Given a universe $S = \{-\Delta, -\Delta + 1, \ldots, \Delta\}^d$ of d-dimensional points, we use a linear sketch to reduce an input set $P \subseteq S$ into a point set $Q = \left\{(1+\varepsilon)^l \cdot \boldsymbol{e} \mid \boldsymbol{e} \in E_{\boldsymbol{p}_c}, l \in \log_{1+\varepsilon} d\right\}$, where $E_{\boldsymbol{p}_c}$ is a metric ε-net centered at a point $\boldsymbol{p}_c \in P$ and d is an $O(1)$ approximation for the distance of the furthest point in P to $\boldsymbol{p}_c$. Our algorithm is also applicable when $S = \left\{-1, -1 + \frac{1}{\Delta}, \ldots, 1\right\}^d$. Now, we reduce P directly into the point set $Q = E$ where E is the metric ε-net centered at the origin. This linear sketch can be though of as an $N \times |S|$ matrix A for $N = l\,|E|$ where each row corresponds to one point possible point $(1+\varepsilon)^l \cdot \boldsymbol{e}$, containing 1's for all the points $\boldsymbol{p} \in S$ with direction closest to $\boldsymbol{e}$ and norm closest to $(1+\varepsilon)^l$. Intuitively, $A \cdot P$ has the effect of taking each point $\boldsymbol{p} \in P$ and snapping them in the direction of the closest metric ε-net point, and then rounding their norm to the nearest power of $(1+\varepsilon)$. When $Q = E$, $A \cdot P$ has the effect of snapping each point $\boldsymbol{p} \in P$ to the closest metric ε-net point. The major benefit of using this linear sketch is that we don't need to actually store our metric ε-net. We can calculate where each point is snapped to quickly on the fly, so we do not suffer the exponential in d space complexity of ε-kernel based formulations.

Our algorithm works in the presence of input point deletions. So, for metric ε-nets centered around a point, we present a method to find $\boldsymbol{p}_c$ and d using only non-deleted points in P. This is important as our approximation guarantees depend on our metric ε-net. We achieve this with ℓ_0 samplers, as they work in the presence of point deletions. Using an ℓ_0 sampler, we can sample a non-deleted input point which becomes our center point $\boldsymbol{p}_c$. Now, by construction of P, we have an upper and lower bound on the distance of other points from $\boldsymbol{p}_c$. Thus, we can binary search over powers of 2 in this range, again using an ℓ_0 sampler to check for the presence of a point with distance to $\boldsymbol{p}_c$ larger than the current power of 2 in each iteration. By the end of the binary search, which takes $\log\log(\Delta d)$ iterations, we end up with a 2-approximation for d, considering only non-deleted points.

This linear sketch of the data can now be used to run our algorithm on, treating it as a smaller virtual input compared to the original input. Our general algorithm for solving LP-type problems follows the sampling idea presented by [8], where input points are assigned weights, and a weighted sampling procedure is used in order to solve the LP-type problem on smaller samples from the input. By modifying the weights throughout multiple iterations, the algorithm quickly converges on samples that contain a basis for the LP-type problem, for which the solution is the solution of the entire input set. By using a metric ε-net for our linear sketch, we can ensure that our virtual input can be accessed quickly on the fly, and that a solution for virtual input is a $(1+\varepsilon)$-approximation solution for the original input.

This approach brings a complication to our weighted sampling procedure. Because our linear sketch is formed by snapping each input point $\boldsymbol{p} \in P$ to a point $\boldsymbol{q} \in Q$, where $|Q| = N \ll |P| = n$, we can have distinct original points $\boldsymbol{p} \neq \boldsymbol{p}'$ that are snapped to the same point $\boldsymbol{q}$. Since our weight function deals with weights for the snapped points $\boldsymbol{q}$, and there is no easy way to recognize duplicates in large (and possibly distributed) input sets, sampling using reservoir sampling over P will favor these duplicated points. Thus, we utilize ℓ_0 estimators and ℓ_0 samplers in order to sample proportional to the number of distinct snapped points $\boldsymbol{q}$. There is a large body of work on these estimators and samplers in big data models, see [12, 31, 32] for some examples. An advantage is that these estimators and samplers are linear sketches, and thus can be implemented in our algorithm. The tradeoff is that they come with approximations and error probabilities, however we show that the distribution we sample from by using their approximate weights is within a small variation distance of the distribution we would obtain from using true weights. As such, we retain the same analysis and approximate bounds using these estimators and samplers. Note also that this allows our algorithm to work in the presence of duplicated points in the input, which generally can prove problematic for sampling based algorithms.

As our algorithm works in the multiple linear sketch model, we can solve LP-type problems in the following big data models.

- **Multipass Streaming**: In this model, the input P is presented as a large stream of points. Our algorithm can make multiple linear scans of this stream, reading one input point at a time. The goal for our algorithm in the model is to minimize the number of passes over the stream while maintaining a small space complexity.
- **Strict Turnstile**: In this model, there is an underlying vector $\boldsymbol{v}$ where $\boldsymbol{v_p}$ corresponds to the point $\boldsymbol{p} \in U$. The input P is presented as a stream of additive updates $\boldsymbol{v} \leftarrow \boldsymbol{v} + \boldsymbol{e_p}$ or $\boldsymbol{v} \leftarrow \boldsymbol{v} - \boldsymbol{e_p}$, where we are guaranteed that $\boldsymbol{v}$ is itemwise nonnegative. This can be thought of as a stream of operations insert($\boldsymbol{p}$) and delete($\boldsymbol{p}$) with the guarantee that no point $\boldsymbol{p} \in P$ has negative copies. Similarly to the multipass streaming model our algorithm can make multiple linear scans of this stream, with the goal of minimizing the number of passes over the stream and maintaining a small space complexity.

- **Coordinator**: In this model, there are k distributed machines and a separate coordinator machine connected to each machine via a two-way communication channel. The input P is distributed among the machines, with the goal of computing a joint solution on P. Communication between the machines and the coordinator proceeds in rounds, where each round the coordinator can send a message to each machine, and then each machine can send a message back. The final computation is done by the coordinator. The goal for our algorithm in the model is to minimize the number of communication rounds, and maximum bits of information sent or received in any round.

- **Parallel Computation**: In this model, there are k distributed machines, connected via two-way communication channels. Similar to the coordinator model, the input is distributed among the machines which compute a joint solution on P over rounds of communication. This solution can be on the union of each machine's dataset, however one can also use different combinations. For example, the symmetric difference of data over time across servers can capture the change in that data. Thus, one might want to optimize over this difference to predict current trends, see e.g. [26]. The goal for our algorithm again is to minimize the number of communication rounds, and maximum bits of information sent or received in any round, known as the load.

We give the following result for LP-type problems that are bounded in combinatorial and VC-dimension. More detailed bounds and time bounds are given in Sections 3.1, 3.2, 3.3, and 3.4.

Theorem 1.1. *For any $s \in [1, \log N]$, there exists a randomized algorithm to compute a $(1 + O(\varepsilon))$-approximation solution to an LP-type problem with VC and combinatorial dimensions in $O(d)$ with high probability in the following models, using a linear sketch of size N where $\log N \in \operatorname{poly}(d, \log(1/\varepsilon))$, and where $\operatorname{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$ and a solution $f(\cdot)$.*

- ***Multipass Streaming**: An $O(ds)$ pass algorithm with $O\left(sN^{3/s}\right) \cdot \operatorname{poly}(d, \log N) + \left(O(ds) + \operatorname{poly}\left(d, N^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p})$ space complexity.*
- ***Strict Turnstile**: An $O(ds + \log\log(\Delta d))$ pass algorithm with $O\left(sN^{3/s}\right) \cdot \operatorname{poly}(d, \log N) + \left(O(ds) + \operatorname{poly}\left(d, N^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p})$ space complexity.*
- ***Coordinator**: An $O(ds)$ round algorithm with $O\left(k\left(d \log N + \log\left(dN^{1/s}\right)\right)\right) + O\left(k + \operatorname{poly}\left(d, N^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p})$ load.*
- ***Parallel Computation**: An $O(ds)$ round algorithm with $O\left(k\left(d \log N + \log\left(dN^{1/s}\right)\right)\right) + O\left(k + \operatorname{poly}\left(d, N^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p})$ load.*

We can use this general algorithm to find approximate solutions for many LP-type problems which satisfy some elementary properties, as described in Section 2. We emphasize that most LP-type problems fulfill these properties, and we explain how each of the problems we present fulfill the properties. We present applications of our general algorithm for the following problems. Note that for the Linear SVM, Bounded LP, and Bounded SDP problems, we do not have the $\log\log(\Delta d)$ pass increase in the strict turnstile model.

- **MEB**: Given n points $P \subseteq \{-\Delta, -\Delta+1, \ldots, \Delta\}^d$, the objective of the MEB problem is to find a d-dimensional ball of minimal radius that encloses all of the input points. We present an algorithm to solve the $(1+\varepsilon)$-approximation MEB problem, outputting a ball that encloses P and has radius at most $(1+O(\varepsilon))$ of the optimal radius. For the MEB problem, we have $N \in O\left(\frac{\sqrt{d}}{\varepsilon}\right)^d$.

- **Linear SVM**: Given n labeled points $P \subseteq \left\{-1. -1 + \frac{1}{\Delta}, \ldots, 1\right\}^d \times \{-1, +1\}$ and some $\gamma > 0$ for which the poins are either linearly inseparable or γ-separable, the objective of the linear SVM problem is to find a separating hyperplane of P with the largest margin. We present an algorithm to solve the $(1+\varepsilon)$-approximation linear SVM problem with high probability, outputting either that the points are linearly inseparable, or outputting a separating hyperplane $(\boldsymbol{u}, b)$ where $\|\boldsymbol{u}\|^2$ is at most $(1+O(\varepsilon))$ of the optimal separating hyperplane. For the linear SVM problem, we have $N \in O\left(\frac{\sqrt{d}}{\varepsilon\gamma}\right)^d$.

- **Bounded LP**: Given an objective vector $\boldsymbol{c} \in \left\{-1. -1 + \frac{1}{\Delta}, \ldots, 1\right\}^d$ such that $\|\boldsymbol{c}\| \in O(1)$ and n constraints $P \subseteq \left\{-1. -1 + \frac{1}{\Delta}, \ldots, 1\right\}^d \times \left\{-1, -1 + \frac{1}{\Delta}, \ldots, 1\right\}$, where for each constraint $(\boldsymbol{a}_i, b_i) \in P$ $\|\boldsymbol{a}_i\|, |b_i| \in O(1)$, the objective of bounded LP problems are to find a solution $\boldsymbol{x}$ with $\|\boldsymbol{x}\| \in O(1)$ that maximizes the objective $\boldsymbol{c}^T\boldsymbol{x}$ while satisfying the constraints. We present an algorithm to solve additive ε-approximation bounded LP problems, outputting a solution that is $O(\varepsilon)$ within each constraint, and for which the objective is at most $O(\varepsilon)$ smaller than the optimal solution. For bounded LP problems, we have $N \in O\left(\frac{\sqrt{d}}{\varepsilon}\right)^d$.

- **Bounded SDP**: Given an objective matrix $C \in \left\{-1. -1 + \frac{1}{\Delta}, \ldots, 1\right\}^{d \times d}$ such that $\|C\|_F \leq 1$ and n constraints $P \subseteq \left\{-1. -1 + \frac{1}{\Delta}, \ldots, 1\right\}^{d \times d} \times \left\{-1, -1 + \frac{1}{\Delta}, \ldots, 1\right\}$, where for each constraint $(A^{(i)}, b^{(i)}) \in P$ $\left\|A^{(i)}\right\|_2, \left|b^{(i)}\right| \leq 1$, $\left\|A^{(i)}\right\|_F \leq F$, and the number of nonzero entries of $A^{(i)}$ is bounded by S, the objective of bounded SDP problems are to find a positive semidefinite solution X of unit trace that maximizes the objective $\langle C, X\rangle_F$ while satisfying the constraints. We present an algorithm to solve additive ε-approximation bounded SDP problems, outputting a solution that is $O(\varepsilon)$ within each constraint, and for which the objective is at most $O(\varepsilon)$ smaller than the optimal solution. For bounded SDP problems, as we are working with $d \times d$ matrices, the combinatorial and VC dimensions are $O(d^2)$, and we have $N \in \binom{d^2}{S} \cdot O\left(\left(\frac{\min(d,S)}{\varepsilon}\right)^S \frac{1}{\varepsilon}\right) + O\left(\left(\frac{\sqrt{d}}{\varepsilon}\right)^d\right)$.

Our algorithm provides powerful results for these problems, and for our general LP-type problem framework, when we set the parameter s equal to $\log N$. As we have $\log N \in \operatorname{poly}(d, \log(1/\varepsilon))$, this allows us to achieve algorithms with pass and space complexities polynomial in d in polylogarithmic in $1/\varepsilon$. Table 1.1 compares our results to those of prior work.

For the approximate MEB problem, there are many results, as discussed in Section 1.2. Here, we highlight two results for $(1+\varepsilon)$-approximation MEB, by Zarrabi-Zadeh and Bâdoiu & Clarkson [9, 51]. Zarrabi-Zadeh achieves a 1 pass algorithm with $O\left((1/\varepsilon)^{\frac{d-1}{2}} \log(1/\varepsilon)\right)$ space com-

Table 1.1: Result Comparisons for Various LP-type Problems

Problem	Ref.	Previous Complexity	Our Complexity
MEB	[51]	1 pass $O\left(\left({}^{1}/_{\varepsilon}\right)^{\frac{d-1}{2}} \log\left({}^{1}/_{\varepsilon}\right)\right)$ space	$O\left(d^2 \log\left({}^{\sqrt{d}}/_{\varepsilon}\right)\right)$ passes poly $(d, \log({}^{1}/_{\varepsilon}))$ space
	[9]	$\lceil {}^{2}/_{\varepsilon} \rceil$ passes $O\left({}^{d}/_{\varepsilon}\right)$ space	
Linear Classification	[25]	$\tilde{O}\left({}^{1}/_{\varepsilon}\right)$ passes $\tilde{O}\left({}^{1}/_{\varepsilon^2}\right)$space	$O\left(d^2 \log\left({}^{\sqrt{d}}/_{\varepsilon}\right)\right)$ passes poly $(d, \log({}^{1}/_{\varepsilon}))$ space
Bounded SDP	[28]	$O\left(\frac{1}{\varepsilon^2}\log n \begin{pmatrix} F^2(n+\log d)+\frac{1}{\varepsilon}S\log d \\ +\min\left(\frac{1}{\varepsilon^2}S\log d, d^2\right)\frac{1}{\varepsilon^2}\log d \end{pmatrix}\right)$ time	$\tilde{O}\left(d^2 S \log\left(\frac{d}{\varepsilon}\right)\left(n+d^6\right)\right)$ time
	[42]	$O\left(\sqrt{d}\log\left({}^{d}/_{\varepsilon}\right)\right)$ passes $O\left(n^2+d^2\right)$ space	$O\left(d^4 \log\left({}^{\sqrt{d}}/_{\varepsilon}\right)\right)$ passes poly $(d, \log({}^{1}/_{\varepsilon}))$ space

plexity, and Bâdoiu & Clarkson achieve a $\lceil {}^{2}/_{\varepsilon} \rceil$ pass algorithm with $O\left({}^{d}/_{\varepsilon}\right)$ space complexity. In contrast. we achieve an $O\left(d^2 \log\left({}^{\sqrt{d}}/_{\varepsilon}\right)\right)$ pass algorithm with $\mathrm{poly}\left(d, \log\left({}^{1}/_{\varepsilon}\right)\right)$ space complexity. The Zarrabi-Zadeh result is among many $(1+\varepsilon)$-approximation algorithms that utilize an ε-kernel. While these algorithms require one pass, they require space complexity of ${}^{1}/_{\varepsilon}{}^{O(d)}$. Thus, we present a large increase in space efficiency over this method. We also present a lower bound on the space complexity for any one-pass algorithm for the MEB problem of $\left({}^{1}/_{\varepsilon}\right)^{\Omega(d)}$ in Section 4, which further motivates our multi-pass approach. While the result by Bâdoiu & Clarkson has space complexity polynomial in d, it is also polynomial in ${}^{1}/_{\varepsilon}$ in passes and space. Thus, in the high accuracy regime, we present an exponential improvement over their result.

Another approach to the $(1+\varepsilon)$-MEB problem is by using the ellipsoid algorithm with a similar metric ε-net construction. While this algorithm also can achieve $\mathrm{poly}\left(d, \log\left({}^{1}/_{\varepsilon}\right)\right)$ pass and space complexity, we present two main improvements over using this approach. Most importantly, as our algorithm is in the multiple linear sketch model, we are able to apply our result to the turnstile model, where there is addition and removal of points in the input stream, and the coordinator and parallel computation models, where the input is distributed. As the ellipsoid algorithm is not in the multiple linear sketch model, it cannot, for example, find the MEB of the symmetric difference of pointsets from two machines. Further, the bulk of our algorithm utilizes ℓ_0 samplers and estimators and simple arithmetic, which can allow our algorithm to be practical to implement.

As we present in Section 3.7.1, our algorithm for additive ε-approximation bounded LPs can be used to solve the linear classification problem. Note that our result finds an exact classifier. We compare our result to the result by [25], who achieve an algorithm with $\tilde{O}\left({}^{1}/_{\varepsilon}\right)$ passes and $\tilde{O}\left({}^{1}/_{\varepsilon^2}\right)$ space complexity. In contrast. we achieve an $O\left(d^2 \log\left({}^{\sqrt{d}}/_{\varepsilon}\right)\right)$ pass algorithm with $\mathrm{poly}\left(d, \log\left({}^{1}/_{\varepsilon}\right)\right)$ space complexity. In the high accuracy regime, this represents an exponential improvement over their result in pass and space complexity.

Another application of our framework is for additive ε-approximation bounded SDP problems.

As we present in Section 3.8.1, our formulation can be used to solve for the saddle point problem within the unit simplex. For this problem, Garber & Hazan provide a result of

$$O\left(\frac{1}{\varepsilon^2}\log n\left(F^2(n+\log d)+\frac{1}{\varepsilon}S\log d+\min\left(\frac{1}{\varepsilon^2}S\log d, d^2\right)\frac{1}{\varepsilon^2}\log d\right)\right)$$

operations [28]. In contrast, we are able to achieve a time bound of $\tilde{O}\left(d^2 S \log\left(\frac{d}{\varepsilon}\right)(n+d^6)\right)$, which gives us an exponential improvement in $1/\varepsilon$ dependence, significant in the high accuracy regime. Another result is by [42] who show an $O\left(\sqrt{d}\log\left(d/\varepsilon\right)\right)$ pass algorithm with $\tilde{O}\left(n^2+d^2\right)$ space complexity. In contrast, we achieve an $O\left(d^4\log\left(d/\varepsilon\right)\right)$ pass algorithm with $\mathrm{poly}\left(d, \log 1/\varepsilon\right)$ space complexity. In the high accuracy regime with a large number of constraints $n \gg 1/\varepsilon$, this presents a loss in number of passes polynomial in d, while achieving a significant decrease in space complexity, as our result does not depend on n at all.

1.2 Related Work

Approximation algorithms for LP-type problems in big data models have been well-studied, where the trade-off between approximation ratio, number of passes, and space complexity has given rise to different techniques being used for different regimes.

A classic result for the MEB problem is a 1-pass $3/2$-approximation algorithm by Zarrabi-Zadeh & Chan, that uses $O(d)$ space complexity [52]. Allowing $O(d)$ passes, an algorithm by Agarwal & Sharathkumar, with further analysis by Chan & Pathak, achieves a 1.22-approximation with $O\left(\left(d/\varepsilon^3 \log\left(1/\varepsilon\right)\right)\right)$ space complexity [1, 21]. For the $(1+\varepsilon)$-approximation problem, much work is done in the single-pass model. These algorithms are based on storing extremal points, called an ε-kernel, and achieve a space complexity of $\left(1/\varepsilon\right)^{O(d)}$ [2, 3, 20, 51]. There is also a body of work on bounding the size of these ε-kernels, as they are useful to achieve low-space representations of large data [10, 24, 33, 50]. [7] present a lower bound of $\left(1/\varepsilon\right)^{\Omega(d)}$ on the size of an ε-kernel. In the multi-pass model, there is a classic result by Bâdoiu & Clarkson, which presents a $\lceil 2/\varepsilon \rceil$-pass algorithm with $O(d/\varepsilon)$ space complexity [9]. This algorithm is widely used in big data models, for its linear dependence on d/ε for space and $1/\varepsilon$ for pass complexity, and simplicity [38, 44].

Linear programs are frequently studied in big data models, with a major focus on exact multi-pass LP results. One recent result is by [8] who give an $O(ds)$ pass algorithm with $O(n^{1/s}) \cdot \mathrm{poly}\left(d, \log n\right)$ space complexity for any $s \in [1, \log n]$. Setting $s = \log n$, their algorithm takes $O(d \log n)$ passes and has $\mathrm{poly}\left(d, \log n\right)$ space usage. This result gives a large improvement over previous results, and is applicable to other LP-type problems as well as the coordinator and massively parallel computation models. Their algorithm builds on Clarkson's sampling based method [23], and utilizes reservoir sampling to sample a μ-net [22]. We build on the ideas presented by [8], to solve LP-type problem approximations. Certain approximation LP algorithms in big-data models are studied by Ahn & Guha who give a $(1+\varepsilon)$-approximation algorithm for packing LPs [4], and [30] who give a $(1+\varepsilon)$-approximation algorithm for covering LPs. The linear classification problem is also well studied, as a fundamental machine learning problem. The classic result for this problem is mistake bound of the perceptron algorithm [39]. A more recent result is by [25], which achieves $O\left({}^{1}\!/\!{}_{\varepsilon^2}\log n\right)$ iterations to find a linear classifier.

There is a lot of study on approximate semidefinite programs, which focus on different regimes. A result that focuses on bounded SDP problems is by Garber & Hazan [28], that give bounds on solving saddle point problems with certain assumptions about the problem input. We use similar assumptions to compare our results for the saddle point optimization problem. For the regime with a low number of constraints and a large dimension, a recent result is by [42]. which achieves an $O\left(\sqrt{d}\log {}^{d}\!/\!{}_{\varepsilon}\right)$ pass algorithm with $\tilde{O}\left(n^2+d^2\right)$ space complexity.

1.3 Preliminaries

Throughout this work we employ the use of various sketching primitives as well as concepts. This section provides some descriptions and examples.

ℓ_0 Estimator There is a 1-pass streaming algorithm which given additive $+1$ updates to an underlying vector $\boldsymbol{v}$ in $\{0,1,\dots,\text{poly}\,(n)\}^n$, gives a $(1\pm\zeta)$-approximation to the number of non-zero coordinates of v, denoted $\ell_0(v)$, with error probability δ, and using $O(\frac{\log\delta^{-1}}{\zeta^2}+\log n)$ bits of memory [12].

ℓ_0 Sampler There is a 1-pass streaming algorithm which given additive $+1$ updates to an underlying vector $\boldsymbol{v}$ in $\{0,1,\dots,\text{poly}\,(n)\}^n$, outputs a coordinate i in the set of non-zero coordinates of $\boldsymbol{v}$, where for each non-zero coordinate j in $\boldsymbol{v}$, we have $i=j$ with probability $\frac{1}{\ell_0(v)}\pm\frac{1}{\text{poly}(n)}$, using $O(\text{polylog}\,(n))$ bits of memory [31].

TV Distance The Total Variation (TV) distance is a metric on the space of probability distributions over a discrete set X. If $\mathcal{P}$ and $\mathcal{Q}$ are distributions over X, the TV distance between $\mathcal{P}$ and $\mathcal{Q}$ is given by

$$\text{TV}(\mathcal{P},\mathcal{Q})=\frac{1}{2}\sum_{x\in X}|\Pr_{\mathcal{P}}(x)-\Pr_{\mathcal{Q}}(x)|.$$

The TV distance is also an integral probability metric with respect to the class of functions with co-domain $\{0,1\}$. That is,

$$\text{TV}(\mathcal{P},\mathcal{Q})=\sup\left\{\left|\mathop{\mathbb{E}}_{x\sim\mathcal{P}}[f(x)]-\mathop{\mathbb{E}}_{x\sim\mathcal{Q}}[f(x)]\right|:f:X\to\{0,1\}\right\}.$$

Chernoff Bound We use the following formulation of the Chernoff bound. Given t independent random variables $X_1,\dots,X_t$ that take values in the range $[0,1]$, and letting $X:=\sum_{i=1}^t X_i$ denote their sum, for any $\delta>0$

$$\Pr(X\geq(1+\delta)\,\mathbb{E}\,[X])\leq e^{\frac{-\delta^2\mathbb{E}[X]}{2+\delta}}.$$

Metric ε-net Given a metric space (M, d), a metric ε-net is a set of points $E \subseteq M$ such that for any point $\boldsymbol{p} \in M$, there exists a point $\boldsymbol{e} \in E$ for which

$$d(\boldsymbol{p}, \boldsymbol{e}) \leq \varepsilon.$$

For large metric spaces, i.e. when $M = \{-\Delta, -\Delta + 1, \dots, \Delta\}^d$, this formulation requires too many points. In this case, we give an alternate formulation. In the case where M is large, given a center point $\boldsymbol{p}_c$, which we consider to be the origin, and a corresponding norm $\|\boldsymbol{q}\| = d(\boldsymbol{q}, \boldsymbol{p}_c)$ for each point $\boldsymbol{q} \in M$, a metric ε-net is a set of points where for any point $\boldsymbol{q} \in M$, there exists a point $\boldsymbol{e} \in E$ for which

$$d\left(\frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}, \boldsymbol{e}\right) \leq \varepsilon.$$

This inequality can be multiplied by $\|\boldsymbol{q}\|$ on both sides for the equivalent definition

$$d(\boldsymbol{q}, \|\boldsymbol{q}\| \, \boldsymbol{e}) \leq \varepsilon \, \|\boldsymbol{q}\| \, .$$

Combinatorial Dimension The combinatorial dimension of a problem is the maximum cardinality of a basis for the problem, denoted ν. For LP-type problems, we consider a basis as the minimum number of points required to define a solution to the problem.

Set System A set system is a pair $(X, \mathcal{R})$ where X is a set of elements, and $\mathcal{R}$ is a collection of subsets of X. For example, in the MEB problem, the set system can be thought of as $(P, \mathcal{B})$ where $P \in \mathbb{R}^d$ is a set of points and $\mathcal{B}$ is a collection of all balls that enclose a subset of points.

VC Dimension Given a set system $(X, \mathcal{R})$ and $Y \subseteq X$, the projection of $\mathcal{R}$ on Y is defined as $\mathcal{R}|_Y := \{Y \cap R \mid R \in \mathcal{R}\}$. The VC dimension of $(X, \mathcal{R})$ is the minimum integer λ where $|\mathcal{R}|_Y| < 2^{|Y|}$ for any finite subset $Y \subseteq X$ such that $|Y| > \lambda$ [37].

ϵ-net Given a set system $(X, \mathcal{R})$, a weight function $w : X \to \mathbb{R}$, and a parameter $\epsilon \in [0, 1]$, a set $N \subseteq X$ is an ϵ-net with respect to $w(\cdot)$ if $N \cap R \neq \emptyset$ for all sets $R \in \mathcal{R}$ such that $w(R) \geq \epsilon \cdot w(X)$ [37].

Lemma 1.2 (Corollary 3.8 from [29])**.** *For any set system $(X, \mathcal{R})$ of VC dimension $d < \infty$, finite $A \subseteq X$, and $\epsilon, \delta > 0$, if N is the set of distinct elements of A obtained by*

$$m \geq \max\left(\frac{4}{\epsilon} \log \frac{2}{\delta}, \frac{8d}{\epsilon} \log \frac{8d}{\epsilon}\right)$$

random independent draws from A, then N is an ϵ-net for $\mathcal{R}$ with probability $1 - \delta$.

While this construction in its original presentation by Haussler & Welzl [29] applies to the trivial weight function $w(x) = 1$, by drawing with probability proportional to an element's weight, an ϵ-net with respect to $w(\cdot)$ can be obtained for any weight function [8].

Notation. To not cause confusion between metric ε-nets and ϵ-nets, we refer to the latter as μ-nets.

Vector and Matrix Norms For vectors we use the 2-norm, where for a length n vector $\boldsymbol{v}$

$$\|\boldsymbol{v}\| = \sqrt{\sum_{i\in[n]} v_i}.$$

For matrices, we use several definitions of norms. For an $n \times m$ matrix A, we use the following definitions of norm.

- Entry-wise 1-norm: $\|A\| = \sum_{i\in[n],j\in[m]} A_{ij}$.
- Spectral norm: $\|A\|_2 = \max_{\boldsymbol{v}\in\mathbb{R}^m,\|\boldsymbol{v}\|\neq 0} \frac{\|A\boldsymbol{v}\|}{\|\boldsymbol{v}\|}$.
- Frobenius norm: $\|A\|_F = \sqrt{\sum_{i\in[n],j\in[m]} A_{ij}^2}$.
- Schatten 1-norm: $\|A\|_* = \sum_{i=1}^{\min(n,m)} \sigma_i(A)$, where $\sigma_i(A)$ are the singular values of A.

1.4 Organization

The rest of this work is organized as such. Chapter 2 presents our algorithm in the multiple linear sketching model for solving general LP-type problems. Chapter 3 presents applications of our algorithm to various big data models, namely the multipass streaming, strict turnstile, coordinator, and parallel computation models. Further, it shows the application of our algorithm to solving various LP-type problems, namely the MEB, Linear SVM, Bounded LP, and Bounded SDP problems. Chapter 4 presents lower bounds for the MEB and Linear SVM problems in the single pass streaming model.

Chapter 2

Algorithm

In this chapter, we present our algorithm for Theorem 1.1, in the multiple linear sketch model. It is inspired by the algorithm presented in [8]. However, we remove a dependence on the number of points in the original stream, n, by employing the use of linear sketches, where we snap the space of input points to a smaller space of discrete points formed by a metric ε-net. This allows us to compute ε-approximation solutions to the problems while using much lower storage complexity.

As we utilize the ideas of [8], we retain the needed properties for LP-type problems from their result, which we present as properties (P1) and (P2). Further, we add a third property that must be fulfilled, (P3). We stress that most natural LP-type problems follow properties (P1) and (P2), and that property (P3) follows naturally for most if not all $(1+\varepsilon)$-approximation solutions to these problems. Our algorithm therefore requires an LP-type problem (S, f) to satisfy the following properties:

(P1) Each element $\boldsymbol{x} \in S$ is associated with a set of elements $S_{\boldsymbol{x}} \subseteq R$ where R is the range of f.

(P2) For all $P \subseteq S$, $f(P)$ is the minimal element of $\bigcap_{\boldsymbol{x} \in P} S_{\boldsymbol{x}}$.

(P3) For all $P, Q \subseteq S$ where Q is the set of points resultant of snapping all the points $\boldsymbol{p} \in P$ to a metric ε-net, a $(1 + O(\varepsilon))$-approximation to $f(Q)$ is a $(1 + O(\varepsilon))$-approximation to $f(P)$.

As an example, consider the MEB problem where S is the set of all d-dimensional points, R is the set of all d-dimensional balls, and f is the function that returns the minimum enclosing ball of the points in its input. For properties (P1) and (P2), each point in the MEB problem is associated with the set of all balls that enclose it. For any set of points, the intersection of these sets of balls are precisely the set of balls that enclose all of them, and thus the MEB of the points is the minimal element of this set. Property (P3) follows intuitively because "unsnapping" a set of points from a metric ε-net can only increase the size of their MEB by a small amount. A formal proof of this property for the MEB problem is given in section 3.5.

Our algorithm is as such. First, we define the metric ε-net we use. We start by taking any point to use as the center of our metric ε-net. For example, in the multipass streaming model, this can be the first point. We call this point $\boldsymbol{p}_c$. Now, we can shift our input points so that this point acts as the origin. This will also give us the useful fact that now any other point's norm is defined as its distance to $\boldsymbol{p}_c$. We can use this point to create a metric ε-net by allowing, for each d dimensions, the values $-1 + i\frac{\varepsilon}{\sqrt{d}}$ where i ranges in $\left[0, 2\frac{\sqrt{d}}{\varepsilon}\right]$. It is clear to see that this creates a metric ε-net as it is is a lattice covering the unit d-cube (which contains the unit d-sphere), where the distance between net points $\boldsymbol{e}_i$ and $\boldsymbol{e}_{i+1}$, where i denotes the value for each coordinate, is ε. Thus, any point that is scaled to the unit d-sphere will be at most $\frac{\varepsilon}{2}$ away from a point in our metric ε-net. The size of this metric ε-net is $\left(1 + \frac{2\sqrt{d}}{\varepsilon}\right)^d$. It is important to note that while this net is sub-optimal in its size, its major advantage as a lattice is that it can be described succinctly, so its storage complexity is small. Further, it is easy to calculate the closest metric ε-net point to an arbitrary point $\boldsymbol{p}_i$.

The crux of our algorithm is therefore to use this metric ε-net to create a linear sketch of the input. Given any point in the input $\boldsymbol{p} \in P$, we can get the point $\frac{\boldsymbol{p}}{\|\boldsymbol{p}\|}$. This point resides in our metric ε-net, and thus we can snap it to its closest metric ε-net point. Now, we can scale it back up, but we round down the norm to $\left(1 + \frac{2}{\varepsilon}\right)^l$ for some $l \in \mathbb{N}$. This has the effect of discretizing our point space along the metric ε-net directions, so that our linear sketch has size bounded by $\lceil \log_{1+\varepsilon}\left(\max_{\boldsymbol{p}\in P} \|\boldsymbol{p}\|\right)\rceil \left(1 + \frac{2\sqrt{d}}{\varepsilon}\right)^d$. In some problems, we will be presented with input points already inside the unit d-cube, or a d-cube of bounded size. In these cases, we only need to snap the input points to the closest metric ε-net point. We refer to the size of our linear sketch as N, and maintain that it is small compared to the size of the input, n.

Notation. For the sake of clarity, we will refer to the input points $\boldsymbol{p} \in P$ as original points, and these new points $\boldsymbol{q} \in Q$ as snapped points.

Before we explore the algorithm proper, we note a side effect of creating this sketch. Shrinking the point space means that distinct points in the original input can now map to the same point. As our algorithm includes weighted sampling over the snapped points, we must treat these as one point and not two overlapping points. In order to achieve this, we utilize ℓ_0 estimators and samplers, as defined in [12] and [31] respectively, which allow us to consider estimates on points and sample from them without running into the problem of overlapping. We note that these ℓ_0 estimators and samplers are linear sketches, and such are able to be used in the models present in our algorithm.

Our algorithm proceeds in iterations. Throughout, we maintain a weight function $w : Q \to \mathbb{R}$ that maps each snapped point to a weight. Each snapped point starts at weight 1, and may be multiplied by $N^{1/s}$ each iteration. In each iteration, the goal of the algorithm is to sample a small subset of the snapped points that contain a basis for the optimal solution to the problem, and is thus able to give us that solution while keeping a low space complexity. This goal is achieved by sampling a μ-net with respect to $w(\cdot)$ from the snapped points using Lemma 1.2. As we explain previously, this sampling cannot be done directly from the snapped points Q because of overlaps, and so we need to utilize ℓ_0 estimators and samplers. Since each point can increase in weight at most once in each iteration, we have a number of discrete weight classes, for which we create estimators and samplers. This allows us to randomly sample each unique snapped point with probability proportional to its weight by randomly selecting a weight class in accordance to the class' total estimated weight, and then using its sampler to uniformly sample a point from that weight class.

Given a sample B, we can calculate its solution $f(B)$, and find the set V of points that violate $f(B)$ (for example, in the MEB problem, violators are points that are outside of the ball). Two sets of ℓ_0 estimators are now used to calculate estimated weights of the snapped points Q and the violators V, overestimating the weight of Q and underestimating the weight of V. If the estimated weight of V is at most μ times the estimated weight of S, then we denote this iteration as successful, and multiply the weight of each violator by $N^{1/s}$. The intuition behind this is that our algorithm tries to find a basis for the problem. Thus, samples from successful iterations can be thought of as good representations of Q, and violators of those samples are more likely to be points in that basis, and should thus be given a higher weight in sampling. The algorithm finishes when there are no violators, at which point it has successfully found $f(Q)$, and as such we can return a $(1 + O(\varepsilon))$-approximation to $f(P)$ using Property (P3).

Pseudocode for for the algorithm is provided in Algorithm 1, which uses Algorithms 2 and 3 as helper functions to sample a μ-net and to check the success of an iteration.

Algorithm 1: An Algorithm for LP-Type Problems

```
input : A stream $P$ of $n$ points $\boldsymbol{p} \in \{-\Delta, -\Delta + 1, \dots, \Delta\}^d$
output: A $1 + O(\varepsilon)$ approximation for $f(P)$
1  Take a point $\boldsymbol{p}_1 \in P$, as the origin
2  Create the metric $\varepsilon$-net $\left\{(v_1, v_2, \dots, v_d) \in \mathbb{R}^d \mid v_i = -1 + i\frac{\varepsilon}{\sqrt{d}} \text{ where } i \in \left[0, 2\frac{\sqrt{d}}{\varepsilon}\right]\right\}$
3  $r_m \leftarrow \max_{\boldsymbol{p}_i \in P} \|\boldsymbol{p}_i\|$
4  Define $N := \lceil \log_{1+\varepsilon} r_m \rceil \left(1 + \frac{2\sqrt{d}}{\varepsilon}\right)^d$
5  Choose $s \leq \ln N$
6  Define $\mu := \frac{1}{10\nu N^{1/s}}$ where $\nu$ is the combinatorial dimension of the LP-type problem.
7  repeat
8      $B \leftarrow$ SampleMPoints ($P$, $\mu$, $w(\cdot)$, current iteration number $t$)
9      if CheckViolatorsWeight ($P$, $\mu$, $w(\cdot)$, current iteration number $t$, $f(B)$) then
10         Denote iteration successful
11         Set $w(\boldsymbol{q}) = w(\boldsymbol{q}) \cdot N^{1/s}$ for each violator $\boldsymbol{q} \in V$
12     end
13 until $V = \emptyset$
14 Let $f(B)$ be the last solution
15 return $(1 + O(\varepsilon))$ approximation to $f(B)$
```

Algorithm 2: `SampleMPoints`

```
input : A stream $P$ of $n$ points $\boldsymbol{p} \in \{-\Delta, -\Delta+1, \dots, \Delta\}^d$, a parameter $\mu$, weight function
        $w(\cdot)$, and integer $t$
output: A set $B$ of unique points in $m := \max\left(\frac{8\lambda}{\mu}\log\frac{8\lambda}{\mu}, \frac{4}{\mu}\log\frac{2}{\frac{1}{4}}\right)$ samples drawn with
        replacement according to $w(\cdot)$
1  Define $m := \max\left(\frac{8\lambda}{\mu}\log\frac{8\lambda}{\mu}, \frac{4}{\mu}\log\frac{2}{\frac{1}{4}}\right)$
2  Create $t$ $(1 \pm \frac{1}{m^{3/2}})$-approximation $\ell_0$ estimators with error probability $\frac{1}{\mathrm{poly}(N)}$ and $t$ $\ell_0$
     samplers with error probability $\frac{1}{\mathrm{poly}(N)}$, (one for each weight class), where the underlying
     vectors are of size $N$ and indexed by points $\boldsymbol{q} \in Q$.
3  foreach $\boldsymbol{p}_i \in P$ do
4      $\boldsymbol{q}_j \leftarrow \boldsymbol{p}_i$ snapped to nearest metric $\varepsilon$-net direction and rounded down
5      Increment $w(\boldsymbol{q}_j)^{\text{th}}$ estimator and sampler by $1$ in the $\boldsymbol{q}_j{}^{\text{th}}$ position
6  end
7  $f_i \leftarrow$ estimate for each estimator $i \in [t]$
8  $p_i \leftarrow \frac{f_i N^{i/s}}{\sum_{j=1}^{t} f_j N^{j/s}}$
9  $B \leftarrow \emptyset$
10 repeat $m$ times
11     Pick $j \in [t]$ randomly according to probabilities $p_i$
12     $\boldsymbol{q} \leftarrow$ sample point from $j^{\text{th}}$ sampler
13     $B \leftarrow B \cup \{\boldsymbol{q}\}$
14 end
15 return $B$
```

Algorithm 3: `CheckViolatorsWeight`

```
input  : A stream $P$ of $n$ points $\boldsymbol{p} \in \{-\Delta, -\Delta+1, \ldots, \Delta\}^d$, a parameter $\mu$, weight function
         $w(\cdot)$, integer $t$, and $f(B)$
output: A boolean representing if the weight of violators for $f(B)$ is small enough
1  Create two sets of $t$ $(1 \pm \frac{1}{4})$-approximation $\ell_0$ estimators with error probability $\frac{1}{\mathrm{poly}(N)}$, (two
     for each weight class), where the underlying vectors are of size $N$ and indexed by points
     $\boldsymbol{q} \in Q$
2  foreach $\boldsymbol{p}_i \in P$ do
3      $\boldsymbol{q}_j \leftarrow \boldsymbol{p}_i$ snapped to nearest metric $\varepsilon$-net direction and rounded down
4      Increment $w(\boldsymbol{q}_j)^{\text{th}}$ estimator in the first set by 1 in the $\boldsymbol{q}_j{}^{\text{th}}$ position
5      if $f\left(B \cup \{\boldsymbol{q}_j\}\right) > f(B)$ then                    // $\boldsymbol{q}_j$ is a violator
6          Increment $w(\boldsymbol{q}_j)^{\text{th}}$ estimator in the second set by 1 in the $\boldsymbol{q}_j{}^{\text{th}}$ position
7      end
8  end
9  $f'_i \leftarrow$ estimate for each estimator $i \in [t]$ in the first set
10 $f''_i \leftarrow$ estimate for each estimator $i \in [t]$ in the second set
11 return $\frac{1}{(1+\frac{1}{4})} \sum_i f''_i N^{i/s} \leq \mu \cdot \frac{1}{(1-\frac{1}{4})} \sum_i f'_i N^{i/s}$
```

2.1 Algorithm Correctness

In this section, we show that our algorithm correctly computes a valid $(1 + O(\varepsilon))$-approximation to the solution of the LP-type problem, $f(P)$.

Claim 2.1. *Given an LP-type problem (S, f) and a set of input points P, Algorithm 1 correctly returns a $(1 + O(\varepsilon))$-approximation to $f(P)$.*

Proof. The algorithm only returns when a computed solution $f(B)$ has no violators. Thus, for any point $\boldsymbol{q}_j \notin B$, $f\left(B \cup \{\boldsymbol{q}_j\}\right) \leq f(B)$. Further, since $B \subseteq B \cup \{\boldsymbol{q}_j\}$, by the monotonicity property for LP-type problems, we have that $f(B) \leq f\left(B \cup \{\boldsymbol{q}_j\}\right)$. Such, we have that $f(B) = f\left(B \cup \{\boldsymbol{q}_j\}\right)$.

We can further this by using the locality property for LP-type problems, so that for any $\boldsymbol{q}_k \notin B$, $f(B) = f\left(B \cup \{\boldsymbol{q}_j\} \cup \{\boldsymbol{q}_k\}\right)$. Proceeding in an inductive setting for all $\boldsymbol{q}_j \in Q \setminus B$, we will have that $f(B) = f(B \cup (Q \setminus B)) = f(Q)$.

Finally, we use property (P3), in order to conclude that our result $f(B)$ is a $(1 + O(\varepsilon))$-approximation to $f(P)$. □

2.2 Algorithm Iteration Count

In this section, we bound the total number of iterations our algorithm takes at $O(\nu s)$ where ν is the combinatorial dimension of the LP-type problem, and s is our parameter with range $[1, \log N]$. To do this, we first bound the number of successful iterations our algorithm takes, which we then use to achieve a total bound on the number of iterations.

Claim 2.2. *Each iteration of Algorithm 1 is denoted successful with probability at least* $2/3$.

Proof. The intuition behind this claim is that when points are sampled in accordance to their weights, we can use Lemma 1.2 to state that our sample is a μ-net with probability $\frac{3}{4}$. While our sampling is not exact and based on ℓ_0 estimators and samplers, we show using a TV distance argument that we retain this $\frac{3}{4}$ probability. Using this, we show that our condition for success is fulfilled with probability $\frac{3}{4} - \frac{1}{\text{poly}(N)}$, where the error is due to the error of our estimators, which still gives an overall $\frac{2}{3}$ success probability. A full proof is provided hereafter.

The first step towards showing this is getting a probability bound on how many iterations will have the sample be a μ-net. By Lemma 1.2, a sample obtained from $m := \max\left(\frac{8\lambda}{\mu}\log\frac{8\lambda}{\mu}, \frac{4}{\mu}\log\frac{2}{\frac{1}{4}}\right)$ draws with probability proportional to the points' weights will result in a μ-net with respect to $w(\cdot)$ with probability at least $3/4$. However, we have to account for the approximations and error probabilities of our ℓ_0 estimators and samplers, as we use them to get our draws.

To account for this, we will look at the TV distance between the distributions where the m drawn samples are taken with probabilities according to the actual weights, and with probabilities according to our estimates.

Let $\mathcal{P}$ be the actual distribution, and $\mathcal{Q}$ be our estimated distribution. Then, looking at the TV distance, we have that

$$\begin{aligned}\text{TV}(\mathcal{P}, \mathcal{Q}) &= \frac{1}{2}\sum_{B} |\mathbf{Pr}_{\mathcal{P}}(B) - \mathbf{Pr}_{\mathcal{Q}}(B)| \\ &= \frac{1}{2}\sum_{B} \mathbf{Pr}_{\mathcal{Q}}(B)\left|\frac{\mathbf{Pr}_{\mathcal{P}}(B)}{\mathbf{Pr}_{\mathcal{Q}}(B)} - 1\right|. \end{aligned} \tag{2.1}$$

We can now bound the quotient of these two probabilites. For simplicity and generality, we will refer to the approximation ratio of the ℓ_0 samplers as $(1+\zeta)$.

Lemma 2.3. $\frac{\mathbf{Pr}_{\mathcal{P}}(B)}{\mathbf{Pr}_{\mathcal{Q}}(B)} \leq \left(1 + \frac{1}{\text{poly}(N)}\right)^m \left(\frac{1+\zeta}{1-\zeta}\right)^m$.

Proof. We proceed by induction on m.

Base Case Consider a singleton sample B_1, and let K_1 be the event of B_1's weight class being chosen. Then we get the quotient

$$\begin{aligned}
\frac{\mathbf{Pr}_{\mathcal{P}}(B_1)}{\mathbf{Pr}_{\mathcal{Q}}(B_1)} &= \frac{\mathbf{Pr}_{\mathcal{P}}(B_1 \mid K_1)}{\mathbf{Pr}_{\mathcal{Q}}(B_1 \mid K_1)} \frac{\mathbf{Pr}_{\mathcal{P}}(K_1)}{\mathbf{Pr}_{\mathcal{Q}}(K_1)} \\
&\leq \left(1 + \frac{1}{\text{poly}\,(N)}\right) \frac{\mathbf{Pr}_{\mathcal{P}}(K_1)}{\mathbf{Pr}_{\mathcal{Q}}(K_1)} \qquad \text{by the errors of the } \ell_0 \text{ samplers and estimators} \\
&\leq \left(1 + \frac{1}{\text{poly}\,(N)}\right) \left(\frac{1+\zeta}{1-\zeta}\right) \qquad \text{by the error of the } \ell_0 \text{ estimators}
\end{aligned}$$

as desired.

Induction Step Consider now the sample B of m sample points $B_1, \ldots, B_m$, and let K_i be the event of S_i's weight class being chosen. Then we get the quotient

$$\begin{aligned}
\frac{\mathbf{Pr}_{\mathcal{P}}(B)}{\mathbf{Pr}_{\mathcal{Q}}(B)} &= \frac{\mathbf{Pr}_{\mathcal{P}}(B_m)}{\mathbf{Pr}_{\mathcal{Q}}(B_m)} \frac{\mathbf{Pr}_{\mathcal{P}}(B_1, \ldots, B_{m-1})}{\mathbf{Pr}_{\mathcal{Q}}(B_1, \ldots, B_{m-1})} \\
&\leq \frac{\mathbf{Pr}_{\mathcal{P}}(B_m)}{\mathbf{Pr}_{\mathcal{Q}}(B_m)} \left(1 + \frac{1}{\text{poly}\,(N)}\right)^{m-1} \left(\frac{1+\zeta}{1-\zeta}\right)^{m-1} \qquad \text{by the induction hypothesis.}
\end{aligned}$$

We can now notice that $\frac{\mathbf{Pr}_{\mathcal{P}}(B_m)}{\mathbf{Pr}_{\mathcal{Q}}(B_m)} = \frac{\mathbf{Pr}_{\mathcal{P}}(B_1)}{\mathbf{Pr}_{\mathcal{Q}}(B_1)}$ since the samples are drawn independently with replacement. Thus, we get that

$$\frac{\mathbf{Pr}_{\mathcal{P}}(B)}{\mathbf{Pr}_{\mathcal{Q}}(B)} = \left(1 + \frac{1}{\text{poly}\,(N)}\right)^{m} \left(\frac{1+\zeta}{1-\zeta}\right)^{m}$$

completing our induction proof. □

Now, we can plug this back into Equation 2.1, and use the fact that the sum of all probabilities of a distribution is 1, to get that

$$\text{TV}(\mathcal{P}, \mathcal{Q}) \leq \frac{1}{2}\left(\left(1 + \frac{1}{\text{poly}\,(N)}\right)^{m} \left(\frac{1+\zeta}{1-\zeta}\right)^{m} - 1\right).$$

Using our ℓ_0 sampler approximation ratios of $\left(1 + \frac{1}{m^{3/2}}\right)$, we get that this TV distance converges to 0, which means that our distribution will be sufficiently close to the actual distribution, and so we will retain that each sample will be a μ-net with probability at least $3/4$. We will now case on this event when our sample B is a μ-net.

Let $f(B)$ be the solution of the sample B. First, we claim that for the set of violators V, we have that $w(V) \leq \mu \cdot w(Q)$. Assume for the sake of contradiction that $w(V) > \mu \cdot w(Q)$. Since $V \in \mathcal{R}$ and B is a μ-net with respect to $w(\cdot)$, this implies that $B \cap V \neq \emptyset$. So, consider a point $\boldsymbol{e} \in B \cap V$. Since $\boldsymbol{e} \in V$, $f(B \cup \{\boldsymbol{e}\}) > f(B)$. However, since $\boldsymbol{e} \in B$, $B \cup \{\boldsymbol{e}\} = B$, and such it cannot be that $f(B \cup \{\boldsymbol{e}\}) > f(B)$. Thus, due to this contradiction, we have that $w(V) \leq \mu \cdot w(S)$.

Now, we want to show that $\frac{1}{(1+\eta)}\sum_i f_i'' N^{i/s} \le \mu \cdot \frac{1}{(1-\eta)}\sum_i f_i' N^{i/s}$, as that is when we denote an iteration successful. First, we define $f_{i,V}$ as the number of violators in the i^{th} weight class, and similarly $f_{i,Q}$ as the number of snapped points in the i^{th} weight class. Using these definitions, we can rewrite $w(V)$ and $w(Q)$, separating by weight class, as $\sum_i f_{i,V} N^{i/s}$ and $\sum_i f_{i,Q} N^{i/s}$ respectively. Plugging these new expressions in, we have that

$$\sum_i f_{i,V} N^{i/s} \le \mu \sum_i f_{i,Q} N^{i/s}. \tag{2.2}$$

By the definition of our estimators, for any i we have that

$$(1-\eta) f_{i,Q} \le f_i' \le (1+\eta) f_{i,Q}.$$

So, we have that $f_{i,Q} \le \frac{1}{(1-\eta)} f_i'$, and so multiplying by the weight of the class and summing over all weight classes we have that

$$\sum_i f_{i,Q} N^{i/s} \le \frac{1}{(1-\eta)} \sum_i f_i' N^{i/s}.$$

Similarly, as we have that

$$(1-\eta) f_{i,V} \le f_i'' \le (1+\eta) f_{i,V},$$

we have that

$$\frac{1}{(1+\eta)} \sum_i f_i'' N^{i/s} \le \sum_i f_{i,V} N^{i/s}.$$

Plugging these into (2.2), we get that

$$\frac{1}{(1+\eta)} \sum_i f_i'' N^{i/s} \le \mu \cdot \frac{1}{(1-\eta)} \sum_i f_i' N^{i/s},$$

with probability at least $1 - \frac{1}{\text{poly}(N)}$, which comes from the error probability of the estimators.

So we know that each iteration will be denoted a successful iteration with probability at least $3/4 - \frac{1}{\text{poly}(N)}$. Given a sufficiently large N such that $\frac{1}{\text{poly}(N)} \le 1/12$, we have that each iteration will be denoted a successful iteration with probability at least $2/3$. □

We now note that the weight function $w(\cdot)$ is only updated when there is an iteration that is deemed successful. By Claim 2.2 each iteration is deemed successful with probability at least $\frac{2}{3}$. By the Chernoff bound, if the algorithm takes t iterations, then with probability at least $1 - e^{-\Omega(t)}$, at least $\frac{t}{2}$ of these iterations are denoted as successful. It follows that bounding the number of successful iterations using the weight function gives us a bound on the total number of iterations.

Claim 2.4. *Algorithm 1 takes $O(\nu s)$ iterations with probability at least* $1 - \frac{1}{\text{poly}(N)} - e^{-\Omega(t)}$.

Proof. The proof follows by upper and lower bounding the weight of the set Q after the t^{th} iteration denoted successful, which can then be used together to bound t at $O(\nu s)$ with high probability. The proof is provided hereafter.

For the sake of simplicity and generality, with we will refer of the approximation ratio of the ℓ_0 samplers as $(1+\eta)$.

Note that since for all $\boldsymbol{q} \in Q$ $w_0(\boldsymbol{q}) = 1$, $w_0(Q) \leq N$.

Now, we claim that for any integer $t \geq 1$ after the t^{th} iteration denoted successful,

$$N^{t/\nu s} \leq w_t(Q) \leq e^{\frac{(1+\eta)^2 t}{10(1-\eta)^2 \nu}} \cdot N. \tag{2.3}$$

Lemma 2.5. *For any integer $t \geq 1$, after the t^{th} iteration denoted successful, $N^{t/\nu s} \leq w_t(Q)$.*

Proof. Consider an arbitrary basis B^* of Q, defined by k points for some $k \leq \nu$. Since $B^* \subseteq Q$, $w_t(B^*) \leq w_t(Q)$. Thus, it suffices to show that $N^{t/\nu s} \leq w_t(B^*)$.

Now, observe that for the set of violators V for an iteration, if $V \neq \emptyset$, then $V \cap B^* \neq \emptyset$. This is because if we assume for contradiction that $V \cap B^* = \emptyset$, then for any point $\boldsymbol{e} \in B^*$, $\boldsymbol{e} \notin V$, meaning that $f(B \cup \{\boldsymbol{e}\}) \leq f(B)$. Further, from the monotonicity property, $f(B \cup \{\boldsymbol{e}\}) \geq f(B)$, and so $f(B \cup \{\boldsymbol{e}\}) = f(B)$.

Inductively repeating this for all points in B^* using the locality property, we get that $f(B \cup B^*) = f(B)$. Further, by the monotonicity property, we have that

$$f(B^*) \leq f(B \cup B^*) \leq f(Q).$$

But since B^* is a basis for Q, it must be that $f(B^*) = f(Q)$. Connecting these all together, we get that $f(B) = f(Q)$.

Finally, to achieve our contradiction, we can notice that in this case it cannot be that B has any violating points $\boldsymbol{e}'$, since by monotonicity $f(B \cup \{\boldsymbol{e}'\}) \leq f(Q) = f(B)$. So, we have a contradiction to the fact that $V \neq \emptyset$, and thus $V \cap B^* \neq \emptyset$.

Now for each $i \in [t]$, define B_i as the sample in the i^{th} iteration denoted successful. For any $l \in [k]$, let a_l be the number of samples B_i that are violated by $\boldsymbol{q}_l \in B^*$, i.e.,

$$a_l = \left|\{i \in [t] \mid f(B_i \cup \{\boldsymbol{q}_l\}) > f(B_i)\}\right|.$$

Since in each iteration denoted successful, there have been some violators (as the algorithm did not return after it), we know that $V \cap B^* \neq \emptyset$ for these iterations, and so there must exist at least one $\boldsymbol{q}_l$ which violated B_i for each $i \in [t]$. Thus, we have that $\sum_{l=1}^{k} a_l \geq t$.

Now, looking at B^*, we can see that

$$\begin{aligned}
w_t(B^*) &= \sum_{l=1}^{k} w_t(\boldsymbol{q}_l) \\
&= \sum_{l=1}^{k} \left(N^{1/s}\right)^{a_l} \\
&\geq k\left(N^{1/s}\right)^{\sum_{l=1}^{k} a_l/k} && \text{by Jensen's inequality} \\
&\geq k\left(N^{1/s}\right)^{t/k} \\
&\geq N^{t/\nu s} && \text{as } k \leq \nu
\end{aligned}$$

proving the lemma. □

Lemma 2.6. *For any integer $t \geq 1$, after the t^{th} iteration denoted successful, $w_t(Q) \leq e^{\frac{(1+\eta)^2 t}{10(1-\eta)^2\nu}} \cdot N$.*

Proof. First, by how the weights are set, we have that

$$w_{t+1}(Q) = w_t(SQ) + (N^{1/s} - 1) \cdot w_t(V) \leq w_t(Q) + N^{1/s} \cdot w_t(V). \tag{2.4}$$

We also know from our definition of a successful iteration that

$$\frac{1}{(1+\eta)} \sum_i f''_{i,t} N^{i/s} \leq \mu \cdot \frac{1}{(1-\eta)} \sum_i f'_{i,t} N^{i/s} \tag{2.5}$$

where the estimate f'_i on the t^{th} iteration is denoted $f'_{i,t}$ and similarly for $f''_{i,t}$.

We rewrite $w_t(Q)$ and $w_t(V)$ in this form, as $\sum_i f_{i,t,Q} N^{i/s}$ and $\sum_i f_{i,t,V} N^{i/s}$ respectively.

By the definition of our estimators, for any i we have that

$$(1-\eta) f_{i,t,Q} \leq f'_{i,t} \leq (1+\eta) f_{i,t,Q}$$

So, we have that $\frac{1}{(1-\eta)} f'_{i,t} \leq \frac{(1+\eta)}{(1-\eta)} f_{i,t,Q}$, and so multiplying by the weight of the class and summing over all weight classes we have that

$$\frac{1}{(1-\eta)} \sum_i f'_{i,t} N^{i/s} \leq \frac{(1+\eta)}{(1-\eta)} \sum_i f_{i,t,Q} N^{i/s} = \frac{(1+\eta)}{(1-\eta)} w_t(Q).$$

Similarly, as we have that

$$(1-\eta) f_{i,t,V} \leq f''_{i,t} \leq (1+\eta) f_{i,t,V},$$

we have that

$$\frac{(1-\eta)}{(1+\eta)} w_t(V) = \frac{(1-\eta)}{(1+\eta)} \sum_i f_{i,t,V} N^{i/s} \leq \frac{1}{(1+\eta)} \sum_i f''_{i,t} N^{i/s}.$$

Plugging these into (2.5), we get that

$$\frac{(1-\eta)}{(1+\eta)}w_t(V) \leq \mu \cdot \frac{(1+\eta)}{(1-\eta)}w_t(Q)$$

with probability at least $\frac{1}{\mathrm{poly}(N)}$, which comes from the error probability of the estimators. This can further be rearranged into

$$w_t(V) \leq \mu \cdot \frac{(1+\eta)^2}{(1-\eta)^2}w_t(Q).$$

Now, plugging this into (2.4), we get that

$$\begin{aligned}
w_t(Q) &\leq w_{t-1}(Q) + N^{1/s} \cdot \mu \cdot \frac{(1+\eta)^2}{(1-\eta)^2}w_{t-1}(Q) \\
&= \left(1 + N^{1/s} \cdot \mu \cdot \frac{(1+\eta)^2}{(1-\eta)^2}\right) w_{t-1}(Q) \\
&= \left(1 + N^{1/s} \cdot \frac{1}{10 \cdot \nu \cdot N^{1/s}} \cdot \frac{(1+\eta)^2}{(1-\eta)^2}\right) w_{t-1}(Q) \\
&= \left(1 + \frac{1}{10 \cdot \nu} \cdot \frac{(1+\eta)^2}{(1-\eta)^2}\right) w_{t-1}(Q).
\end{aligned}$$

We can further unroll this inequality until we get

$$w_t(Q) \leq \left(1 + \frac{(1+\eta)^2}{10\nu(1-\eta)^2}\right)^t w_0(Q) \leq e^{\frac{(1+\eta)^2 t}{10\nu(1-\eta)^2}} \cdot N$$

with probability at least $1 - \frac{1}{\mathrm{poly}(N)}$, proving the lemma. □

Connecting both sides of (2.3), we get $N^{t/\nu s} \leq e^{\frac{(1+\eta)^2 t}{10(1-\eta)^2\nu}} \cdot N$, which can be rearranged to get

$$\frac{t}{\nu} \leq \frac{10(1-\eta)^2}{10(1-\eta)^2 - (1+\eta)^2}s.$$

Now, taking $\eta = 1/4$, we can simplify this as $\frac{t}{\nu} \leq \frac{3}{2}s$, bounding the number of successful iterations t at $\frac{3}{2}\nu s$. Thus, the total number of iterations the algorithm takes is bounded by $O(\nu s)$ with probability at least $1 - \frac{1}{\mathrm{poly}(N)} - e^{-\Omega(t)}$. □

Chapter 3

Applications

In this section, we present some applications of our algorithm. As our algorithm is built in the multiple linear sketch model, it can be utilized for many big data models. We highlight the main challenges of adapting to these models and how we adapt our algorithm for them, achieving the results of Theorem 1.1.

Note that during the weighted sampling procedure, the large size of the input, and even our linear sketch, makes it prohibitively expensive to store the weight of each point, or to send it as part of messages. Notice however that the weight of a point only changes when it is a violator point for a successful iteration. Thus, instead of keeping the weight directly, we can store the solutions $f(B)$ of successful iterations. Since these solutions require little space (for example, the MEB solution is a center point and radius) they take significantly less storage space, and also are extremely cheap to send and receive in distributed models. Our algorithm can then check any point against these stored solutions to calculate its weight on the fly. We explain this idea further in Section 3.1

When adapting our sampling procedure to distributed models, there are two hurdles. First, one cannot take a sample of m points by asking machines to sample m points each, as this would lead to a load of $m \cdot k$ points. Further, since we are doing weighted sampling, machines need to sample in accordance with other machines' weights. In order to fix both of these problems, we employ the following protocol, which We describe for the coordinator model, noting that it can be implemented in the parallel computation model similarly by having one machine assume the role of the coordinator. Each machine sends the coordinator the total weight of their subset of points. The coordinator then calculates how many of the m sample points B each machine shall sample, and sends them this number. Now, the machines can sample a subset B_i, and send them to the coordinator. Notice that m points are sent in total now, instead of per machine, so the load is reduced to m. A similar procedure is followed to see if an iteration is successful, where each machine calculates and sends their total violator weight. This protocol is fully laid out in Section 3.3.

Finally, as our algorihm handles models with deletion of points, such as the strict turnstile model, we must make sure when constructing our metric ε-net around a central point, that the point will exist at the end of the stream. We also need to find an approximation for the distance of the furthest point from it, to define the size of our net. Failing to do so can lead to our net failing to follow Property (P3). The idea behind our fix is to use ℓ_0 samplers, which work in the presence of deletions. We first sample any non-deleted point, which becomes the center of our net. Now, as our input points are from the set $\{-\Delta, -\Delta+1, \ldots, \Delta\}^d$, we have an upper and lower bound on the distance any point can be from our sampled center. Such, we simply binary search over the powers of 2 in this range and sample a non-deleted point with norm larger than the current power of 2 in each iteration. When we find the largest power of 2 where a non-deleted point exists, the distance to that point gives us a 2-approximation of the largest distance from our center point. As the binary search takes $O\left(\log\log\left(\Delta d\right)\right)$ iterations, we get only a miniscule increase in pass complexity. We explain this procedure in detail in Section 3.2.

Using these techniques, we are able to utilize our algorithm in many big data models. We now present the specifics of these models, as well as our results for them. Then, we present different problems our algorithm can be used to solve, and the results for each problem.

3.1 Application in the Multipass Streaming Model

The first model we consider is the multipass streaming model. In this model, our input is presented as a stream of points P, and our algorithm is allowed to make multiple linear scans of this stream, while maintaining a small space complexity.

Our algorithm is well suited for handling a large stream of data, as it creates a linear sketch of the data points. Therefore, all of our sampling and estimation can easily be done with the aid of our ℓ_0 samplers and ℓ_0 estimators. The major consideration we have to make is in how we calculate and store the weights of each snapped point, because we use that to decide which estimators and samplers to use for each point. We clearly cannot store the weight of each snapped point, as that would take up space linear in the number of snapped points. However, one can notice that the weight of a snapped point changes only when it is a violator of a successful iteration, at which point it is multiplied by $N^{1/s}$. Thus, it suffices to instead store the solution $f(B)$ for each successful iteration. Now, the weight of a snapped point is simply $N^{v/s}$, where v is the number of stored solutions $f(B)$ it violates, which can be calculated on the fly efficiently.

Given this, we can calculate the number of passes needed by our algorithm. First, we see that each iteration of our algorithm requires two passes over the stream. As seen in the last paragraph, getting the weight of each snapped point while creating our sample can be done in one pass. Further, after calculating a solution $f(B)$, checking for violators can also be done in one pass. Finally, we require one initial pass over the data to set our origin point as well as to get the value of N.

The space complexity of our algorithm comes from the four quantities it stores, those being the ℓ_0 estimators, ℓ_0 samplers, previous solutions $f(B)$, and the current sample B. Note that we do not actually need to store the metric ε-net points, as with our construction being a lattice, we can quickly access the closest metric ε-net point to any point $\boldsymbol{p}_i \in P$.

In each iteration we have t $\left(1 \pm \frac{1}{m^{3/2}}\right)$-approximation ℓ_0 estimators with $\frac{1}{\mathrm{poly}(N)}$ error. Each of these estimators has a space usage of $O(m^3 \log N)$. We also have $2t$ $\left(1 \pm \frac{1}{4}\right)$-approximation ℓ_0 estimators with $\frac{1}{\mathrm{poly}(N)}$ error. Each of these estimators has a space usage of $O(\log N)$. Finally, we have t ℓ_0 samplers with with $\frac{1}{\mathrm{poly}(N)}$ error, each of which has a space usage of $\mathrm{polylog}\,(N)$. Given that t is bounded by $O(\nu s)$, and that m is in $O\left(\nu\lambda N^{1/s} \log\left(\nu\lambda N^{1/s}\right)\right)$, we have a total storage from all of these of $O\left(\nu^4 s \lambda^3 N^{3/s}\right) \cdot \mathrm{polylog}\,(\lambda, N)$. The t previous solutions can be stored in $O(\nu s) \cdot S_f$ space, where S_f is the storage needed to store a solution $f(B)$. Finally, a sample of m points can be stored in $O\left(\nu\lambda N^{1/s} \log\left(\nu\lambda N^{1/s}\right)\right) \cdot \mathrm{bit}(\boldsymbol{p})$ space, where $\mathrm{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$. Combining all of these, the total space complexity for our algorithm is

$$O\left(\nu^4 s \lambda^3 N^{3/s}\right) \cdot \mathrm{polylog}\,(\lambda, N) + O(\nu s) \cdot S_f + O\left(\nu\lambda N^{1/s} \log\left(\nu\lambda N^{1/s}\right)\right) \cdot \mathrm{bit}(\boldsymbol{p}).$$

We are mostly interested in the multi-pass case where $s = \log N$. First, note that in this case $N^{3/s}$ becomes constant. Further, we have that $\log N = \log\left(\lceil \log_{1+\varepsilon} r_m \rceil\right) + d \log\left(1 + \frac{2\sqrt{d}}{\varepsilon}\right)$. The first term diminishes quickly. Thus, we obtain a space complexity of

$$O\left(\nu^4 \lambda^3 d \log\left(\frac{d}{\varepsilon}\right)\right) \cdot \mathrm{polylog}\left(\lambda, d, \log\left(\frac{d}{\varepsilon}\right)\right) + O\left(\nu d \log\left(\frac{d}{\varepsilon}\right)\right) \cdot S_f + O\left(\nu\lambda \log\left(\nu\lambda\right)\right) \cdot \mathrm{bit}(\boldsymbol{p}).$$

The time complexity of our algorithm depends on three operations we perform in each iteration. The first operation is to create our sample B. This can be done in $O(m)$ time per iteration since each point can be sampled from our ℓ_0 samplers in linear time. The second operation is to calculate the solution for our sample, $f(B)$, which is done once per iteration. The third operation is to check if a snapped point is a violator, i.e. to calculate if $f\left(B \cup \{\boldsymbol{q}\}\right) > f(B)$. This operation is done for all points each iteration. For a given LP-type problem, we can denote the time bounds for these as T_B and T_V. Together, this gives us a total time complexity over $O(\nu s)$ iterations of

$$O\left(\nu s \left(T_V \cdot n + m + T_B\right)\right).$$

In order for our algorithm to be efficient in time complexity, we want T_V to be constant time, and T_B to be polynomial in d, and logarithmic in $\frac{1}{\varepsilon}$.

All together, we achieve the following result for the multipass streaming model.

Theorem 3.1. *For any $s \in [1, d \log({}^{1}/_{\varepsilon})]$, there exists a randomized algorithm to compute a $(1 + O(\varepsilon))$-approximation solution to an LP-type problem in the multipass streaming model, that takes $O(\nu s)$ passes over the input, with*

$$O\left(\nu^4 s \lambda^3 N^{3/s}\right) \cdot \mathrm{polylog}\,(\lambda, N) + O(\nu s) \cdot S_f + O\left(\nu\lambda N^{1/s} \log\left(\nu\lambda N^{1/s}\right)\right) \cdot \mathrm{bit}(\boldsymbol{p})$$

space complexity and $O\left(\nu s \left(T_V \cdot n + m + T_B\right)\right)$ time complexity, where S_f is the storage needed to store a solution $f(B)$, and $\mathrm{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$.

3.2 Application in the Strict Turnstile Model

In the strict turnstile model, there is an underlying vector $\boldsymbol{v}$ where the i^{th} element of the vector corresponds to a point $\boldsymbol{p}_i \in P$. This vector is initialized to $\mathbf{0}$. The input is then presented a stream of additive updates to the coordinates of $\boldsymbol{v}$, presented as $\boldsymbol{v} \leftarrow \boldsymbol{v} + \boldsymbol{e}_i$ or $\boldsymbol{v} \leftarrow \boldsymbol{v} - \boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the i^{th} standard unit vector. At the end of the stream, we are guaranteed that $\boldsymbol{v}$ is itemwise nonnegative. This stream can also be thought of as operations insert($\boldsymbol{p}_i$) and delete($\boldsymbol{p}_i$), with the guarantee that at the end of the stream, no point $\boldsymbol{p}_i$ has negative copies. Our algorithm is allowed to make multiple linear scans of this stream, while maintaining a small space complexity.

The application of our algorithm to this model is very similar to the multipass streaming model, because it is based on a linear sketch of the data points. Note that ℓ_0 samplers and estimators work in the presence of insertions and deletions of points. We again can use the same way of calculating the weights as in the multipass streaming model, meaning that the number of passes over the input stream is the same in each iteration.

The only additional hurdle presented by the strict turnstile model is that we cannot simply pick the first point in the data stream as our origin and center of our metric ε-net and get the maximum distance point in order to define N. This is because we need to make sure that the points we use here are not ones that will end up at zero copies at the end of the input stream. Instead, we present the following scheme. We first use an ℓ_0 sampler in order to sample a distinct non-deleted point $\boldsymbol{p}$, which we can then think of as our origin, and set up our metric ε-net around. We also notice that it suffices to get an $O(1)$-approximation of the largest norm of a (shifted) non-deleted point. Further, note that as each input point is in $\{-\Delta, -\Delta + 1, \ldots, \Delta\}^d$, this norm is upper bounded by $O\left(\Delta\sqrt{d}\right)$ and lower bounded by 1. We perform a binary search over the powers of 2 in this range. Since there are $O(\log(\Delta d))$ powers of 2 in the range, the binary search will take $O(\log\log(\Delta d))$ iterations. In each iteration of this search, we will use an ℓ_0 sampler in order to sample a distinct non-deleted point with norm above the current power of 2. When we find the largest such power of 2 where there exists a non-deleted point, we will have a 2-approximation of the largest norm from our origin point $\boldsymbol{p}_1$, and this can set our N. Each iteration of the binary search requires one pass over the data, and there is the initial pass to find $\boldsymbol{p}_1$, which gives us a total increase in the pass count of $O(\log\log(\Delta d))$ over the multipass streaming model.

The space and time complexities of our algorithm in the strict turnstile model match the multipass model for each iteration of the algorithm. Fort he initial creation of the net, we do constant time work to sample one point each iteration, giving us the following result.

Theorem 3.2. *For any $s \in [1, d\log(1/\varepsilon)]$, there exists a randomized algorithm to compute a $(1 + O(\varepsilon))$-approximation solution to an LP-type problem in the multipass streaming model, that takes $O(\nu s + \log\log(\Delta d))$ passes over the input, with*

$$O\left(\nu^4 s \lambda^3 N^{3/s}\right) \cdot \operatorname{polylog}(\lambda, N) + O(\nu s) \cdot S_f + O\left(\nu\lambda N^{1/s}\log\left(\nu\lambda N^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p})$$

space complexity and $O\left(\nu s\left(T_V \cdot n + m + T_B\right) + \log\log(\Delta d) \cdot n\right)$ time complexity, where S_f is the storage needed to store a solution $f(B)$, and $\operatorname{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$.

3.3 Application in the Coordinator Model

In the coordinator model, there are k distributed machines and a separate coordinator machine, which is connected to each machine via a two-way communication channel. The input set P is distributed among the machines, with each machine i getting a part P_i, and the goal is to jointly compute the solution $f(P)$ for the LP-type problem. Communication between the machines proceeds in rounds, where each round the coordinator can send a message to each machine, and then each machine can send a message back. The final computation is done by the coordinator. In the coordinator model, our algorithm aims to minimize the number of communication rounds and maximum bits of information sent or received in any round.

Again, notice that we can use the same idea of samplers and estimators at each machine, as well as the idea of on-the-fly weight calculation by storing previous solutions. The major implementation detail now is how to sample m points in the distributed setting and denote success of iterations. We will achieve this by the coordinator calculating what share of the m point sample each machine should be responsible for, and then building up an m point sample from the samples of each machine.

Our algorithm can do both of these in each round with three rounds of communication between the coordinator and the machines. First, at the start of each round, if the last round was deemed successful, the coordinator sends each machine the solution $f(B)$, for each machine to store and use in their weight calculations. The machines send the coordinator the weight of their subset of snapped points, $w(Q_i) = \sum_{\boldsymbol{q}_j \in Q_i} w(\boldsymbol{q}_j)$. Now, the coordinator generates m i.i.d. numbers $x_1, \ldots, x_m$ where each number $i \in k$ has probability of being picked according to the weight share of machine i, i.e. $\mathbf{Pr}\left[i \text{ is generated}\right] = \frac{w(Q_i)}{w(Q)}$ where $w(Q) = \sum_{i \in [k]} w(Q_i)$ is calculated by the coordinator. Starting the second round of communication, the coordinator sends each machine i a number $y_i = |\{l \mid x_l = i\}|$. Each machine then uses its ℓ_0 samplers to sample y_i points from its subset Q_i, according to the weight share of each snapped point $\boldsymbol{q}_j$, i.e. $\mathbf{Pr}\left[\boldsymbol{q}_j \text{ is sampled}\right] = \frac{w(\boldsymbol{q}_j)}{w(Q_i)}$, and sends this sample to the coordinator. This sampling system ensures that in total, $\mathbf{Pr}\left[\boldsymbol{q}_j \text{ is sampled}\right] = \frac{w(\boldsymbol{q}_j)}{w(Q_i)} \cdot \frac{w(Q_i)}{w(Q)} = \frac{w(\boldsymbol{q}_j)}{w(Q)}$, and thus the sample is correctly weighted. The coordinator then combines each machine's sample to create the total sample B, and calculates $f(B)$. For the final round of communication, the coordinator sends $f(B)$ to all machines, which they use to calculate their subset's violators. They send back the weight of their violator set, $w(V_i)$. If all of these $w(V_i)$'s are 0, then the algorithm is done. If not, the coordinator can calculate whether the iteration is successful, and continue to the next.

To complete our analysis, notice that the start of the algorithm requires two rounds of communication, one round for any machine to send a point to take as the origin, and a second round for the coordinator to send that point to the machines, and for the machines to send back their furthest distance from it.

To calculate the maximum load over a round of communication, we look at the messages being sent. The start of the algorithm has each message representable as a point, so the maximum load is $k \cdot \text{bit}(\boldsymbol{p})$. Then, in each iteration, we have four different messages to account for. A solution has size S_f, so sending it has load $k \cdot S_f$. Each weight being sent is bounded by $w(Q)$, which itself is upper bounded by $N \cdot \left(N^{1/s}\right)^t$ on the t^{th} iteration. Since we have $O(\nu s)$ iterations, we can bound the load of sending weights at $k \cdot O(\nu \log N)$. Each number y_i is bounded by m, giving a total load of $k \cdot \log m$. Finally, the samples being sent by each machine have in total m elements, thus the load is $m \cdot \operatorname{bit}(\boldsymbol{p})$.

All together, we achieve the following result for the coordinator model.

Theorem 3.3. *For any $s \in [1, d \log(1/\varepsilon)]$, there exists a randomized algorithm to compute a $(1 + O(\varepsilon))$-approximation solution to an LP-type problem in the coordinator model, that takes $O(\nu s)$ rounds of communication, with*

$$O\left(\left(k + \nu\lambda N^{1/s} \log\left(\nu\lambda N^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p}) + k \cdot S_f + k \log\left(\nu\lambda N^{1/s}\right) + k\nu \log N\right)$$

load, where S_f is the storage needed to store a solution $f(B)$, and $\operatorname{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$. The local computation time of the coordinator is $O(\nu s (m + T_B + k))$, and the local computation time of each machine i is $O(\nu s (n_i T_V))$ where $n_i = |P_i|$.

3.4 Application in the Parallel Computation Model

In the parallel computation model, there are k distributed machines, which are each connected via two-way communication channels. The input set P is distributed among the machines, with each machine i getting a part P_i, and the goal is to jointly compute the solution $f(P)$ for the LP-type problem. Communication between the machines proceeds in rounds, where each round the machines can communicate with each other in messages. In the coordinator model, our algorithm aims to minimize the number of communication rounds and maximum bits of information sent or received in any round.

Our procedure for running our algorithm in the parallel computation model is simply to run it as our algorithm for the coordinator model, assigning one of the machines to act as the coordinator. Thus, we can achieve the same bounds on the rounds of communication and maximum load. For the computation times, one machine will have a computation time on the order of the coordinator time added to the machine time, and other machines will have computation times same as in the coordinator model.

3.5 Solving the MEB Problem

One important application of our algorithm is in solving the MEB problem. Given n points $\boldsymbol{p}_i \in \{-\Delta, -\Delta + 1, \ldots, \Delta\}^d$, the objective of the problem is to find a d-dimensional ball $(\boldsymbol{c}, r) \in \mathbb{R}^d \times \mathbb{R}$ with center c and radius r that encloses all of the input points, i.e. where for all points $\boldsymbol{p}_i$, $d(\boldsymbol{c}, \boldsymbol{p}_i) \leq r$, and which has the smallest such radius r.

The MEB problem is an LP-type problem where S is the set of d-dimensional points, and $f(\cdot)$ is the function that maps from a set of points to their MEB. The combinatorial dimension of the

MEB problem is $\nu = d+1$, as any MEB in d dimensions defined by $d+2$ points P can be defined by any $d+1$ point subset of P. However, there are MEB instances defined by $d+1$ points that cannot be defined by a d point subset, e.g., 3 vertices of an equilateral triangle in 2 dimensions. Further, the VC dimension of the MEB set system is $\lambda = d+1$ [46].

Now, we show that the MEB problem satisfies the properties (P1), (P2), and (P3). For (P1), each point can be associated with the set of all balls that enclose it. For (P2), it can be seen that for a set of points, the intersection of sets of balls that enclose them is the set of balls that enclose the whole set. Thus, it is natural that the MEB is the minimal such ball. Finally, for (P3), we show that for an MEB $(\boldsymbol{c}, r)$ of a set of snapped points Q, the ball $(\boldsymbol{c}, (1+4\varepsilon)r)$ is a $(1+O(\varepsilon))$-approximation MEB of the set of original points P. We show this using two lemmas.

Lemma 3.4. *All points $\boldsymbol{p}_i \in P$ are inside the ball $(\boldsymbol{c}, (1+4\varepsilon)r)$, i.e., $d(\boldsymbol{c}, \boldsymbol{p}_i) \leq (1+4\varepsilon)r$.*

Proof. Say point $\boldsymbol{p}_i$ was snapped to $\|\boldsymbol{p}_i\| \, \boldsymbol{e}_k$ for some point $\boldsymbol{e}_k$ on the metric ε-net and rounded up to $\boldsymbol{q}_j$. We know $\|\boldsymbol{p}_i\| \, \boldsymbol{e}_k$ is between $(1+\varepsilon)^l \boldsymbol{e}_k$ and $(1+\varepsilon)^{l+1}\boldsymbol{e}_k$ for some l, and so $\|\boldsymbol{p}_i\| \in \left[(1+\varepsilon)^l, (1+\varepsilon)^{l+1}\right]$.

We first bound $d(\boldsymbol{c}, \boldsymbol{p}_i)$.

$$\begin{aligned}
d(\boldsymbol{c}, \boldsymbol{p}_i) &\leq d(\boldsymbol{c}, \boldsymbol{q}_j) + d(\boldsymbol{p}_i, \boldsymbol{q}_j) && \text{by the triangle inequality} \\
&\leq r + d(\boldsymbol{p}_i, \boldsymbol{q}_j) && \text{as } (\boldsymbol{c}, r) \text{ is the MEB of } Q \\
&\leq r + d(\boldsymbol{p}_i, \|\boldsymbol{p}_i\| \, \boldsymbol{e}_k) + d(\|\boldsymbol{p}_i\| \, \boldsymbol{e}_k, \boldsymbol{q}_j) && \text{by the triangle inequality} \\
&\leq r + \varepsilon \|\boldsymbol{p}_i\| + d(\|\boldsymbol{p}_i\| \, \boldsymbol{e}_k, \boldsymbol{q}_j) && \text{by the metric } \varepsilon\text{-net} \\
&\leq r + \varepsilon \|\boldsymbol{p}_i\| + (1+\varepsilon)^{l+1} - (1+\varepsilon)^l \\
&= r + \varepsilon \|\boldsymbol{p}_i\| + \varepsilon(1+\varepsilon)^l \\
&\leq r + \varepsilon \|\boldsymbol{p}_i\| + \varepsilon \|\boldsymbol{p}_i\| \\
&= r + 2\varepsilon \|\boldsymbol{p}_i\| .
\end{aligned}$$

Now, we know $\|\boldsymbol{q}_j\|$ is $(1+\varepsilon)^{l+1}$ as it was snapped down. Thus, since $\|\boldsymbol{p}_i\| \in \left[(1+\varepsilon)^l, (1+\varepsilon)^{l+1}\right]$, we can say that $\|\boldsymbol{p}_i\| \leq \|\boldsymbol{q}_j\|$. Plugging this in, we get

$$d(\boldsymbol{c}, \boldsymbol{p}_i) \leq r + 2\varepsilon \|\boldsymbol{q}_j\| .$$

Now, since $\|\boldsymbol{q}_j\|$ is $d(\boldsymbol{q}_j, \boldsymbol{p}_1)$ and $\boldsymbol{p}_1$ was unaffected by the snapping and rounding as the center of the ε-net, both $\boldsymbol{q}_j$ and $\boldsymbol{p}_1$ are in $(\boldsymbol{c}, r)$. This means their distance is bounded by $2r$, meaning $\|\boldsymbol{q}_j\| \leq 2r$. Plugging this in, we get

$$\begin{aligned}
d(\boldsymbol{c}, \boldsymbol{p}_i) &\leq r + 4\varepsilon r \\
&\leq (1+4\varepsilon)r.
\end{aligned}$$

□

Lemma 3.5. $(\boldsymbol{c}, r)$ *is no larger than* $(1+O(\varepsilon))$ *times the size of the MEB of the original points* $(\boldsymbol{c}^*, r^*)$*, i.e.,* $r \leq (1+O(\varepsilon))r^*$.

Proof. First, we know from the proof of Lemma 3.4 that $d(\boldsymbol{p}_i, \boldsymbol{q}_j) \leq 2\varepsilon \|\boldsymbol{p}_i\|$. As $\|\boldsymbol{p}_i\|$ is $d(\boldsymbol{p}_i, \boldsymbol{p}_1)$, we can use the fact the fact that the MEB $(\boldsymbol{c}^*, r^*)$ contains all points $\boldsymbol{p}_i$, to bound this distance by $2r^*$. Thus, $\|\boldsymbol{p}_i\| \leq 2r^*$. So we can say that $d(\boldsymbol{p}_i, \boldsymbol{q}_j) \leq 4\varepsilon r^*$.

Thus, since $(\boldsymbol{c}^*, r^*)$ encloses all the original points $\boldsymbol{p}_i \in P$, the ball $(\boldsymbol{c}^*, r^* + 4\varepsilon r^*)$ encloses all the snapped and rounded points $\boldsymbol{q}_j \in Q$.

Thus, $r \leq r^* + 4\varepsilon r^*$ as $(\boldsymbol{c}, r)$ is the MEB for Q, and so $r \leq (1 + 4\varepsilon)r^*$. $\square$

Together, these lemmas show that the ball $(\boldsymbol{c}, (1+4\varepsilon)r)$ is an enclosing ball for P, and is within $1 + O(\varepsilon)$ of the MEB for P, which together show (P3).

Thus, our algorithm can be utilized to solve the $(1 + 4\varepsilon)$-approximation MEB problem. Notice that a solution to the MEB problem is a ball $(\boldsymbol{c}, r)$ and can thus be stored in $2\,\mathrm{bit}(\boldsymbol{p})$ space. Further, violator checks can easily be done on a stored solution by checking whether for a snapped point $\boldsymbol{q}$, whether $d(\boldsymbol{c}, \boldsymbol{q}) \leq r$, giving us $T_V = O(1)$ for the MEB problem. The MEB of a sample of m points can be found in $O\left((m + d)^3\right)$ [49], which gives us T_B for the MEB problem. Therefore, we can achieve the following result for the MEB problem, using the results from Theorems 3.1, 3.2, and 3.3.

Theorem 3.6. *For any $s \in [1, d \log({}^1\!/\!\varepsilon)]$, there exists a randomized algorithm to compute a $(1 + 4\varepsilon)$-approximation solution to the MEB problem with high probability in the following models, where $\mathrm{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$, and $N = \lceil \log_{1+\varepsilon} r_m \rceil \left(1 + \frac{2\sqrt{d}}{\varepsilon}\right)^d$.*

- *Multipass Streaming: An $O(ds)$ pass algorithm with $O\left(d^7 s N^{3/s}\right) \cdot \mathrm{polylog}(d, N) + O\left(ds + d^2 N^{1/s} \log(dN^{1/s})\right) \cdot \mathrm{bit}(\boldsymbol{p})$ space complexity and $O\left(ds\left(n + (m + d)^3\right)\right)$ time complexity.*
- *Strict Turnstile: An $O(ds + \log\log(\Delta d))$ pass algorithm with $O\left(d^7 s N^{3/s}\right) \cdot \mathrm{polylog}(d, N) + O\left(ds + d^2 N^{1/s} \log(dN^{1/s})\right) \cdot \mathrm{bit}(\boldsymbol{p})$ space complexity and $O\left(ds\left(n + (m + d)^3\right)\right)$ time complexity.*
- *Coordinator / Parallel Computation: An algorithm with $O(ds)$ rounds of communication and $O\left(\left(k + d^2 N^{1/s} \log\left(dN^{1/s}\right)\right) \cdot \mathrm{bit}(\boldsymbol{p}) + k \log\left(dN^{1/s}\right) + kd \log N\right)$ load. The local computation time of the coordinator is $O\left(ds\left((m + d)^3 + k\right)\right)$, and the local computation time of each machine i is $O(dsn_i)$ where $n_i = |P_i|$.*

3.6 Solving the Linear SVM Problem

A seperate application of our algorithm is as a Monte Carlo application in solving the Linear SVM Problem. In the problem, we are given n points $\boldsymbol{p}_i = (\boldsymbol{x}_i, y_i)$ where $\boldsymbol{x}_i \in \left\{-1, -1 + \frac{1}{\Delta}, \ldots, 1\right\}^d$ and $y_i \in \{-1, +1\}$. We are also given some $\gamma > 0$ for which the points are either inseparable or γ-separable. The objective of the problem is to determine either that the points are inseparable, or to compute a d-dimensional hyperplane $(\boldsymbol{u}, b)$ where b is the bias term, which separates the points with the largest margin, i.e. to solve the quadratic optimization problem

$$\min_{\boldsymbol{u}\in\mathbb{R}^d} \|\boldsymbol{u}\|^2 \quad \text{subject to} \quad y_i\left(\boldsymbol{u}^T\boldsymbol{x}_i - b\right) \geq 1 \quad \text{for all} \quad i \in [n]. \tag{3.1}$$

The linear SVM problem is an LP-type problem where S is the set of tuples of d-dimensional points and ± 1 y values, and $f(\cdot)$ is the function that maps from a set of points to their optimal separating hyperplane, or to the value infeasible. The combinatorial dimension of the linear SVM problem is $\nu = d+1$ for a separable set, and $\nu = d+2$ for an inseparable set. For a separable set, this can be seen as any basis is in the form of a hyperplane with $y = \pm 1$ and a point with $y = \mp 1$. For an inseparable set, another point with $y = \mp 1$ on the opposite side of the hyperplane is needed. Thus, $\nu \leq d+2$. Further, the VC dimension of the linear SVM set system is $\lambda = d+1$ [46].

For the linear SVM problem, there is an additional hurdle to overcome. That is, if the points are too close together, running our algorithm with a metric ε-net might not retain separability. Therefore, we instead create a metric $\frac{\varepsilon\gamma}{2}$-net centered at the origin, where $\varepsilon < \frac{1}{2}$. This means that we have snapped points $\boldsymbol{q}_i = (\boldsymbol{z}_i, y_i)$ where $\boldsymbol{z}_j = \boldsymbol{x}_i + \boldsymbol{e}_i$ for some metric net point $\boldsymbol{e}_i$ with $\|\boldsymbol{e}_i\| \leq \frac{\varepsilon\gamma}{2}$. Our algorithm then finds the hyperplane that solves the quadratic optimization problem

$$\min_{\boldsymbol{u}\in\mathbb{R}^d} \|\boldsymbol{u}\|^2 \quad \text{subject to} \quad y_i\left(\boldsymbol{u}^T\boldsymbol{z}_i - b\right) \geq 1 \quad \text{for all} \quad i \in [N]. \tag{3.2}$$

The added benefit of this is that we do not need to use the additional search in the turnstile model, and can use the same metric net. We also run our algorithm as a Monte Carlo algorithm, where if we do not find a solution in the first $O(ds)$ iterations, we return that the point set is not separable. This gives us an algorithm with a one-sided error. That is, if a point set is inseparable, we will always output as such. If it is separable however, we will output a $(1+O(\varepsilon))$-approximation for the optimal separating hyperplane with high probability.

Given this new way of running our algorithm, we now show that the linear SVM problem satisfies the properties (P1), (P2), and (P3). As a constrained optimization problem, it satisfies (P1) and (P2) naturally. Each point is associated with the set of hyperplanes $(\boldsymbol{u}, b)$ that satisfy its constraint, and for a set of constraints, the solution minimizes $\|\boldsymbol{u}\|^2$ over hyperplanes that satisfy all constraints. Finally, for (P3), we show that for the optimal hyperplane $(\boldsymbol{u}, b)$ separating a set of snapped points Q, the hyperplane $((1+2\varepsilon)\boldsymbol{u}, (1+2\varepsilon)b)$ is a $(1+O(\varepsilon))$-approximation of the optimal hyperplane separating the set of original points P. We show this using two lemmas.

Lemma 3.7. *The hyperplane $((1+2\varepsilon)\boldsymbol{u}, (1+2\varepsilon)b)$ separates all original points $\boldsymbol{p}_i \in P$, i.e., $y_i\left((1+2\varepsilon)\,\boldsymbol{u}^T\boldsymbol{x}_i - (1+2\varepsilon)\,b\right) \geq 1$ for all $i \in [n]$.*

Proof. First, for any i, from our problem description, we know that $y_i\left(\boldsymbol{u}^T\boldsymbol{z}_i - b\right) \geq 1$. Working with that, we get

$$\begin{aligned} & y_i\left(\boldsymbol{u}^T\boldsymbol{z}_i - b\right) \geq 1 \\ \implies & y_i\left(\boldsymbol{u}^T\left(\boldsymbol{x}_i + \boldsymbol{e}_i\right) - b\right) \geq 1 \\ & \implies y_i\left(\boldsymbol{u}^T\boldsymbol{x}_i - b\right) \geq 1 - y_i\boldsymbol{u}^T\boldsymbol{e}_i. \end{aligned}$$

Now, we look at $y_i \boldsymbol{u}^T \boldsymbol{e}_i$. Since $y_i \in \{-1, +1\}$, we can say that $y_i \boldsymbol{u}^T \boldsymbol{e}_i \leq \left|\boldsymbol{u}^T \boldsymbol{e}_i\right|$, which itself is bounded by $\|\boldsymbol{u}\| \, \|\boldsymbol{e}_i\|$ by the Cauchy-Schwarz inequality.

By construction, we know that $\|\boldsymbol{e}_i\| \leq \frac{\varepsilon\gamma}{2}$.

Moreover, since the original points are γ-separable, and snapping to the metric net moves each point at most $\frac{\varepsilon\gamma}{2}$, we know that the snapped points are $(\gamma - \varepsilon\gamma)$-separable, and so $\frac{\gamma}{2}$-separable as $\varepsilon \leq \frac{1}{2}$. Thus, using the definitions of linear SVM

$$\begin{aligned} \frac{2}{\|\boldsymbol{u}\|} &\geq \frac{\gamma}{2} \\ \implies \|\boldsymbol{u}\| &\leq \frac{4}{\gamma}. \end{aligned}$$

So, plugging these two results in we have that $y_i \boldsymbol{u}^T \boldsymbol{e}_i \leq 2\varepsilon$. Thus, we have that

$$\begin{aligned} & y_i \left(\boldsymbol{u}^T \boldsymbol{x}_i - b\right) \geq 1 - 2\varepsilon \\ \implies & y_i \left(\frac{1}{1-2\varepsilon}\left(\boldsymbol{u}^T \boldsymbol{x}_i - b\right)\right) \geq 1 \\ \implies & y_i \left((1+2\varepsilon)\left(\boldsymbol{u}^T \boldsymbol{x}_i - b\right)\right) \geq 1 && \text{for small } \varepsilon \end{aligned}$$

giving us the desired result. □

Lemma 3.8. *The objective of Problem 3.2 is no larger than $(1 + O(\varepsilon))$ times the objective of Problem 3.1, i.e., $\|\boldsymbol{u}\|^2 \leq (1 + O(\varepsilon)) \|\boldsymbol{u}^*\|^2$.*

Proof. For small ε, $(1 + O(\varepsilon)) \|\boldsymbol{u}^*\|^2 \approx \|(1 + O(\varepsilon)) \boldsymbol{u}^*\|^2$, so it suffices to show that

$$\|\boldsymbol{u}\|^2 \leq \|(1 + O(\varepsilon)) \boldsymbol{u}^*\|^2 .$$

We know that for any i, $y_i \left(\boldsymbol{u}^{*T} \boldsymbol{x}_i - b^*\right) \geq 1$. So, by the same argument as Lemma 3.7

$$y_i \left((1+2\varepsilon) \boldsymbol{u}^{*T} \boldsymbol{z}_i - (1+2\varepsilon) b\right) \geq 1.$$

Thus, as $((1+2\varepsilon) \boldsymbol{u}^*, (1+2\varepsilon) b^*)$ is a solution to Problem 3.2, and $(\boldsymbol{u}, b)$ is the minimal solution to the optimization problem, $\|\boldsymbol{u}\|^2 \leq \|(1 + O(\varepsilon)) \boldsymbol{u}^*\|^2$ as desired. □

Together, these lemmas show that the hyperplane $((1+2\varepsilon)\boldsymbol{u}, (1+2\varepsilon)b)$ is a separating hyperplane for P, and is within $1 + O(\varepsilon)$ of the objective function for Problem 3.1, which together show (P3).

Thus, our algorithm can be utilized to solve the $(1+2\varepsilon)$-approximation linear SVM problem. Notice that a solution to the linear SVM problem is a hyperplane $(\boldsymbol{u}, b)$ and can thus be stored in $2\,\mathrm{bit}(\boldsymbol{p})$ space. Further, violator checks can easily be done on a stored solution by checking whether for a snapped point $\boldsymbol{q} = (\boldsymbol{z}, y)$, whether $y\left(\boldsymbol{u}^T \boldsymbol{z} - b\right) \geq 1$, giving us $T_V = O(1)$ for the linear SVM problem. The optimal separating hyperplane of a sample of m points can be found in $O\left((m+d)^3\right)$ [49], which gives us T_B for the linear SVM problem. Therefore, we can achieve the following result for the linear SVM problem, using the results from Theorems 3.1, 3.2, and 3.3.

Theorem 3.9. *For any $s \in [1, d\log({}^{1}/_{\varepsilon})]$, there exists a randomized algorithm to compute a $(1+2\varepsilon)$-approximation solution to the linear SVM problem with high probability in the following models, where $\operatorname{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$, and $N = \left(1 + \frac{4\sqrt{d}}{\varepsilon\gamma}\right)^d$.*

- *Multipass Streaming: An $O(ds)$ pass algorithm with $O\left(d^7 s N^{3/s}\right) \cdot \operatorname{polylog}(d, N) + O\left(ds + d^2 N^{1/s} \log(dN^{1/s})\right) \cdot \operatorname{bit}(\boldsymbol{p})$ space complexity and $O\left(ds\left(n + (m+d)^3\right)\right)$ time complexity.*
- *Strict Turnstile: An $O(ds)$ pass algorithm with $O\left(d^7 s N^{3/s}\right) \cdot \operatorname{polylog}(d, N) + O\left(ds + d^2 N^{1/s} \log(dN^{1/s})\right) \cdot \operatorname{bit}(\boldsymbol{p})$ space complexity and $O\left(ds\left(n + (m+d)^3\right)\right)$ time complexity.*
- *Coordinator / Parallel Computation: An algorithm with $O(ds)$ rounds of communication and $O\left(\left(k + d^2 N^{1/s} \log\left(dN^{1/s}\right)\right) \cdot \operatorname{bit}(\boldsymbol{p}) + k\log\left(dN^{1/s}\right) + kd\log N\right)$ load. The local computation time of the coordinator is $O\left(ds\left((m+d)^3 + k\right)\right)$, and the local computation time of each machine i is $O(dsn_i)$ where $n_i = |P_i|$.*

3.7 Solving Bounded Linear Programming Problems up to Additive Epsilon Error

So far, we have dealt with applications with multiplicative error, i.e. where our solution is within $(1 + O(\varepsilon))$ times the optimal solution. We can also use our algorithm to solve problems up to additive error. One application for this usage is in certain linear programming applications we will explore. Linear programs in general are optimization problems of the form

$$\max_{x \in \mathbb{R}^d} \boldsymbol{c}^T \boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{a}_i^T \boldsymbol{x} \le b_i \quad \text{for all} \quad i \in [n] \tag{3.3}$$

where the input is the objective vector $\boldsymbol{c}$ as well as n input constraints $\boldsymbol{p}_i = (\boldsymbol{a}_i, b_i)$.

We will work with a certain class of linear programs, where the input points, as well as the solution vector $\boldsymbol{x}$ is bounded, i.e., for all constraints i, $\|\boldsymbol{a}_i\|, |b_i| \in O(1)$, as well as $\|\boldsymbol{c}\|, \|\boldsymbol{x}\| \in O(1)$. Given this, we can run our algorithm by snapping each constraint point $\boldsymbol{p}_i$ to a metric $O(\varepsilon)$-net to get $\boldsymbol{a}'_i = \boldsymbol{a}_i + \boldsymbol{e}_i$ where $\|\boldsymbol{e}_i\| \le O(\varepsilon)$ and $b'_i = b_i + f_i$ where $|f_i| \le O(\varepsilon)$. As a note, this construction means that we require no additional passes in the turnstile model to create our net, similar to the linear SVM problem. This snapping then gives us the LP

$$\max_{\boldsymbol{x} \in \mathbb{R}^d} \boldsymbol{c}^T \boldsymbol{x} \quad \text{subject to} \quad \boldsymbol{a}'^T_i \boldsymbol{x} \le b'_i \quad \text{for all} \quad i \in [N]\,. \tag{3.4}$$

The optimal solution to LP (3.4), $\boldsymbol{x}$, then gives an additive ε-approximation solution to LP (3.3) as well.

Linear programs are the eponymous LP-type program, where S is the set of constraints of the LP and $f(\cdot)$ is the function that maps the set of constraints to their optimal solution, or to the value infeasible. An important note is that there might be multiple optimal solutions for a set of constraints, in which case any tie-breaking system may be used (a common one is to take the lexicographically smallest optimal point). The combinatorial dimension of linear programming is $\nu = d$, as d is the number of variables of the LP, and the VC dimension is $\lambda = d + 1$, since each constraint for an LP induces a feasible half-space [36, 46].

We now show that linear programs satisfy the properties (P1), (P2), and (P3). As a constrained optimization problems, they satisfy (P1) and (P2) naturally. Each constraint is associated with the halfspace that satisfies it, and for a set of constraints, the objective function is maximized over points that satisfy all constraints. Finally, we show a modified (P3) as we are now deal with additive approximations. We show that for the optimal solution $\boldsymbol{c}^T\boldsymbol{x}$ satisfying a set of snapped constraints Q, the solution $\boldsymbol{x}$ gives an additive ε-approximation to the optimal solution $\boldsymbol{c}^T\boldsymbol{x}^*$ approximately satisfying the original constraints P. We show this using two lemmas.

Lemma 3.10. *The solution $\boldsymbol{x}$ approximately satisfies all original constraint points $\boldsymbol{p}_i \in P$, i.e., $\boldsymbol{a}_i^T\boldsymbol{x} \leq b_i + O(\varepsilon)$ for all $i \in [n]$.*

Proof. First, for any i, from our problem description, we know that $\boldsymbol{a}_i'^T\boldsymbol{x} \leq b_i'$. Working with that, we get

$$\begin{aligned} \boldsymbol{a}_i'^T\boldsymbol{x} &\leq b_i' \\ \implies (\boldsymbol{a}_i + \boldsymbol{e}_i)^T\boldsymbol{x} &\leq b_i + f_i \\ \implies \boldsymbol{a}_i^T\boldsymbol{x} &\leq b_i - \boldsymbol{e}_i^T\boldsymbol{x} + f_i \end{aligned}$$

Now, we look at $-\boldsymbol{e}_i^T\boldsymbol{x}$. It is bounded by $\left|\boldsymbol{e}_i^T\boldsymbol{x}\right|$, which itself is bounded by $\|\boldsymbol{e}_i\|\,\|\boldsymbol{x}\|$ by the Cauchy-Schwarz inequality. By construction, we know that $\|\boldsymbol{e}_i\| \leq O(\varepsilon)$, and as given by our class of problem, $\|\boldsymbol{x}\| \in O(1)$. So, plugging these two results in we have that $-\boldsymbol{e}_i^T\boldsymbol{x} \leq O(\varepsilon)$. Further, we have that f_i is bounded by $O(\varepsilon)$ by construction. Thus, together we have that

$$\boldsymbol{a}_i^T\boldsymbol{x} \leq b_i + O(\varepsilon)$$

as desired. □

Lemma 3.11. *The objective of LP* (3.4) *is at most $O(\varepsilon)$ smaller than the objective of LP* (3.3)*, i.e., $\boldsymbol{c}^T\boldsymbol{x} \geq \boldsymbol{c}^T\boldsymbol{x}^* - O(\varepsilon)$.*

Proof. We know that for any i, $\boldsymbol{a}_i^T\boldsymbol{x}^* \leq b_i$. So, by the same argument as Lemma 3.10

$$\boldsymbol{a}_i'^T\boldsymbol{x}^* \leq b_i' + O(\varepsilon)$$

Thus, $\boldsymbol{x}^*$ is a solution to the additive ε-approximation of LP (3.4). So, if we denote the optimal solution to it as $\boldsymbol{x}'$, we have that $\boldsymbol{c}^T\boldsymbol{x}' \geq \boldsymbol{c}^T\boldsymbol{x}^*$. It then only remains to show that the optimal solution to the additive ε-approximation of LP (3.4) $\boldsymbol{x}'$ is not too much larger than the optimal solution to the exact LP (3.4). This follows because both $\|\boldsymbol{c}\|$ and $\|\boldsymbol{x}\|$ are bounded in $O(1)$. Allowing the constraints to be approximately satisfied by $O(\varepsilon)$ can make the optimal point increase in norm by at most a $(1+O(\varepsilon))$ multiplicative factor, and thus an additive $O(\varepsilon)$ factor for bounded norm. Thus, we have that $\boldsymbol{c}^T\boldsymbol{x}' \le \boldsymbol{c}^T\boldsymbol{x} + O(\varepsilon)$. Plugging this in, we get

$$\boldsymbol{c}^T\boldsymbol{x} \ge \boldsymbol{c}^T\boldsymbol{x}^* - O(\varepsilon)$$

as desired. □

Together, these lemmas show that the solution $\boldsymbol{x}$ satisfies the constraints P, and is within additive $O(\varepsilon)$ of the objective function for LP (3.3), which together show (P3).

Thus, our algorithm can be utilized to solve additive ε-approximation LP problems. Notice that a solution to an LP is a point $\boldsymbol{x}$ and can thus be stored in $\mathrm{bit}(\boldsymbol{p})$ space. Further, violator checks can easily be done on a stored solution by checking whether it satisfies a given constraint, giving us $T_V = O(1)$ for LP problems. The optimal solution to an LP with d variables and m constraints can be found in $\tilde{O}\left(md + d^{2.5}\right)$ [45], which gives us T_B for LP problems. Therefore, we can achieve the following result for bounded LP problems, using the results from Theorems 3.1, 3.2, and 3.3.

Theorem 3.12. *For any $s \in [1, d\log(1/\varepsilon)]$, there exists a randomized algorithm to compute an additive ε-approximation solution to bounded LP problems with high probability in the following models, where $\mathrm{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$, and $N \in O\left(\frac{\sqrt{d}}{\varepsilon}\right)^d$.*

- *Multipass Streaming: An $O(ds)$ pass algorithm with $O\left(d^7 s N^{3/s}\right) \cdot \mathrm{polylog}(d,N) + O\left(ds + d^2 N^{1/s}\log(dN^{1/s})\right)\cdot \mathrm{bit}(\boldsymbol{p})$ space complexity and $\tilde{O}\left(ds\left(n + md + d^{2.5}\right)\right)$ time complexity.*
- *Strict Turnstile: An $O(ds)$ pass algorithm with $O\left(d^7 s N^{3/s}\right) \cdot \mathrm{polylog}(d,N) + O\left(ds + d^2 N^{1/s}\log(dN^{1/s})\right)\cdot \mathrm{bit}(\boldsymbol{p})$ space complexity and $\tilde{O}\left(ds\left(n + md + d^{2.5}\right)\right)$ time complexity.*
- *Coordinator / Parallel Computation: An algorithm with $O(ds)$ rounds of communication and $O\left(\left(k + d^2 N^{1/s}\log\left(dN^{1/s}\right)\right)\cdot \mathrm{bit}(\boldsymbol{p}) + k\log\left(dN^{1/s}\right) + kd\log N\right)$ load. The local computation time of the coordinator is $\tilde{O}\left(ds\left(md + d^{2.5} + k\right)\right)$, and the local computation time of each machine i is $O(dsn_i)$ where $n_i = |P_i|$.*

3.7.1 Solving the Linear Classification Problem

One example of a bounded LP where our algorithm can be used to achieve a useful result is the linear classification problem. In the problem, we are given n labeled examples $\boldsymbol{p}_i$, comprising of a bounded d-dimensional feature vector $\boldsymbol{x}_i \in \left\{-1, -1+\frac{1}{\Delta}, \ldots, 1\right\}^d$ and a corresponding label $y_i \in \{-1, +1\}$. The goal of the problem is to find a separating hyperplane $\boldsymbol{u}$, which can be thought of as a normal vector $\boldsymbol{u} \in \mathbb{R}^d$ where $\|\boldsymbol{u}\| = 1$, such that $y_i\left(\boldsymbol{x}_i^T\boldsymbol{u}\right) \ge 0$ for all points $\boldsymbol{p}_i$. We will consider the related approximate optimization problem, where we assume that there is an optimal classifier $\boldsymbol{u}^*$ such that $y_i\left(\boldsymbol{x}_i^T\boldsymbol{u}^*\right) \ge \varepsilon$ for all points $\boldsymbol{p}_i$. We thus want to find a classifier that is within additive ε of the separation of this optimal classifier.

This problem can be written as a bounded LP as such. First, notice that we have two types of constraints, $\boldsymbol{x}_i^T \boldsymbol{u} \leq 0$ for any point $\boldsymbol{p}_i$ with $y_i = -1$, and $\boldsymbol{x}_i^T u \geq 0$ for any point $\boldsymbol{p}_i$ with $y_i = +1$. For simplicity, we will actually consider the negation of any such example, so we will actually consider points $\boldsymbol{p}'_i = (\boldsymbol{x}'_i, y'_i)$ where $\boldsymbol{x}'_i = -\boldsymbol{x}_i$ and $y'_i = -y_i$ if $y_i = +1$, and $\boldsymbol{x}'_i = \boldsymbol{x}_i$ and $y'_i = y_i$ otherwise. This allows us to only have constraints of the type

$$\boldsymbol{x}_i'^T \boldsymbol{u} \leq 0 \quad \text{for all} \quad i \in [n] \, .$$

Our objective is to maximize the separation of $\boldsymbol{u}$, which is $min_{i \in [n]} \boldsymbol{x}_i'^T \boldsymbol{u}$. While this objective isn't linear as is, it can be made linear by utilizing an additional variable representing the separation, and n additional constraints.

Thus, we can use our general framework for bounded LP problems in order to solve this problem with separation that is within additive ε of the optimal hyperplane's separation. This gives us a very important benefit. Namely, since the optimal hyperplane's separations is ε, our hyperplane will in fact be a separating hyperplane for the original points, meaning that our solution fully satisfies the original constraints P. Thus, we are able to solve the linear classification problem with an additive ε-approximation of the largest separation, in the bounds given in Theorem 3.12.

3.8 Solving Bounded SDP Problems up to Additive Epsilon Error

Another additive ε-approximation application of our algorithm is in solving bounded semidefinite programming problems. These problems are optimization problems of the form

$$\max_{X \in \mathbb{R}^{d \times d}} \langle C, X \rangle_F \quad \text{subject to} \quad \begin{array}{c} \langle A^{(i)}, X \rangle_F \leq b^{(i)} \quad \text{for all} \quad i \in [n] \\ X \succeq 0 \end{array} \tag{3.5}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product, i.e. $\langle A, B \rangle_F = \sum_{i,j} A_{ij} B_{ij}$, and $X \succeq 0$ denotes X as a positive semidefinite matrix. The input will be the objective matrix C as well as n input constraints $\boldsymbol{p}_i = \left(A^{(i)}, b^{(i)}\right)$.

We will work with a certain bounded class of SDP problems. Namely, we will have the following boundedness assumptions.

- X has unit trace, i.e. $\mathrm{Tr}(X) = 1$,
- $\|C\|_F \leq 1$ where $\|\cdot\|_F$ is the Frobenius norm.

Further, for all constraints i

- $\left\|A^{(i)}\right\|_2 \leq 1$ where $\|\cdot\|_2$ is the spectral norm,
- $\left\|A^{(i)}\right\|_F \leq F$,
- The number of nonzero entries of $A^{(i)}$ is bounded by S,

- $\left|b^{(i)}\right| \leq 1$.

Our algorithm solves these problems by solving them as bounded LP problems with the solution vector $\boldsymbol{x} \in \mathbb{R}^{d^2}$ where $\boldsymbol{x} = \operatorname{Vec}(X)$. This gives us the natural objective function $\max_{\boldsymbol{x}\in\mathbb{R}^{d^2}} \boldsymbol{c}^T\boldsymbol{x}$ where $\boldsymbol{c} = \operatorname{Vec}(C)$. For the constraints of these LPs, we consider the two different type of SDP constraints separately. First, we have constraints of the type $\langle A^{(i)}, X\rangle_F \leq b^{(i)}$, which we can represent as $\boldsymbol{a}^{(i)T}\boldsymbol{x} \leq b^{(i)}$ where $\boldsymbol{a}^{(i)} = \operatorname{Vec}(A^{(i)})$. We also have the constraint that X must be positive semidefinite, i.e. $X \succeq 0$. This is equivalent to the constraints $\boldsymbol{y}^T X \boldsymbol{y} \geq 0$ for all vectors $\boldsymbol{y} \in \mathbb{R}^d$ where $\|\boldsymbol{y}\| = 1$. We can represent each of these as constraints $(\boldsymbol{y} \otimes \boldsymbol{y})^T x$. The problem however is that we would need an infinite amount of these constraints, as there are infinite $\boldsymbol{y}$ vectors to consider.

Thus, we again utilize metric ε-nets. For each $A^{(i)}$, we note that a naive lattice where each coordinate $A^{(i)}_{jk}$ is snapped to the nearest multiple of $\frac{\varepsilon}{d}$ would suffice, however we can achieve a tighter bound because of the assumption that $A^{(i)}$ has at most S nonzero entries. We can therefore create a net for each of the $\binom{d^2}{S}$ possible combinations, where each net size is exponential in S rather than d^2. Now, for any $A^{(i)}$, its nonzero coordinates must be fully captured by at least one of these nets, and so we can deterministically choose one and snap it to a point in that net. We can further snap to the nearest multiple of $\frac{\varepsilon}{\min(d,S)}$ in order to decrease the size of each net. Finally, notice that for each coordinate of $A^{(i)}$, $\left|A^{(i)}_{jk}\right| \leq \left\|A^{(i)}\right\|_2 \leq 1$. Thus each net is of size $\left(\frac{2\min(d,S)}{\varepsilon} + 1\right)^S$, and therefore we have a total size of $\binom{d^2}{S} \cdot O\left(\left(\frac{\min(d,S)}{\varepsilon}\right)^S\right)$ for the nets. For each b_i, we can snap it to the nearest multiple of ε, giving a net of size $O\left(\frac{1}{\varepsilon}\right)$. In total, this means that in our snapped LP, we have $\binom{d^2}{S} \cdot O\left(\left(\frac{\min(d,S)}{\varepsilon}\right)^S \frac{1}{\varepsilon}\right)$ constraints of the form $\langle A'^{(i)}, X\rangle_F \leq b'^{(i)}$.

For the positive semidefinite constraints, we create a lattice where each coordinate is a multiple of $\frac{\varepsilon}{d\sqrt{d}}$. This gives us a metric ε-net of size $O\left(\left(\frac{d\sqrt{d}}{\varepsilon}\right)^d\right)$, and so we can reduce the infinite constraint space into that many constraints of the form $\boldsymbol{z}^T X \boldsymbol{z} \geq 0$ where $\boldsymbol{z} \in \mathbb{R}^d$ are points on the metric ε-net. We can call the original (infinitely numerous) constraints Y, and the new constraints Z.

Because our algorithm now solves an LP, our combinatorial dimension is $\nu = d^2$, and our VC dimension is $\lambda = d^2 + 1$. Further, the problems satisfy the properties (P1) and (P2). Finally, because of the different formulation of our nets, we have to show (P3), again. We show that for the optimal solution $\langle C, X\rangle_F$ satisfying a set of snapped constraints Q and Z, the solution $X + \frac{3\varepsilon}{d}I$, where I is the $d \times d$ identity matrix, gives an additive ε-approximation to the optimal solution $\langle C, X^*\rangle_F$ approximately satisfying the original constraints P and Y. We show this using three lemmas.

Lemma 3.13. *The solution $X + \frac{3\varepsilon}{d}I$ approximately satisfies all original constraint points $\boldsymbol{p}_i \in P$, i.e., $\langle A^{(i)}, X + \frac{3\varepsilon}{d}I\rangle_F \leq b^{(i)} + O(\varepsilon)$ for all $i \in [n]$.*

Proof. First, for any i, from our problem description, we know that $\langle A'^{(i}, X\rangle_F \le b'^{(i)}$. Working with that, we get

$$\begin{aligned}
\langle A'^{(i}, X\rangle_F &\le b'^{(i)} \\
\implies \sum_{j,k} A'^{(i)}_{jk} X_{jk} &\le b'^{(i)} \\
\implies \sum_{j,k} \left(A^{(i)}_{jk} + e^{(i)}_{jk}\right) X_{jk} &\le b^{(i)} + f^{(i)} \\
\implies \sum_{j,k} A^{(i)}_{jk} X_{jk} &\le b^{(i)} + f^{(i)} + \sum_{j,k} -e^{(i)}_{jk} X_{jk}.
\end{aligned}$$

First, we look at $f^{(i)}$. It is bounded by $\left|f^{(i)}\right|$, which by construction is bounded by ε. Secondly, we look at $\sum_{j,k} -e^{(i)}_{jk} X_{jk}$. That is bounded by $\sum_{j,k} \left|e^{(i)}_{jk} X_{jk}\right| = \sum_{j,k} \left|e^{(i)}_{jk}\right| |X_{jk}|$. We can bound this quantity in two ways. First, by construction, $\left|e^{(i)}_{jk}\right| \le \frac{\varepsilon}{\min(d,S)}$. Thus, we get that

$$\begin{aligned}
\sum_{j,k} \left|e^{(i)}_{jk}\right| |X_{jk}| &\le \sum_{j,k} \frac{\varepsilon}{\min(d,S)} |X_{jk}| \\
&= \frac{\varepsilon}{\min(d,S)} \sum_{j,k} |X_{jk}| \\
&= \frac{\varepsilon}{\min(d,S)} \|X_{jk}\| && \text{where } \|\cdot\| \text{ is the Entry-wise 1-norm} \\
&\le \frac{\varepsilon}{\min(d,S)} d \|X_{jk}\|_* && \text{where } \|\cdot\|_* \text{ is the Schatten 1-norm} \\
&= \frac{d\varepsilon}{\min(d,S)}.
\end{aligned}$$

Secondly, for all j, k, we have that $|X_{jk}| \le \|X\|_2 \le \|X\|_F \le \|X\|_* = 1$. We can thus plug this in to get

$$\begin{aligned}
\sum_{j,k} \left|e^{(i)}_{jk}\right| |X_{jk}| &\le \sum_{j,k} \left|e^{(i)}_{jk}\right| \\
&\le S \frac{\varepsilon}{\min(d,S)} && \text{since at most } S\ e^{(i)}_{jk}\text{'s are nonzero.}
\end{aligned}$$

Thus, putting these together, we have that $\sum_{j,k} \left|e^{(i)}_{jk}\right| |X_{jk}| \le \varepsilon$. So, plugging these in, we get the result

$$\langle A^{(i)}, X\rangle_F \le b^{(i)} + 2\varepsilon.$$

Now, we can explore $\langle A^{(i)}, X + \frac{3\varepsilon}{d} I\rangle_F$. We can see that

$$\begin{aligned}
\langle A^{(i)}, X + \frac{3\varepsilon}{d} I\rangle_F &= \sum_j A^{(i)}_{jj}\left(X_jj + \frac{3\varepsilon}{d}\right) + \sum_{j\neq k} A^{(i)}_{jk} X_{jk} \\
&= \langle A^{(i)}, X\rangle_F + \frac{3\varepsilon}{d}\sum_j A^{(i)}_{jj} \\
&\leq \langle A^{(i)}, X\rangle_F + \frac{3\varepsilon}{d} d \left\|A^{(i)}\right\|_2 \\
&\leq b^{(i)} + 2\varepsilon + 3\varepsilon \\
&= b^{(i)} + 5\varepsilon
\end{aligned}$$

as desired. □

Lemma 3.14. *The solution $X + \frac{3\varepsilon}{d} I$ satisfies all constraints $\boldsymbol{y} \in Y$, i.e., $\boldsymbol{y}^T(X + \frac{3\varepsilon}{d} I)\boldsymbol{y} \geq 0$ for all $\boldsymbol{y} \in \mathbb{R}^d$ such that $\|\boldsymbol{y}\| = 1$.*

Proof. First, from our lattice construction, we know that $\boldsymbol{z}^T X \boldsymbol{z} \geq 0$ for $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{e}$, where $\|\boldsymbol{e}\| \leq \frac{\varepsilon}{d}$. If $\boldsymbol{e} = \vec{0}$ then we directly have the desired bounds. Assuming $\boldsymbol{e} \neq \vec{0}$, we get

$$\begin{aligned}
&\boldsymbol{z}^T X \boldsymbol{z} \geq 0 \\
\implies &(\boldsymbol{y} + \boldsymbol{e})^T X (\boldsymbol{y} + \boldsymbol{e}) \geq 0 \\
\implies &\boldsymbol{y}^T X \boldsymbol{y} \geq -\boldsymbol{e}^T X \boldsymbol{y} - \boldsymbol{y}^T X \boldsymbol{e} - \boldsymbol{e}^T X \boldsymbol{e}.
\end{aligned}$$

First, we look at $\boldsymbol{e}^T X \boldsymbol{e}$. Notice that this is bounded by $\left|\boldsymbol{e}^T X \boldsymbol{e}\right|$, which is bounded by $\|\boldsymbol{e}\| \|X\boldsymbol{e}\|$ by Cauchy-Schwarz. Further, $\|X\boldsymbol{e}\|$ is bounded by $\|X\|_2 \|\boldsymbol{e}\|$. We already saw in Lemma 3.13 that $\|X\|_2$ is bounded by 1, and by construction $\|\boldsymbol{e}\| \leq \frac{\varepsilon}{d}$. Thus $\boldsymbol{e}^T X \boldsymbol{e} \leq \left(\frac{\varepsilon}{d}\right)^2 \leq \frac{\varepsilon}{d}$.

Now, looking at $\boldsymbol{e}^T X \boldsymbol{y}$ and $\boldsymbol{y}^T X \boldsymbol{e}$, we can use similar logic to bound both by $\left\|\boldsymbol{e}^T\right\| \|\boldsymbol{y}\| \|X\|$. Since $\|\boldsymbol{y}\| = 1$ by construction, these terms are then bounded by $\frac{\varepsilon}{d}$.

Plugging these in, we get the result

$$\boldsymbol{y}^T X \boldsymbol{y} \geq -3\frac{\varepsilon}{d}.$$

Now, we can look at $\boldsymbol{y}^T(X + \frac{3\varepsilon}{d} I)\boldsymbol{y}$. We see that

$$\begin{aligned}
\boldsymbol{y}^T(X + \frac{3\varepsilon}{d} I)\boldsymbol{y} &= \boldsymbol{y}^T X \boldsymbol{y} + \frac{3\varepsilon}{d}\boldsymbol{y}^T I \boldsymbol{y} \\
&= \boldsymbol{y}^T X \boldsymbol{y} + \frac{3\varepsilon}{d}\|\boldsymbol{y}\|^2 \\
&\geq -\frac{3\varepsilon}{d} + \frac{3\varepsilon}{d} \\
&= 0
\end{aligned}$$

as desired. □

Lemma 3.15. *The objective of the snapped LP is at most $O(\varepsilon)$ smaller than the objective of SDP* (3.5)*, i.e.,* $\langle C, X + \frac{3\varepsilon}{d} I\rangle_F \geq \langle C, X^*\rangle_F - O(\varepsilon)$.

Proof. We know that for any i, $\langle A^{(i)}, X^*\rangle_F \leq b^{(i)}$. Further, $X^* \succeq 0$ So, by the same argument as Lemmas 3.13 and 3.14

$$\langle A'^{(i)}, X^*\rangle_F \leq b'^{(i)} + 2\varepsilon \quad \text{and} \quad \boldsymbol{z}^T X^* \boldsymbol{z} \geq -\frac{3\varepsilon}{d}.$$

Thus, X^* is a solution to the additive ε-approximation of the snapped LP. So, if we denote the optimal solution to it as X', we have that $\langle C, X'\rangle_F \geq \langle C, X^*\rangle_F$. It then only remains to show that the optimal solution to the additive ε-approximation of the snapped LP X' is not too much larger than the optimal solution to the exact snapped LP. This follows because both $\|C\|_F$ and $\|X\|_f$ are bounded by 1. the LP formulation of the SDP has bounded vectors c and x. So, we can use the same analysis as done for the proof of 3.11. So, we have that $\langle C, X'\rangle_F \leq \langle C, X\rangle_F + O(\varepsilon)$. Plugging this in, we get

$$\langle C, X\rangle_F \geq \langle C, X^*\rangle_F - O(\varepsilon).$$

Finally, it only remains to show that

$$\langle C, X + \frac{3\varepsilon}{d} I\rangle_F \geq \langle C, X\rangle_F - O(\varepsilon),$$

This also follows similarly to the analysis of Lemma 3.13, such that

$$\begin{aligned}\langle C, X + \frac{3\varepsilon}{d} I\rangle_F &= \langle C, X\rangle_F + \frac{3\varepsilon}{d}\sum_j C_{jj} \\ &\geq \langle C, X\rangle_F - \frac{3\varepsilon}{d}\left|\sum_j C_{jj}\right| \\ &\geq \langle C, X\rangle_F - \frac{3\varepsilon}{d} d \|C\|_2 \\ &\geq \langle C, X\rangle_F - 3\varepsilon \|C\|_F \\ &\geq \langle C, X\rangle_F - 3\varepsilon\end{aligned}$$

as desired. □

Together, these lemmas show that the solution $X + \frac{3\varepsilon}{d} I$ satisfies the constraints P and Y, and is within additive $O(\varepsilon)$ of the objective function for SDP (3.5), which together show (P3).

Thus, our algorithm can be utilized to solve additive ε-approximation SDP problems. Notice that a solution to an SDP is a point $(X + \frac{3\varepsilon}{d} I)$ and can thus be stored in $\mathrm{bit}(\boldsymbol{p})$ space. Further, violator checks can easily be done on a stored solution by checking whether it satisfies a given constraint, giving us $T_V = O(1)$ for SDP problems. For the bounds of the problem, notice that we convert an SDP problem into an LP problem on d^2 dimensions. Thus, we can use the T_B for LPs with d^2 variables and m constraints. Therefore, we can achieve the following result for bounded SDP problems, using the results from Theorems 3.1, 3.2, and 3.3.

Theorem 3.16. *For any $s \in [1, d^2 \log(1/\varepsilon)]$, there exists a randomized algorithm to compute an additive ε-approximation solution to bounded SDP problems with high probability in the following models, where $\mathrm{bit}(\boldsymbol{p})$ is the bit complexity to store one point $\boldsymbol{p} \in P$, and $N \in \binom{d^2}{S} \cdot O\left(\left(\frac{\min(d,S)}{\varepsilon}\right)^S \frac{1}{\varepsilon}\right) + O\left(\left(\frac{d\sqrt{d}}{\varepsilon}\right)^d\right)$.*

- *Multipass Streaming: An $O(d^2 s)$ pass algorithm with $O\left(d^{14} s N^{3/s}\right) \cdot \mathrm{polylog}(d, N) + O\left(d^2 s + d^4 N^{1/s} \log(dN^{1/s})\right) \cdot \mathrm{bit}(\boldsymbol{p})$ space complexity and $\tilde{O}\left(d^2 s\left(n + md^2 + d^5\right)\right)$ time complexity.*
- *Strict Turnstile: An $O(d^2 s)$ pass algorithm with $O\left(d^{14} s N^{3/s}\right) \cdot \mathrm{polylog}(d, N) + O\left(d^2 s + d^4 N^{1/s} \log(dN^{1/s})\right) \cdot \mathrm{bit}(\boldsymbol{p})$ space complexity and $\tilde{O}\left(d^2 s\left(n + md^2 + d^5\right)\right)$ time complexity.*
- *Coordinator/Parallel Computation: An algorithm with $O(d^2 s)$ rounds of communication and $O\left(\left(k + d^4 N^{1/s} \log\left(dN^{1/s}\right)\right) \cdot \mathrm{bit}(\boldsymbol{p}) + k \log\left(dN^{1/s}\right) + kd^2 \log N\right)$ load. The local computation time of the coordinator is $O\left(d^2 s\left(md^2 + d^5 + k\right)\right)$, and the local computation time of each machine i is $O(d^2 s n_i)$ where $n_i = |P_i|$.*

3.8.1 Solving SDP Saddle Point Problems in the Unit Simplex

One example of a bounded SDP where our algorithm can be used to achieve a useful result is the saddle point problem

$$\max_X \min_{\boldsymbol{p} \in \Delta_n} \sum_{i \in [n]} p_i \left(\langle A^{(i)}, X\rangle_F - b^{(i)}\right) \tag{3.6}$$

where for all $i \in [n]$, $A^{(i)} \in \mathbb{R}^{d \times d}$ are symmetric and $b^{(i)} \in \mathbb{R}$. $\Delta_n = \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{x} \geq 0, \|\boldsymbol{x}\|_1 = 1\}$ is the $n-1$ dimensional unit simplex, and $X \in \mathbb{R}^{d \times d}$ is a positive semidefinite matrix of trace 1. In the case where the optimal solution to Equation 3.6 is nonnegative, then solving it up to additive ε error is equivalent to finding an X that satisfies all constraints $\langle A^{(i)}, X\rangle_F \geq b^{(i)}$ up to additive ε error. The optimization version of this is maximizing a margin σ where $\langle A^{(i)}, X\rangle_F \geq b^{(i)} + \sigma$.

Notice that the constraints of the problem directly fit into our framework for bounded SDP problems, and the margin σ can be represented as an aditional variable in our LP formulation. The maximization objective $\max \sigma$ now requires a 1-hot vector c, which fits into our boundedness constraint.

Thus, we can use our general framework for bounded SDP problems in order to solve this problem up to an additive ε-approximation, in the bounds given in Theorem 3.16.

Chapter 4

Lower Bounds

In this section, we motivate the usage of multipass algorithms for achieving subexponential space complexity in $(1+\varepsilon)$-approximations for LP-type problems in the high accuracy regime, where $d < ({}^1\!/\!_\varepsilon)^{0.999}$. We establish these lower bounds for the MEB and linear SVM problems by analyzing the communication complexity with reductions from the Indexing problem.

It is a well-known result that a lower bound on the 1-round communication complexity for a problem gives a lower bound on the space complexity of 1-pass streaming algorithms, as a 1-pass streaming algorithm can be turned into a 1-round communication protocol by having Alice run the streaming algorithm on her points and send the state of the algorithm to Bob, who finishes running the algorithm on his points [6], meaning that our communication complexity lower bounds give the same space complexity lower bounds for the MEB and linear SVM problems.

We use the standard two party communication model. A problem in this model is a function $P : \mathcal{A} \times \mathcal{B} \to \mathcal{C}$. Alice receives an input $A \in \mathcal{A}$ and Bob receives an input $B \in \mathcal{B}$. In an r-round protocol, Alice and Bob communicate up to r messages with each other, where the sender and receiver alternate with each round. In particular, if r is even then Bob sends the first message to Alice. If r is odd then Alice sends the first message to Bob. After r rounds of communication, Bob outputs some $C \in \mathcal{C}$. The goal is for Bob to output $P(A, B)$.

The r-round communication complexity of a problem P, denoted $\mathsf{CC}^r(P)$, is the minimum worst-case cost over all protocols in which Bob correctly outputs $P(A, B)$ with probability at least $2/3$. In this model, cost is measured by the total number of bits sent between Alice and Bob throughout the r messages.

In the Indexing Problem, denoted Ind_n, Alice is given a binary string $b \in \{0,1\}^n$ and Bob is given an index $i \in [n]$. The goal is for Bob to output the i^{th} bit of b. It is well-known that the 1-round randomized communication complexity of Ind_n is $\mathsf{CC}^1(\mathsf{Ind}_n) = \Omega(n)$.

4.1 Lower Bound for the MEB Problem

In this section, we provide the following lower bound for the communication complexity of the 1-round MEB problem.

Theorem 4.1. *For $d < (1/\varepsilon)^{0.999}$, any 1-round communication protocol which yields a $(1+\varepsilon)$-approximation for the MEB problem requires $\left(\frac{1}{\varepsilon}\right)^{\Omega(d)}$ bits of communication.*

Proof. The proof proceeds by reduction from Ind_n to MEB. Given an instance of the indexing problem, Alice uses a scheme in order to transform her bitstring b into a set of at most n points $P \in \mathbb{R}^d$, one point $\boldsymbol{p}_j$ for each $b_j = 1$. Separately, Bob uses a scheme in order to transform his index i into a single point $\boldsymbol{q} \in \mathbb{R}^d$. We then show that if $b_i = 1$, then the MEB of the at most $n+1$ points $P \cup \{\boldsymbol{q}\}$ obtained by Alice and Bob has radius 2, while if $b_i = 0$ then the MEB has radius at most $2 - \Omega(\varepsilon)$ where ε satisfies $n = \left(\frac{1}{\varepsilon}\right)^{\lfloor d/4 \rfloor}$. This therefore gives a $\left(\frac{1}{\varepsilon}\right)^{\Omega(d)}$ lower bound on the communication complexity of any 1-round communication protocol for a $(1+\varepsilon)$-approximation, which gives the same lower bound on the space complexity of any 1-pass streaming algorithm for the $(1+\varepsilon)$-approximation MEB problem. A full proof is provided hereafter, which first shows a lower bound for $d = 2$, and then extends it to higher dimensions.

First we proceed with the proof when $d = 2$.

Let $\varepsilon = \left(\frac{1}{n}\right)^2$ and Alice's points $\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}, \ldots, \boldsymbol{p}^{(n)} \in \mathbb{R}^2$ be equally spaced points on the unit circle, where

$$\boldsymbol{p}^{(j)} = \left(\cos\left(2\pi j\sqrt{\varepsilon}\right), \sin\left(2\pi j\sqrt{\varepsilon}\right)\right).$$

For each $j \in [n]$, if $b_j = 1$, then Alice adds $\boldsymbol{p}^{(j)}$ to her portion of the input stream to the MEB problem. Bob adds to his portion of the input stream the point $\boldsymbol{q} \in \mathbb{R}^2$, given by

$$\boldsymbol{q} = \left(3\cos\left(\pi + 2\pi i\sqrt{\varepsilon}\right), 3\sin\left(\pi + 2\pi i\sqrt{\varepsilon}\right)\right),$$

such that $\boldsymbol{q}$ is the point opposite $\boldsymbol{p}^{(i)}$ through the origin such that $\left\|\boldsymbol{p}^{(i)} - \boldsymbol{q}\right\| = 4$. Without loss of generality, we assume that $i = n$. Then, we have $\boldsymbol{p}^{(i)} = (1, 0)$ and $\boldsymbol{q} = (-3, 0)$. If $b_i = 1$, then both $\boldsymbol{p}^{(i)}$ and $\boldsymbol{q}$ are part of the input stream to Alice and Bob's instance of MEB. In this case, it is easy to see that the MEB is that which has a diameter from $\boldsymbol{p}^{(i)}$ to $\boldsymbol{q}$, meaning that the MEB has radius 2.

However, if $b_i = 0$, then $\boldsymbol{p}^{(i)}$ is not part of the input to the MEB instance. Thus, it remains to show that $\left\|\boldsymbol{p}^{(j)} - \boldsymbol{q}\right\| \leq 4 - \Omega(\varepsilon)$ for all other j where $b_j = 1$. It is clear that we only need to consider j such that $\boldsymbol{p}^{(j)}$ is to the right of the origin, as any point $\boldsymbol{p}^{(j)}$ left of the origin clearly satisfies this inequality. We can bound the square of this distance using the Pythagorean theorem as

$$\begin{aligned}\|\boldsymbol{p}^{(j)} - \boldsymbol{q}\|^2 &= \left(\cos\left(2\pi j\sqrt{\varepsilon}\right) + 3\right)^2 + \left(\sin\left(2\pi j\sqrt{\varepsilon}\right)\right)^2 \\ &= 10 + 6\cos\left(2\pi j\sqrt{\varepsilon}\right).\end{aligned}$$

Thus, as this distance only depends on the cosine, and thus only on the x coordinate of $\boldsymbol{p}^{(j)}$, we can see that the distance from $\boldsymbol{q}$ is larger the further right a point is. Thus, the radius of the MEB is bounded above by the distance from $\boldsymbol{q}$ to $\boldsymbol{p}^{(1)}$ (or symmetrically $\boldsymbol{p}^{(n-1)}$). So, plugging in for j we have that

$$\|\boldsymbol{p}^{(1)} - \boldsymbol{q}\|^2 = 10 + 6\cos\left(2\pi\sqrt{\varepsilon}\right).$$

It is a known result that $\cos(\theta) < 1 - \frac{4\theta^2}{\pi^2}$ for all $\theta \in (0, \frac{\pi}{2})$ [11]. Thus we have that

$$\|\boldsymbol{p}^{(1)} - \boldsymbol{q}\|^2 = 16 - 96\varepsilon$$

which yields that the distance between $\boldsymbol{p}^{(1)}$ and $\boldsymbol{q}$ is $2 - \Omega(\varepsilon)$, giving our desired result.

We now extend this idea to an arbitrary dimension d satisfying $d < \left(\frac{1}{\varepsilon}\right)^{0.999}$.

Without loss of generality, we assume d is even. If it is not, we can simply disregard one of the dimensions. Let $\varepsilon = \left(\frac{1}{n}\right)^{4/d}$. As in the 2 dimensional case, we define Alice's n points $\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}, \ldots, \boldsymbol{p}^{(n)} \in \mathbb{R}^d$ where each point $\boldsymbol{p}^{(j)}$ is given by a sequence $\{j_k\}_{k=1}^{d/2}$ in $\{1, 2, \ldots, \frac{1}{\sqrt{\varepsilon}}\}$, so that the ℓ-th coordinate of $\boldsymbol{p}^{(j)}$ is given by

$$\boldsymbol{p}_\ell^{(j)} = \begin{cases} \sqrt{\frac{2}{d}}\cos\left(2\pi j_{\frac{\ell+1}{2}}\sqrt{\varepsilon}\right) & \text{if } \ell \text{ is odd,} \\ \sqrt{\frac{2}{d}}\sin\left(2\pi j_{\frac{\ell}{2}}\sqrt{\varepsilon}\right) & \text{if } \ell \text{ is even.} \end{cases}$$

Informally, we pair up the axes of d-dimensional space, and choose the points on the unit circle which when projected onto the plane of a pair of axes will resemble the 2-dimensional case. Alice and Bob choose their points as in the 2-dimensional case. Without loss of generality, we assume that $i = n$ and that $\boldsymbol{p}^{(i)}$ is given by the sequence $i_k = \sqrt{\varepsilon}$ for all $k \in [\frac{d}{2}]$. Then, we have that $\boldsymbol{p}^{(i)}$ is the point where the ℓ-th coordinate is given by

$$\boldsymbol{p}_\ell^{(i)} = \begin{cases} \sqrt{\frac{2}{d}} & \text{if } \ell \text{ is odd,} \\ 0 & \text{if } \ell \text{ is even.} \end{cases}$$

We then have that $\boldsymbol{q}$ is point such that the ℓ-th coordinate is given by

$$\boldsymbol{q}_\ell = \begin{cases} -3\sqrt{\frac{2}{d}} & \text{if } \ell \text{ is odd,} \\ 0 & \text{if } \ell \text{ is even.} \end{cases}$$

As in the 2-dimensional case, it can easily be seen that if $b_i = 1$, then the MEB has radius 2. If $b_i = 0$, then the radius of the MEB is bounded above by the distance from $\boldsymbol{q}$ to $\boldsymbol{p}^{(j)}$, where $\boldsymbol{p}^{(j)}$ is given by $j_1 = 1$ and $j_k = \frac{1}{\sqrt{\varepsilon}}$ for all $k \neq 1$. We can now bound the squared distance.

$$\begin{aligned}
\|\boldsymbol{p}^{(j)} - \boldsymbol{q}\|^2 &= \sum_{k=1}^{d/2} \left[\sqrt{\frac{2}{d}} \cos\left(2\pi j_k \sqrt{\varepsilon}\right) + 3\sqrt{\frac{2}{d}}\right]^2 + \left[\sqrt{\frac{2}{d}} \sin\left(2\pi j_k \sqrt{\varepsilon}\right)\right]^2 \\
&= \sum_{k=1}^{d/2} \frac{20}{d} + \frac{12}{d} \cos\left(2\pi j_k \sqrt{\varepsilon}\right) \\
&= \left(\frac{20}{d} + \frac{12}{d}\cos\left(2\pi\sqrt{\varepsilon}\right)\right) + \sum_{k=2}^{d/2} \left(\frac{20}{d} + \frac{12}{d}\cos\left(2\pi j_k \sqrt{\varepsilon}\right)\right) \\
&\leq \left(\frac{20}{d} + \frac{12}{d}\cos\left(2\pi\sqrt{\varepsilon}\right)\right) + \sum_{k=2}^{d/2} \left(\frac{20}{d} + \frac{12}{d}\right) && \text{bounding } \cos \text{ by } 1 \\
&= \left(\frac{20}{d} + \frac{12}{d}\cos\left(2\pi\sqrt{\varepsilon}\right)\right) + \left(16 - \frac{32}{d}\right) \\
&\leq \left(\frac{20}{d} + \frac{12}{d} - \frac{196\varepsilon}{d}\right) + \left(16 - \frac{32}{d}\right) && \text{bounding } \cos \text{ by [11]} \\
&= 16 - \frac{192\varepsilon}{d}.
\end{aligned}$$

This yields that the distance between $\boldsymbol{p}^{(j)}$ and $\boldsymbol{q}$ is $2 - \Omega(\frac{\varepsilon}{d})$, giving a lower bound of $\left(\frac{1}{\varepsilon d}\right)^{\Omega(d)}$. In the regime where $d < \left(\frac{1}{\varepsilon}\right)^{0.999}$, this is $\left(\frac{1}{\varepsilon}\right)^{\Omega(d)}$, which is our desired result.

$\square$

4.2 Lower Bounds for the Linear SVM Problem

In this section, we provide the following lower bounds for the communication complexity of the 1-round linear SVM problem.

Theorem 4.2. *For $d < (1/\varepsilon)^{0.999}$, any 1-round communication protocol which yields a $(1+\varepsilon)$-approximation for the linear SVM problem requires $\left(\frac{1}{\varepsilon}\right)^{\Omega(d)}$ bits of communication.*

Proof. The proof follows the structure of the proof for Theorem 4.1, with a reduction from Ind_n to linear SVM. Alice transforms her bitstring into at most n points labeled -1, and Bob transforms his index into a point labeled $+1$. We show then that if $b_i = 1$, then the separating hyperplane $(\boldsymbol{u}, b)$ of $P \cup \{\boldsymbol{q}\}$ obtained by Alice and Bob has $\|\boldsymbol{u}\| = 4$, while if $b_i = 0$ then $\|\boldsymbol{u}\| \leq 4 - \Omega\left(\frac{\varepsilon}{d}\right)$. This therefore gives a $\left(\frac{1}{\varepsilon}\right)^{\Omega(d)}$ lower bound on the communication complexity of any 1-round communication protocol for a $(1+\varepsilon)$-approximation, which gives the same lower bound on the space complexity of any 1-pass streaming algorithm for the $(1+\varepsilon)$-approximation linear SVM problem. A full proof is provided hereafter.

Again, without loss of generality, we assume d is even. If it is not, we can simply disregard one of the dimensions. Let $\varepsilon = \left(\frac{1}{n}\right)^{4/d}$. We define Alice's n points, which are all labelled -1, $\left(\boldsymbol{p}^{(1)}, -1\right), \left(\boldsymbol{p}^{(2)}, -1\right), \ldots, \left(\boldsymbol{p}^{(n)}, -1\right) \in \mathbb{R}^d \times \{-1, +1\}$ where each point $\boldsymbol{p}^{(j)}$ is given by a sequence $\{j_k\}_{k=1}^{d/2}$ in $\{1, 2, \ldots, \frac{1}{\sqrt{\varepsilon}}\}$, so that the ℓ-th coordinate of $\boldsymbol{p}^{(j)}$ is given by

$$\boldsymbol{p}_\ell^{(j)} = \begin{cases} \sqrt{\frac{2}{d}} \cos\left(2\pi j_{\frac{\ell+1}{2}} \sqrt{\varepsilon}\right) & \text{if } \ell \text{ is odd,} \\ \sqrt{\frac{2}{d}} \sin\left(2\pi j_{\frac{\ell}{2}} \sqrt{\varepsilon}\right) & \text{if } \ell \text{ is even.} \end{cases}$$

Alice and bob construct an input to the Linear SVM problem as follows: Alice adds to the input the labeled point $(\boldsymbol{p}^{(j)}, -1)$ for each $j \in [n]$ such that $b_j = 1$, and Bob adds to the input the labeled point $(2\boldsymbol{p}^{(i)}, +1)$. Call this instance of Linear SVM $\mathcal{I}$, and let $(\boldsymbol{u}^*, b^*)$ be the optimal solution to this instance. We make use of the following two lemmas, which will be proven later.

Lemma 4.3. *If* $b_i = 1$*, then* $\|\boldsymbol{u}^*\| = 4$.

Lemma 4.4. *If* $b_i = 0$*, then* $\|\boldsymbol{u}^*\| \leq 4 - \Omega(\frac{\varepsilon}{d})$.

It then follows from the above lemmas that Alice and Bob can solve the indexing problem by obtaining a $(1 + \Theta(\frac{\varepsilon}{d}))$-approximation for $\mathcal{I}$. Indexing has a lower bound of $\Omega(n)$ bits of communication, so we have that in general, a $(1 + \varepsilon)$-approximation of linear SVM must have a lower bound of $\left(\frac{1}{\varepsilon d}\right)^{\Omega(d)}$ bits of communication. In the regime where $d < \left(\frac{1}{\varepsilon}\right)^{0.999}$, this is a lower bound of $\left(\frac{1}{\varepsilon}\right)^{\Omega(d)}$. □

Proof of Lemma 4.3. Suppose that $b_i = 1$. Let $\boldsymbol{v}$ be the vector given by

$$\boldsymbol{v} = \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots & 1 & 0 \end{bmatrix}^T.$$

Without loss of generality, assume that $\boldsymbol{p}^{(i)}$ is the point characterized by $j_1 = j_2 = \cdots = j_{d/2} = 0$, that is, $\boldsymbol{p}^{(i)} = \sqrt{\frac{1}{2d}}\boldsymbol{v}$. This can always be achieved via a rotation of the working space. Consider the hyperplane given by $\sqrt{\frac{32}{d}}\boldsymbol{v}^T\boldsymbol{x} - 3 = 0$. For any point $\boldsymbol{p}^{(j)}$ which was potentially inserted into $\mathcal{I}$ with label -1 by Alice, we have

$$\begin{aligned} \sqrt{\frac{32}{d}}\boldsymbol{v}^T\boldsymbol{p}^{(j)} - b &= \sqrt{\frac{16}{d^2}} \sum_{\ell=1}^{d/2} \cos(2\pi j_\ell \sqrt{\varepsilon}) - 3 \\ &\leq \frac{d}{2}\sqrt{\frac{16}{d^2}} - 3 \\ &= 1. \end{aligned}$$

Similarly, for the point $2\boldsymbol{p}^{(i)}$ inserted into $\mathcal{I}$ with label $+1$ by Bob, we have

$$\begin{aligned} 2\sqrt{\frac{32}{d}}\boldsymbol{v}^T\boldsymbol{p}^{(i)} - 3 &= d\sqrt{\frac{16}{d^2}} - 3 \\ &= 1. \end{aligned}$$

Thus, $(\sqrt{\frac{32}{d}}\boldsymbol{v}, 3)$ is a feasible solution to $\mathcal{I}$. To see that this is optimal, observe

$$\|2\boldsymbol{p}^{(i)} - \boldsymbol{p}^{(i)}\| = \|\boldsymbol{p}^{(i)}\| = \frac{1}{2} = \frac{2}{\left\|\sqrt{\frac{32}{d}}\boldsymbol{v}\right\|}.$$

Thus, we must have that $\|\boldsymbol{u}^*\|_2 = 4$. □

Proof of Lemma 4.4. Suppose that $b_i = 0$. Let $\boldsymbol{v}$ be as defined in the proof of Lemma 4.3. For each $j \in [n]$, let $\boldsymbol{q}^{(j)}$ be the orthogonal projection of $\boldsymbol{p}^{(j)}$ onto the line spanned by $\boldsymbol{v}$. Let $\mathcal{J}$ be the instance of linear SVM consisting of the pairs $(\boldsymbol{q}^{(j)}, -1)$ for all $j \in [n]$ such that $b_j = 1$ and the pair $(2\boldsymbol{p}^{(i)}, +1)$. It follows from orthogonality that if the hyperplane $\alpha\boldsymbol{v}^T\boldsymbol{x} - \beta = 0$ is a feasible solution to $\mathcal{J}$, then it must also be a feasible solution to $\mathcal{I}$. Using this fact, we proceed by showing there exists a hyperplane $\alpha\boldsymbol{v}^T\boldsymbol{x} - \beta = 0$ which is a feasible solution to $\mathcal{J}$ and is such that $\|\alpha\boldsymbol{v}\|_2 \leq 4 - \Omega(\frac{\varepsilon}{d})$. First, we observe that for all $j \in [n]$

$$\begin{aligned} \boldsymbol{q}^{(j)} &= \text{proj}_{\boldsymbol{v}}(\boldsymbol{p}^{(j)}) \\ &= \frac{\langle \boldsymbol{p}^{(j)}, \boldsymbol{v}\rangle}{\langle \boldsymbol{v}, \boldsymbol{v}\rangle}\boldsymbol{v} \\ &= \left(\sum_{\ell=1}^{d/2} \frac{2}{d}\sqrt{\frac{1}{2d}}\cos(2\pi j_\ell\sqrt{\varepsilon})\right)\boldsymbol{v}. \end{aligned}$$

With this, we obtain that for each j such that $b_j = 1$, we have

$$\begin{aligned}
\left\|2\boldsymbol{p}^{(i)} - \boldsymbol{q}^{(j)}\right\| &= \|\boldsymbol{v}\| \left| \sqrt{\frac{2}{d}} - \sum_{\ell=1}^{d/2} \frac{2}{d}\sqrt{\frac{1}{2d}} \cos(2\pi j_\ell \sqrt{\varepsilon}) \right| \\
&= \sqrt{\frac{d}{2}} \left| \sqrt{\frac{2}{d}} - \sum_{\ell=1}^{d/2} \frac{2}{d}\sqrt{\frac{1}{2d}} \cos(2\pi j_\ell \sqrt{\varepsilon}) \right| \\
&= \left| 1 - \sum_{\ell}^{d/2} \frac{1}{d} \cos(2\pi j_\ell \sqrt{\varepsilon}) \right| \\
&\geq \left| 1 - \left(\frac{1}{d}\left(\frac{d}{2} - 1\right) + \frac{1}{d}\cos(2\pi\sqrt{\varepsilon}) \right) \right| \\
&= \left| \frac{1}{2} + \frac{1}{d}\left(1 - \cos(2\pi\sqrt{\varepsilon})\right) \right| \\
&\geq \left| \frac{1}{2} + \frac{1}{d}\left(1 - (1 - 16\varepsilon)\right) \right| \\
&= \frac{1}{2} + \frac{16\varepsilon}{d}.
\end{aligned}$$

Since all points in $\mathcal{J}$ are in the line spanned by $\boldsymbol{v}$, and the points are separable, there must exist a feasible hyperplane $\alpha \boldsymbol{v}^T \boldsymbol{x} - \beta = 0$ such that

$$\frac{2}{\|\alpha \boldsymbol{v}\|} = \min\left\{ \left\|2\boldsymbol{p}^{(i)} - \boldsymbol{q}^{(j)}\right\| : j \in [n] \text{ and } b_j = 1 \right\} \geq \frac{1}{2} + \frac{16\varepsilon}{d}.$$

Fnally, this gives that

$$\|\alpha \boldsymbol{v}\| \leq \frac{4d}{d + 32\varepsilon} \leq 4 - \frac{8\varepsilon}{d} = 4 - \Omega\left(\frac{\varepsilon}{d}\right).$$

giving us the desired result. □

Theorem 4.5. *For $d < (1/\varepsilon)^{0.999}$, any 1-round communication protocol which determines is a set of binary labeled points is γ-separable requires $\left(\frac{1}{\gamma}\right)^{\Omega(d)}$ bits of communication.*

Proof. The proof is similar to that of Theorem 4.2, with a reduction from Ind_n to determining γ-separability. Alice transforms her bitstring exctly as in the proof of Theorem 4.2, and Bob chooses a set of d points, labeled $+1$, which lie in the unique hyperplane that is orthogonal to the line spanned by the point $\boldsymbol{p}^{(i)}$, which is the point Alice would transform b_i in the case where $b_i = 1$, and also contains $\boldsymbol{p}^{(i)}$, such that $\boldsymbol{p}^{(i)}$ lies in the segment between some pair of points chosen by Bob.

Since $\boldsymbol{p}^{(i)}$ can be written as a convex combination of two of Bob's points, it is clear that the points chosen by Alice and Bob are inseparable if $b_i = 1$, as then Alice will include $\boldsymbol{p}^{(i)}$ in P.

If however $b_i = 0$, then there will be a separating hyperplane parallel to Bob's set of points, and midway between that and the closest point of Alice. This hyperplane induces a margin of size $\Omega\left(\frac{16\gamma}{d}\right)$, which gives a lower bound on the communication complexity of determining γ-separability of $\left(\frac{1}{\gamma}\right)^{\Omega(d)}$. $\square$

Chapter 5

Conclusion

In this work, we explore solving $(1+\varepsilon)$-multiplicative and ε-additive approximation LP-type problems in the multiple linear sketching model. We achieve results with pass and space complexities polynomial in the dimensionality of the problem d and polylogarithmic in ε. This provides exponential improvements on many current results in the high accuracy regime, i.e. when $d < (1/\varepsilon)^{0.999}$. We apply our algorithm to various big data models and LP-type problems, namely the multipass streaming, strict turnstile, coordinator, and parallel computation models, as well as the MEB, Linear SVM, Bounded LP, and Bounded SDP problems. We also provide lower bounds on the MEB and Linear SVM problem in the single pass model, motivating our multi-pass approach.

Bibliography

[1] Pankaj K. Agarwal and R. Sharathkumar. *Streaming Algorithms for Extent Problems in High Dimensions*, pages 1481–1489. 2010. doi: 10.1137/1.9781611973075.120. URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611973075.120`.

[2] Pankaj K. Agarwal and Hai Yu. A space-optimal data-stream algorithm for coresets in the plane. In *Proceedings of the Twenty-Third Annual Symposium on Computational Geometry*, SCG '07, page 1–10, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937056. doi: 10.1145/1247069.1247071. URL `https://doi.org/10.1145/1247069.1247071`.

[3] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Approximating extent measures of points. *J. ACM*, 51(4):606–635, jul 2004. ISSN 0004-5411. doi: 10.1145/1008731.1008736. URL `https://doi.org/10.1145/1008731.1008736`.

[4] Kook Jin Ahn and Sudipto Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Information and Computation*, 222:59–79, 2013. ISSN 0890-5401. doi: https://doi.org/10.1016/j.ic.2012.10.006. URL `https://www.sciencedirect.com/science/article/pii/S0890540112001460`. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[5] Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New characterizations in turnstile streams with applications. In *Proceedings of the 31st Conference on Computational Complexity*, CCC '16, Dagstuhl, DEU, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 9783959770088.

[6] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.1997.1545. URL `https://www.sciencedirect.com/science/article/pii/S0022000097915452`.

[7] Sunil Arya, Guilherme D. da Fonseca, and David M. Mount. Near-Optimal epsilon-Kernel Construction and Related Problems. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry (SoCG 2017)*, volume 77 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:15, Dagstuhl, Germany,

2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-038-5. doi: 10.4230/LIPIcs.SoCG.2017.10. URL `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SoCG.2017.10`.

[8] Sepehr Assadi, Nikolai Karpov, and Qin Zhang. Distributed and streaming linear programming in low dimensions. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '19, page 236–253, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362276. doi: 10.1145/3294052.3319697. URL `https://doi.org/10.1145/3294052.3319697`.

[9] Mihai Bădoiu and Kenneth L. Clarkson. Smaller core-sets for balls. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, page 801–802, USA, 2003. Society for Industrial and Applied Mathematics. ISBN 0898715385.

[10] Mihai Bădoiu and Kenneth L. Clarkson. Optimal core-sets for balls. *Computational Geometry*, 40(1):14–22, 2008. ISSN 0925-7721. doi: https://doi.org/10.1016/j.comgeo.2007.04.002. URL `https://www.sciencedirect.com/science/article/pii/S0925772107000454`.

[11] Yogesh J. Bagul and Satish K. Panchal. Certain inequalities of Kober and Lazarevic type. *The Journal of the Indian Mathematical Society*, 89(1-2):01–07, Jan. 2022. doi: 10.18311/jims/2022/20737. URL `https://www.informaticsjournals.com/index.php/jims/article/view/20737`.

[12] Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. *ACM Trans. Algorithms*, 16(1), December 2019. ISSN 1549-6325. doi: 10.1145/3309193. URL `https://doi.org/10.1145/3309193`.

[13] A. Blum, A. Frieze, R. Kannan, and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 330–338, 1996. doi: 10.1109/SFCS.1996.548492.

[14] Yaroslav Bulatov, Sachin Jambawalikar, Piyush Kumar, and Saurabh Sethia. Hand recognition using geometric classifiers. In David Zhang and Anil K. Jain, editors, *Biometric Authentication*, pages 753–759, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-25948-0.

[15] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998. doi: 10.1023/A:1009715923555. URL `https://doi.org/10.1023/A:1009715923555`.

[16] Tom Bylander. Learning linear threshold functions in the presence of classification noise. In *Proceedings of the Seventh Annual Conference on Computational Learning Theory*, COLT '94, page 340–347, New York, NY, USA, 1994. Association for Computing Machinery. ISBN 0897916557. doi: 10.1145/180139.181176. URL `https://doi.org/10.1145/180139.181176`.

[17] Emmanuel Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, June 2012. ISSN 0001-0782. doi: 10.1145/2184319.2184343. URL https://doi.org/10.1145/2184319.2184343.

[18] Emmanuel J. Candes and Terence Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010. doi: 10.1109/TIT.2010.2044061.

[19] Jair Cervantes, Farid Garcia-Lamont, Lisbeth Rodríguez-Mazahua, and Asdrubal Lopez. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing*, 408:189–215, 2020. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.10.118. URL https://www.sciencedirect.com/science/article/pii/S0925231220307153.

[20] Timothy M. Chan. Faster core-set constructions and data-stream algorithms in fixed dimensions. *Computational Geometry*, 35(1):20–35, 2006. ISSN 0925-7721. doi: https://doi.org/10.1016/j.comgeo.2005.10.002. URL https://www.sciencedirect.com/science/article/pii/S0925772105000970. Special Issue on the 20th ACM Symposium on Computational Geometry.

[21] Timothy M. Chan and Vinayak Pathak. Streaming and dynamic algorithms for minimum enclosing balls in high dimensions. In Frank Dehne, John Iacono, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, pages 195–206, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-22300-6.

[22] M. T. Chao. A general purpose unequal probability sampling plan. *Biometrika*, 69(3):653–656, 1982. ISSN 00063444. URL http://www.jstor.org/stable/2336002.

[23] Kenneth L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42(2):488–499, mar 1995. ISSN 0004-5411. doi: 10.1145/201019.201036. URL https://doi.org/10.1145/201019.201036.

[24] Kenneth L. Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Trans. Algorithms*, 6(4), sep 2010. ISSN 1549-6325. doi: 10.1145/1824777.1824783. URL https://doi.org/10.1145/1824777.1824783.

[25] Kenneth L. Clarkson, Elad Hazan, and David P. Woodruff. Sublinear optimization for machine learning. *J. ACM*, 59(5), November 2012. ISSN 0004-5411. doi: 10.1145/2371656.2371658. URL https://doi.org/10.1145/2371656.2371658.

[26] Graham Cormode and S. Muthukrishnan. What's hot and what's not: tracking most frequent items dynamically. In *Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '03, page 296–306, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136706. doi: 10.1145/773153.773182. URL https://doi.org/10.1145/773153.773182.

[27] John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for

solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008. doi: 10.1007/s10107-007-0095-7. URL https://doi.org/10.1007/s10107-007-0095-7.

[28] Dan Garber and Elad Hazan. Sublinear time algorithms for approximate semidefinite programming. *Mathematical Programming*, 158(1):329–361, 2016. doi: 10.1007/s10107-015-0932-z. URL https://doi.org/10.1007/s10107-015-0932-z.

[29] David Haussler and Emo Welzl. Epsilon-nets and simplex range queries. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, page 61–71, New York, NY, USA, 1986. Association for Computing Machinery. ISBN 0897911946. doi: 10.1145/10515.10522. URL https://doi.org/10.1145/10515.10522.

[30] Piotr Indyk, Sepideh Mahabadi, Ronitt Rubinfeld, Jonathan Ullman, Ali Vakilian, and Anak Yodpinyanee. Fractional Set Cover in the Streaming Model. In Klaus Jansen, José D. P. Rolim, David P. Williamson, and Santosh S. Vempala, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, volume 81 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:20, Dagstuhl, Germany, 2017. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-044-6. doi: 10.4230/LIPIcs.APPROX-RANDOM.2017.12. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX-RANDOM.2017.12.

[31] Hossein Jowhari, Mert Saglam, and Gábor Tardos. Tight bounds for ℓ_p samplers, finding duplicates in streams, and related problems. *CoRR*, abs/1012.4889, 2010. URL http://arxiv.org/abs/1012.4889.

[32] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *Proceedings of the Twenty-Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '10, page 41–52, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450300339. doi: 10.1145/1807085.1807094. URL https://doi.org/10.1145/1807085.1807094.

[33] Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM J. Exp. Algorithmics*, 8:1.1–es, dec 2004. ISSN 1084-6654. doi: 10.1145/996546.996548. URL https://doi.org/10.1145/996546.996548.

[34] Yi Li, Huy L. Nguyen, and David P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 174–183, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450327107. doi: 10.1145/2591796.2591812. URL https://doi.org/10.1145/2591796.2591812.

[35] Nantia Makrynioti, Nikolaos Vasiloglou, Emir Pasalic, and Vasilis Vassalos. Data science with linear programming. 2017. URL https://api.semanticscholar.org/CorpusID:198233428.

[36] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4):498–516, 1996. doi: 10.1007/BF01940877. URL `https://doi.org/10.1007/BF01940877`.

[37] Nabil H. Mustafa and Kasturi R. Varadarajan. Epsilon-approximations and epsilon-nets. *CoRR*, abs/1702.03676, 2017. URL `http://arxiv.org/abs/1702.03676`.

[38] Frank Nielsen and Richard Nock. Approximating smallest enclosing balls. In Antonio Laganá, Marina L. Gavrilova, Vipin Kumar, Youngsong Mun, C. J. Kenneth Tan, and Osvaldo Gervasi, editors, *Computational Science and Its Applications – ICCSA 2004*, pages 147–157, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24767-8.

[39] A. B. Novikoff. On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, New York, NY, USA, 1962. Polytechnic Institute of Brooklyn.

[40] Benjamin Recht. A simpler approach to matrix completion. *J. Mach. Learn. Res.*, 12(null): 3413–3430, December 2011. ISSN 1532-4435.

[41] Micha Sharir and Emo Welzl. A combinatorial bound for linear programming and related problems. In Alain Finkel and Matthias Jantzen, editors, *STACS 92*, pages 567–579, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg. ISBN 978-3-540-46775-5.

[42] Zhao Song, Mingquan Ye, and Lichen Zhang. Streaming semidefinite programs: $O(\sqrt{n})$ passes, small space and fast runtime, 2023. URL `https://arxiv.org/abs/2309.05135`.

[43] Xiaoming Sun, David P. Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. *ACM Trans. Algorithms*, 17(4), October 2021. ISSN 1549-6325. doi: 10.1145/3470566. URL `https://doi.org/10.1145/3470566`.

[44] Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *J. Mach. Learn. Res.*, 6:363–392, dec 2005. ISSN 1532-4435.

[45] Jan van den Brand, Yin Tat Lee, Yang P. Liu, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Minimum cost flows, mdps, and 1-regression in nearly linear time for dense instances. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 859–869, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380539. doi: 10.1145/3406325.3451108. URL `https://doi.org/10.1145/3406325.3451108`.

[46] V. N. Vapnik and A. Ya. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, pages 11–30. Springer International Publishing, Cham, 2015. ISBN 978-3-319-21852-6. doi: 10.1007/978-3-319-21852-6_3. URL `https://doi.org/10.1007/978-3-319-21852-6_3`.

[47] David P. Woodruff. Sketching as a tool for numerical linear algebra. *Found. Trends Theor.*

Comput. Sci., 10(1–2):1–157, October 2014. ISSN 1551-305X. doi: 10.1561/0400000060. URL `https://doi.org/10.1561/0400000060`.

[48] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL `https://proceedings.neurips.cc/paper_files/paper/2002/file/c3e4035af2a1cde9f21e1ae1951ac80b-Paper.pdf`.

[49] Yinyu Ye and Edison Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical Programming*, 44(1):157–179, 1989. doi: 10.1007/BF01587086. URL `https://doi.org/10.1007/BF01587086`.

[50] E. Alper Yildirim. Two algorithms for the minimum enclosing ball problem. *SIAM J. on Optimization*, 19(3):1368–1391, nov 2008. ISSN 1052-6234. doi: 10.1137/070690419. URL `https://doi.org/10.1137/070690419`.

[51] Hamid Zarrabi-Zadeh. An almost space-optimal streaming algorithm for coresets in fixed dimensions. *Algorithmica*, 60(1):46–59, 2011. doi: 10.1007/s00453-010-9392-2. URL `https://doi.org/10.1007/s00453-010-9392-2`.

[52] Hamid Zarrabi-Zadeh and Timothy M. Chan. A simple streaming algorithm for minimum enclosing balls. In *Proceedings of the 18th Annual Canadian Conference on Computational Geometry, CCCG 2006, August 14-16, 2006, Queen's University, Ontario, Canada*, 2006. URL `http://www.cs.queensu.ca/cccg/papers/cccg36.pdf`.