Geometric Tools for Algorithms

Santosh Vempala

August 1997

CMU-CS-97-167

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Thesis Committee:
Avrim Blum (Chair)
Alan Frieze
Ravi Kannan
László Lovász, Yale University

This research was sponsored by an NSF Young Investigator Award under grant no. CCR-9357793. Views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the United States Government.

Abstract

Our thesis is that a *geometric* perspective yields insights into the structure of fundamental problems, and thereby suggests efficient algorithms for them. As evidence, we develop new geometric models and general-purpose tools for *removing outliers* from numeric data, *reducing dimensionality*, and *counting combinatorial sets*. Then we apply these techniques to a set of old problems to obtain polynomial-time algorithms. These include: (1) learning noisy linear-threshold functions (half-spaces), (2) learning the intersection of half-spaces, (3) clustering text corpora, and (4) counting lattice points in a convex body. We supplement some of our theorems with experimental studies.

To my dear parents,
for giving me a love for learning,
and to two people who have shared with me
their marvellous eye for beauty,
Laci Lovász and Ardi Eggleston.

Acknowledgements

First of all, I want to thank my dear Amma and Nanagaru[1] for their love and their dedication to my education, and for making clear to me the value of learning.

Since I came to CMU, I have had the opportunity to work with some wonderful mathematicians and computer scientists.

I want to thank Avrim Blum, for many things. He was the perfect advisor – encouraging always, supportive in difficult times, thoughtful to keep me on track, and brilliant when we needed ideas.

Thanks to Alan Frieze, for always being willing to listen to my questions and for his witty answers.

I want to thank Ravi Kannan for initiating me into a geometric style of thinking and sharing with me his tremendous intuition, and for making seemingly complicated concepts look easy.

I am very grateful to Laci Lovász for all the beautiful things I learnt from him; it is truly thrilling to work with him!

I am also grateful to Hui Chen, Chandra Chekuri, R. Ravi, Christos Papadimitriou, Danny Sleator, Hisao Tamaki, Prabhakar Raghavan, Prasad Tetali, Piotr Indyk, Andrew Kotlov, Goran Konjevod, Debbie Goldman, Prasad Chalasani, Vijay Vazirani, Naveen Garg, Claudson Bornstein, Rachel Rue, Russ Bubley, Vivek Arora, Neeraj Aggarwal, Sean Hallgren, and Arianna for many stimulating discussions.

Thanks to my housemates and colleagues, Neil Heffernan, Doug Beeferman, Doug Rohde, Carl Burch, Adam Kalai, and Prem Janardhan for all the relaxing hours I spent with them.

A special thanks to my friend Ardi for listening enthusiastically to my haphazard descriptions of mathematical phenomena. I have enjoyed mathematics even more since I met her.

Finally, I want to thank my younger brother Naresh, and my friends from Vizag, Sridhar, Prasad, Prem, Ram, Ramesh, Gopi and Sadiq.

[1] The Telugu words for mother and father

Contents

Chapter 1

Introduction

In algorithm discovery, a *geometric* point of view is often an insightful one. A wonderful example of this is *Linear Programming* (LP). Algorithms for LP such as the *Simplex* method, the *Ellipsoid* method and *Interior point* methods can all be presented and explained in purely algebraic terms. However, the ideas and intuition behind them become particularly transparent when viewed geometrically. We illustrate this in more detail. Consider the following general linear program:

$$\mathbf{max} \quad c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

subject to the constraints:

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n & \leq & b_2 \\ & . & \\ & . & \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n & \leq & b_m \end{array}$$

This is a linear program in n variables, $x_1, x_2, \ldots, x_n$, so the problem is in $\boldsymbol{R}^n$. The c_j's, a_{ij}'s and b_i's are specified real numbers. Each linear constraint corresponds to a half-space (one side of a hyperplane) in $\boldsymbol{R}^n$. Hence their intersection, the *feasible region*, is a polyhedron (the higher-dimensional analogue of a polygon). The *objective function* which we are trying to maximize corresponds to a direction, and its value of at a point x is simply (a scaling of) its distance from the origin in that direction. From this perspective it is intuitively clear that the maximum will be achieved at a point that is farthest away from the origin in the direction specified by the objective function. Further, since the feasible region is a polyhedron, the maximum is achieved at some corner or facet (if there is a finite maximum) of the polyhedron. The Simplex method is then a very natural one: start at some vertex of polyhedron[1], and move to an adjacent vertex that improves the value of the objective function, i.e., is farther away in that direction till this is not possible and then declare that point the maximum! With a little more work the Ellipsoid and Interior-point methods can also be explained in a similar fashion.

The models and methods presented in this thesis are all motivated from a geometric perspective. In some cases, the original statement of the problem is *not* in geometric terms, yet recasting it in such terms helps us find efficient algorithms. In some case we derive the first polynomial-time algorithms, in other cases where polynomial algorithms were already known, we improve their efficiency.

1.1 Overview of new results

Outliers. The first scenario we consider is a rather general one. We are presented with a set of points. Each point has a fixed set of numeric attributes. This data could be the input to an algorithm, e.g., the training set of a learning algorithm. It is possible that such a data set has *outliers*. Typically, this might be due to some error in collecting the data etc., or it might actually correspond to an interesting pattern. In any case a useful thing to do would be to find (and separate) outliers in the data.

We address this situation by first posing the question: *what precisely is an outlier?* At first sight, our definition might seem rather strong. We call a point an outlier (with respect to a given set of points) if there exists *any* direction in which the squared distance of the point from the origin is more than a fixed ratio times the average squared distance of the data set in that direction [12].

Given this definition, two questions arise: (1) Is it possible to quickly detect such outliers? (2) Is it possible to remove a small subset of the points so there are no outliers left? We are able give a polynomial-time algorithm for detecting outliers in n-dimensional data, i.e., points in $\boldsymbol{R}^n$, and show that for a reasonable ratio of outlier to average, the number of outliers is at most a small fraction of the total number of points. Formally,

Lemma 1 *For any set of points in $\boldsymbol{R}^n$, each given to b bits of precision, in polynomial*

[1]This is typically achieved by using additional *slack* and *surplus* variables; we omit the details.

time it is possible to remove less than $\frac{1}{n}$ fraction of the set, so that in the remaining set T, for every vector $v \in \boldsymbol{R}^n$,

$$\max_{x \in T}(v \cdot x)^2 \leq poly(n, b)\frac{1}{|T|}\sum_{x \in T}(v \cdot x)^2,$$

i.e., the maximum squared distance in any *direction is at most a polynomially bounded number times the average.*

Learning noisy perceptrons. As an application of the lemma, we consider the classical problem of learning a linear threshold function (a half-space in n dimensions, also called a "perceptron"). Methods for solving this problem generally fall into two categories. In the absence of noise, this problem can be formulated as a Linear Program and solved in polynomial time. Alternatively, simple greedy algorithms such as the Perceptron Algorithm [50] are often used in practice and have certain provable noise-tolerance properties; but, their running time depends on a separation parameter, which quantifies the amount of "wiggle room" available for a solution, and can be exponential in the description length of the input.

We show that after removing outliers from the training data, a simple modification of the Perceptron Algorithm finds a weak hypothesis in polynomial time *without* dependence on any separation parameter. Suitably combining these hypotheses results in a polynomial-time algorithm for learning linear threshold functions even in the presence of random classification noise, i.e., when the labels of examples presented to us are inverted with some fixed noise probability. The chapter includes some experimental studies and lists other potential applications.

Random Projection. The next scenario we examine is one where, as above, the data points presented to us are in a considerably high dimensional space, i.e., they have a large number of attributes. What we would like to do now, however, is to represent the points in a suitable lower dimensional subspace. Of course, what constitutes a *suitable* subspace depends on the specific application in mind. We show that a very simple idea — project the points to a random lower dimensional subspace, i.e., a random hyperplane through the origin — is very useful in identifying a good subspace quickly. We analyze this technique as applied to two different problems: learning the intersection of half-spaces, an old problem, and clustering text corpora, a relatively new problem. Below we discuss these examples in some detail.

Learning the intersection of half-spaces. The first examples is again from learning theory. An excellent illustration of the complexity of learning is Blum and Rivest's result about training a 3-node neural network: we are given points in n-dimensional space each colored with one of two colors, red and blue. It is known that the red points can be separated from the blue points using two half-spaces. Finding these half-spaces, equivalent to training a 3-node neural network, is NP-hard [14]. In spite of this apparent hardness, it cannot be ignored that the intersection of 2 or, more generally, k half-spaces, is a natural generalization of a perceptron that approximates a simple neural network used in many machine learning applications.

How do we get around the difficulty? We restrict the distributions from which labeled examples are drawn. Under the assumption that the distribution from which points are presented is "non-concentrated"[2], we present a simple algorithm to learn the intersection of k half-spaces in $\boldsymbol{R}^n$ [60]. Generalizing all previous algorithms for the problem, our approach works for k up to $\log n / \log\log n$ in polynomial-time. In addition it explicitly finds a set of $O(k)$ planes which agree on $1-\epsilon$ of the distribution with the true set of k planes (with high probability). Our algorithm is inspired by the the following observation. The true complexity of the problem is determined *not* by the dimension n or the number of half-spaces k, but by the dimension of the subspace spanned by their normal vectors to the defining hyperplanes (the "relevant" subspace). The key step is a projection to random lines (1-dimensional subspaces) to identify the relevant subspace.

Fast Latent Semantic Indexing. Our second example is drawn from the burgeoning field of Information Retrieval.

Latent Semantic Indexing (LSI) [19] is an information retrieval method which attempts to capture the hidden structure in a corpus of documents by using techniques from linear algebra. Vectors representing the documents are projected in a new, low-dimensional space obtained by *singular value decomposition* of the term-document matrix A. This low-dimensional space is spanned by the eigenvectors of A^TA that correspond to the few largest eigenvalues — and thus, presumably, to the few most striking correlations between terms Queries are also projected and processed in this low-dimensional space. This results not only in great savings in storage and query time (at the expense of some considerable preprocessing), but also, according to empirical evidence reported in the literature, to *improved information retrieval* [10, 22, 23]. Indeed, it has been repeatedly reported that LSI outperforms, with regard to precision and recall in standard collections and query workloads, more conventional vector-based methods.

We use a probabilistic corpus model and probabilistic analysis to *prove rigorously* that, under certain conditions, LSI succeeds in capturing the underlying semantics of

[2] The probability density is polynomially bounded from above and inverse polynomially from below.

the corpus and achieves improved retrieval performance.

Then, we propose the technique of random projection as a way of speeding up latent semantic indexing. This idea yield an interesting improvement on LSI: we can perform the LSI precomputation not on the original term-document matrix, but on a low-dimensional projection, at great computational savings and no great loss of accuracy.

The last result can be seen as an alternative to (and to some extent, a justification of) *sampling* in LSI. Reports on LSI experiments in the literature seem to suggest that LSI is often done not on the entire corpus, but on a randomly selected subcorpus (both terms and documents may be sampled, although it appears that most often documents are). There is very little non-empirical evidence of the accuracy of such an approach. Our result suggests a different and somewhat more elaborate approach — projection on a random low-dimensional subspace — which can be proved to be accurate. We supplement some of our theorems with experiments on corpora derived from our statistical model.

Sampling lattice points. An old question in mathematics concerns the size of a convex body: how to compute its volume ?

One could imagine placing the body in a grid of equally spaced points and then counting up the number of points that were *inside* it. Intuitively, this should approximate the volume of the body. More than 150 years ago, the mathematician Gauss turned this into a precise question: Exactly when does the volume of a convex body approximate the number of (unit-spaced) lattice points inside it?

Using a celebrated algorithm of Dyer, Frieze and Kannan [24], we derive a sufficient condition in answer to Gauss' question: roughly speaking, if the body contains a ball of radius at least as large as the dimension of the space, then the volume and number of lattice points are within a constant factor of each other (In fact, this condition is tight) [40].

From this general condition, we are able to derive polynomial-time sampling (and counting) algorithms for various special cases of the problem, such as *contingency tables*, *multi-dimensional knapsack* problems, and *integral flows*.

1.2 Organization of this dissertation

In the rest of this chapter we give some basic mathematical background. In chapter 2 we define outliers, show how to remove them, and apply it learning perceptrons and noisy perceptrons.

In chapters 3 and 4 we apply random projection in two different scenarios, first to the intersection of half-spaces, then to quickly approximating the eigenspace of a matrix.

In chapter 5 we describe an algorithm to sample lattice points in a convex body and give some applications.

Chapters 2-5 can be read independently of each other. The background section ahead might be a useful reference for all of them.

1.3 Mathematical background

We recollect some basic definitions and well-known theorems from probability, geometry, algebra, and learning theory. The material in this chapter is not meant to be comprehensive. It concentrates on results that we will employ in the chapters ahead.

Probability

Let $X_1, X_2, \ldots, X_n$ be independent random variables with finite expectations and variances.

Let

$$\begin{aligned} S &= X_1 + \ldots + X_n, & \overline{X} &= S/n, \\ \mu &= \mathbf{E}[\overline{X} = \mathbf{E}[S/n], & \sigma^2 &= n\mathbf{var}(\overline{X}) = (\mathbf{var}S)/n. \end{aligned}$$

Then the following upper bounds can be placed on the probability that the sum of the random variables deviates from its expectation. The first inequality below is the Bienaymé-Chebyshev inequality and the latter two are Hoeffding's inequalities [35].

$$Pr[|\overline{X} - \mu| \geq t] \quad \leq \quad \frac{\sigma^2}{nt^2} \tag{1.1}$$

Assuming that for all i, $0 \leq X_i \leq 1$,

$$Pr[\overline{X} - \mu \geq t] \quad \leq \quad e^{-2nt^2} \tag{1.2}$$

An extension of the previous bound where we assume that for each i, $a_i \leq X_i \leq b_i$,

$$Pr[\overline{X} - \mu \geq t] \quad \leq \quad e^{-2n^2t^2/\sum_{i=1}^{n}(b_i - a_i)^2} \tag{1.3}$$

Geometry

B_n refers to the ball of unit radius in $\boldsymbol{R}^n$.

The *volume* of B_n is

$$\frac{2r^n \pi^{n/2}}{n\Gamma(n/2)}.$$

The volume of a *cone* in $\boldsymbol{R}^n$ of height h and base radius r is

$$\frac{\mathbf{vol} B_{n-1} r^n h}{n}.$$

Linear algebra

Given a real square matrix A with n rows and n columns, a vector $v \in \boldsymbol{R}^n$ is an eigenvector of A with *eigenvalue* λ if

$$Av = \lambda v.$$

For more on the theory of eigenspaces we refer the reader to [32, 61].

Learning theory

We recall two basic definitions in learning theory. One is Valiant's notion of *Probably Approximately Correct* learning (PAC learning) [57].

In the PAC-model we assume that examples are being provided from some fixed (but possibly unknown) distribution. Given an example distribution D, the error of a hypothesis h with respect to a target concept c is $\mathbf{Prob}_{x \in D}[h(x) \neq c(x)]$.

In the definition below, n is the size of a single example.

An algorithm $\mathcal{A}$ PAC-learns a concept class C by hypothesis H if, for any $c \in C$, any distribution D over the instance space, any $\epsilon, \delta > 0$, and for some polynomial p, the following is true. Algorithm $\mathcal{A}$ with access to labelled examples of c drawn from distribution D produces with probability at least $1 - \delta$, a hypothesis $h \in H$ with error at most ϵ. In addition, $\mathcal{A}$ should do so after running for time at most $p(n, 1\epsilon, 1\delta, size(c))$ (this trivially puts the same upper bound on the number of examples seen by the algorithm).

The second important definition is the *VC-dimension* of a concept class [59].

For this we say that a set of points S is *shattered* by a concept class C if there are concepts in C that partition S in all of the $2^{|S|}$ possible ways, i.e., all possible ways of classifying S are achievable using concepts in C.

Then the VC-dimension of a concept class is the size of the largest set of points that can be shattered by C.

Let a hypothesis be a *bad* hypothesis if its error is more than ϵ. Then the following nice theorem places an upper bound on the number of examples required for uniform convergence (i.e., till all bad hypotheses see a wrong example).

Theorem 1 *[59]*

$$m = O(\frac{1}{\epsilon}(VCdim(C)\log(\frac{1}{\epsilon}) + \log\frac{1}{\delta})).$$

Chapter 2

Removing Outliers from Numeric Data

We present a robust notion of outliers in numeric data, and a polynomial-time algorithm to remove a small fraction of any set of points so that the remaining set has no such outliers.

2.1 Introduction

The term "outlier" is a familiar one in many contexts. Statisticians have several notions of outliers. Typically these notions quantify how far the outlier is from other values, e.g., the difference between the outlier and the mean of all points, the difference between the outlier and the mean of the remaining values, or the difference between the outlier and the next closest value. In addition this difference might be normalized by some measure of the "scatter" of the set, such as the range or the *standard deviation*. Points that are outside some cut-off are labelled outliers.

One possible cause for the presence of outliers is experimental error. In this case, of course, it is desirable to detect and remove them. Even if this is not the case, removing outliers often gives a much clearer or simpler explanation for the remaining set. Outliers in the data given to a computer program could affect its performance. Conceivably they could slow down or even *mislead* an algorithm. Machine learning algorithms are an example where outliers in the *training* data might lead the algorithm to find a wayward hypothesis.

How does one detect outliers? Simple heuristics for this are based on defining them as above. In the univariate or bivariate cases (1-dimensional or 2-dimensional) one could simply plot the points, and visually decide which ones are astray, perhaps because they are too deviant in one of the coordinates. In general, i.e., in the multivariate case, this need not be true. An outlier could be far away from the rest of the points without being so in any one coordinate.

We develop a robust definition of outliers for a point set (or a probability distribution) in n-dimensional space, $\boldsymbol{R}^n$. Roughly speaking, our notion is that a point is an outlier if it deviates by some prescribed amount from the average in *any* direction (not just one of the coordinate axis directions).

Given this definition, two questions arise: (1) Is it possible to quickly detect such outliers? (2) Is it possible to remove a small subset of the points so there are no outliers left?

The second question is related to the following concern. Suppose we find the outliers and remove them from the data. It is then possible that points that were previously not outliers now become outliers. Can this happen repeatedly, so that we end up removing most of the data set?

In this chapter, we show that for a reasonable ratio of outlier:average, the number of outliers is at most a small fraction of the total number of points, i.e., on removing this fraction of points the remaining data set has no outliers. We give a polynomial time algorithm for detecting outliers in $\boldsymbol{R}^n$.

This chapter is organized into the following sections. First we state our *Outlier Removal Lemma* precisely. Then we give an algorithmic proof of the lemma, i.e., we show that it is indeed possible to remove a small number of points so that the remaining data has no outliers, and describe how to do this efficiently. In the next section we apply the lemma to obtain an efficient algorithm for learning a noisy half-space. The following section contains a discussion of some experiments we conducted that suggest that the technique might have wide-ranging applications. We conclude with some remarks about possible improvements.

2.2 The Outlier Removal Lemma

We assume that all points are given to some b bits of precision. More precisely, we define $I_b = \{p/q : |p|, |q| \in \{0, 1, 2, \ldots, 2^b - 1\}, q \neq 0\}$, and assume that our point set S is restricted to I_b^n (i.e., $I_b \times \cdots \times I_b$). Our main lemma states that given a set of data points in I_b^n, one can remove a small portion and guarantee that the remainder contains no outliers in a certain well-defined sense.

Lemma 2 (Outlier Removal Lemma) *For any set $S \subseteq I_b^n$ and any $\varepsilon > 0$, there exists a subset $S' \subseteq S$ such that:*

(i) $|S'| \geq (1 - \varepsilon - 2^{-nb})|S|$, *and*

(ii) for every vector $w \in \boldsymbol{R}^n$, $\max_{x \in S'} (w \cdot x)^2 \leq \beta \mathbf{E}_{x \in S'}[(w \cdot x)^2]$,

where $\beta = O(n^7 b/\varepsilon)$. Moreover, such a set S' can be computed in polynomial time.

The algorithm for computing the set S' of Lemma 2 is quite simple. It is as follows:

First, we may assume that the matrix X of points in S has rank n; otherwise we simply drop to the subspace spanned. Next we calculate a symmetric factorization of XX^T as $A^2 = XX^T$ which can be done by a standard eigenvalue/eigenvector computation. Then, we perform the linear transformation $A^{-1}X$. The new set of points (or viewed alternatively, the transformed space) has the property that, for every unit vector w, $\mathbf{E}_{x \in S}[(w \cdot x)^2] = 1$. Then we remove all points $x \in S$ such that $|x|^2 \geq \beta/144n$. If S now satisfies the condition of the theorem we stop. Otherwise, we repeat.

The difficult issue is proving that this algorithm will in fact halt before removing too many points from S. To do this we show that at each iteration as described above the volume of an associated *dual* ellipsoid doubles. From our assumption that points are given to a fixed number of bits, we can derive an absolute upper bound on the maximum volume of the ellipsoid, thus bounding the number of iterations. Although the idea is simple, the proof involves some detailed calculation. The reader could proceed directly to section 2.3 without loss of continuity.

2.2.1 An algorithmic proof of the lemma

For a set $S \subseteq \boldsymbol{R}^n$ (S need not be finite) and a distribution μ on R^n, let

$$W_\mu(S) = \{w \in \boldsymbol{R}^n : \mathbf{E}_\mu[(w^T x)^2 \mid x \in S] \leq 1\}.$$

We will drop the subscript μ (on both W and on the expectation $\mathbf{E}$) when the distribution is clear from context. The key to our proof is the following lemma.

Lemma 3 *Let μ be a measure on $\boldsymbol{R}^n$ which is not concentrated on a subspace of dimension less than n (i.e. the total measure on any subspace of dimension less than n is less than 1). Then, for any $0 < \alpha < 1/3n$, $\beta = 36n^3/\alpha$ and n sufficiently large, there exists an ellipsoid $S \subseteq \boldsymbol{R}^n$ such that*

(a) $\Pr(x \notin S) \leq \alpha$.

(b) *Either*

(i) *for all* $w \in \boldsymbol{R}^n$, $\max\{(w^T x)^2 : \ x \in S\} \leq \beta \mathbf{E}((w^T x)^2 \mid x \in S)$, *or*

(ii) $vol(W(S)) \geq 2 vol(W(\boldsymbol{R}^n))$.

Proof. In the entire proof, probabilities and expectations are w.r.t. the distribution μ. Let

$$\begin{aligned} M &= \mathbf{E}(xx^T) \\ &= A^2, \end{aligned}$$

where A is symmetric, and non-singular by assumption. Then

$$\mathbf{E}((w^T x)^2) = w^T M w$$

for all $w \in \boldsymbol{R}^n$. Now let

$$\begin{aligned} E &= \{x \in \boldsymbol{R}^n : (w^T x)^2 \leq w^T M w, \forall w \in \boldsymbol{R}^n\} \\ &= \{x \in \boldsymbol{R}^n : ((Aw)^T (A^{-1} x))^2 \leq |Aw|^2, \forall w \in \boldsymbol{R}^n\} \\ &= \{x \in \boldsymbol{R}^n : |A^{-1} x| \leq 1\}. \end{aligned}$$

Note that this shows that E is an ellipsoid. Putting $z = A^{-1} x$ we see that for any $\gamma > 0$,

$$\begin{aligned} \Pr(x \notin \gamma E) &= \Pr(|z| > \gamma) \\ &\leq \sum_{j=1}^{n} \Pr(|z_j| \geq \gamma/\sqrt{n}) \\ &\leq n\gamma^{-2} \sum_{j=1}^{n} \mathbf{E}(z_j^2), \end{aligned}$$

by the Chebychev inequality.

But,

$$\begin{aligned} \mathbf{E}(zz^T) &= \mathbf{E}(A^{-1} x x^T A^{-1}) \\ &= I \end{aligned}$$

and so

$$\Pr(x \notin \gamma E) \leq n^2/\gamma^2.$$

We now take $\gamma = n/\alpha^{1/2}$, $S = \gamma E$ and we see that (a) of the lemma is satisfied.

We now consider two possibilities:

Case (i) For all $w \in \boldsymbol{R}^n$,

$$\mathbf{E}((w^T x)^2 \mid x \in S) \geq \gamma^2 \mathbf{E}((w^T x)^2)/\beta.$$

In this case, for any $w \in \boldsymbol{R}^n$, by the definitions of $\mathbf{E}$ and S,

$$\begin{aligned} \max_{x \in S}\{(w^T x)^2) &\leq \gamma^2 \mathbf{E}((w^T x)^2) \\ &\leq \beta \mathbf{E}((w^T x)^2 \mid x \in S). \end{aligned}$$

Case (ii) There exists $\hat{w} \in \boldsymbol{R}^n$ such that

$$\mathbf{E}((\hat{w}^T x)^2 \mid x \in S) < \gamma^2 \mathbf{E}((\hat{w}^T x)^2)/\beta. \tag{2.1}$$

Let

$$M_1 = \mathbf{E}(xx^T \mid x \in S).$$

So

$$\mathbf{E}((w^T x)^2 \mid x \in S) = w^T M w.$$

We complete the lemma by showing that

$$\mathrm{vol}(T_1) \geq 2\mathrm{vol}(T), \tag{2.2}$$

where

$$\begin{aligned} T &= W(\boldsymbol{R}^n) \\ &= \{w \in \boldsymbol{R}^n : \ w^T M w \leq 1\} \end{aligned}$$

and

$$\begin{aligned} T_1 &= W(S) \\ &= \{w \in \boldsymbol{R}^n : \ w^T M_1 w \leq 1\} \end{aligned}$$

It will be convenient to show that

$$\mathrm{vol}(AT_1) \geq 2\mathrm{vol}(AT), \tag{2.3}$$

which is equivalent to (2.2) because the linear transformation A multiplies volumes by $|det(A)|$.

Note next that by substituting $v = Aw$ we see that

$$\begin{aligned} AT &= \{v \in \boldsymbol{R}^n : \ v^T A^{-1} M A^{-1} v \le 1\} \\ &= \{v \in \boldsymbol{R}^n : \ v^T v \le 1\} \\ &= B_n, \end{aligned}$$

where B_n is the unit ball in $\boldsymbol{R}^n$.

Furthermore,

$$\mathbf{E}((w^T x)^2 \mid x \in S) \le (1-\alpha)^{-1} \mathbf{E}((w^T x)^2)$$

which follows from $\mathbf{E}((w^T x)^2) \ge \mathbf{E}((w^T x)^2 \mid x \in S) \Pr(x \in S)$. So,

$$\begin{aligned} AT_1 &= \{v \in \boldsymbol{R}^n : \ v^T A^{-1} M_1 A^{-1} v \le 1\} \\ &\supseteq \{v \in \boldsymbol{R}^n : \ v^T A^{-1} M A^{-1} v \le 1 - \alpha\} \\ &= (1-\alpha) B_n. \end{aligned} \tag{2.4}$$

Also, AT_1 contains a vector of length $\lambda = \beta^{1/2}/\gamma$. Indeed, let

$$\hat{v} = \lambda \frac{A\hat{w}}{|A\hat{w}|}.$$

Then, $\hat{v} \in AT_1$ because, from (2.1),

$$\begin{aligned} \hat{v}^T A^{-1} M_1 A^{-1} \hat{v} &= \frac{\lambda^2}{|A\hat{w}|^2} \hat{w} M_1 \hat{w}^T \\ &\le \frac{\lambda^2}{|A\hat{w}|^2} \frac{\gamma^2}{\beta} \hat{w} M \hat{w}^T \\ &= 1. \end{aligned}$$

Since AT_1 contains an $n-1$ dimensional ball around the origin and a point at a distance of $1-\alpha$ from the center of the ball, from the convexity of AT_1 it follows then that AT_1 contains a cone with base an $(n-1)$-dimensional ball of radius $1-\alpha$ and height λ.

Thus if V_n denotes the volume of B_n we see, using the bound on α, that

$$\begin{aligned} \frac{\mathrm{vol}(AT_1)}{\mathrm{vol}(AT)} &\ge \frac{\lambda V_{n-1}(1-\alpha)^{n-1}}{n V_n} \\ &\ge \frac{\lambda (1-\alpha)^{n-1}}{2\sqrt{n}} \\ &\ge 2. \end{aligned}$$

□

We now specialize the above result to the case where μ is concentrated on I_b^n. Let $L_0 = \{x \in I_b^n : \mu(x) \geq 2^{-4nb}\}$. So, $\mu(L_0) \geq 1 - |I_b^n| 2^{-4nb} \geq 1 - 2^{-nb}$.

Let μ_0 denote the measure induced on L_0 by μ i.e. $\mu_0(x) = \mu(x)/\mu(L_0)$ for $x \in L_0$. We consider applying the construction of Lemma 3, K times starting with μ_0. In general we would expect to construct a sequence of ellipsoids S_i This assumes Case (bii) always occurs. Let μ_i denote the measure induced on $S_1 \cap S_2 \cap \cdots \cap S_i$ by μ_0. It is possible that μ_i is concentrated on a subspace V_i of lower dimension. If so, we simply work within V_i from then on. This cannot happen more than n times.

Suppose that Case (bi) never occurs. Then there exists a subspace V_K of dimension ν and ellipsoids $S_1, S_2, \ldots, S_K$ such that if $T_K = L_0 \cap S_1 \cap S_2 \cap \cdots \cap S_K \cap V_K$ then

(a) $\dim(T_K) = \nu$.

(b) $\mu_0(T_K) \geq 1 - \alpha K$.

(c) $\mathrm{vol}_\nu(W(T_K)) \geq 2^{K/n}$,

where in (c),

$$W(T_K) = \{w \in V_K : \mathbf{E}((w^T x)^2 \mid x \in T_K) \leq 1\}.$$

Part (c) takes into account the doubling of volume K times, and restarting each time we move to a lower dimensional subspace (at most n times).

The above is not possible for sufficiently large K as we will now show by bounding the length of each $w \in T_K$. By assumption, T_K contains ν linearly independent vectors $v_1, v_2, \ldots, v_\nu \in I_b^n$. For any such set of vectors,

$$\begin{aligned} \mathbf{E}((w^T x)^2 \mid x \in T_K) &\geq \sum_{i=1}^{\nu} (w^T v_i)^2 \mu_K(v_i) \\ &\geq 2^{-4nb} \sum_{i=1}^{\nu} (w^T v_i)^2. \end{aligned}$$

So if $w \in W(T_K)$ then

$$\sum_{i=1}^{\nu} (w^T v_i)^2 \leq 2^{4nb}. \tag{2.5}$$

Let B denote the $n \times n$ matrix $\sum_{i=1}^{\nu} v_i v_i^T$ so that

$$w^T B w = \sum_{i=1}^{\nu} (w^T v_i)^2. \tag{2.6}$$

Let B have eigenvalues $0 = \lambda_1 = \lambda_2 = \cdots = \lambda_{n-\nu} < \bar{\lambda} = \lambda_{n-\nu+1} \leq \lambda_{n-\nu+2} \leq \cdots \lambda_n$. Let $a_1, a_2, \ldots, a_n$ be a corresponding orthonormal basis of eigenvectors. Now if $w \in$

$V_K, w \neq 0$, then

$$\frac{w^T B w}{w^T w} \geq \bar{\lambda}. \tag{2.7}$$

Since $w^T v_i \neq 0$ for at least one i, we can apply (2.6). But $\bar{\lambda} \neq 0$ is a root of a polynomial of degree at most $n-1$ with rational coefficients α_i/β_i where $|\alpha_i|, |\beta_i| \leq n!2^{nb}$. By a simple computation, this implies that $\bar{\lambda} \geq (n!2^{nb})^{-2n}$ and so (2.5), (2.6 and (2.7) imply that if $w \in W(T_K)$ then

$$|w|^2 \leq 2^{4nb} 2^{2n^2 b} (n!)^{2n} \leq 2^{4n^2 b}$$

(for $b > \log n$) and so

$$\mathrm{vol}_\nu(W(T_K)) \leq (2^{4n^2 b})^{n/2}.$$

This is a contradiction for $K \geq K_0 = 2n^4 b$. We deduce then that

Theorem 2 *For any $0 < \alpha < 1/3n$ and $\beta = 36n^3/\alpha$ and μ concentrated on I_b^n, there exist $k \leq K_0$ ellipsoids S_i such that if $S = \bigcap_{i=1}^k S_i$*

(i) $\mu(S) \geq 1 - k\alpha - 2^{-nb}$.

(ii) $\max\{(w^T x)^2 : x \in S\} \leq \beta \mathbf{E}((w^T x)^2 \mid x \in S)$, *for all* $w \in \boldsymbol{R}^n$.

The previous discussion has been existential in nature and we now show how to make it constructive. This is relatively easy for a finite set of m points (i.e μ is concentrated on the m points). Now if we apply the above theorem to μ then all of the ellipsoids and subspaces are computable in polynomial time.

One way to view the algorithm is the following. We wish to find a set of points with the property that in any direction w, the maximum squared value of the projection of points in that direction is not much more than the average. If initially there is a direction where this is not true, we apply a transformation to the points ($A^{-1}x$, above) that results in their inertial ellipsoid becoming the unit ball. Then we drop all points outside a multiple γ of this ellipsoid and repeat on the smaller set of points (with their original coordinates). This cannot go on forever since we assume that the points are represented by bounded rationals and an associated ellipsoid is doubling in volume at each iteration.

Note that we can make the method constructive for the infinite case as well by picking a sample of points and applying VC-dimension arguments.

2.3 An application: learning a noisy half-space

The problem of learning a linear threshold function is one of the oldest problems studied in machine learning. Typically, this problem is solved by using simple greedy methods. For instance, one commonly-used greedy algorithm for this task is the Perceptron Algorithm [54, 5], described below in Section 2.3.1. These algorithms have running times that depend on the amount of "wiggle room" available to a solution. In particular, the Perceptron Algorithm has the following guarantee [50]. Given a collection of data points in $\boldsymbol{R}^n$, each labeled as *positive* or *negative*, the algorithm will find a vector w such that $w \cdot x > 0$ for all positive points x and $w \cdot x < 0$ for all negative points x, if such a vector exists.[1] Moreover, the number of iterations made by the algorithm is at most $1/\sigma^2$ where σ is a "separation parameter" defined as the largest value such that for some vector w^*, all positive x satisfy $\cos(w^*, x) > \sigma$, and all negative x satisfy $\cos(w^*, x) < -\sigma$, where $\cos(a, b) = \frac{a \cdot b}{|a||b|}$ is the cosine of the angle between vectors a and b.

Unfortunately, it is possible for the separation parameter σ to be exponentially small, and for the algorithm to take exponential time, even if all the examples belong to $\{0, 1\}^n$. A classic setting in which this can occur is a data set labeled according to the function "if $x_1 = 1$ then positive else if $x_2 = 1$ then negative else if $x_3 = 1$ then positive, ...". This function *has* a linear threshold representation, but it requires exponentially large weights and can cause the Perceptron Algorithm to take exponential time. (In practice, though, the Perceptron Algorithm and its variants tend to do fairly well; e.g., see [7].)

Given this difficulty, one might propose instead to use a polynomial-time linear programming algorithm to find the desired vector w. Each example provides one linear constraint and one could simply apply an LP solver to solve them [44, 41, 48]. In practice, however, this approach is less often used in machine learning applications. One of the main reasons is that the data often is not consistent with *any* vector w and the goal is simply to do as well as one can. And, even though finding a vector w that minimizes the number of misclassified points is NP-hard, variants on the Perceptron Algorithm typically do well in practice[31, 6]. In fact, it is possible to provide guarantees for variations on the Perceptron Algorithm in the presence of inconsistent data (e.g., see [15, 16, 42][2]), under models in which the inconsistency is

[1] If a non-zero threshold is desired, this can be achieved by adding one extra dimension to the space.

[2] The word "polynomial" in the title of [15] means polynomial in the inverse of the separation parameter, which as noted above can be exponential in n even when points are chosen from $\{0, 1\}^n$.

produced by a sufficiently "benign" process, such as the *random classification noise model* discussed below.

We are given access to examples (points) drawn from some distribution D over R^n. Each example is labeled as positive or negative. The labels on examples are determined by some unknown target function $w^* \cdot x > 0$ (i.e., x is positive if $w^* \cdot x > 0$ and is negative otherwise) but each label is then flipped independently with some fixed probability $\eta < 1/2$ before it is presented to the algorithm. η is called the *noise rate*.

Our goal is to find an algorithm that for any (unknown) distribution D, any (unknown) target concept $w^* \cdot x > 0$, any (unknown) $\eta < 1/2$, and any inputs $\epsilon, \delta > 0$, with probability at least $1 - \delta$ produces a hypothesis whose error with respect to the target function is at most ϵ. The algorithm may request a number of examples polynomial in $n, b, 1/\epsilon, \log(\frac{1}{\delta})$, and $\frac{1}{1-2\eta}$, and should run in time polynomial in these parameters as well.

Here we present a version of the Perceptron Algorithm that maintains its properties of noise-tolerance, while providing polynomial-time guarantees. Specifically, the algorithm we present is guaranteed to provide a weak hypothesis (one that correctly classifies noticeably more than half of the examples) in time polynomial in the description length of the input and *not* dependent on any separation parameter. The output produced by the algorithm can be thought of as a "thick hyperplane," satisfying the following two properties:

1. Points outside of this thick hyperplane are classified with high accuracy (points inside can be viewed as being classified as "I don't know").

2. At least a $1/poly$ fraction of the input distribution lies outside of this hyperplane.

This sort of hypothesis can be easily boosted in a natural way (by recursively running the algorithm on the input distribution restricted to the "don't know" region) to achieve a hypothesis of arbitrarily low error.[3] This yields the following theorem.

[3]Thanks to Rob Schapire for pointing out that standard Boosting results [55, 29] do not apply in the context of random classification noise. (It is an open question whether arbitrary weak-learning algorithms can be boosted in the random classification noise model.) Thus, we use the fact that the hypothesis produced by the algorithm can be viewed as a high-accuracy hypothesis over a known, non-negligible portion of the input distribution. Alternatively, Aslam and Decatur [3] have shown that Statistical Query (SQ) algorithms *can*, in fact, be boosted in the presence of noise. Since our algorithm can be made to fit the SQ framework (see Section 2.3.5), we could also apply their results to achieve strong learning.

Theorem 3 *The class of linear threshold functions in $\boldsymbol{R}^n$ can be learned in polynomial time in the PAC prediction model in the presence of random classification noise.*

Remark: The learning algorithm can be made to fit the Statistical Query learning model [42].

The main idea of our result is as follows. First, we modify the standard Perceptron Algorithm to produce an algorithm that succeeds in weak learning unless an overwhelming fraction of the data points lie on or very near to some hyperplane through the origin. Specifically, the algorithm succeeds unless there exists some "bad" vector w such that most of the data points x satisfy $|\cos(w, x)| < \delta$ for some small $\delta > 0$. Thus, we are done if we can somehow preprocess the data to ensure that no such bad vector w exists. We apply the Outlier Removal Lemma to ensure this.

Recall that the lemma tells us that given any set S of points in n dimensional space, each requiring b bits of precision, one can remove only a small fraction of those points and then guarantee that in the set T remaining, for every vector v,

$$\max_{x \in T}(x \cdot v)^2 \leq poly(n, b)\mathbf{E}_{x \in T}[(x \cdot v)^2].$$

In this sense, the set remaining has no outliers with respect to any hyperplane through the origin. In addition, these outliers can be removed in polynomial time. After removing the outliers, we can then apply a linear transformation so that in the transformed space, for every unit vector v,

$$\mathbf{E}_{x \in T}[(x \cdot v)^2] = 1 \quad \textit{and} \quad \max_{x \in T}(x \cdot v)^2 \leq poly(n, b).$$

Because the maximum is bounded, having the expectation equal to 1 means that for every hyperplane through the origin, at least a $1/poly(n, b)$ fraction of the examples are at least a $1/poly(n, b)$ distance away, which then allows us to guarantee that the modified Perceptron Algorithm will be a weak learner.

2.3.1 The Perceptron Algorithm

The Perceptron Algorithm[54, 5] operates on a set S of labeled data points in n dimensional space. Its goal is to find a vector w such that $w \cdot x > 0$ for all positive points x and $w \cdot x < 0$ for all negative points x. We will say that such a vector w *correctly classifies* all points in S. If a non-zero threshold value is desired, this can be handled by simply creating an extra $(n + 1)$st coordinate and giving all examples a value of 1 in that coordinate.

For convenience, define $\ell(x)$ (the label of x) to be 1 if x is positive and -1 if x is negative. So, our goal is to find a vector w such that $\ell(x)(w \cdot x) > 0$ for all $x \in S$. Also, for a point x let $\hat{x} = x/|x|$. I.e., $\hat{x}$ is the vector x normalized to have length 1.

2.3.2 The standard algorithm

The standard algorithm proceeds as follows. We begin with $w = \vec{0}$. We then perform the following operation until all examples are correctly classified:

Pick some arbitrary misclassified example $x \in S$ and let $w \leftarrow w + \ell(x)\hat{x}$.

A classic theorem (see [50]) describes the convergence properties of this algorithm.

Theorem 4 *[50] Suppose the data set S can be correctly classified by some unit vector w^*. Then, the Perceptron Algorithm converges in at most $1/\sigma^2$ iterations, where $\sigma = min_{x \in S}|w^* \cdot \hat{x}|$.*

Proof. Consider the cosine of the angle between the current vector w and the unit vector w^* given in the theorem. That is, $\frac{w \cdot w^*}{|w|}$. In each step of the algorithm, the numerator of this fraction increases by at least σ because $(w + \ell(x)\hat{x}) \cdot w^* = w \cdot w^* + \ell(x)\hat{x} \cdot w^* \geq w \cdot w^* + \sigma$. On the other hand, the square of the denominator increases by at most 1 because $|w + \ell(x)\hat{x}|^2 = |w|^2 + 2\ell(x)(w \cdot \hat{x}) + 1 < |w|^2 + 1$ (since x was misclassified, this means the crossterm is negative). Therefore, after t iterations, $w \cdot w^* \geq t\sigma$ and $|w| < \sqrt{t}$. Notice that the former cannot be larger than the latter. Thus, $t \leq 1/\sigma^2$. □

2.3.3 A modified version

We now describe a modified version of the Perceptron Algorithm that will be needed for our construction. Recall our notation that $\cos(a, b)$ is the cosine of the angle between vectors a and b, or equivalently $\frac{a \cdot b}{|a||b|}$.

The reason we need to modify the algorithm is this: In the standard algorithm, if some of the points are far from the target plane (in the sense that $\cos(w^*, x)$ is large) and some are near, then eventually the hypothesis will correctly classify the far away points but may make mistakes on the nearby ones. This is simply because the points far from $w^* \cdot x = 0$ cause the algorithm to make substantial progress but the others do not. Unfortunately, we cannot test for points being far or near to the target plane. So, we cannot produce the rule: "if $|\cos(w^*, x)|$ is large then predict based on $x \cdot w$, else say 'I don't know'." What we want instead is an algorithm that does well on points that are far from the *hypothesis* plane, because $|\cos(w, x)|$ is something that the algorithm *can* calculate. If we then can guarantee that a reasonable fraction of points will have this property, we will have our desired weak hypothesis (just replacing w^* by w in the above rule).

Specifically, our modified algorithm takes as input a quantity σ and its goal is to produce a vector w such that every misclassified $x \in S$ should satisfy $|\cos(w, x)| \leq \sigma$. The algorithm proceeds as follows.

The Modified Perceptron Algorithm

1. Begin with w as a random unit vector.

2. If every misclassified $x \in S$ satisfies $|\cos(w, x)| \leq \sigma$ (i.e., if $|w \cdot \hat{x}| \leq \sigma|w|$) then halt.

3. Otherwise, pick the misclassified $x \in S$ maximizing $|\cos(w, x)|$ and update w using:
$$w \leftarrow w - (w \cdot \hat{x})\hat{x}.$$
In other words, we add to w the appropriate multiple of x so that w is now orthogonal to x, i.e., we add the multiple of x that shrinks w as much as possible.

4. If we have made fewer than $(1/\sigma^2)\ln n$ updates then go back to Step 2. Otherwise, go back to Step 1 (begin anew with a new random unit starting vector).

Theorem 5 *If the data set S is linearly separable, then with probability $1-\delta$ the modified perceptron algorithm halts after $O((1/\sigma^2)\ln(n)\ln(\frac{1}{\delta}))$ iterations, and produces a vector w such that every misclassified $x \in S$ satisfies $|\cos(w, x)| \leq \sigma$.*

Proof. Let w^* be a unit vector that correctly classifies all $x \in S$. Suppose it is the case that the initial (random unit) vector w satisfies $w \cdot w^* \geq 1/\sqrt{n}$. Notice that in each update made in Step (3), $w \cdot w^*$ does not decrease because

$$(w - (w \cdot \hat{x})\hat{x}) \cdot w^* = w \cdot w^* - (w \cdot \hat{x})(w^* \cdot \hat{x}) \geq w \cdot w^*$$

where the last inequality holds because w misclassifies x. On the other hand, $|w|^2$ *does* decrease significantly because (this is just the Pythagorean Theorem)

$$\begin{aligned} |(w - (w \cdot \hat{x})\hat{x})|^2 &= |w|^2 - 2(w \cdot \hat{x})^2 + (w \cdot \hat{x})^2 \\ &\leq |w|^2(1 - \sigma^2). \end{aligned}$$

Thus, after t iterations, $|w| \leq (1-\sigma^2)^{t/2}$. Since $|w|$ cannot be less than $w \cdot w^*$, this means that the number of iterations t satisfies $(1-\sigma^2)^{t/2} \geq 1/\sqrt{n}$, which implies $t \leq (\ln n)/\sigma^2$.

Each time we choose a random initial unit vector for w, there is at least a constant > 0 probability that w satisfies our desired condition that $w \cdot w^* > 1/\sqrt{n}$. Thus, the theorem follows. □

We have described the algorithm as one that runs in *expected* polynomial time. Alternatively we could stop the algorithm after a suitable number of iterations and have a high probability of success. In Section 2.3.5 we will alter this algorithm slightly to make it tolerant to random classification noise.

2.3.4 A new guarantee for an old algorithm

The Modified Perceptron Algorithm can be combined with the Outlier Removal Lemma in a natural way. Given a data set S, we use the Lemma to produce a set S' with $|S'| \geq \frac{1}{2}|S|$ and such that for all vectors w, $\max_{S'}(w \cdot x)^2 \leq \beta \mathbf{E}_{S'}[(w \cdot x)^2]$ where β is polynomial in n and b.

We then reduce dimensionality if necessary to get rid of any vectors w for which the above quantity is zero. That is, we project onto the subspace L spanned by the eigenvectors of the XX^T matrix having non-zero eigenvalue (X is the matrix of points in S'). Now, we perform the linear transformation A^{-1} described in Section 2.2 so that in the transformed space, for all unit vectors w, $\mathbf{E}_{S'}[(w \cdot x)^2] = 1$. Our guarantee for set S' implies that in the transformed space, $\max_{S'} |x|^2 \leq \beta n$. Thus, for any unit vector w,

$$\begin{aligned}
\mathbf{E}_{S'}[\cos(w,x)^2] &= \mathbf{E}_{S'} \frac{(w \cdot x)^2}{|x|^2} \\
&\geq \frac{\mathbf{E}_{S'}[(w \cdot x)^2]}{\max_{S'} |x|^2} \\
&\geq 1/(\beta n).
\end{aligned}$$

This implies that in the transformed space, at least a $1/(2\beta n)$ fraction of the points in S' satisfy $\cos(w,x)^2 \geq 1/(2\beta n)$. We can now run the Modified Perceptron Algorithm with $\sigma = 1/\sqrt{2\beta n}$, and guarantee that at the end, at least a $1/(2\beta n)$ fraction of the points in S' satisfy $|\cos(w,x)| \geq \sigma$.

The final hypothesis of the algorithm, in the original untransformed space, is: if $x \notin L$ or $|cos(w, A^{-1}x)| < \sigma$ then guess the label randomly (or say "I don't know"), and otherwise predict according to the hypothesis $w^T A^{-1} x > 0$.

Achieving strong (PAC) learning

The algorithm presented above splits the input space into a classification region

$$\{x : x \in L \text{ and } |\cos(w, A^{-1}x)| \geq \sigma\}$$

and a don't-know region

$$\{x : x \notin L \text{ or } |\cos(w, A^{-1}x)| < \sigma\}.$$

By standard VC-dimension arguments [59], if the sample S is drawn from distribution D, then for any $\epsilon, \delta > 0$, if S is sufficiently (polynomially) large, then with high probability ($\geq 1 - \delta$), the true error of the hypothesis inside the classification region is less than ϵ. Furthermore, the weight under D of the classification region is at least $1/poly(n, b)$; that is, the fraction of S that lies in the classification region is representative of the weight of this region under D. Therefore, we can boost the accuracy of the learning algorithm by simply running it recursively on the distribution D restricted to the don't-know region. The final hypothesis produced by this procedure is a decision list of the form: "if the example lies in the classification region of hypothesis 1, then predict using hypothesis 1, else if the example lies in the classification region of hypothesis 2, then predict using hypothesis 2, and so on".

2.3.5 Learning with noise

We now describe how the Modified Perceptron Algorithm can be converted to one that is robust to *random classification noise*. We do this by recasting the algorithm in the Statistical Query (SQ) model of Kearns [42] as extended by Aslam and Decatur [4], and to use the fact that any SQ algorithm can be made tolerant of random classification noise.

We begin with some observations. For convenience, in the discussion below we will normalize the examples to all have length 1, so that we need not distinguish between x and $\hat{x}$. Recall that $\ell(x) = 1$ if x is a positive example and $\ell(x) = -1$ if x is a negative example.

The first observation is that the only properties of the point x selected in Step 3 of the Modified Perceptron Algorithm that are actually used in the analysis of Theorem 5 are:

$$\cos(w, x)\ell(x) \leq -\sigma, \text{ and} \tag{2.8}$$

$$\cos(w^*, x)\ell(x) \geq 0. \tag{2.9}$$

The second observation is that, in fact, we only need points that *approximately* achieve these two properties. In particular, suppose that every point x we use in Step 3 satisfies the relaxed conditions:

$$\cos(w,x)\ell(x) \leq -\sigma/2, \text{ and} \tag{2.10}$$

$$\cos(w^*,x)\ell(x) \geq \frac{-\sigma^2}{16\sqrt{n}\ln n}. \tag{2.11}$$

The first condition guarantees that after $t = (8\ln n)/\sigma^2$ iterations we have $|w| \leq (1-(\sigma/2)^2)^{t/2} < 1/n$. The second guarantees that if initially $w \cdot w^* \geq \frac{1}{\sqrt{n}}$, then after t iterations $w \cdot w^* \geq \frac{1}{\sqrt{n}} - \frac{t\sigma^2}{16\sqrt{n}\ln n} \geq \frac{1}{2\sqrt{n}}$. Therefore, we are guaranteed to halt before t iterations have been made.

The final observation is that any positive multiple of

$$\mu_{w,S} = \mathbf{E}_S[\ell(x)x : \cos(w,x)\ell(x) \leq -\sigma]$$

will satisfy conditions (2.8) and (2.9), assuming zero noise so that every $x \in S$ satisfies (2.9), and if we define $\ell(\mu_{w,S}) = 1$. Furthermore, any point sufficiently near to $\mu_{w,S}$ will satisfy the relaxed conditions (2.10) and (2.11). Specifically, the definition of $\mu_{w,S}$, the fact that all examples have length 1, and condition (2.8) together imply that $|\mu_{w,S}| \geq \sigma$. So, any point $\tilde{\mu}_{w,S}$ such that $|\tilde{\mu}_{w,S} - \mu_{w,S}| \leq \sigma^3/(16\sqrt{n}\ln n)$ satisfies conditions (2.10) and (2.11).

Statistical queries

Let f be a function from labeled examples to $[0,1]$. That is, in our setting,

$$f : R^n \times \{-1,1\} \longrightarrow [0,1].$$

A *statistical query* is a request for the expected value of f over examples drawn from distribution D and labeled according to the target concept c; i.e., a request for $\mathbf{E}_{x\in D}[f(x,c(x))]$. Assuming that f is polynomial-time computable, it is clear that given access to non-noisy data, this expectation can be estimated to any desired accuracy τ with any desired confidence $1-\delta$ in time $poly(\frac{1}{\tau}, \log(\frac{1}{\delta}))$, by simply calculating the expectation over a sufficiently large sample. Kearns [42] and Aslam and Decatur [4] prove that one can similarly perform such an estimation even in the presence of random classification noise.[4] Specifically, for any noise rate $\eta < 1/2$ and

[4]Kearns [42] considers queries with range $\{0,1\}$. Aslam and Decatur [4] extends these arguments (among other things) to queries with range $[0,1]$, which is more convenient for our purposes.

any accuracy (or *tolerance*) parameter τ, the desired expectation can be estimated with confidence $1-\delta$ in time (and sample size) $poly(\frac{1}{\tau}, \log(\frac{1}{\delta}), \frac{1}{1-2\eta})$. Thus, to prove an algorithm tolerant to random classification noise, it suffices to show that its use of labeled examples can be recast as requests for approximate expectations of this form.

The Modified Perceptron Algorithm uses labeled examples in two places. The first is in Step 2 where we ask if there are any points $x \in S$ such that $\cos(w,x)\ell(x) \le -\sigma$, and we halt if there are none. We can replace this with a statistical query requesting the probability that a random labeled example from D satisfies this property (formally, a request for $\mathbf{E}_{x\in D}[f(x,c(x))]$ where $f(x,\ell)=1$ if $\cos(w,x)\ell \le -\sigma$ and $f(x,\ell)=0$ otherwise) and halting if this probability is sufficiently small. Specifically, we can set $\tau = \frac{1}{3}\epsilon/(2\beta n)$ and halt if the result of the query is at most $\frac{2}{3}\epsilon/(2\beta n)$, where $1/(2\beta n)$ is a lower bound on $\Pr_{x\in D}(|\cos(w,x)| \ge \sigma)$ from the Outlier Removal Lemma.

The second place that labeled examples are used is in Step 3. As noted in the discussion following equations (2.10) and (2.11), it suffices for this step to use a good approximation to $\mu_{w,S}$ instead of using any specific labeled example. We can find such an approximation via statistical queries. Specifically, to approximate the ith coordinate of $\mu_{w,S}$, we ask for $\mathbf{E}_{x\in D}[\ell(x)x_i | \cos(w,x)\ell(x) \le -\sigma]$. This *conditional* expectation can be approximated as the ratio of the answers to the following two statistical queries. One is a request for $\mathbf{E}_{x\in D}[f(x,c(x))]$ where $f(x,\ell)=\ell x$ if $\cos(w,x)\le -\sigma$ and $f(x,\ell)=0$ otherwise. The other is $\Pr[\cos(w,x)\ell(x) \le -\sigma]$ which we saw how to calculate in Step 2. Note that we are guaranteed from Step 2 that $\Pr(\cos(w,x)\ell(x) \le -\sigma)$ is reasonably large. Finally, we combine the approximations for each coordinate into an approximation $\tilde{\mu}_{w,S}$ of $\mu_{w,S}$.

Note that examples are also used in the algorithm for the Outlier Removal Lemma. However, since this algorithm ignores the labels, it is unaffected by classification noise.

2.4 Experiments and other potential applications

Even though the theoretical bounds established in the previous sections are not small enough to directly imply the practicality of our techniques, the algorithms seem to do quite well in practice.

We conducted the two different experiments to gather empirical evidence. Both were based on the perceptron algorithm.

- Efficiency.

DATA. Synthetically generated, linearly-separable set of points, with a small number of outliers (10% - 20%). We generated a set of points at random, then inserted some outliers by hand.

We ran the perceptron algorithm on data sets of various sizes (using a random misclassified example at each iteration). Then we removed the outliers, and ran it again on the outlier-free data, then incorporated the outliers (usually this took only a few extra iterations). We considered a couple of different variations of the algorithm, e.g. with the misclassified examples normalized to unit length at each iteration, and with the outlier-free data transformed so that the expected squared distance was 1 in every direction. In almost every case the number of iterations to convergence dropped to 50% of the original number (and lesser in some cases).

- Quality.

  DATA. Breast Cancer measurements [62] on 569 patients with thirty attributes of measurements on the cell nucleus (such as radius, texture, and smoothness) and one field indicating the diagnosed nature of the cancer, "Benign" or "Malignant".

  On running the perceptron algorithm on the data set for different numbers of iterations, the best half-space we could find correctly classified 79% of the data set. Next we set the the outlier parameter, $\beta = 5$, and removed the outliers. There were 364 points left (the computation took a few seconds in the MATLAB environment on an IBM PowerPC). Then we ran the algorithm on the outlier-free data and tested the half-space obtained on the entire data set. It classified 91% of the data correctly [5].

  Remarks: It is possible that there exists an even better half-space for this data set. On just the outlier-free data, the half-space performed even better.

While these experiments are not in any way conclusive, they indicate that outlier removal might have many more ramifications than our theorems imply. One thing is clear — they call for more extensive experiments.

[5] It is a mystery to me that there should exist such a good half-space for separating the benign and malignant cases of breast cancer

2.5 Conclusion and open problems

In this chapter we have seen evidence that outliers could play a critical role in the performance and speed of an algorithm. We presented a method to find outliers defined in a strong way and proved theoretical guarantees about the method.

Unfortunately these guarantees are rather weak at the moment. Can the ratio between the maximum and the average, β, in the outlier removal lemma be reduced to a smaller polynomial? Our experiments suggest that the true bound might be significantly better.

Can we provide theoretical guarantees for outlier removal in other contexts such as nearest neighbor search? Very recently, Edith Cohen showed that the Outlier Removal Lemma can be used to learn a noisy half-space as a *single* half-space [17].

Chapter 3

Reducing Dimensionality by Random Projection I

We use random projection to 1-dimensional subspaces to identify the relevant subspace in an algorithm for learning the intersection of half-spaces.

3.1 Introduction: the intersection of half-spaces

In this chapter we consider the problem of learning the intersection of k half-spaces in n dimensions from labelled examples. We are presented with points in n-dimensional space each labelled *positive* or *negative*. The problem is to find a set of k half-spaces such that all the positive examples lie in a single region of intersection of the k half-spaces and all the negative examples lie outside this region, if such a set of half-spaces exists. For $k = 1$, this corresponds to learning a single half-space (also called a *perceptron*), which we considered in the previous chapter. As we observed, learning a single perceptron is equivalent to linear programming and hence can be solved in polynomial time. Other solutions to this problem, notably the perceptron algorithm, have also been studied in the literature [12] (as we saw in the last chapter). While perceptrons themselves are highly interesting and in fact have directly found applications, it is often the case that one needs a more complex concept class to accurately model some phenomenon. The intersection of k half-spaces is a natural generalization of a perceptron. Besides its intuitive geometric appeal, in principle any convex concept could be approximated by an intersection of half-spaces. It also approximates a simple neural network which is used in many machine learning applications.

What is the complexity of learning the intersection of k half-spaces? There are several negative results about this [9, 14, 46, 49]. In the distribution-free model, where we make no assumptions on the distribution from which examples are presented to us, and under the requirement that the algorithm must produce a set of k half-spaces, the problem cannot be solved in polynomial time even for $k = 2$ unless $RP = NP$ [14, 49].

On the other hand, for special distributions there are some positive results. Baum [8] gave an algorithm for learning the intersection of two homogenous half-spaces (a half-space is homogenous if the hyperplane defining it passes through the origin) over any distribution $\mathcal{D}$ that is origin-symmetric, i.e., for any $x \in R^n$, $\mathcal{D}(x) = \mathcal{D}(-x)$ (any point x and its reflection through the origin are equally likely). Recently, Blum and Kannan [13] gave a polynomial time algorithm for the problem for any constant number of half-spaces for the uniform distribution on the unit ball in n dimensions. Their algorithm does not explicitly find the half-spaces, instead it finds a prediction rule which can be evaluated in polynomial time (for a constant number of half-spaces) and is probably approximately correct. The running time and number of examples required by the algorithm are doubly exponential in k.

We present a randomized algorithm for the problem. Besides being simpler, our algorithm improves on the previous one in three ways:

- It is faster: the running time and number of examples required are (singly) exponential in k and polynomial in n. Hence we can learn the intersection of up to $O(\log n/\log\log n)$ half-spaces in polynomial time.
- The concept that it reports is shorter: our algorithm explicitly finds an (intersection of a) set of $O(k)$ half-spaces.
- It can handle more general distributions (for a constant number of half-spaces): specifically any distribution on the unit ball that is not "concentrated", i.e., whose probability density is at least $1/poly(n)$ and at most $poly(n)$ everywhere.

Although our algorithm is quite different from that of Blum and Kannan it is inspired by the the same observation: the true complexity of the problem is determined *not* by the dimension n or the number of half-spaces k, but by the dimension of the subspace spanned by their normal vectors to the defining hyperplanes. Indeed our algorithm can learn the intersection of any number of half-spaces so long as their normals span a subspace of dimension $O(\log n/\log\log n)$.

To explain the idea, let us assume that the half-spaces are homogenous, i.e., the hyperplanes defining them pass through the origin (this is actually without loss of generality for us as shown in section 3.3.5). Let the half-spaces be $w_1 \cdot x \leq 0, w_2 \cdot x \leq$

$0, \ldots, w_k \cdot x \leq 0$. The intersection of these half-spaces is the positive region P. For each half-space $w_i \cdot x \leq 0$, w_i is the normal vector to the hyperplane defining the half-space (lying in the region opposite the positive region with respect to this hyperplane). Our goal will be to find a set of normal vectors that are very close in angle to $w_1, \ldots, w_k$. For this we consider the set of all normal vectors that define hyperplanes through the origin which *do not intersect* the positive region P. This is precisely the cone at the origin formed by the vectors $w_1, \ldots, w_k$. Formally, it is the set D_P of all vectors v such that $v = \sum_j \alpha_j w_j$, $\alpha_j \geq 0$. Then each vector $v \in D_P$ has the property that for any positive example x, $v \cdot x \leq 0$. In other words D_P is the set of normal vectors of hyperplanes that do not intersect P. In linear programming theory P and D_P are *dual* to each other.

The first step of the algorithm is to find a good approximation to D_P. Although the minimum dimension of a subspace containing D_P is at most k (it could be less) we first find a good "approximation" to D_P in n dimensions. This is done by simply choosing a large sample of examples and considering the dual D_C of their conical hull C, i.e., the set of homogenous hyperplanes that do not intersect the convex hull of the examples. In the second step we apply a simple procedure based on random sampling to identify a k-dimensional subspace close to the subspace containing D_P. Then we project our n-dimensional approximation of D_P to this relevant subspace. Let this projection be $\hat{D_C}$. The next step is choose vectors from $\hat{D_C}$ to guarantee that for each w_i there is at least one vector in the sample close to it (in angle). We could do this by simply considering all points of a sufficiently fine grid enclosing $\hat{D_C}$. The size of the sample is chosen to be large enough to guarantee that for each w_i there is at least one vector in the sample close to it (in angle). Finally we prune the sample using a greedy heuristic. The half-spaces defined by the vectors in the pruned sample constitute the concept output by the algorithm. In other words, we label a point positive if it lies in the intersection of these half-spaces and negative otherwise.

For most of the discussion we assume that the half-spaces we are trying to learn are homogenous. In the next section we introduce the framework and notation. Then we describe our algorithm in detail. Section 3.3.1 is devoted to proving a property we need of a large sample of examples. Section 3.3.2 outlines a proof of the sampling procedure used to find the relevant subspace. Section 3.3.3 describes the grid we use for sampling and a bound on the size of the sample we need. Section 3.3.4 discusses the final pruning step. Section 3.3.5 shows how to reduce the non-homogenous case to the homogenous one (this does not work in general, only for the restricted distributions we can handle). In a brief concluding section we mention possible extensions of this work and some related questions.

3.1.1 Preliminaries

We adopt the terminology used in the literature. To recap quickly, an *example* is a point in $\boldsymbol{R}^n$; a *concept* is a subset of $\boldsymbol{R}^n$. An example that belongs to a concept is a *positive* example for the concept, and an example that lies outside the concept is a *negative* example. Given a set of labelled examples drawn from an unknown distribution $\mathcal{D}$ in $\boldsymbol{R}^n$, and labelled according to an unknown *target* concept the learning task is to learn the target concept. What this means is that given an error parameter ϵ and a confidence parameter δ, with probability at least $1-\delta$ the algorithm has to find a concept that has error at most ϵ on $\mathcal{D}$.

For us the target concept is the intersection of l half-spaces in $\boldsymbol{R}^n$, such that the normal vectors to the hyperplanes bounding these half-spaces span a subspace of dimension k. For most of the chapter we will assume that the hyperplanes bounding the half-spaces pass through the origin. Each point $x \in R^n$ is labelled *positive* or *negative* according to the following rule:

$$\ell(x) = + \quad if \quad Wx \leq 0$$

$$\ell(x) = - \qquad otherwise.$$

Here W is a real matrix of rank k with l rows and n columns. Each row w_i represents a half-space $w_i \cdot x \leq 0$. Hence a point x is labelled positive if it lies in the intersection of the k half-spaces and it is labelled negative otherwise. Formally, the positive region P is:

$$\{x \in \boldsymbol{R}^n | x \in B_n, Wx \leq 0\}$$

Examples presented to the algorithm are drawn from some unknown distribution on the unit ball in n dimensions, B_n. We assume that the distribution is *non-concentrated*, meaning that its probability density is at least $1/poly(n)$ and at most $poly(n)$ everywhere in the unit ball[1]. We can also assume that P occupies at least an ϵ fraction of the unit ball (otherwise we could simply output the concept that labels every point negative). It is worth noting that such a distribution is a "good" one for

[1] It is worth noting that such a distribution is a "good" one for the perceptron algorithm discussed in the previous chapter. We could set the σ of the algorithm to be $1/poly(n)$ and in polynomial time the algorithm would find a half-space that classified all but a polynomial fraction correctly.

the perceptron algorithm discussed in the previous chapter. We could set the σ of the algorithm to be $1/poly(n)$ and in polynomial time the algorithm would find a

Let D_P denote the dual to the cone formed by the positive region.

$$D_P = \{v \in \boldsymbol{R}^n : v = \sum_i \alpha_i w_i, \quad \alpha_i \geq 0\}$$

We could project down to the subspace spanned by D_P and the projections $w_1', w_2', \ldots, w_l'$ of the w_i's give half-spaces that separate the positives from the negatives in the reduced space.

We say that a convex cone K in $\boldsymbol{R}^k$ is *ϵ-enclosed* by another convex cone K' if $K \subseteq K'$, and for every point $x \in K' \cap B_n$ there is some point $y \in K \cap B_n$ such that the angle between the vectors x and y is at most ϵ. In other words for each $x \in K'$ there is a y in K such that the angle between x and y at the origin is at most ϵ. Inversely we say that K' *ϵ-encloses* K.

The *projection length* of a convex body K along a unit vector (or direction) v is the length of the 1-dimensional projection of K onto v. Intuitively, it is the width of the body in the direction v. Formally it is

$$\max_{x \in K} x \cdot v - \min_{x \in K} x \cdot v$$

3.2 The Algorithm

The input to the algorithm is an error parameter ϵ, a confidence parameter δ, and a set of labelled examples. The output of the algorithm is a set of m half-spaces.

Here we give a high-level description of the algorithm. Details and proofs of individual steps are in later sections. The parameters ϵ_i will be specified later.

1. **Approximate the dual cone.** Let S be the set of positive examples presented to the algorithm. Let C be the cone formed by the vectors in S and D_C be the dual of C, i.e., the set of normal vectors to hyperplanes that do not intersect C. The size of S is chosen so that D_C ϵ_1-encloses D_P.

2. **Identify the "irrelevant" subspace.** We do this by finding a set of $n-k$ orthogonal vectors $\{x_1, x_2, \ldots, x_{n-k}\}$ as follows: Choose a set of random unit vectors and let x_1 be the one among them such that the projection length of $D_C \cap B_n$ in the direction of x_1 is minimum. Now pick unit unit vectors orthogonal to x_1 and let x_2 be the one among them with the minimum projection length

of $D_C \cap B_n$. In this way at step i we find a vector x_i that is orthogonal to $x_1, \ldots, x_{i-1}$ and the projection length of D_C along x_i is small. After $n-k$ steps, the vectors $\{x_1, \ldots, x_{n-k}\}$ span a subspace that approximates the irrelevant subspace. The size of the sample is chosen so that at each vector x_i is a small angle (ϵ_2) away from being orthogonal to D_P.

3. **Reduce dimensionality.** Project D_C (implicitly) to the subspace orthogonal to that spanned by $\{x_1, x_2, \ldots, x_{n-k}\}$. Let $\hat{D_C}$ be the projection.

4. **Sample the dual.** Consider all lattice points spaced at ϵ_3 units in a box that encloses $\hat{D_C}$. In other words, in the original space for each w_i there is a vector corresponding to some lattice point that makes an angle less than ϵ_3 with w_i.

5. **Prune the sample.** Let U be the random sample from the previous step. Let S_1 be a new set of $\Omega(nl)$ examples. Greedily choose a subset of U of size m as follows: let u_1 be the vector from U such that $u_1 \cdot x \leq 0$ separates the largest number of negative examples from the positive examples in S_1. Discard the negative examples that are separated in this way by u_1. Let u_2 be the vector from U that separates the largest number of remaining negative examples in S_1 from the positives. Similarly, let u_i be the vector from U that separates the largest number of remaining negatives. The number of steps m is chosen so that the number of remaining negatives after m steps is less than an $\epsilon/2$ fraction of the initial number. Output the half-spaces $u_1 \cdot x \leq 0, u_2 \cdot x \leq 0, \ldots, u_m \cdot x \leq 0$. Given an unlabelled point x, we project it to $\hat{D_C}$ and then label it positive if it lies in the intersection of the above half-spaces and negative otherwise.

In the last step, alternatively, we could extend the vectors $u_1, \ldots, u_m$ to vectors in $\boldsymbol{R}^n$ and output the corresponding half-spaces in $\boldsymbol{R}^n$.

In order to learn a constant number of half-spaces in polynomial time, step (4) of the algorithm could be replaced by any standard method to learn half-spaces in constant-dimensional space. However, by this approach we can allow the relevant subspace to have dimension greater than a constant.

3.3 The Analysis

Our main theorem is a performance guarantee for this algorithm for a suitable choice of the ϵ_i's. In the statement below $p \geq 1$ is a parameter of the distribution. If $\mathcal{D}$

has parameter p then the probability density everywhere in the unit ball is between $\frac{1}{p}$ and p.

Theorem 6 *The above algorithm PAC-learns with parameter ϵ and δ the intersection of l half-spaces whose normals lie in a k-dimensional subspace, and has a bound of*

$$O(poly(n) l k^k (\frac{p}{\epsilon})^k \log \frac{1}{\delta})$$

on the running time and the number of examples required.

For nearly-uniform distributions, i.e., when p is a constant, this gives us a polynomial-time algorithm for k up to about $\log n / \log\log n$.

3.3.1 A large sample of examples

In this section we derive an upper bound on the number of examples required by the algorithm. Let S be the set of examples. Then $|S|$ should be large enough to guarantee that the dual to the cone formed by S ϵ_1-encloses the dual to the cone formed by P.

Lemma 4 *Let $\mathcal{D}$ be a non-concentrated distribution on B_n Let S be a random sample of positive examples from this distribution and C be the cone at the origin formed by the points in S. Let D be the dual cone of C. If*

$$|S| = \Omega(p(n).(\frac{1}{\epsilon_1})^k. \log \frac{1}{\delta})$$

where $p(n)$ is a fixed polynomial that depends only on $\mathcal{D}$, then with probability at least $1-\delta$, the cone D_P is ϵ_1-enclosed by D_C.

Proof. Let $D_{C'}$, the dual of a cone C', be the maximum body that ϵ_1-encloses D_P. Our goal is to show that the dual D_C of the conical hull C of a large enough sample S will be contained in D'_C with high probability. To prove this we show that for each point h on the boundary of $D_{C'}$, there is a supporting plane of D_C which separates h from D_P.

Let $h \cdot x' = 0$ be a supporting hyperplane of C' such that C' lies in the half-space $h \cdot x' \leq 0$ and consider the convex region $P \cap B_n \cap h \cdot x' \geq 0$. The key idea is to show that the probability that there is a point in the sample from such a region is high. This probability is the total probability mass assigned to this part of the unit ball, i.e., it is at least $\frac{1}{p(n)}$ times the fraction of the volume of the unit ball occupied by this region. To calculate the fraction of the unit ball occupied by this region, we can first go down to the $(k+1)$-dimensional space spanned by $w_1, w_2, \ldots, w_l$ and h. In this space the volume of the region grows with ϵ_1 roughly as at least $\frac{1}{p(n)}\epsilon_1^k Vol(B_k)$ where $Vol(B_k)$ is the volume of the unit ball (this essentially follows from the observation that the positive region P occupies at least $1/p(n)$ fraction of B_k). So the probability that any single example falls in the region is $\Omega(\epsilon_1^k/p(n))$. Now we use the VC-dimension of the intersection of up to $l+1$ half-spaces to complete the proof. The VC theorem implies that if we consider a sample of size $\Omega(\frac{(l+1)n}{\epsilon})$ then with high probability every concept in the class, i.e., an intersection of every set of $l+1$ half-spaces with more than ϵ probability will see at least one example in the sample. We set ϵ in the theorem to be $\epsilon_1^k/p(n)$ to complete the proof. □

3.3.2 Identifying the relevant subspace

In this section we analyze the procedure to approximately identify the irrelevant subspace and hence the subspace spanned by D_P .

The first step is to find a vector x_1 such that the projection length of D_C onto x_1 is small. For this we do a the following. Pick a set of random unit vectors, and find the projection length of $D_C \cap B_n$ along these vectors. Note that computing the projection length of $D_C \cap B_n$ onto a vector x can be done efficiently: we need to calculate $\max_{y \in D_C \cap B_n} x \cdot y$ and $\min_{y \in D_C \cap B_n} x \cdot y$, both of which are convex programs with simple separation oracles [34]

Let x_1 be the direction along which the projection length is minimum. We estimate the probability that x_1 is nearly orthogonal to D_P using the following fact.

Fact 1 *The volume of the n-dimensional ball of radius r is equal to* $\frac{2r^n\pi^{n/2}}{n\Gamma(n/2)}$ *and its surface area is* $\frac{2r^{n-1}\pi^{n/2}}{\Gamma(n/2)}$.

Lemma 5 *Let* $v_1, v_2, \ldots v_k$ *be orthogonal unit vectors in* $\boldsymbol{R}^n$ *and let* $\alpha \leq \frac{1}{\sqrt{k}}$. *Then the probability that a random unit vector x satisfies* $|v_i \cdot x| \leq \alpha$, *for all i, is*

$$\Omega((1-k\alpha^2)^{(n-k)/2}(\alpha^2(n-k))^{k/2}).$$

In other words the lemma lower bounds the probability that a random unit vector is nearly orthogonal to each of a fixed set of k vectors.

Proof. Consider the set of of unit vectors x such that $|v_i \cdot x| \leq \alpha$ for all $i = 1, \ldots, k$. This is the intersection of the unit ball B_n with $2k$ half-spaces. The intersection contains a "band" which has as its base an $n-k$ dimensional ball of radius $\sqrt{(1-k\alpha^2)}$ and a thickness of 2α in the other k orthogonal dimensions. A simple calculation based on fact 1 gives the bound. $\square$

Now we sample only from vectors orthogonal to x_1, record the vector x_2 with the minimum projection length and repeat this to find $x_2, \ldots, x_{n-k}$. Let $\hat{D_C}$ denote the projection of D_C to the subspace orthogonal to $x_1, x_2, \ldots, x_{n-k}$.

From the lemma it follows, for example, that if we pick n/β^k random unit vectors then with high probability one of them will have a dot product of magnitude less than $\frac{\beta}{\sqrt{n-k}}$ with each of a fixed set of k orthogonal unit vectors. By letting the k vectors to be a set of basis vectors of D_P we have that w.h.p. x_1 will be nearly orthogonal to D_P.

Lemma 6 *Assume the projection length of $D_C \cap B_n$ along any direction orthogonal to D_P is at most $\alpha/2$ and along any direction in the subspace spanned by D_P is at least α. Then a sample of n/α^k random unit vectors to find each x_i guarantees that the vectors $x_1, .., x_{n-k}$ are almost orthogonal to D_P, i.e., $|v_i \cdot x_j| \leq \frac{\alpha}{\sqrt{n+1-j-k}}$ for $i = 1, \ldots, k$ and $j = 1, \ldots, n-k$. Further any unit vector $w \in D_P$ has a projection $\hat{w} \in \hat{D_C}$ such that $w \cdot \hat{w} \geq 1 - \alpha^2 k \log n$.*

Proof. From lemma 5 the vector x_1 satisfies $v_i \cdot x_1 \leq \frac{\alpha}{\sqrt{n-k}}$. For the purpose of analysis we can view the second step of the algorithm as first projecting D_C to the subspace orthogonal to x_1 and then sampling from all unit vectors in that subspace. The projection of v_i is $v_i - (v_i \cdot x_1)x_1$. So again from the lemma we have

$$|v_i \cdot x_2| = |[v_i - (v_i \cdot x_1)x_1] \cdot x_2| \leq \frac{\alpha}{\sqrt{n-1-k}}$$

At the t^{th} step, the projection of v_i is $v_i - \sum_{j=1}^{t-1}(v_i \cdot x_j)x_j$ and it follows that $|v_i \cdot x_j| \leq \frac{\alpha}{\sqrt{n+1-t-k}}$.

To prove the second part, for a unit vector $w \in D_P$, $\hat{w} = w - \sum_{j=1}^{n-k}(w \cdot x_j)x_j$. So $w \cdot \hat{w} = 1 - \sum_{j=1}^{n-k}(w \cdot x_j)^2$. We rewrite w as $\sum_{i=1}^{k} \lambda_i v_i$ where $\sum_i \lambda_i^2 = 1$ and then

$$|w \cdot x_j| = |\sum_{i=1}^{k} \lambda_i (v_i \cdot x_j)|$$

$$\leq \frac{\alpha}{\sqrt{n+1-j-k}} \sum_i \lambda_i \leq \alpha \sqrt{\frac{k}{n+1-j-k}}.$$

We plug this into the expression for $w \cdot \hat{w}$ to get that $w \cdot \hat{w} \geq 1 - \alpha^2 k \log n$. $\square$

The first assumption of the lemma can be satisfied by setting $\epsilon_1 = \epsilon_2/2$ in the previous step of the algorithm. The second assumption is not really restrictive. If it is not true, then that means that we can reduce the problem to one in $k-1$-dimensional space. By setting $\alpha = \frac{\epsilon_2}{\sqrt{k \log n}}$ we get that each $w \in D_P$ has a projection $\hat{w} \in \hat{D_C}$ at an angle of no more than ϵ_2 for small values of ϵ_2.

3.3.3 Sampling the dual

The next step is to find a sample the projected dual so that there is a point in the sample within ϵ_3 of each w_i'. We do this by simply finding a sample that has point close to every point of $\hat{D_C} \cap B_k$. Let $\mathbf{Z}^k$ be the integer lattice in k dimensions, and let $\epsilon_3\mathbf{Z}^k$ be a scaled down integer lattice where the least spacing between points is ϵ_3. Then the sample we consider is $\{\hat{D_C} \cap B_k \cap \epsilon_3\mathbf{Z}^k$. It is easy to see that the size of the sample is bounded by $(\frac{1}{\epsilon_3})^k$.

3.3.4 Pruning the sample of normal vectors

Here we show that $m = O(l)$. Let S_1 be a fresh sample of examples. From standard VC-dimension arguments it is enough to find a set of half-spaces that have an error less than ϵ on a sample whose size is the VC-dimension. So we choose $|S_1|$ to be $\Omega(nl)$.

Let $v_1, v_2, \ldots, v_l$ be the vectors in U that are closest in angle to $w'_1, w'_2, \ldots, w'_l$ respectively. Let P_v be the region $v_1 \cdot x \leq 0, \ldots, v_l \cdot x \leq 0$. P_v is the positive region according to these vectors. Assume that the combined error of these half-spaces with respect to the actual set of half-spaces is bounded by $\epsilon/2$. In other words there is a set of l half-spaces in U that correctly classify a $(1-\frac{\epsilon}{2})$ fraction of the distribution. This can be achieved by setting $\epsilon_2 + \epsilon_3 = \frac{\epsilon}{2k}$ in the previous steps of the algorithm.

Our procedure to prune U is the following: pick the best u from U, i.e., the vector u such that the half-space $u \cdot x \leq 0$ separates the maximum number of negatives from the positives in S_1. Call it u_1. Then pick the best u for the remaining negative examples and so on. From fairly standard set cover guarantees it follows that a greedy set of l half-spaces must separate at least half as many as the best set of l half-spaces, and a set of $l \log r$ greedy half-spaces separate at least $1-\frac{1}{r}$ fraction of what the best set of l half-spaces can separate. We formalize this in the following theorem.

Theorem 7 *A greedily chosen set of $2l \log r$ half-spaces will with high probability correctly classify at least $(1-\frac{1}{r})(1-\frac{\epsilon}{2})$ fraction of the distribution and hence achieve*

PAC-learning.

Setting r to be less than $\frac{\epsilon}{3}$ (say) gives us a set of $O(l)$ planes that correctly classify $1-\epsilon$ of the distribution.

3.3.5 The non-homogenous case

The discussion so far has assumed that the half-spaces we are trying to learn are homogenous. Of course this may not be the case in general, and here we show a simple reduction from the general (non-homogenous) case to the homogenous case by going to a representation in one more dimension, i.e., in $\boldsymbol{R}^{n+1}$. The key issue will be to make sure that we can do this while keeping the distribution in the new space non-concentrated. We make two observations for this.

Our first observation is that all the algorithm needs is a distribution on the unit *sphere* (rather than the ball) in $\boldsymbol{R}^n$ such that:

Any convex region of the sphere with more than $1/p(n)$ fraction of the area of the sphere should have probability mass between $1/q(n)$ and $q(n)$ for some polynomials $p(n), q(n)$. (Analogously, any c_1 fraction should have probability mass between c_2 and $1/c_2$ for constants c_1, c_2.)

Our second observation is that we only need the above condition to hold for the part of the unit sphere that is in the positive region. (Negative examples are used only in the last step.)

The following mapping from the unit ball B_n in $\boldsymbol{R}^n$ to the unit sphere S_n in $\boldsymbol{R}^{n+1}$ satisfies these conditions.

$$\begin{aligned} y_i &= \frac{n x_i}{\sqrt{n^2|x|^2+1)^2}} \quad for \quad i=1,\ldots n \\ y_{n+1} &= \frac{1}{\sqrt{n^2|x|^2+1}} \end{aligned}$$

This corresponds to blowing up B_n by a factor of n then placing it on the plane $y_{n+1}=1$ and then mapping it stereographically to the unit sphere (scaling the length of the image).

The new coordinate y_{n+1} lies between 1 and $\frac{1}{\sqrt{n^2+1}}$, i.e., points are mapped only to the part of the sphere in the half-space $y_{n+1} \geq \frac{1}{\sqrt{n^2+1}}$. Suppose the positive region in $\boldsymbol{R}^n$ is given by the set of (possibly non-homogenous) half-spaces $w_i \cdot x \leq b_i$ for $i=1,\ldots,l$.

Then in $\boldsymbol{R}^{n+1}$ the positive region is given by the half-spaces

$$(w_i, -b_i n) \cdot y \leq 0$$

$$y_{n+1} \geq \frac{1}{\sqrt{n^2+1}}$$

This is then approximated by the homogenous half-spaces $(w_i, -b_i n) \cdot y \leq 0$ and $y_{n+1} \geq 0$ and we can run the algorithm to learn them. The "blow-up" factor n could be replaced by any $poly(n)$. Note that it does not matter that only (about) half the sphere is used.

3.4 Conclusion and open problems

We have seen the simplicity and utility of random projection as applied to a classical problem in learning theory. In the next chapter we study random projection in a rather different context, namely information retrieval. Random projection also seems to be a natural scheme for rounding semidefinite relaxations to vertex-ordering problems. Recently, Kleinberg [45] gave an algorithm for finding approximate nearest neighbors using similar techniques. It is my feeling that other applications are waiting to be discovered. We conclude this chapter with some questions about the algorithm presented here.

Can the running time of the algorithm for learning the intersection of half-spaces be improved to polynomial in $\frac{1}{\epsilon}$? Indeed an algorithm that is fully polynomial, i.e., polynomial in k as well, might be possible for non-concentrated distributions. I do not know any hardness reduction that would make this unlikely.

Can we learn the intersection of k hyperplanes when the examples are drawn from the uniform distribution on the vertices of a hypercube? This does not induce a non-concentrated distribution on the sphere and so the methods in this chapter cannot be applied directly. One reason that the problem is interesting for this distribution is that it includes as a special case the problem of learning DNF-formulae.

Chapter 4

Reducing Dimensionality by Random Projection II

We use random projection to quickly approximate the eigenspace of a matrix and apply it to speeding up the information retrieval technique known as Latent Semantic Indexing.

4.1 Introduction: Information Retrieval

The complexity of information retrieval is best illustrated by the two nasty classical problems of *synonymy* (missing documents with references to "automobile" when querying on "car") and *polysemy* (retrieving documents about the Internet when querying on "surfing"). One possible approach to dealing with these two problems would be to represent documents (and queries) not by terms (as in conventional vector-based methods), but by the *underlying (latent, hidden) "concepts"* referred to by the terms. This hidden structure is not a fixed many-to-many mapping between terms and concepts, but depends critically on the *corpus* (document collection) in hand, and the term correlations it embodies.

Latent Semantic Indexing (LSI) [19] is an information retrieval method which attempts to capture this hidden structure by using techniques from linear algebra. Vectors representing the documents are projected in a new, low-dimensional space obtained by *singular value decomposition* of the term-document matrix A. This low-dimensional space is spanned by the eigenvectors of A^TA that correspond to the few largest eigenvalues — and thus, presumably, to the few most striking correlations between terms (see Section 4.1.1 for a brief description of the technique). Queries are also projected and processed in this low-dimensional space. This results not only in great savings in storage and query time (at the expense of some considerable preprocessing), but also, according to empirical evidence reported in the literature, to *improved information retrieval* [10, 22, 23]. Indeed, it has been repeatedly reported that LSI outperforms, with regard to precision and recall in standard collections and query workloads, more conventional vector-based methods.

There is very little in the literature in the way of a mathematical theory that predicts this improved performance. An interesting mathematical fact due to Eckart and Young (stated below as Theorem 8) which is often cited as an explanation of the improved performance of LSI states, informally, that LSI retains as much as possible the relative position of the document vectors. This, however, may only provide an explanation of why LSI *does not deteriorate too much* in performance over conventional vector-space methods; it fails to justify the observed improvement.

This is *a* first attempt at using mathematical techniques to rigorously explain the improved performance of LSI (Section 4.2 starts with a brief comparison with previous uses of probabilistic techniques in information retrieval). Since LSI seems to exploit and reveal the statistical properties of a corpus, we must start with a rigorous probabilistic model of the corpus (that is to say, a mathematical model of how corpora are generated); we do this in Section 4.2. Briefly, we model *topics* as probability distributions on terms. A *document* is then a probability distribution that is the convex combination of a small number of topics. We also include in our framework *style* of authorship, which we model by a stochastic matrix that modifies the term distribution. A *corpus* is then a collection of documents obtained by repeatedly sampling a probability distribution on combinations of topics and styles.

Once we have a corpus model, we would like to determine under what conditions LSI results in enhanced retrieval properties. We would like to prove a theorem stating essentially that *if the corpus is a reasonably focused collection of meaningfully correlated documents, then LSI does well.* The problem is to define these terms so that (1) there is a reasonably close correspondence with what they mean intuitively and in practice, and (2) the theorem can be proved. In Section 4.3 we prove results that, although not totally comprehensive and general, definitely point to this direction. In particular, we show that in the special case in which (a) there is no style modifier; (b) each document is on a single topic; and (c) the terms are partitioned among the topics so that each topic distribution has high probability on its own terms, and low probability on all others; *then* LSI, projecting to a subspace of dimension equal to the number of topics, will discover these topics exactly, with high probability

(Theorem 9).

In Section 4.4 we point out an interesting fact: if we project the term-document matrix on a *completely random* low-dimensional subspace, then with high probability we have a distance-preservation property akin to that enjoyed by LSI. This random projection idea may yield an interesting improvement on LSI: we can perform the LSI precomputation not on the original term-document matrix, but on a low-dimensional projection, at great computational savings and no great loss of accuracy (Theorem 11).

This last result can be seen as an alternative to (and a justification of) *sampling* in LSI. Reports on LSI experiments in the literature seem to suggest that LSI is often done not on the entire corpus, but on a randomly selected subcorpus (both terms and documents may be sampled, although it appears that most often documents are). There is very little non-empirical evidence of the accuracy of such an approach. Our result suggests a different and more elaborate (and computationally intensive) approach — projection on a random low-dimensional subspace — which can be *rigorously proved* to be accurate. We supplement several of our theorems with experiments on corpora derived from our statistical model.

4.1.1 A review of LSI in information retrieval

A *corpus* is a collection of documents. Each document is a collection of *terms* from a universe of n terms. Each document can thus be represented as a vector in $\Re^n$ where each axis represents a term. The i^{th} coordinate represents some function of the number of times the i^{th} term occurs in the document. This is the standard vector-space representation of documents. There are several candidates for the right function to be used here; we assume that it is the relative frequency of the term (number of times the term occurs/total number of terms in the document).

Let A be an $n \times m$ matrix whose rows represent terms and columns represent documents. Let the rank of A be r. Let the singular values of A be $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$ (not necessarily distinct), i.e., $\sigma_1^2, \sigma_2^2, \ldots \sigma_r^2$ are the eigenvalues of AA^T. The *singular value decomposition* of A expresses A as the product of three matrices $A = UDV^T$, where $D = \text{diag}(\sigma_1, \ldots, \sigma_r)$ is an $r \times r$ matrix, $U = (u_1, \ldots, u_r)$ is an $n \times r$ matrix whose columns are orthonormal, and $V = (v_1, \ldots, v_r)$ is an $m \times r$ matrix which is also column-orthonormal.

LSI works by omitting all but the k largest singular values in the above decomposition, for some appropriate k; here k is the dimension of the low-dimensional space alluded to in the informal description of Section 4.1. It should be small enough to enable fast retrieval, and large enough to adequately capture the structure of the corpus. Let $D_k = diag(\sigma_1, \ldots, \sigma_k)$, $U_k = (u_1, \ldots, u_k)$ and $V_k = (v_1, \ldots, v_k)$. Then

$$A_k = U_k D_k V_k^T$$

is a matrix of rank k, which is our approximation of A. The rows of $V_k D_k$ above are then used to represent the documents. In other words, the column vectors of A (documents) are projected to the k-dimensional space spanned by the column vectors of U_k; we sometimes call this space the LSI space of A.

How good is this approximation? The following well-known theorem gives us some idea.

Theorem 8 *(Eckart and Young, see [32].) Among all $n \times m$ matrices C of rank at most k, A_k is the one that minimizes $\|A - C\|^2 = \sum_{i,j}(A_{i,j} - C_{i,j})^2$.*

Therefore, LSI preserves (to the extent possible) the relative distances (and hence, presumably, the retrieval capabilities) in the term-document matrix while projecting it to a lower-dimensional space. It remains to be seen in what way it *improves* these retrieval capabilities.

4.2 The Probabilistic Corpus Model

There are many useful formal models of IR in the literature, and probability plays a major role in many of them — see for instance the surveys and comparisons in [30, 53, 56]. The approach in this body of work is to formulate information retrieval as a problem of learning the concept of "relevance" that relates documents and queries. The corpus and its correlations plays no central role. In contrast, our focus is on the probabilistic properties of the corpus.

Since LSI exploits and brings out the structure of the corpus it will fare well in a meaningful collection of strongly correlated documents, and will produce noise in a random set of unrelated documents. In order to study the dependence of the performance of LSI on the statistical properties of the corpus, we must start with a probabilistic model of a corpus. We state now our basic probabilistic model, which we will work with for much of this chapter. In Section 4.5.1 we will extend this to a more general graph-theoretic model in which the conductances of subsets of vertices will play a role.

Let the universe of all terms be U. A *topic* is a probability distribution on U. A meaningful topic is very different from the uniform distribution on U, and is concentrated on terms that might be used to talk about a particular subject. For example, the topic of "space travel" might favor the terms "galaxy" and "starship", while rarely mentioning "misery" or "spider". A possible criticism against this model is that it does not take into account correlations of terms within the same topic (for example, a document on the topic "Internet " is much more likely to contain the term "search" if it also contains the term "engine").

The structure of documents is also heavily affected by *authorship style.* We model style as a $|U| \times |U|$ stochastic matrix (a matrix with nonnegative entries and row sums equal to 1), denoting the way whereby style modifies the frequency of terms. For example, a "formal" style may map "car" often to "automobile" and "vehicle," and seldom to "car" — and almost never to "wheels." Admittedly, this is not a comprehensive treatment of style; for example, it makes the assumption – not always valid – that this influence is independent of the underlying topic.

A *corpus model* $\mathcal{C}$ is a quadruple $\mathcal{C} = (U, \mathcal{T}, \mathcal{S}, D)$, where U is the universe of terms, $\mathcal{T}$ is a set of topics, and $\mathcal{S}$ a set of styles. $\hat{\mathcal{T}} \times \hat{\mathcal{S}} \times \mathbf{Z}^+$, where by $\hat{\mathcal{T}}$ we denote the set of all convex combinations of topics in $\mathcal{T}$, by $\hat{\mathcal{S}}$ the set of all convex combinations of styles in $\mathcal{S}$, and by $\mathbf{Z}^+$ the set of positive integers (the integers represent the lengths of documents). That is, a corpus model is a probability distribution on topic combinations (intuitively, favoring combinations of a few related topics), style combinations, and document lengths (total number of term occurrences in a document).

A document is generated from a corpus model $\mathcal{C} = (U, \mathcal{T}, \mathcal{S}, D)$ through the following two-step sampling process. In the first step, a convex combination of topics $\hat{T}$ from $\hat{\mathcal{T}}$, a convex combination of styles $\hat{S}$ from $\hat{\mathcal{S}}$, and a positive integer ℓ are sampled according to distribution D. Then terms are sampled ℓ times to form a document, each time according to distribution $\hat{T}\hat{S}$. A *corpus of size* m is a collection of m documents generated from $\mathcal{C}$ by repeating this two-step sampling process m times.

4.3 An attempt to explain LSI's success

We now establish a result based on our model that provides some intuition for LSI's empirical success. We begin with some tools from spectral analysis in Section 4.3.1; following our analysis in Section 4.3.2, we give some experimental results using our synthetic model in support of this analysis. More extensive experimentation using large corpora of "real" documents further supports this analysis (see Section 4.5).

4.3.1 Tools

The following lemma formalizes the intuition that if the k largest singular values of a matrix A are well-separated from the remaining singular values then the subspace spanned by the corresponding singular vectors is preserved well when a small perturbation is added to A.

Lemma 7 *Let A be an $n \times m$ matrix of rank r with singular value decomposition*

$$A = UDV^T,$$

where $D = \mathsf{diag}(\sigma_1, \ldots, \sigma_r)$. Suppose that, for some k, $1 \leq k < r$, $\sigma_k/\sigma_{k+1} > c\sigma_1/\sigma_k$ for sufficiently large constant c. Let F be an arbitrary $n \times m$ matrix with $\|F\|_2 \leq \epsilon$, where ϵ is a sufficiently small positive constant. Let $A' = A + F$ and let $U'D'V'^T$ be its singular-value decomposition. Let U_k and U'_k be $n \times k$ matrices consisting of the first k columns of U and U' respectively. Then, $U'_k = U_k R + G$ for some $k \times k$ orthonormal matrix R and some $n \times k$ matrix G with $\|G\|_2 \leq O(\epsilon)$.

The proof of this lemma, given in the appendix, relies on a theorem of Stewart [33] about perturbing a symmetric matrix.

4.3.2 Analysis of LSI

We now show that in a restricted version of our probabilistic model, LSI brings together documents on the same topic while keeping apart documents on different topics. Let $\mathcal{C} = (U, \mathcal{T}, D)$ be a corpus model. We call $\mathcal{C}$ *pure* if each document talks only about a single topic. We call $\mathcal{C}$ *ϵ-separable*, where $0 \leq \epsilon < 1$, if a set of terms U_T is associated with each topic $T \in \mathcal{T}$ so that (1) U_T are mutually disjoint and (2) for each T, the total probability T assigns to the terms in U_T is at least $1 - \epsilon$. We call U_T the *primary set of terms* of topic T. The assumption that a corpus is ϵ-separable for some small value of ϵ is more realistic if the documents are assumed to be preprocessed to eliminate commonly-occurring stop-words.

Let $\mathcal{C}$ be a pure corpus model and let $k = |\mathcal{T}|$ denote the number of topics in $\mathcal{C}$. Since $\mathcal{C}$ is pure, each document generated from $\mathcal{C}$ is in fact generated from some single topic T: we say that the document *belongs to* the topic T. Let C be a corpus generated from $\mathcal{C}$ and, for each document $d \in C$, let v_d denote the vector assigned to

d by the rank-k LSI performed on C. We say that the rank-k LSI is *δ-skewed* on the corpus instance C if, for each pair of documents d and d', $v_d \cdot v_{d'} \leq \delta \|v_d\| \|v_{d'}\|$ if d and d' belong to different topics and $v_d \cdot v_{d'} \geq 1 - \delta \|v_d\| \|v_{d'}\|$ if they belong to the same topic. Informally, the rank-k LSI is δ-skewed on a corpus (for small δ), if it assigns nearly orthogonal vectors to two documents from different topics and nearly parallel vectors to two documents from a single topic: LSI does a particularly good job of classifying documents when applied to such a corpus. The following theorem states that a large enough corpus (specifically, when the number of documents is greater than the number of terms) generated from our restricted corpus model indeed has this nice property with high probability.

Theorem 9 *Let $\mathcal{C}$ be a pure and ϵ-separable corpus model with k topics such that the probability each topic assigns to each term is at most τ, where $\tau > 0$ is a sufficiently small constant. Let C be a corpus of m documents generated from $\mathcal{C}$. Then, the rank-k LSI is $O(\epsilon)$-skewed on C with probability $1 - O(m^{-1})$.*

Proof. Let C_i denote the subset of the generated corpus C consisting of documents belonging to topic T_i, $1 \leq i \leq k$. To see the main idea, let us first assume that $\epsilon = 0$. Then, each document of C_i consists only of terms in U_i, the primary set of terms associated with topic T_i. Thus, the term-document matrix A representing corpus C consists of blocks B_i, $1 \leq i \leq k$: the rows of B_i correspond to terms in U_i and columns of B_i correspond to documents in C_i; the entire matrix A can have non-zero entries in these rows and columns only within B_i. Therefore, $A^T A$ is block-diagonal with blocks $B_i^T B_i$, $1 \leq i \leq k$. Now focus on a particular block $B_i^T B_i$ and let λ_i and λ_i' denote the largest and the second largest eigenvalues of $B_i^T B_i$. Intuitively, the matrix $B_i^T B_i$ is essentially the adjacency matrix of a random bipartite multigraph and then, from the standard theory of spectra of graphs[18], we have that $\lambda_i'/\lambda_i \to 0$ with probability 1 as $\tau \to 0$ and $|C_i| \to \infty$. Below we give a formal justification of this by showing that a quantity that captures this property, the *conductance* [36] (equivalently, *expansion*) of $B_i^T B_i$ is high. The conductance of an undirected edge-weighted graph $G = (V, E)$ is

$$\min_{S \subset V} \frac{\sum_{i \in S, j \in \overline{S}} wt(i,j)}{\min\{|S|, |\overline{S}|\}}$$

Let $x^1, x^2, \ldots, x^t$ be random documents picked from the topic T_i. Then we will show that the conductance is $\Omega(\frac{|t|}{|T_i|})$, where $|T_i|$ is the number of terms in the topic T_i. Let G be the graph induced by the adjacency matrix $B_i^T B_i$. For any subset S of the vertices (documents),

$$\sum_{i\in S, j\in \overline{S}} wt(i,j) = \sum_{i\in S, j\in \overline{S}} x^i \cdot x^j$$
$$= (\sum_{i\in S} x^i)\cdot(\sum_{j\in\overline{S}} x^j).$$

Assume w.l.o.g. that $|S| \leq |\overline{S}|$. Let p_s be the probability of the s^{th} term in T_i. Then we can estimate, for each term, $\sum_{j\in\overline{S}} x^j_s \geq \min\{p_s/2, p_s - \epsilon\}$ with probability at least $1-\frac{1}{2^t}$ using the independence of the x^j's via a simple application of Chernoff-Hoeffding bound [35]. Using this we lower bound the weight of the cut $(S, \overline{S})$:

$$(\sum_{i\in S} x^i)\cdot(\sum_{j\in\overline{S}} x^j) \geq \sum_{i\in S} x^i_s(\min\{p_s/2, p_s-\epsilon\})$$

which is $\Omega(\frac{|S|}{|T_i|})$ with high probability by a second application of the Chernoff-Hoeffding bound. The desired bound on the conductance follows from this.

Thus, if the sample size $m = |C|$ is sufficiently large, and the maximum term probability τ is sufficiently small (note this implies that the size of the primary set of terms for each topic is sufficiently large), the k largest eigenvalues of A^TA are λ_i, $1 \leq i \leq k$, with high probability. Suppose now that our sample C indeed enjoys this property. Let $\hat{u}_i$ denote the eigenvector of $B_i^TB_i$ corresponding to eigenvalue λ_i (in the space where coordinates are indexed by the terms in T_i) and let u_i be its extension to the full term space, obtained by padding zero entries for terms not in T_i. Then, the k-dimensional LSI-space for corpus C is spanned by the mutually orthogonal vectors u_i, $1 \leq i \leq k$. When a vector v_d representing a document $d \in C_i$ is projected into this space, the projection is a scalar multiple of u_i, because v_d is orthogonal to u_j for every $j \neq i$.

Caveat: Although the above argument might seem to support the speculation that each basis vector of the LSI space found by LSI corresponds to one topic, this is not necessarily the case, at least in our model. If the eigenvalues λ_i, $1 \leq i \leq k$, in the above analysis are all distinct, then LSI indeed finds u_i, $1 \leq i \leq k$, as the basis vectors. However, if some of the eigenvalues are identical, then LSI may find a different set of basis vectors. Note that the argument in the previous paragraph is not relying on LSI finding exactly the eigenvectors u_i, $1 \leq i \leq k$; it relies only on LSI identifying the subspace spanned by those vectors. This observation may sound an inessential side note because in our probabilistic corpus generation model, the probability that two of the eigenvalues λ_i, λ_j are identical goes to zero. In the more general case $\epsilon > 0$ we consider below, however, we may not expect LSI to identify individual eigenvectors corresponding to single topics, even if the eigenvalues are all distinct.

When $\epsilon > 0$, the term-document matrix A can be written as $A = B + F$, where B consists of blocks B_i as above and F is a matrix with small $\|L\|_2$-norm (not exceeding ϵ by much, with high probability). As observed in the above analysis for the case $\epsilon = 0$, the invariant subspace W_k of $B^T B$ corresponding to its largest k eigenvalues is an ideal representation space for representing documents according to their topics. Our hope is that the small perturbation F does not prevent LSI from identifying W_k with small errors. This is where we apply Lemma 7. Let W'_k denote the k-dimensional space the rank-k LSI identifies. The ϵ-separability of the corpus model implies that the two-norm of the perturbation to the document-term matrix is $O(\epsilon)$ and, therefore by the lemma, the two-norm of the difference between the matrix representations of W_k and W'_k is $O(\epsilon)$. Since W'_k is a small perturbation of W_k, projecting a vector representing a document in C_i into W'_k yields a vector close, in its direction, to u_i (the dominating eigenvector of $B_i^T B_i$). Therefore, the LSI representations of two documents are almost in the same direction if they belong to the same topic and are nearly orthogonal if they belong to different topics. A quantitative analysis (Lemma 10) shows that the rank-k LSI is indeed $O(\epsilon)$-skewed on C with high probability. □

4.3.3 Experiments

Even though Theorem 9 gives an asymptotic result and only claims that the probability approaches 1 as the size parameters grow, the phenomenon it indicates can be observed in corpora of modest sizes, as is seen in the following experiment. We generated 1000 documents (each 50 to 100 terms long) from a corpus model with 2000 terms and 20 topics. Each topic is assigned a disjoint set of 100 terms as its primary set. The probability distribution for each topic is such that 0.95 of its probability density is equally distributed among terms from the primary set, and the remaining 0.05 is equally distributed among all the 2000 terms. Thus this corpus model is 0.05-separable. We measured the angle between all pairs of documents in the original space and in the rank 20 LSI space. The following is a typical result. Call a pair of documents *intra-topic* if the two documents are generated from the same topic and *inter-topic* otherwise.

	intra-topic				inter-topic			
	min	max	average	std	min	max	average	std
original space	0.801	1.39	1.09	0.079	1.49	1.57	1.57	0.00791
LSI space	0	0.312	0.0177	0.0374	0.101	1.57	1.55	0.153

Here, angles are measured in radians. It can be seen that the angles of intra-topic pairs are dramatically reduced in the LSI space. Although the minimum inter-topic angle is rather small, indicating that some inter-topic pairs can be close enough to be confused, the average and the standard deviation show that such pairs are extremely rare. Similar results are obtained from ten repeated trials. Results from experiments with different size-parameters are also similar in spirit.

In this and the other experiments reported here, we used SVDPACKC [11] for singular value decomposition.

4.4 Fast LSI via pandom projection

A lemma of Johnson and Lindenstrauss shows that if points in a vector space are projected to a random subspace of suitably high dimension, then the distances between the points are approximately preserved. Although such a random projection can be used to reduce the dimension of the document space, it does not bring together semantically related documents. LSI on the other hand seems to achieve the latter, but its computation time is a bottleneck. This naturally suggests the following approach:

1. Apply a random projection to the initial corpus to l dimensions, for some small $l > k$, to obtain, with high probability, a much smaller representation, which is still very close (in terms of distances and angles) to the original corpus.

2. Apply rank k LSI to the documents in the projected space to get the final result.

We prove that the above approach is good in the sense that the the final representation is very close to what we would get by directly applying LSI. Another way to view this result is that random projection gives us a fast way to *approximate* the eigenspace (eigenvalues, eigenvectors) of a matrix.

We first state the Johnson-Lindenstrauss lemma.

Lemma 8 *(Johnson and Lindenstrauss, see [28, 38].) Let $v \in R^n$ be a unit vector, let H be a random l-dimensional subspace through the origin, and let the random* *variable X denote the square of the length of the projection of v onto H. Suppose $0 < \epsilon < \frac{1}{2}$, and $24 \log n < l < \sqrt{n}$. Then, $\mathbf{E}[X] = l/n$, and*

$$Pr(|X - l/n| > \epsilon l/n) < 2\sqrt{l}e^{-(l-1)\epsilon^2/4}.$$

Using the above lemma, we can infer that with high probability, all pairwise Euclidean distances are approximately maintained under projection to a random subspace. By choosing l to be $\Omega(\frac{\log m}{\epsilon^2})$ in Lemma 8, we have with high probability that the projected vectors, after scaling by a factor $\sqrt{n/l}$, $\{v_i'\}$, satisfy

$$\|v_i - v_j\|_2(1-\epsilon) \leq \|v_i' - v_j'\|_2 \leq \|v_i - v_j\|_2(1+ \ eps).$$

Similarly inner products are also preserved approximately: $2v_i \cdot v_j = v_i^2 + v_j^2 - (v_i - v_j)^2$. So the projected vectors satisfy

$$2v_i' \cdot v_j' \leq (v_i^2 + v_j^2)(1+\epsilon) - (v_i - v_j)^2(1-\epsilon)$$

Therefore, $v_i' \cdot v_j' \leq v_i \cdot v_j(1-\epsilon) + \epsilon(v_i^2 + v_j^2)$. In particular, if the v_i's are all of length at most 1, then any inner product $v_i \cdot v_j$ changes by at most 2ϵ.

Consider again the term-document matrix A generated by our corpus model. Let R be a random column-orthonormal matrix with n rows and l columns, used to project A down to an l-dimensional space. Let $B = \sqrt{\frac{n}{l}}R^T A$ be the matrix after random projection and scaling, where,

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

and

$$B = \sum_{i=1}^{t} \lambda_i a_i b_i^T$$

are the SVD's of A and B respectively.

Theorem 10 *Let ϵ be an arbitrary positive constant. If $l \geq c\frac{\log n}{\epsilon^2}$ for a sufficiently large constant c then, for $p = 1, \ldots, t$*

$$\lambda_p^2 \geq \frac{1}{k}[(1-\epsilon)\sum_{i=1}^{k} \sigma_i^2 - \sum_{j=1}^{i-1} \lambda_j^2].$$

Proof. The p^{th} eigenvalue of B can be written as

$$\lambda_p^2 = max_{|v|=1} v^T[B - \sum_{j=1}^{p-1} a_j b_j^T]^T[B - \sum_{j=1}^{p-1} a_j b_j^T]v$$

Consider the above expression for $v_1, \ldots, v_k$, the first k eigenvectors of A. For the i^{th} eigenvector v_i it can be reduced to

$$v_i^T(B^TB - \sum_{j=1}^{p-1} \lambda_j^2 b_j b_j^T)v_i$$

$$v_i^T B^T B v_i - \sum_{j=1}^{p-1} \lambda_j^2 (b_j \cdot v_i)^2$$

$$\sigma_i^2 |u_i^T R|^2 - \sum_{j=1}^{p-1} \lambda_j^2 (b_j \cdot v_i)^2$$

$$\geq (1-\epsilon)\sigma_i^2 - \sum_{j=1}^{p-1} \lambda_j^2 (b_j \cdot v_i)^2$$

Summing this up for $i = 1, \ldots, k$,

$$\sum_{i=1}^{k} v_i^T B^T B v_i \geq (1-\epsilon)\sum_{i=1}^{k} \sigma_i^2 - \sum_{j=1}^{p-1} \lambda_j^2 \sum_{i=1}^{k} (b_j \cdot v_i)^2$$

Since the v_i's are orthogonal and the b_j's are unit vectors,

$$\geq (1-\epsilon)\sum_{i=1}^{k} \sigma_i^2 - \sum_{j=1}^{p-1} \lambda_j^2$$

Hence

$$\lambda_p^2 \geq max_{v_i} v_i^T B^T B v_i \geq \frac{1}{k}[(1-\epsilon)\sum_{i=1}^{k} \sigma_i^2 - \sum_{j=1}^{p-1} \lambda_j^2]$$

□

Theorem 11

$$\|B_{2k}\|_2{}^2 \geq (1-\epsilon)\|A_k\|_2{}^2$$

In other words the matrix obtained by RP+LSI recovers most of the matrix obtained by direct LSI.

If we further assume that only the top k eigenvalues of A^TA are dominant, than we can prove more.

How much faster is the two step method? Let A be an $n \times m$ matrix. Then the time to compute LSI is $O(mn^2)$ for a dense matrix. On the other hand if A is sparse, this goes down to $O(mnc)$ where c is the (average) number of non-zero entries in a column of A, i.e. the number of terms in a document. The time to compute the random projection to l dimensions is $O(mnl)$ for dense matrices and $O(mcl)$ for sparse matrices. After the projection, the time to compute LSI is $O(ml^2)$. So the total time is $O(ml(l+c))$. To obtain an ϵ approximation we need l to be $O(\frac{\log n}{\epsilon^2})$. Thus the running time of the two-step method is asymptotically superior: $O(m(\log^2 n + c\log n))$ compared to $O(mnc)$.

4.5 Conclusion and further work

Recently, in personal communication, S. Vathyanathan and D. Modha (at IBM Almaden) give preliminary reports of success on real-life corpora with methods involving RP and SVD.

A theoretician's first reaction to an unexpected (positive *or* negative) empirical phenomenon is to understand it in terms of mathematical models and rigorously proved theorems; this is precisely what we have tried to do, with substantial if partial success. What we have been able to prove should be seen as a mere *indication* of what might hold; we expect the true positive properties of LSI to go far beyond the theorems we are proving here.

There are several specific issues to be pursued here. Two of them are, a model where documents could belong to several topics, and one where term occurrences are not independent. Another issue is, does LSI address polysemy? We have seen some evidence that it handles synonymy.

Theory should ideally go beyond the *ex post facto* justification of methods and explanation of positive phenomena, it should point the way to new ways of exploiting them and improving them. Section 4.4, in which we propose a random projection technique as a way of speeding up LSI (and possibly as an alternative to it), is an attempt in this direction.

4.5.1 A more general model

In this model we will view the corpus as an edge-weighted graph. There is a vertex in the graph for each document. The weight of an edge between two documents u and v denotes the similarity between the documents, with higher weight indicating greater similarity, e.g. $u \cdot v$. Documents that constitute a *topic* form an induced subgraph with high *conductance* [36] in this graph. This addresses the intuitive idea that two documents could be related through other documents even if they do not have many terms in common directly. Also notice that, in general, a single document could belong to many different topics, which calls for a substantial extension of the theory and techniques developed here.

We now extend our analysis of LSI to this more general setting. Assume that each document belongs to a single topic. The graph can then be partitioned into the topics, so that the subgraph induced by each topic has high conductance. The adjacency matrix A^TA can thus be written as $A'^TA'+F$ with the following properties:

- A'^TA' is a block diagonal matrix
- Each block corresponds to a topic
- The first two eigenvalues of each block are well-separated.
- The matrix F is a perturbation of low norm.

Let there be k topics in the corpus. Applying lemma 7 we see that a rank k LSI will closely approximate the eigenspace spanned by the first k eigenvectors of A'^TA'. Assuming that the first eigenvalues of the blocks are all greater than any of the second eigenvalues, this implies that LSI will separate the documents according to their topics.

4.6 Appendix: Proof of Lemma 7

In the following version of Lemma 7, we take some specific values for some constants to facilitate the proof; note that the choice of those values are arbitrary to a large extent.

Lemma 9 *Let A be an $n \times m$ matrix of rank r with singular value decomposition*

$$A = UDV^T,$$

where $D = \mathsf{diag}(\sigma_1, \ldots, \sigma_r)$. Suppose that, for some k, $1 \leq k < r$, $21/20 \geq \sigma_1 \geq \ldots \geq \sigma_k \geq 19/20$ and $1/20 \geq \sigma_{k+1} \geq \ldots \geq \sigma_r$. Let F be an arbitrary $n \times m$ matrix with $\|F\|_2 \leq \epsilon \leq 1/20$. Let $A' = A + F$ and let $U'D'V'^T$ be its singular-value decomposition. Let U_k and U'_k be $n \times k$ matrices consisting of the first k columns of U and U' respectively. Then, $U'_k = U_kR + G$ for some $k \times k$ orthonormal matrix R and some $n \times k$ matrix G with $\|G\|_2 \leq 9\epsilon$.

The proof of this lemma relies on a theorem of Stewart [33] about perturbing a symmetric matrix.

Theorem 12 *Suppose B and $B+E$ are $n \times n$ symmetric matrices and*

$$Q = \begin{array}{cccc} [& Q_1 & Q_2 &] \\ & k & n-k & \end{array}$$

is an $n \times n$ orthogonal matrix such that range(Q_1) is an invariant subspace for B. Partition the matrices $Q^T BQ$ and $Q^T EQ$ as follows, where B_{11} and E_{11} are $k \times k$ matrices:

$$Q^T BQ = \begin{bmatrix} B_{11} & 0 \\ 0 & B_{22} \end{bmatrix}$$

$$Q^T EQ = \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix}$$

If

$$\delta = \lambda_{min} - \mu_{max} - \|E_{11}\|_2 - \|E_{22}\|_2 > 0,$$

where λ_{min} is the smallest eigenvalue of B_{11} and μ_{max} is the largest eigenvalue of B_{22}, and $\|E_{12}\|_2 \le \delta/2$ then there exists an $(n-k) \times k$ real matrix P such that

$$\|P\|_2 \le \frac{2}{\delta}\|E_{21}\|_2$$

and the columns of $Q_1' = (Q_1 + Q_2 P)(I + P^T P)^{-1/2}$ form an orthonormal basis for a subspace that is invariant for $B+E$.

Proof. of Lemma 9. We apply Theorem 12 to $B = AA^T$, $E = A'(A')^T - B$. We choose the block-diagonalizing matrix Q in the theorem to be U followed by $n-r$ zero-columns. Thus, when we write $Q = [Q_1 Q_2]$, $Q_1 = U_k$, the first k columns of U, and Q_2 consists of remaining columns of U followed by zero-columns. Since $U^T BU$ is a diagonal matrix, $Q^T BQ$ is also a diagonal matrix. Let $Q^T EQ$ be decomposed into blocks E_{ij}, $1 \le i,j \le 2$, as in Theorem 12. To apply the theorem, we need to bound $\|E_{i,j}\|_2$. We do this simply by bounding $\|E\|_2$. Since $E = (A+F)(A+F)^T - AA^T = AF^T + FA^T + FF^T$, we have $\|E\|_2 \le 2\|A\|_2\|F\|_2 + {\|F\|_2}^2 \le 2(21/20)\epsilon + \epsilon^2 < (43/20)\epsilon$. Therefore, $\|E_{ij}\|_2 \le (43/20)\epsilon$, $1 \le i,j \le 2$. The non-zero eigenvalues of B are $\sigma_1^2, \ldots, \sigma_r^2$. Of these, $\sigma_1^2, \ldots, \sigma_k^2 \ge 361/400$ and $\sigma_{k+1}^2, \ldots, \sigma_r^2 \le 1/400$. Hence $\delta = \lambda_{min} - \mu_{max} - \|E_{11}\|_2 - \|E_{22}\|_2$ is positive: $\delta > 361/400 - 1/400 - (43/10)\epsilon \ge 137/200$. Also we have $\|E_{12}\|_2 \le (43/20)\epsilon \le 43/400 < \delta/2$ and all the assumptions of Theorem 12 are satisfied. It follows that there exists an $((n-k) \times k$ matrix P satisfying $\|P\|_2 \le \frac{2}{\delta}\|E_{21}\|_2 \le 7\epsilon$ such that

$$Q_1' = (Q_1 + Q_2 P)(I + P^T P)^{-1/2} \tag{4.1}$$

forms an orthonormal basis for a subspace that is invariant for $B+E$. This invariant subspace corresponds to the k largest singular values of $A+F$. Therefore, the column vectors of U'_k, (the first k eigenvectors of $B+E$) span the same invariant subspace as that spanned by the column vectors of Q'_1. In other words, there is a $k \times k$ orthonormal matrix R such that $U'_k = Q'_1 R$.

Since $\|Q_1\|_2 \leq 1$, $\|Q_1\|_2 \leq 1$, and

$\|P\|_2 \leq 7\epsilon$, it follows from (4.1) that $Q'_1 = Q_1 + H$ for some H with $\|H\|_2 \leq 9\epsilon$. Therefore, $U'_k = U_k R + HR$, with $\|HR\|_2 \leq 9\epsilon$, as claimed. □

The following lemma is also used in the proof of Theorem 9.

Lemma 10 *Let $U \in \boldsymbol{R}^{n\times k}$ be a matrix with orthonormal columns and let $W \in \boldsymbol{R}^{n\times k}$ be a matrix with $\|W-U\|_2 \leq \epsilon$. Let $u, v, w \in \boldsymbol{R}^n$ be vectors such that $\|U^T u\|_2 = \|U^T v\|_2 = \|U^T w\|_2 = 1$, $(U^T u, U^T v) = 1$ and $(U^T u, U^T w) = 0$. Let $u', v', w' \in \boldsymbol{R}^n$ be arbitrary vectors with $\|u-u'\|_2, \|v-v'\|_2, \|w-w'\|_2 \leq \epsilon$. Then,*

$$(W^T u', W^T v') \geq (1-4\epsilon)\|W^T u'\|_2 \|W^T v'\|_2, \quad \textit{and}$$

$$(W^T u', W^T w') \leq 4\epsilon.$$

Chapter 5

Sampling Lattice Points

When is the volume of a convex polytope in $\boldsymbol{R}^n$ close to the number of lattice points in the polytope? We show, algorithmically, that if the polytope contains a ball of radius $n\sqrt{\log m}$, where m is the number of facets, then the volume approximates the number of lattice points to within a constant factor.

5.1 Introduction

In recent years, *random sampling* has become ubiquitous in its applicability. In this chapter we return to a classic theme of random sampling, namely *uniform generation and approximate counting.* The connection between uniformly generating from combinatorial sets and approximately counting them was observed more than a decade ago [37]. Following that, the *Markov Chain Monte-Carlo* method has yielded efficient algorithms based on *random walks* for a variety of generation (and counting) problems.

Here we consider the problem of counting approximately the number of lattice points in an $n-$dimensional polytope of the form

$$P = \{x \in \mathbf{R}^n : Ax \leq b\},$$

where A is an $m \times n$ matrix of nonnegative reals and b is an $m-$ vector of nonnegative reals. Letting $\mathbf{Z}^n$ denote the set of integer points (points with all integer coordinates), we are interested in the problem of estimating $|P \cap \mathbf{Z}^n|$ given A, b. Closely related to it is the problem of sampling nearly uniformly from the set $P \cap \mathbf{Z}^n$ [37].

This problem includes as a special case several combinatorial counting problems that have been studied - like that of estimating the permanent of a 0-1 matrix [36], the number of contingency tables [26], solutions to knapsack problems [25] etc.. It is well-known that the finding $|P \cap \mathbf{Z}^n|$ exactly is # P-hard [58]. It is also easy to show that it is NP-hard to estimate $|P \cap \mathbf{Z}^n|$ to any polynomial (in input length) factor. For completeness, we include a proof of this folklore result here.

An approach to the problem is to reduce it to the tractable problem of estimating the volume of P or a related polytope (see [24]). Intuitively, it is easy to argue that if each entry of b is sufficiently large, then the number of integer points in the polytope is close to the volume of the polytope. In fact, this general principle can be dated back to Gauss, and has been the subject of many fascinating studies (see e.g. [27]).

In this chapter, we prove that if the polytope contains a ball of radius $\Omega(n\sqrt{\log m})$, then the volume of P approximates $|P \cap \mathbf{Z}^n|$ well. We also show that this is essentially tight. The proof is relatively simple and is algorithmic, so it actually yields an algorithm to sample nearly uniformly from $P \cap \mathbf{Z}^n$ under the condition. We give a simple class of examples to show that the bound is tight to within constants.

From this general result, several interesting special cases follow.

One interesting case is that of sampling uniformly from the set of nonnegative integer matrices with specified row and column sums (called "contingency tables"). Specifically, the problem is the following : for natural numbers m, n, we are given "row sums" $r_1, r_2, \ldots r_m$ and column sums $c_1, c_2, \ldots c_n$ which are all nonnegative integers with $\sum r_i = \sum c_j$. We are to pick a sample nearly uniformly from the integer points in the polytope

$$P = \{x \in \mathbf{R}_+^{mn} : \sum_j x_{ij} = r_i \quad \text{for } i = 1, 2, \ldots m \quad \sum_i x_{ij} = c_j \quad \text{for } j = 1, 2 \ldots n\}.$$

This problem arises in Statistics and Combinatorics [20, 21]. Dyer, Kannan and Mount [26] gave a polynomial time algorithm for this problem provided $r_i \in \Omega(mn^2)$ and $c_j \in \Omega(nm^2)$. Our general result here implies the earlier result, with a simple proof (and with slightly weaker assumptions).

A second special case is that of sampling nearly uniformly from the set of integral $s - t$ flows in a network $G = (V, E)$. We show that if each edge capacity is at least $|E|\sqrt{|E|}$, we can do the sampling in polynomial time and hence also solve approximately the problem of counting the number of integral flows. In contrast, we recapitulate the folklore result that it is NP-hard to estimate the number of integral flows when the capacities are all 1. It is an interesting open problem to reduce this gap.

A third special case is the b-matching problem. In this problem, we are given nonnegative integers $b(v)$ associated with the vertices of a graph G. A b-matching is an edge-weighted subgraph of G whose degree at vertex v is at most $b(v)$. Optimization over the set of b-matchings of a graph is studied e.g. in [34]. Here we show that if all the $b(v)$'s satisfy $b(v) \geq |E|deg(v)$, where $deg(v)$ is the degree of vertex v in G, then we can sample uniformly from the set of b-matchings.

Another special case is that of the multidimensional knapsack problem. Here the polytope P is of the form

$$P = \{x \in \mathbf{R}^n : Ax \leq b, 0 \leq x_i \leq d, \quad for \quad i = 1, \ldots, n\},$$

where A is a nonnegative integer matrix and d is a vector of "upper bounds". Without loss of generality, we may assume that $d_j A_{ij} \leq b_i$ for all i, j. It is shown in [25] that if all $d_j \geq n^2$, then there is a polynomial time algorithm to count approximately the number of integer points in P. We show that our general result gives a polynomial time algorithm with slightly better bounds.

5.2 The Sampling Theorem

We call a point in $\mathbf{R}^n$ with all integer coordinates an integer point. If x is any point, and λ a positive real, we denote by $C(x, \lambda)$ the cube of side 2λ with x as center.

Suppose A is an $m \times n$ matrix of reals and b is an $m \times 1$ of nonnegative reals and

$$P = \{x \in \mathbf{R}^n : Ax \leq b\}.$$

Let r be the maximum number (over integral points x) of facets of P that intersect any $C(x, 1)$ for x an integral point. Let A_i denote the i th row of A. Then our main result can be summarized as follows.

Theorem 13 *For any polytope P satisfying $b_i \in \Omega(n\sqrt{\log r}|A^{(i)}|)$ for all i, there exists a polynomial time algorithm for nearly-uniformly sampling $P \cap \mathbf{Z}^n$.*

The running time of the algorithms will be inversely proportional to the desired accuracy. The rest of this section is devoted to proving this theorem by constructing a sampling algorithm.

Let c be any positive real to be specified later and let b' be a $m-$vector defined by

$$b'_i = b_i + (c + \sqrt{2 \log r})|A_i|.$$

Let $P' = \{x : Ax \leq b'\}$.

Our idea will be to pick a point p from P' from a probability density close to the uniform. We will then "round" p to obtain an integer point; if the integer point is in P, we accept, otherwise, we reject and repeat. We will use a natural probabilistic rounding procedure which we describe presently. This simple rounding procedure used in a different context by Raghavan and Thompson e.g. [52], is the main new ingredient here.

For $p \in \mathbf{R}^n$, we define a vector-valued random variable $X(p)$ by

$$X(p)_i = \begin{cases} \lfloor p_i \rfloor + 1 & \text{with probability} \quad p_i - \lfloor p_i \rfloor \\ \lfloor p_i \rfloor & \text{with probability} \quad 1 - p_i + \lfloor p_i \rfloor \end{cases}$$

where the $X(p)_i, i = 1, 2, \ldots n$ are chosen independently.

Theorem 14 *Suppose p is picked from a probability density $\mathcal{P}$ whose variational distance to the uniform density on P' is at most ε. Then for any $x \in P \cap \mathbf{Z}^n$, we have*

$$\frac{1 - 2e^{-c^2} - \varepsilon}{vol(P')} \quad \leq \quad \mathbf{Pr}\left(X(p) = x\right) \quad \leq \quad \frac{1}{vol(P')}.$$

Proof. First we establish the lower bound. The idea is to bound the probability of picking an integer point x in terms of the probability of picking x given that we pick a continuous point in $C(x, 1)$.

Define $\{Y_i\}_{i=1}^n$ to be independent identically distributed real valued random variables each distributed according to the density function $1 - |t|$ on the real interval $t \in [-1 \;\; +1]$. In the calculations below, dp is an infinitesimal n-dimensional volume and $d\mathcal{P}$ is the probability of picking a point from dp. Then for any $x \in P \cap \mathbf{Z}^n$, and $p \in \mathbf{R}^n$, with $|p_i - x_i| \leq 1$, we have that

$$\text{Prob. density } (x + Y = p) = \prod_{i=1}^n (1 - |p_i - x_i|).$$

$$\text{Also, } \mathbf{Pr}(X(p) = x | p) = \prod_{i=1}^n (1 - |p_i - x_i|).$$

$$\int_{p \in P'} \mathbf{Pr}(X(p) = x | p) d\mathcal{P} = \frac{1}{\text{vol}(P')} \int_{p \in P'} \mathbf{Pr}(X(p) = x | p) dp + \varepsilon',$$

where $|\varepsilon'| \leq \varepsilon$.

$$\text{Now, } \int_{p \in P'} \text{ Prob. density } (x + Y = p) dp =$$

$$\int_{p \in C(x,1)} \text{ Prob. density } (x + Y = p) dp - \int_{C(x,1) \setminus P'} \text{ Prob. density } (x + Y = p) dp$$

$$\geq 1 - \sum_{i=1}^{r} \mathbf{Pr}(A_i Y \geq b_i' - b_i).$$

Now for a fixed i, consider the random variables

$$Z_j = \sum_{k=1}^{j} A_{ik} Y_k.$$

It is easy to see that $E(Z_j | Z_{j-1}) = Z_{j-1}$, so the $\{Z_j\}$ form a Martingale. Also, $|Z_j - Z_{j-1}| \leq A_{ij}$. So applying Azuma's inequality [1], we get that

$$\mathbf{Pr}(A_i Y \geq b_i' - b_i) \leq 2e^{-c^2}/r.$$

This proves the lower bound in the Theorem.

The upper bound on the probability follows by

$$\int_{p \in C(x,1)} \text{Prob. density } (x + Y = p) dp = 1.$$

We show that each $x \in P \cap \mathbf{Z}^n$ gets picked with about the same probability by the following algorithm.

Sampling Algorithm

Suppose $\epsilon > 0$ is given.

- Pick a point p from P' with $c = \sqrt{\ln \frac{4}{\epsilon}}$, according to a probability density $\mathcal{P}$ whose variational distance to uniform is at most $\frac{\epsilon}{2}$.
- If $X(p)$ is in P, then return it; otherwise reject and repeat.

It remains to show that the probability of rejection is not too high assuming bounds on b_i.

Lemma 11 *If $b_i \geq 8n\sqrt{\log \frac{r}{\epsilon}} |A_i|$ for all i, then the probability of acceptance of the sampling algorithm is at least $\frac{1}{32}$.*

Proof. The probability of acceptance is

$$\sum_{x \in P \cap \mathbf{Z}^n} \mathbf{Pr}(X(p) = x) \geq \frac{1}{4} \frac{|P \cap \mathbf{Z}^n|}{\text{vol}(P')}.$$

We show that

$$|P \cap \mathbf{Z}^n| \geq \frac{1}{8} \text{vol}(P').$$

To this end, suppose we pick p from the uniform density from

$$P'' = \{x : A_i x \le b_i - (c + \sqrt{2\log r})|A_i| \text{for } i = 1, 2, \ldots m\}$$

and let $x = X(p)$. Then by the same argument as before, with probability at least $\frac{1}{4}$, x will be in $P \cap \mathbf{Z}^n$. Also, for a fixed $x \in P \cap \mathbf{Z}^n$, the probability that we get $X(p) = x$ by this process is at most $1/\mathrm{vol}(P'')$. So,

$$\frac{|P \cap \mathbf{Z}^n|}{\mathrm{vol}(P'')} \ge \text{ Prob. that } X(p) \in P \ge \frac{1}{4}.$$

Thus, $|P \cap \mathbf{Z}^n| \ge \frac{1}{4}\mathrm{vol}(P'')$. Now,

$$\frac{\mathrm{vol}(P')}{\mathrm{vol}(P'')} \le \max_i \left(\frac{b_i + (c + \sqrt{2\log r})|A_i|}{b_i - (c + \sqrt{2\log r})|A_i|} \right)^n \le 2.$$

□

For algorithmic purposes we can weaken the lower bound to be that each component of b is $\Omega(n|A^{(i)}|\frac{\sqrt{\log r}}{\log n})$.

Note that we assumed as hypothesis that $b_i \in \Omega(n\sqrt{\log r}|A^{(i)}|)$. It is easy to see that this is equivalent to saying that a ball of radius $\Omega(n\sqrt{\log r})$ with the origin as center is contained in P. It is then a simple matter to remove the restriction that the ball have the origin as its center.

The latest algorithms for estimating the volumes of convex sets (or sampling from them) are quite fast: $O^*(n^5)$ [39]. This follows a long series of improvements starting from [24] and [47]. It is worth mentioning that these "continuous" random walks are now provably much faster than their discrete counterparts.

The arguments used in proving Theorem 13 also prove

Theorem 15 *Let P be a polytope in $\boldsymbol{R}^n$ with m facets containing a ball of radius $\Omega(n\sqrt{\log m})$. Then there is a constant c such that*

$$c|P \cap \mathbf{Z}^n| \le \mathit{vol}(P) \le \frac{1}{c}|P \cap \mathbf{Z}^n|.$$

5.3 Special cases of the theorem

5.3.1 Contingency tables

The problem is as described in the Introduction. But it will be more convenient to deal with a full dimensional polytope. With some simple manipulation, it is easy to see that as in [26], we can define the polytope P with $\{x_{ij}\}, 1 \leq i \leq m-1 \quad 1 \leq j \leq n-1$ as the variables defined by the following constraints :

$$\sum_{j=1}^{n-1} x_{ij} \leq r_i \quad \forall i$$

$$\sum_{i=1}^{m-1} x_{ij} \leq c_j \quad \forall j$$

$$\sum_{i=1}^{m-1} \sum_{j=1}^{n-1} x_{ij} \geq \sum_{i=1}^{m-1} r_i - c_n \quad x_{ij} \geq 0.$$

In the above, as well as in the rest of the section, i will run through $1, 2, \ldots m-1$ and j will run through $1, 2, \ldots n-1$ unless otherwise specified.

To apply Theorem 13, we will reformulate the problem with the substitution

$$y_{ij} = x_{ij} - mn.$$

Then the polytope P in y-space is defined by

$$\sum_{j=1}^{n-1} y_{ij} \leq r_i - mn(n-1) \quad \forall i$$

$$\sum_{i=1}^{m-1} y_{ij} \leq c_j - mn(m-1) \quad \forall j$$

$$\sum_{i=1}^{m-1} \sum_{j=1}^{n-1} y_{ij} \geq \sum_{i=1}^{m-1} r_i - c_n - mn(n-1)(m-1)$$

$$y_{ij} \geq -mn.$$

Now if $r_i \geq 2mn^2$ for each r_i and $c_j \geq 2nm^2$ for each c_j, P satisfies the theorem's requirements. We can relax these conditions slightly, namely it suffices to have $r_i \geq 2mn^2/\sqrt{\log mn}$ and $c_j \geq 2nm^2/\sqrt{\log mn}$ for each r_i and each c_j respectively.

5.3.2 Integral flows

In this section we consider the problem of sampling (nearly) uniformly from the set of integral $s-t$ flows in a network. Although our approach could be used for directed graphs, for simplicity we assume here that we are given an undirected graph $G = (V, E)$ with capacities on the edges $C : E \rightarrow \boldsymbol{R}^+$, and two distinguished vertices $s, t \in V$ called the source and the sink respectively. An *integral flow* is an assignment of integers to the edges corresponding to a flow from s to t. Such an assignment must respect the capacity constraints, namely the flow through an edge must be less than the capacity, and the conservation constraints, namely the net flow at a vertex must be zero. To set up the problem as a polytope, we make two modifications to G. First we add the edge (t,s) to the graph (if it is not present) so that we can enforce the conservation constraints at all vertices (i.e. including s and t). Then we arbitrarily direct every edge so that E is now a set of directed edges. The resulting polytope of feasible solutions, P, has the following constraints:

$$-C(i,j) \le x_{ij} \le C(i,j) \quad \forall (i,j) \in E$$
$$\sum_{j:(i,j)\in E} x_{ij} = \sum_{j:(j,i)\in E} x_{ji} \quad \forall i \in V.$$

This polytope is not full-dimensional in $\boldsymbol{R}^{|E|}$. It will be more convenient to work with a full-dimensional polytope, so we apply some further transformations. Choose a spanning tree T of G (such a spanning tree exists because in order to have a non-zero flow we can assume that s and t are in the same connected component; any component that does not contain s or t is irrelevant and can be deleted). For simplicity, we can assume that T is an arborescence, rooted at s (say), by directing the edges of T first, when we chose directions, and then the rest of the edges. Number the vertices in postfix order along T, so that each vertex gets a higher number than any descendant of it. Consider the conservation constraint for a leaf vertex i, let $(k,i) \in T$ be the leaf edge.

$$x_{ki} = \sum_{j:(i,j)\in E\setminus T} x_{ij} - \sum_{j:(j,i)\in E\setminus T} x_{ji}.$$

By substituting these expressions for leaf edges in the conservation constraints for next-to-leaf vertices, and recursing, we get the following equations, one for each tree edge (k,l):

$$x_{kl} = \sum_{i\in S(l)} \left(\sum_{j:(i,j)\in E\setminus T} x_{ij} - \sum_{j:(j,i)\in E\setminus T} x_{ji} \right) \quad \forall (k,l) \in T$$

Here $S(l)$ denotes the set of vertices in the subtree rooted at l in T. So the polytope P can be reformulated in $\boldsymbol{R}^{|E|-|V|+1}$ space as

$$-C(i,j) \le x_{ij} \le C(i,j) \quad \forall (i,j) \in E \setminus T$$
$$-C(k,l) \le \sum_{i\in S(l)} \left(\sum_{j:(i,j)\in E\setminus T} x_{ij} - \sum_{j:(j,i)\in E\setminus T} x_{ji} \right) \le C(k,l) \quad \forall (k,l) \in T.$$

Since each constraint trivially has at most E variables, if each capacity is at least $|E|^{\frac{3}{2}}$ then we can apply the theorem to sample the integer points of P nearly uniformly (and thus also count the number of integral flows). Note that a simple modification will allow us to sample the number of flows of a specified value f.

5.3.3 Hardness of counting flows

To contrast with the above result, we recount the folklore result that it is NP-hard to count approximately the number of flows. We do this by reducing the NP-complete problem of deciding whether a graph has a Hamilton cycle to this approximate counting problem.

Theorem 16 *It is NP-hard to count the number of $s-t$ simple paths in a graph with n vertices to any $poly(n)$ factor.*

Proof. Let $k, l \in \mathbf{Z}_+$ be two numbers which we will fix later. We replace each edge of G by lk edges: first divide the edge into l edges by introducing $l-1$ new vertices, and then replace each resulting edge with k parallel edges.

The number of cycles in the new graph G' will tell us if the original graph G is hamiltonian. If G is Hamiltonian then G' has at least $(k^l)^n = k^{ln}$ cycles (corresponding to a single Hamilton cycle). On the other hand, if G has no cycles of length n, an upper bound on the number of cycles in G' is $k^{l(n-1)}2^n n!$ because there are fewer than $2^n n!$ cycles in G and each has at most $k^{l(n-1)}$ representatives in G'. By a suitable choice of k, l with each only $O(n)$, we can make the first number higher than any fixed $poly(n)$ times the second number. Thus counting the number of cycles in G' up to any $poly(n)$ factor would let us decide if G is Hamiltonian.

Finally, the number of cycles involving a particular edge (u, v) is just the number of paths/flows from u to v in the graph with the edge deleted and all capacities set to 1. $\square$

5.3.4 b-matchings

Given an undirected graph, $G = (V, E)$, and a function $b : V \to \mathbf{Z}^+$, a *b-matching* is an assignment of positive integers to edges, $x : E \to \mathbf{Z}^+$, so that the sum of the weights on edges incident at a vertex v is at most $b(v)$. A b-matching is *perfect* if it has a weight of exactly $b(v)$ at every vertex. A perfect b-matching can be found in polynomial time (if one exists) using the ellipsoid algorithm [34]. Here we consider the problem of sampling from the set of b-matchings uniformly at random.

Let P be the polytope defined by the following constraints.

$$x(e) \geq 0 \quad \forall e \in E$$

$$x(\delta(v)) \leq b(v) \quad \forall v \in V$$

In the second set of constraints, $\delta(v)$ represents the edges incident to v and $x(\delta(v))$ is the sum of the weights on these edges. Any integer solution satisfying the above constraints is a valid b-matching of G.

Let $|E| = m$, and $d(v)$ be the degree of a vertex v. To apply the theorem, we use the following substitution,

$$y(e) = x(e) - m.$$

Then in y-space, we have the following polytope

$$y(e) \geq -m$$

$$y(\delta(v)) \leq b(v) - md(v).$$

Now for any b such that $b(v) \geq 2md(v)$ for all $v \in V$, and in particular for b such that each $b(v) \geq 2mn$, we can sample b-matchings of G nearly uniformly. This can be extended to sampling capacitated b-matchings, where there are additional constraints on the maximum weights assigned to edges [34].

5.4 Tight examples

In this section we give examples to show that our bound on the components of b is tight (up to a $\sqrt{\log r}$ factor). Consider the simplex in n dimensions,

$$P = \{x \in \mathbf{R}^n : \sum_{i=1}^{n} x_i \leq b, \quad x_i \geq 0 \quad \forall i\}$$

To derive an upper bound, we can reformulate P (thereby centering it around the origin). We substitute,

$$y_i = x_i - n.$$

Then in y-space, P becomes,

$$\sum_{i=1}^{n} y_i \leq b - n^2, \quad y_i \geq -n \quad \forall i.$$

So now for $b \geq n^2 + n\sqrt{n}$, P satisfies the conditions of the theorem.

Let us examine what happens if b is slightly smaller. The volume of the simplex is $b^n/n!$ and the number of lattice points in P, i.e., the number of ways of dividing b into n or fewer parts is $\binom{b+n-1}{n-1}$. So the ratio of the number of lattice points to the volume is

$$\frac{n(b+n-1)!}{b^n b!}$$

which is the same as

$$\frac{n}{b}(1+\frac{n-1}{b})(1+\frac{n-2}{b})..(1+\frac{1}{b}).$$

The latter is lower bounded by

$$\frac{n}{b}(1+\frac{n-1}{2b})^{\frac{n}{2}}.$$

For any c such that $b \leq cn^2$, this is exponential in $1/c$ (the ratio is about $e^{\frac{1}{c}}$), showing that the volume is no longer a good approximation to the number of lattice points.

5.5 Conclusion and open problems

Is the dependence of our main theorem on m, the number of facets, necessary? It would be nice to get rid of it.

Although we have presented a (nearly) tight condition for sampling lattice points, it is possible that a different point of view would yield more general algorithms. For example, for *lattice point based random walks*, what are some simple, easy-to-verify conditions that guarantee rapid-mixing?

Bibliography

[1] N. Alon and J. Spencer *The probabilistic method.* John Wiley (1992).

[2] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 402–409, May 1991.

[3] J. A. Aslam and S. E. Decatur. General bounds on statistical query learning and PAC learning with noise via hypothesis boosting. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 282–291, November 1993.

[4] J. A. Aslam and S. E. Decatur. Improved noise-tolerant learning and generalized statistical queries. Technical Report TR-17-94, Harvard University, July 1994.

[5] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6(3):382–392, 1954.

[6] E. Amaldi. *From finding maximum feasible subsystems of linear systems to feed-forward neural network design.* PhD thesis, Swiss Federal Institute of Technology at Lausanne (EPFL), October 1994. (Ph.D. dissertation No. 1282, Department of Mathematics).

[7] J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research.* MIT Press, 1988.

[8] E. B. Baum. Polynomial time algorithms for learning neural nets. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 258-272. Morgan Kaufmann (1990).

[9] Eric B. Baum. On learning a union of half spaces. *Journal of Complexity*, 6(1):67-101, March 1990.

[10] M. W. Berry, S. T. Dumais, and G. W. O'Brien. Using linear algebra for intelligent information retrieval. SIAM Review, 37(4), 1995, 573-595, 1995.

[11] M. W. Berry, T. Do, G. W. O'Brien, V. Krishna, and S. Varadhan. SVDPACKC (Version 1.0) User's Guide. University of Tennessee, April 1993.

[12] A. Blum, A. Frieze, R. Kannan and S. Vempala. A polynomial-time algorithm for learning noisy linear threshold functions. In *Proceedings of the 37th Symposium on the Foundations of Computer Science*, 330-338, 1996.

[13] A. Blum and R. Kannan. Learning an intersection of k halfspaces over a uniform distribution. In *Proceedings of the 34th Symposium on the Foundations of Computer Science*, 312-320, 1993.

[14] A. Blum and R. Rivest. Training a 3-node neural network is NP-hard. *Neural Networks*, 5:117-127, 1992.

[15] T. Bylander. Polynomial learnability of linear threshold approximations. In *Proceedings of the Sixth Annual Workshop on Computational Learning Theory*, pages 297–302. ACM Press, New York, NY, 1993.

[16] T. Bylander. Learning linear threshold functions in the presence of classification noise. In *Proceedings of the Seventh Annual Workshop on Computational Learning Theory*, pages 340–347. ACM Press, New York, NY, 1994.

[17] E. Cohen. Learning a noisy perceptron by a perceptron in polynomial time. To appear in *Proc. of the Thirty-Eighth Annual IEEE Symposium on the Foundations of Computer Science*, 1997.

[18] D.M. Cvetković, M. Doob, and H. Sachs, Spectra of Graphs, Academic Press, 1979.

[19] S. Deerwester, S. T. Dumais, T.K. Landauer, G.W. Furnas, and R.A. Harshman. Indexing by latent semantic analysis. Journal of the Society for Information Science, 41(6), 391-407, 1990.

[20] P. Diaconis and B. Efron. Testing for independence in a two-way table. *Annals of Statistics*, **13**, pp. 845-913, (1985).

[21] P. Diaconis and A. Ganguly. Rectangular arrays with fixed margins. in *Proceedings of the workshop on Markov Chains*, (1994).

[22] S.T. Dumais, G.W. Furnas, T.K. Landauer, and S. Deerwester. Using latent semantic analysis to improve information retrieval. In Proceedings of CHI'88: Conference on Human Factors in Computing, New York: ACM, 281-285, 1988.

[23] S.T. Dumais. Improving the retrieval of information from external sources. Behavior Research Methods, Instruments and Computers, 23(2), 229-236, 1991.

[24] M. Dyer, A. Frieze and R. Kannan. A random polynomial time algorithm for estimating the volumes of convex bodies. *Journal of the ACM*, 38, 1991.

[25] M. E. Dyer, A. Frieze, R. Kannan, A. Kapoor, L. Perkovic and U. Vazirani. A mildly exponential time algorithm for approximating the number of solutions to a multidimensional knapsack problem. To appear in *Combinatorics, Probability, and Computation.*

[26] M. Dyer, R. Kannan and J. Mount. Sampling contingency tables. To appear in *Random Structures and Algorithms.*

[27] P.Erdös, P. M. Gruber and J. Hammer. *Lattice points.* Longman (1989).

[28] P. Frankl and H. Maehara. The Johnson-Lindenstrauss Lemma and the Sphericity of some graphs, J. Comb. Theory B **44** (1988), 355-362.

[29] Y. Freund. An improved boosting algorithm and its implications on learning complexity. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 391–398. ACM Press, 1992.

[30] N. Fuhr "Probabilistic models of information retrieval," *Computer Journal, 35*, 3, pp. 244–255, 1992.

[31] S. Gallant. Perceptron-based learning algorithms. *IEEE Transactions on Neural Networks*, 1(2):179–191, 1990.

[32] G. Golub and C. Reinsch. Handbook for matrix computation II, Linear Algebra. Springer-Verlag, New York, 1971.

[33] G. H. Golub and C. F. Van Loan. Matrix computations. Johns Hopkins University Press, London, 1989.

[34] M. Grotchel, L. Lovász and A. Schrijver. *Geometric algorithms and combinatorial optimization.* Springer-Verlag, (1988).

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

[36] M. Jerrum and A. Sinclair. Approximating the permanent. *Siam J. Comp.* **18**, pp. 1149-1178, (1989).

[37] M. Jerrum, L. G. Valiant and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comp. Sci.* **43**, pp. 169-188,(1986).

[38] W. B. Johnson and J. Lindenstrauss. Extensions of Lipshitz mapping into Hilbert space, Contemp. Math. **26** (1984), 189–206.

[39] R. Kannan, L. Lovász and M. Simonovits. Random walks and and an $O^*(n^5)$ volume algorithm for convex sets. Preprint, 1996.

[40] R. Kannan and S. Vempala. Sampling lattice points. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing.*

[41] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

[42] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 392–401, 1993.

[43] M. Kearns and U. Vazirani. *An Introduction to Computational Learning Theory.* MIT Press, 1994.

[44] L. G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[45] J. Kleinberg. Two algorithms for nearest-neighbor search in high dimensions. In *Proc. 29th ACM Symposium on Theory of Computing*, 1997.

[46] P. M. Long and M. K. Warmuth. Composite geometric concepts and polynomial predictability. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 273-287. Morgan Kaufmann (1990).

[47] L. Lovász and M. Simononovits. Mixing rate of Markov chains, an isoperimetric inequality, and computing the volume. In *Proceedings of the 31st Symposium on the Foundations of Computer Science*, pp. 346-355, (1990).

[48] W. Maass and G. Turán. On the complexity of learning from counterexamples. In *Proceedings of the Thirtieth Annual Symposium on Foundations of Computer Science*, pages 262–267, October 1989.

[49] N. Megiddo. On the complexity of polyhedral separability. Technical Report RJ 5252, IBM Almaden Research Center, August 1996.

[50] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 1969.

[51] C. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala. Latent Semantic Indexing: A probablistic analysis. Preprint 1997.

[52] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4), pp. 365-374, (1987).

[53] C. J. van Rijsbergen *Information Retrieval* Butterworths, London 1979.

[54] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, 1962.

[55] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[56] H. R. Turtle and W. B. Croft "A comparison of text retrieval methods," *The Computer Journal, 35*, 3, pp. 279–289, 1992.

[57] L. G. Valiant. A theory of the learnable. *Comm. of the ACM*, 27(11):1134-1142, 1984.

[58] L. G. Valiant. The complexity of computing the permanent. *Theor. Comp. Sci.* **8**, pp. 189-201, (1979).

[59] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its applications*, XVI(2):264–280, 1971.

[60] S. Vempala. A random sampling based algorithm for learning the intersection of half-spaces. To appear in *Proc. of the Thirty-Eighth Annual IEEE Symposium on the Foundations of Computer Science*, 1997.

[61] J.H. Wilkinson. The Algebraic Eigenvalue Problem. Oxford University Press, London 1965.

[62] Wisconsin Diagnostic Breast Cancer Database.