A Polynomial-time Rescaling Algorithm for Solving Linear Programs

John Dunagan
Microsoft Research
jdunagan@microsoft.com

Santosh Vempala
Mathematics Department, MIT
vempala@math.mit.edu

Technical Report
MSR-TR-2002-92

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

Abstract

We show that the perceptron algorithm along with periodic rescaling solves linear programs in polynomial time. The algorithm requires no matrix inversions and no barrier functions.

1 Introduction

Linear programs arise naturally in many areas. The standard form is

$$\begin{array}{rcl} \max & c^T x & \\ Ax & \leq & b \\ x & \geq & 0 \end{array}$$

where b, c are in $\mathbb{R}^n$, and A is an $m \times n$ matrix of reals. Algorithms for solving them efficiently (i.e., in polynomial-time) include the Ellipsoid method [10, 12, 19], interior point methods [11, 18] and the random walk method [2]. In practice, though, variants of the simplex algorithm [1, 7] (perhaps the most popular algorithm for the problem) are often very efficient, although they lack worst-case guarantees.

Another classical algorithm which is quite practical for problems arising in machine learning is the perceptron algorithm [13, 15]. The algorithm is a simple greedy method that is guaranteed to converge, but could take an exponential number of iterations in the worst case. However, it has many useful properties, including noise tolerance [4, 5]. The perceptron algorithm has also been related to boosting in the PAC model of learning [16, 17]. It was recently shown that the algorithm is polynomial with high probability for randomly perturbed linear programs [3]. It has been an open question as to whether there is a variant of the perceptron algorithm that is polynomial in the worst case.

In this paper, we consider only the feasibility version of the linear programming problem in standard form, neglecting the objective function $\max\ c^T x$. Polynomial time reductions of the optimization problem to the feasibility version of the problem are well-known [10]. A typical approach is to replace $\max\ c^T x$ by the constraint $c^T x \geq c_0$ for some value of c_0. Binary search can be used to determine a nearly optimal value of c_0. A solution that is feasible for the problem with c_0 sufficiently close to optimal can be *rounded* to an optimal vertex solution through basis reduction [10].

We show that a perceptron-like algorithm along with periodic rescaling applied to a strictly feasible linear program will return a feasible point in polynomial time. As in all variants of the perceptron algorithm, it does not use any matrix inversions or barrier functions. Our main Theorem (Theorem 3.3) is that a strictly feasible linear program with m constraints in $\mathbb{R}^n$ is solved in time $O(mn^4 \log n \log(1/\rho))$ where ρ is a parameter that roughly corresponds to the radius of a ball that fits in the feasible region, and $\log(1/\rho)$ is guaranteed to be polynomial in the input description.

2 The Algorithm

In this section we present an algorithm for the linear feasibility problem

$$Ax \geq 0, x \neq 0$$

consisting of m constraints in n dimensions, i.e. $x \in \mathbb{R}^n$ and A is $m \times n$. In Section 4 we show how well-known methods can be used to reduce the standard form feasibility problem to this *homogenized* form.

The algorithm is iterative and each iteration consists of three phases, a *perceptron* phase, a *perceptron improvement* phase, and a *rescaling* phase. The perceptron phase uses the classical perceptron algorithm. The perceptron improvement phase uses a modified version of the basic perceptron algorithm. This modified version was first described in [4].

The classical perceptron algorithm for the linear feasibility problem is to find a violated constraint, move the trial solution x one unit in the direction normal to the violated constraint, and repeat if necessary (this is step 2 in the algorithm).

In rest of the paper, we let $\bar{x}$ denote the unit vector in the direction of x.

Algorithm.

Input: An $m \times n$ matrix A.
Output: A point x such that $Ax \geq 0$ and $x \neq 0$.

1. Let $B = I, \sigma = 1/(32n)$.
2. (Perceptron)
   (a) Let x be the origin in $\mathbb{R}^n$.
   (b) Repeat at most $16n^2$ times:
   If there exists a row a such that $a \cdot x \leq 0$, set $x = x + \bar{a}$.
3. If $Ax \geq 0$, then output Bx as a feasible solution and stop.
4. (Perceptron Improvement)
   (a) Let x be a random unit vector in $\mathbb{R}^n$.
   (b) Repeat at most $\ln n/\sigma^2$ times:
   If there exists a row a such that $\bar{a} \cdot \bar{x} < -\sigma$, set $x = x - (\bar{a} \cdot x)\bar{a}$.
   If $x = 0$, go back to step (a).
   (c) If there still exists a row a such that $\bar{a} \cdot \bar{x} < -\sigma$, restart at step (a).
5. If $Ax \geq 0$, then output Bx as a feasible solution and stop.
6. (Rescaling)

$$\text{Set} \qquad A = A(I + \bar{x}\bar{x}^T) \quad \text{and} \quad B = B(I + \bar{x}\bar{x}^T).$$

7. Go back to step 2.

Figure 1 depicts the Rescaling step.

3 Analysis

Let A_0 denote the initial A input to the algorithm, and let A_* denote the A that the algorithm has when it terminates. When the algorithm terminates, it produces a non-zero vector $x' = Bx$ such that $A_*x = (A_0B)x \geq 0$, i.e., $A_0(Bx) \geq 0$, as desired.

The perceptron improvement phase may be re-run a number of times, but it terminates quickly with high probability. In [4] it was shown that with probability $\frac{1}{4}$, the vector x at the end of step 4(b) satisfies

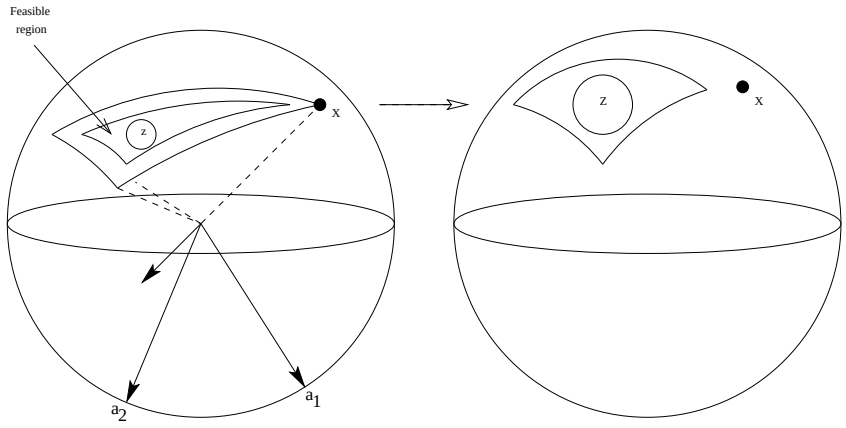

Figure 1: A constraint system before and after rescaling.

$\bar{a} \cdot \bar{x} \geq -\sigma$. We recall this proof for convenience as Lemma 3.4(a) below.

Thus the main question is, how many iterations does the algorithm make? To answer this, we define the following quantity which measures the "roundness" of the feasible region:

$$\rho = \max_{x:|x|=1, Ax \geq 0} \min_i (\bar{a}_i \cdot x).$$

We call ρ the *radius* of A. This quantity was related to the *condition number* of a linear program in [8], and also in [6], and the quantity appears in other work on linear programs as well. We review the relationship between ρ and the condition number in Section 4. The unit vector realizing the above maximum is called the *center* and denoted by z. Roughly speaking, the point z is located centrally in the feasible cone. Lemma 3.4(b) asserts that the vector x at the end of step 4(b) satisfies $z \cdot \bar{x} \geq 1/\sqrt{n}$ with probability at least 1/4.

The classical analysis of the perceptron algorithm [13] (repeated in lemma 3.5) shows that the classical perceptron algorithm (our perceptron phase) applied to a linear feasibility problem with radius ρ will terminate in at most $1/\rho^2$ iterations. Our main lemma shows that in any iteration of our algorithm where ρ is small, it will increase in expectation by a multiplicative factor. Combining our main lemma with the classical analysis will yield that if ρ is small, it gets bigger (guaranteed by the perceptron improvement and scaling phases), and if ρ is big, the algorithm finds a feasible point (guaranteed by the perceptron phase).

Lemma 3.1 *Suppose $\rho \leq 1/4n$. Let $\sigma \leq 1/32n$. Let A' be obtained from A by one iteration of the*

algorithm (one on which the problem was not solved). Let ρ' and ρ be the radii of A' and A respectively. Then,

(a) $\rho' \geq (1 - \frac{1}{16n})\rho$.

(b) With probability at least $\frac{1}{4}$, $\rho' \geq (1 + \frac{1}{4n})\rho$.

Proof: Let a_i, $i = 1, \ldots m$ be the rows of A at the beginning of some iteration. Let z be a unit vector satisfying $\rho = \min_i \bar{a}_i \cdot z$, and let $\sigma_i = \bar{a}_i \cdot \bar{x}$. After a perceptron improvement phase, we get a vector x such that

$$\bar{a}_i \cdot \bar{x} = \sigma_i \geq -\sigma.$$

As in the theorem statement, let A' be the matrix obtained after the rescaling step, i.e.

$$a'_i = a_i + (a_i \cdot \bar{x})\bar{x}.$$

Finally, define

$$z' = z + \alpha(z \cdot \bar{x})\bar{x}.$$

where α will be specified shortly. Although z' is not necessarily the center of A', ρ' is a maximum over a set, and so considering one element ($\bar{z}'$) of the set suffices to lower bound ρ'. We have

$$\rho' \geq \min_i \bar{a}'_i \cdot \bar{z}' = \frac{\bar{a}'_i \cdot z'}{|z'|}.$$

We will first prove that $\bar{a}'_i \cdot z'$ cannot be too small.

$$\begin{aligned} \bar{a}'_i \cdot z' &= \left(\frac{\bar{a}_i + (\bar{a}_i \cdot \bar{x})\bar{x}}{\|\bar{a}_i + (\bar{a}_i \cdot \bar{x})\bar{x}\|} \right) \cdot z' \\ &= \frac{[\bar{a}_i + (\bar{a}_i \cdot \bar{x})\bar{x}][z + \alpha(z \cdot \bar{x})\bar{x}]}{\sqrt{1 + 3(\bar{a}_i \cdot \bar{x})^2}} \\ &\geq \frac{\rho + \sigma_i(z \cdot \bar{x})(1 + 2\alpha)}{\sqrt{1 + 3\sigma_i^2}} \end{aligned}$$

We choose :

$$\alpha = \frac{1}{2}(\frac{\rho}{\bar{x} \cdot z} - 1)$$

so that $2\alpha + 1 = \rho/(\bar{x} \cdot z)$. We have not ensured that $\bar{x} \cdot z \neq 0$, but substituting in for $\alpha(\bar{x} \cdot z)$ in the definition of z' using the value we have chosen for α would remove this boundary case. We have chosen not to do this to aid the exposition. We proceed to calculate

$$\bar{a}'_i \cdot z' \geq \rho \frac{1 + \sigma_i}{\sqrt{1 + 3\sigma_i^2}} \geq \rho \frac{1 - \sigma}{\sqrt{1 + 3\sigma^2}}. \tag{1}$$

where the second inequality follows from $\sigma_i \in [-\sigma, 1]$. Next, observe that

$$|z'|^2 = 1 + (\alpha^2 + 2\alpha)(\bar{x} \cdot z)^2 = 1 + \frac{\rho^2}{4} + (z \cdot \bar{x}) \left(\frac{\rho}{2} - \frac{3}{4}(z \cdot \bar{x}) \right).$$

We consider two cases:

1. $|z \cdot \bar{x}| < \frac{1}{\sqrt{n}}$. This happens with probability at most 3/4.

   Viewing $|z'|^2$ as a quadratic polynomial in $(z' \cdot x)$, we see that it is maximized when $(z' \cdot x) = \frac{\rho}{3}$. In this case, we have

   $$|z'|^2 \leq 1 + \frac{\rho^2}{4} + \frac{\rho^2}{12} \leq 1 + \frac{1}{48n^2}.$$

   Using the elementary identity $\frac{1}{\sqrt{1+\beta}} \geq 1 - \frac{\beta}{2}$ for $\beta \in (-1, 1)$, we find

   $$\begin{aligned} \rho' &\geq \rho \frac{1 - \frac{1}{32n}}{\sqrt{1 + \frac{3}{2^{10}n^2}}\sqrt{1 + \frac{1}{48n^2}}} \\ &\geq \rho\left(1 - \frac{1}{32n}\right)\left(1 - \frac{3}{2^{11}n^2}\right)\left(1 - \frac{1}{96n^2}\right) \\ &\geq \rho\left(1 - \frac{1}{16n}\right). \end{aligned}$$

2. $|z \cdot \bar{x}| \geq \frac{1}{\sqrt{n}}$. This happens with probability at least 1/4.

   In this case,

   $$|z'|^2 = 1 + \frac{\rho^2}{4} + (z \cdot \bar{x})\frac{\rho}{2} - \frac{3}{4}(z \cdot \bar{x})^2 \leq 1 + \frac{1}{64n^2} + \frac{1}{8n\sqrt{n}} - \frac{3}{4n} \leq 1 - \frac{39}{64n}.$$

   Using the same elementary identity as above, we find

   $$\begin{aligned} \rho' &\geq \rho \frac{1 - \frac{1}{32n}}{\sqrt{1 + \frac{3}{2^{10}n^2}}\sqrt{1 - \frac{39}{64n}}} \\ &\geq \rho\left(1 - \frac{1}{32n}\right)\left(1 - \frac{3}{2^{11}n^2}\right)\left(1 + \frac{39}{128n}\right) \\ &\geq \rho\left(1 + \frac{1}{4n}\right). \end{aligned}$$

This proves both parts of the Lemma.

□

Theorem 3.2 *With high probability, the algorithm finds a feasible solution in $O(n \log(1/\rho))$ iterations.*

Proof: From our previous discussion, it suffices to show that ρ has grown to at least $1/4n$ in this many iterations. Let X_i be a random variable for the i'th iteration, with value 1 if ρ grows by a factor of $(1 + 1/4n)$ or more and value 0 otherwise. Let X be the sum of the X_i's over T iterations. Then by Lemma 3.1(b)

$$\mathbf{E}[X] \geq \frac{T}{4}.$$

The Chernoff bound gives,

$$\Pr(X < (1 - \epsilon)\mathbf{E}[X]) \leq e^{-\epsilon^2 E[X]/2}.$$

Let ρ_T be the ρ value after T iterations. Let $T = 2048n \log(1/\rho)$ and $\epsilon = 1/16$. Then

$$e^{-\epsilon^2 T/8} \leq e^{-n}$$

which satisfies the definition of high probability. Analyzing ρ_T in the case that X is within ϵ of its expectation, we have

$$\begin{aligned}
\rho_T &\geq \rho\left(1+\frac{1}{4n}\right)^{X}\left(1-\frac{1}{16n}\right)^{T-X} \\
&\geq \rho\left(1+\frac{1}{4n}\right)^{\frac{T}{4}-\epsilon\frac{T}{4}}\left(1-\frac{1}{16n}\right)^{\frac{3T}{4}+\epsilon\frac{T}{4}} \\
&\geq \rho\left(1+\frac{1}{4n}\right)^{\frac{15T}{64}}\left(1-\frac{1}{16n}\right)^{\frac{47T}{64}} \\
&\geq \rho e^{T/512n}.
\end{aligned}$$

We summarize: with probability at least $1-e^{-n}$, before the algorithm runs more than T iterations, ρ grows to at least $1/(4n)$, at which point the perceptron phase succeeds in finding a feasible point. □

Theorem 3.3 *With high probability, the algorithm finds a feasible solution in time $O(mn^4 \log n \log(1/\rho))$.*

Proof: The inner loop of either the perceptron phase or the perceptron improvement phase requires at most one matrix-vector multiplication (time $O(mn)$), and a constant number of vector manipulations (time $O(n)$). The number of times we repeat the inner loop is at most $16n^2$ in the perceptron phase, and at most $\log n/\sigma^2 = O(n^2 \log n)$ in the perceptron improvement phase. The bound on the total number of times through the perceptron improvement phase is probabilistic, but it can be shown to be close to its expectation using a Chernoff bound just as in the previous theorem. The scaling phase takes time $O(n^2)$. Calculating Bx similarly takes time $O(n^2)$.

By the previous theorem, the number of iterations is at most $O(n \log(1/\rho))$. This yields the overall time bound $O(mn^4 \log n \log(1/\rho))$. □

Lemma 3.4 (BFKV [4]) *Let z and A be as before. With probability at least $\frac{1}{4}$, in at most $\ln n/\sigma^2$ steps, the modified perceptron algorithm (our perceptron improvement phase) returns a vector x for which*

(a) $\bar{a} \cdot x \geq -\sigma$ for every row a of A and

(b) $z \cdot \bar{x} \geq 1/\sqrt{n}$.

Proof: The proof of both parts is similar. By elementary geometry, $z \cdot x \geq 1/\sqrt{n}$ with probability at least $1/4$. We now show that if this is the case, we terminate in the desired number of iterations. If it is not the case, we might terminate anyways, or we might not.

Note that in each update step, $z \cdot x$ does not decrease

$$(x - (x \cdot \bar{a}_i)\bar{a}_i) \cdot z = x \cdot z - (x \cdot \bar{a}_i)(\bar{a}_i \cdot z) \geq x \cdot z$$

because $x \cdot \bar{a}_i$ had to be negative in order for $\bar{a}_i$ to be used in an update step, and $\bar{a}_i \cdot z = \rho \geq 0$ by assumption. (This also implies that if $z \cdot x \geq 1/\sqrt{n}$ initially, x will never be set to zero.) On the other hand $x \cdot x$ does decrease significantly because

$$\begin{aligned}
(x - (x \cdot \bar{a}_i)\bar{a}_i) \cdot (x - (x \cdot \bar{a}_i)\bar{a}_i) &= x \cdot x - 2(\bar{a}_i \cdot x)^2 + (\bar{a}_i \cdot x)^2 = x \cdot x - (\bar{a}_i \cdot x)^2 \\
&\leq x \cdot x(1-\sigma^2)
\end{aligned}$$

Thus after t iterations $\|x\| \leq (1-\sigma^2)^{t/2}$. If $t > (\ln n)/\sigma^2$, we would have $\frac{x \cdot z}{\|x\|} > 1$, which cannot happen. Therefore, every time we start through the algorithm, we finish with probability at least $1/4$, and every time we finish, with probability at least $1/4$ we return x such that $\frac{x \cdot z}{\|x\|} \geq 1/\sqrt{n}$. □

Lemma 3.5 (Block-Novikoff [13]) *If $\rho \geq 1/(4n)$, the classical perceptron algorithm (our perceptron phase) returns a feasible point in at most $16n^2$ iterations.*

Proof: Consider the potential function $\frac{x \cdot z}{\|x\|}$ as in the proof of Lemma 3.4. The numerator increases by at least ρ on each step:

$$(x + \bar{a}_i) \cdot z = x \cdot z + \bar{a}_i \cdot z \geq x \cdot z + \rho$$

While the square of the denominator increases by at most 1:

$$(x + \bar{a}_i) \cdot (x + \bar{a}_i) = x \cdot x + 2x \cdot \bar{a}_i + \bar{a}_i \cdot \bar{a}_i \leq x \cdot x + 1$$

since $x \cdot \bar{a}_i \leq 0$. After t iterations, the potential function is at least $\frac{t\rho}{\sqrt{t}}$ and thus the classical perceptron algorithm must terminate before $1/\rho^2$ iterations. If the algorithm terminates, it must have found a feasible point. □

4 The Standard Form

In this section we show how to reduce the standard form linear feasibility problem

$$Ax \leq b, x \geq 0$$

into the linear feasibility problem we study in the previous sections of the paper. This technique is typically referred to as *homogenization*. We also relate ρ, the condition number, and the "bit-length" of the problem before and after homogenization. We conclude from this that the algorithm is polynomial in the traditional sense [10].

Introduce the variable x_0 and consider the problem

$$Ax \leq bx_0, x \geq 0, x_0 > 0$$

To convert a solution for the standard form to one for the homogenized form, set $x_0 = 1$. To convert a solution from the homogenized form to a solution for the standard form, divide x by x_0. To rewrite the homogenized form as just

$$A'x' \geq 0, x' \neq 0$$

Let $x' = [x\ x_0]$ and

$$A' = \begin{bmatrix} -A & b \\ I & \end{bmatrix}$$

A valid solution to $A'x' \geq 0$ might have $x_0 = 0$. However, because the classic perceptron algorithm (our perceptron phase) always produces solutions in the strict interior of the feasible region, our algorithm will always return a solution with $x_0 > 0$.

The traditional measure of the difficulty of a linear programming problem in the Turing model of computation is called the "bit-length" and denoted by L. The quantity L is never more than polynomial in the input length of the linear program. The condition number of a linear program was defined by Renegar as the normalized distance to ill-posedness [14]. Renegar showed that the log of the condition number is upper-bounded by L, and so linear programming algorithms that are polynomial in the log of the condition number are also polynomial in L. The quantity ρ was related to the condition number in [8], and also in [6], and the quantity appeared in the literature even earlier. In particular, ρ of the homogenized program is no more than n times the condition number of the original program. We conclude from this that the algorithm is polynomial in L.

5 Acknowledgements

The authors would like to thank Dan Stefankovic and Adam Smith for helping these ideas through their formative stages, and Adam Klivans and Rocco Servedio for bringing [9] to their attention.

References

[1] D. Bertsimas and J. Tsitsiklis, *Introduction to Linear Optimization*, Athena Scientific, 1997.

[2] D. Bertsimas and S. Vempala, Solving convex programs by random walks, *Proc. STOC*, 2002, 109-115.

[3] A. Blum and J. Dunagan, Smoothed Analysis of the Perceptron Algorithm for Linear Programming, *Proc. SODA*, 2002, http://citeseer.nj.nec.com/539616.html

[4] A. Blum, A. Frieze, R. Kannan, and S. Vempala, A polynomial-time algorithm for learning noisy linear threshold functions, *Algorithmica*, 22(1), 35-52, 1998.

[5] T. Bylander, Learning linear threshold functions in the presence of classification noise, *Proc. 7th Workshop on Computational Learning Theory*, 1994.

[6] F. Cucker and D. Cheung, A new condition number for linear programming, *Math. Programming*, 91(1 (Ser. A)):163–174, 2001.

[7] V. Chvatal, *Linear Programming*, W.H. Freeman, 1983.

[8] J. Dunagan, S. Teng, and D. A. Spielman, Smoothed Analysis of Renegar's Condition Number for Linear Programming, *SIAM Conference on Optimization 2002*.

[9] J. Forster, A Linear Lower Bound on the Unbounded Error Probabilistic Communication Complexity, *Sixteenth Annual IEEE Conference on Computational Complexity 2001*, http://citeseer.nj.nec.com/forster01linear.html

[10] L. Grotchel, L. Lovasz, and A. Schrijver, *Geometric algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin, 1988.

[11] N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica*, vol. 4 (1984), pp. 373–396.

[12] L. G. Khachiyan, A polynomial algorithm in linear programming, (in Russian), *Doklady Akedamii Nauk SSSR*, 244, 1093–1096, 1979 (English translation: *Soviet Mathematics Doklady*, 20, 191–194, 1979).

[13] M. L. Minsky and S. A. Papert, *Perceptrons: An introduction to computational geometry*, 1969.

[14] J. Renegar, Incorporating condition measures into the complexity theory of linear programming, *SIAM J. Optim.*, 5(3):506–524, 1995.

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

[16] R. Servedio. On PAC Learning using Perceptron, Winnow and a Perceptron-Like Algorithm. SIAM Journal on Computing 31(5), 2002, pp. 1358-1369. Twelfth Annual Conference on Computational Learning Theory (COLT), 1999, pp. 296-307.

[17] R. Servedio. Smooth Boosting and Learning with Malicious Noise. Fourteenth Annual Conference on Computational Learning Theory (COLT), 2001, pp. 473-489.

[18] P. M. Vaidya, A new algorithm for minimizing convex functions over convex sets, *Mathematical Programming*, 291–341, 1996.

[19] D. B. Yudin and A. S. Nemirovski, Evaluation of the information complexity of mathematical programming problems, (in Russian), Ekonomika i Matematicheskie Metody 12, 128-142, 1976 (English translation: Matekon 13, 2, 3-45, 1976).