Rescaling in the Perceptron Algorithm for
Solving Linear Programs in Polynomial Time

Olof Barr
Centre for Mathematical Sciences
Lund University
Sweden
barr@maths.lth.se

Oskar Wigelius
Centre for Mathematical Sciences
Lund University
Sweden
oskarw@maths.lth.se

Abstract

We show that a polynomial-time rescaling algorithm, in [5], for finding feasible solutions to linear programs, has flaws in its theoretical framework. With new estimates for the algorithm, we show that it indeed terminates in polynomial time.

Also, we present a modification of the algorithm, where we suggest a data dependent way of choosing a rescaling direction, instead of choosing a vector at random.

1 Introduction

1.1 Linear Programs

Linear programs (LP) arise naturally in many areas. The standard form of LP is

\[
\begin{align*}
\max & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
x & \geq 0
\end{align*}
\]

where \(x, c \in \mathbb{R}^n, b \in \mathbb{R}^m\) and \(A\) is an \(m \times n\) matrix of real numbers. The feasibility version of a standard form LP neglects the objective function \(c^T x\). It can be shown (cf. [7]) that an approximate solution to a LP always can be found by repeatedly solving its feasibility version and performing a binary search. The feasibility standard form of a LP, in its turn, can be reduced to the homogenised form

\[
Ax \geq 0, \quad x \neq 0.
\]

These methods of rewriting standard LP allow for a wide variety of solution techniques for linear programs to exist.

1.2 Different Algorithms, their Complexity and Computational Cost

Popular algorithms for solving standard LP are versions of the simplex algorithm [12]; ellipsoid methods [7]; interior point algorithms [11]; and the perceptron algorithm (with possible modifications) [5]. Which method to choose depends on the parameters of the problem; for example, the simplex and perceptron algorithms cannot be guaranteed to converge reasonably fast (in polynomial time in \(n\) and \(m\)) but nonetheless they are efficient in many practical examples [5].

Regarding the computational complexity of different methods, the following figures can be found in the literature:

- Simplex algorithm: exponential in worst case, but polynomial with high probability [12].
- Interior point methods: \(O(m^3 \log(m/\sigma))\) (smoothed complexity) [11].
- Ellipsoid methods: \(O(mn^3 \langle A, b\rangle)\) where \(\langle A, b\rangle \geq mn\) [7].
- Perceptron algorithm with rescaling: (smoothed complexity [2]) \(O(mn^4 \log(n) \log(1/\rho))\) where \(\rho\) 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 [5]. We show below that there are mistakes in the theoretical framework for this method, but save the method by doing new estimates for the performance.

These figures tell us that the perceptron-rescaling algorithm will be competitive if \(n\) is small compared to \(m\), which is often the case in practice. In [3] it is also shown that the perceptron-rescaling algorithm has good noise tolerance properties. Moreover the algorithm is fairly easy to
implement and requires not as much storage space as other algorithms. These facts give us a reason for studying the perceptron-rescaling algorithm and try to improve its performance.

1.3 The Algorithm of Dunagan & Vempala

In [5] the authors present an algorithm for the homogenised linear feasibility problem. The algorithm is iterative and each iteration consists of three phases, a perceptron phase, a perceptron improvement phase, and a rescaling phase. No matrix inversions and no barrier functions are required.

Algorithm 1.1 (Dunagan & Vempala)

Input: An m \times n matrix \(A\).

Output: A point (vector) \(x\) such that \(Ax \geq 0\) and \(x \neq 0\).

1. Let \(B = I\) and \(\sigma = \frac{1}{2\sqrt{n}}\).

2. (Perceptron)
   a) Let \(x\) be the origin in \(\mathbb{R}^n\).
   b) Repeat at most \(16n^2\) times:
      i) If there exists a row \(a\) such that \(a \cdot x \leq 0\),
         then \(x \leftarrow x + a/\|a\|\).
      ii) If \(x = 0\), go back to step (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 \(\frac{\ln n}{\sigma^2}\) times:
      i) If there exists a row \(a\) such that \(\frac{a \cdot x}{\|a\| \cdot \|x\|} < -\sigma\),
         then \(x \leftarrow x - \frac{a}{\|a\| \cdot \|x\|}\).
      ii) If \(x = 0\), go back to step (a).
   c) If there still exists a row \(a\) such that \(\frac{a \cdot x}{\|a\| \cdot \|x\|} < -\sigma\),
      restart at step (a).
5. If \(Ax \geq 0\), then output \(Bx\) as a feasible solution and stop.

6. (Rescaling)
   Let \(A \leftarrow A(I + \frac{x}{\|x\| \cdot \|x\|}^T)\) and \(B \leftarrow B(I + \frac{x}{\|x\| \cdot \|x\|}^T)\).

7. Go back to step 2.

Dunagan and Vempala use the probability to show that the complexity of the algorithm is polynomial. With new estimates we show that the lesser probability is still enough for the behaviour of the algorithm.

1.4 Reformulation of data

For linearly separable data, the classical perceptron algorithm finds a separating hyperplane. If we assume that we have \(m\) data points in \(\mathbb{R}^n\), each of the data points belonging to two classes and hence labeled 1 or -1, finding such a hyperplane is equivalent to finding a solution

\[
\hat{x} = \begin{bmatrix} x \\ b \end{bmatrix}\quad \text{to } L \cdot (Ax + bE) \geq 0,
\]

where \(A\) is an \(m \times n\)-matrix representing the data points, \(E\) is an \(m \times 1\)-matrix with all entries equal to 1, and \(L\) is an \(m \times 1\)-matrix containing the labels of each of the data points. By letting \(\hat{A} = [A E]\), that is, adding a coordinate to each of the data points, \(\hat{x}\) solves the homogenised LP

\[
L \otimes (\hat{A}\hat{x}) \geq 0.
\]

Finally, by rescaling the data, we see that \(\hat{x}\) is also a solution to \(L \otimes (\hat{A}\hat{x}) \geq 0\), where \(\hat{A}\) is obtained by normalising the rows of \(\hat{A}\) (this corresponds to all data points lying on the unit sphere). Finally, by switching the sign of all data points with negative labels, we have proved that finding a separating hyperplane for a linearly separable two-class dataset is equivalent to finding a vector \(x\) such that \(x \cdot a \geq 0\) if all \(a\) lie on half of a unit sphere in \(\mathbb{R}^n\).

Thus, the problem of finding a separating hyperplane has been reformulated to a feasibility standard form of LP where all rows in the matrix \(A\) are of length one. This is the problem that will be considered in the remainder of this article.

2 The inner product between randomly chosen unit vectors

In the algorithm described above, the key to prove the performance is to find a unit vector \(v\) having inner product with the optimal, but not known, unit vector \(w\) of at least \(\frac{1}{\sqrt{n}}\), with a high probability. The value \(\frac{1}{\sqrt{n}}\) originates from a proof made in [5]. Here it is of great importance to, at first, know the probability of two unit vectors, picked from a uniform distribution, having an inner product of at least \(\frac{1}{\sqrt{n}}\).

First we conclude that we can fix one of the two \(n\)-dimensional vectors, say \(w = e_1 = (1, 0, 0, \ldots, 0)\) which is a vector on the unit sphere. Now, let \(v = (x_1, \ldots, x_n)\) be a randomly chosen unit vector in our \(n\)-dimensional space,
chosen from a uniform distribution on the \(n\)-dimensional sphere.

In simulations where we pick a random vector of length one, with uniform distribution on the unit ball surface, we first construct \(\tilde{v} = (\tilde{x}_1, \ldots, \tilde{x}_n)\), where each \(\tilde{x}_i\) is chosen with the same Gaussian distribution, in our case \(\tilde{x} \in N(0, 1)\). Our vector \(v\) is a scaling of \(\tilde{v}\) such that it has length one, see [9] for more details on the construction of a uniform distribution on the \(n\)-dimensional sphere. The above construction will also be used in the calculations below.

Let \(\Pi_n\) denote the probability that two \(n\)-dimensional vectors have an inner product of at least \(\frac{1}{\sqrt{n}}\). We calculate the probability of having the desired inner product and assume that \(n \geq 2\) since the case \(n = 1\) is trivial.

\[
\Pr(v \cdot w \geq \frac{1}{\sqrt{n}}) = \Pr(x_1 \geq \frac{1}{\sqrt{n}}) = \frac{1}{2} \Pr(x_1^2 \geq \frac{1}{n}) = \ldots
\]

\[
= \frac{1}{2} \Pr\left(\frac{\sum_{i=1}^{n} x_i^2}{n} \geq \frac{1}{n}\right) = \frac{1}{2} \Pr\left(\bar{x}_1^2 \geq \frac{\sum_{i=1}^{n} x_i^2}{n} \right) = \ldots
\]

\[
= \frac{1}{2} \Pr\left(\bar{x}_1^2 \geq \frac{\sum_{i=1}^{n} x_i^2}{n} \right) = \frac{1}{2} \Pr\left(\bar{x}_1^2 \geq \frac{\sum_{i=1}^{n} \tilde{x}_i^2}{n-1}\right).
\]

This can be used for calculating the probability \(\Pi_n\) when \(n\) tends to infinity. We get the following proposition:

**Proposition 2.1**

\[
\lim_{n \to \infty} \Pi_n = \frac{1}{2} (1 - \text{erf}(\frac{1}{\sqrt{2}}))
\]

**Proof:** When \(n\) tends to infinity we get that the distribution

\[
\frac{\sum_{i=1}^{n} x_i^2}{n-1}
\]

tends to 1, having zero variance, and so,

\[
\lim_{n \to \infty} \Pi_n = \frac{1}{2} \Pr\left(\bar{x}_1^2 \geq 1\right).
\]

\(\bar{x}_1^2\) is \(\chi^2(1)\)-distributed, hence

\[
\lim_{n \to \infty} \Pi_n = \frac{1}{2} \int_1^{\infty} \frac{1}{\sqrt{2\pi}} \frac{e^{-\frac{9}{2}}}{\sqrt{x}} dx = \ldots = \frac{1}{2} (1 - \text{erf}(\frac{1}{\sqrt{2}})).
\]

Moreover, there is an easy way of calculating this probability, which also gives us an analytical expression for every \(n\). By elementary geometry, we get that the probability can be expressed by considering sectors of the unit sphere in \(\mathbb{R}^n\). The surface element \(dS\) on this hypersphere is given by

\[
dS = C_{n-1} \sqrt{(1 - x^2)^{(n-3)}} dx,
\]

where \(C_{n-1}\) is the area of the \((n-1)\)-dimensional unit sphere. We get the following theorem:

**Theorem 2.2.**

\[
\Pi_n = \Pr(v \cdot w \geq \frac{1}{\sqrt{n}}) = \frac{\int_1^{\frac{1}{\sqrt{n}}} \sqrt{(1 - x^2)^{(n-3)}} dx}{\int_{-1}^{1} \sqrt{(1 - x^2)^{(n-3)}} dx}.
\]

Both integrals can be calculated by using the substitution \(x = \cos t\).

We list the first known exact values and present their approximations for a few finite cases. A proof showing that \(\Pi_n\) always is greater than \(\frac{1}{2}(1 - \text{erf}(1/\sqrt{2}))\) will be included in 2.1.

<table>
<thead>
<tr>
<th>(n)</th>
<th>(\Pi_n)</th>
<th>appr. value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1/2)</td>
<td>0.5000…</td>
</tr>
<tr>
<td>2</td>
<td>(1/4)</td>
<td>0.2500…</td>
</tr>
<tr>
<td>3</td>
<td>(1/3 - \sqrt{3}/(4\pi))</td>
<td>0.2113…</td>
</tr>
<tr>
<td>4</td>
<td>(1/3 - \sqrt{3}/(4\pi))</td>
<td>0.1955…</td>
</tr>
<tr>
<td>5</td>
<td>(1/2 - 7/(5\sqrt{5}))</td>
<td>0.1870…</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>(\infty)</td>
<td>(1/2(1 - \text{erf}(1/\sqrt{2})))</td>
<td>0.1587…</td>
</tr>
</tbody>
</table>

Here we see that the earlier proof of Algorithm 1.1 in [5] will fail for \(n \geq 3\). This is the case since they need a probability of 1/4 in the estimations made in the proof. Below we will prove that the lower, but correct, probability is enough for proving the algorithm.

**2.1 \(\Pi_n\) is always greater than \(\frac{1}{2}(1 - \text{erf}(1/\sqrt{2}))\)**

In order to show that \(\Pi_n\) always is big enough for our purposes, we first show the following:

**Theorem 2.3 \(\Pi_n\) is decreasing.**

**Proof:** Define \(I_n\) and \(J_n\) by

\[
I_n = \int_1^{\frac{1}{\sqrt{n}}} \sqrt{(1 - x^2)^{(n-3)}} dx,
\]

\[
J_n = \int_{-1}^{1} \sqrt{(1 - x^2)^{(n-3)}} dx.
\]

Then \(\Pi_n = \frac{I_n}{J_n}\) and we want to show that

\[
\frac{I_n}{J_n} \geq \frac{I_{n+1}}{J_{n+1}},
\]

which is equivalent to

\[
\frac{I_n}{I_{n+1}} \geq \frac{J_n}{J_{n+1}}.
\]
By the substitution \( z = \cos t \), we want to show the inequality

\[
\int_0^{\arccos \frac{1}{\sqrt{n}}} \sin^{n-2} t \, dt \leq \int_0^{\pi/2} \sin^{n-2} t \, dt.
\]

The right-hand side has the following upper estimate:

\[
\frac{\int_0^{\pi/2} \sin^{n-2} t \, dt}{\int_0^{\pi/2} \sin^{n-1} t \, dt} = \frac{n}{n-1}
\]

(the last equality can be found in e.g. [6]). Now we want to find a lower estimate of the left-hand side, or an upper estimate to the inverse of it. For convenience in notation, we let \( a = \arccos \frac{1}{\sqrt{n}} \), \( b = \arccos \frac{1}{\sqrt{n+1}} \), and proceed with the Left Hand Side (LHS):

\[
(LHS)^{-1} = \frac{\int_a^b \sin^{n-1} t \, dt}{\int_0^{\pi/2} \sin^{n-2} t \, dt} = \frac{\int_a^\alpha \sin^{n-1} t \, dt + \int_\alpha^b \sin^{n-1} t \, dt}{\int_0^{\pi/2} \sin^{n-2} t \, dt}
\]

But \( \int_a^b \sin^{n-1} t \, dt \leq (\int_a \sin^{n-2} t)(\int_a \sin t) \) if \( \sin t \geq 1 \) on \( I \) (by Cauchy-Schwarz’ inequality), and hence

\[
\alpha \leq \int_0^a \sin t \, dt = [-\cos t]_0^{\arccos \frac{1}{\sqrt{n}}} = 1 - \frac{1}{\sqrt{n}}.
\]

Now, we use Cauchy-Schwarz’ inequality once more to estimate the nominator of \( \beta \):

\[
\int_a^b \sin^{n-1} t \, dt \leq (\int_a \sin t \, dt)^{n-1} = (\frac{1}{\sqrt{n}} - \frac{1}{\sqrt{n+1}})^{n-1} = (\frac{n + 1 - n}{\sqrt{n} \sqrt{n+1}})^{n-1} \leq \left( \frac{2n^3/2}{n} \right)^{n-1} = \gamma.
\]

Proceeding with the denominator, we have

\[
\int_0^a \sin^{n-2} t \, dt \geq \int_0^{\pi/2} \left( \frac{3\sqrt{3}}{2\pi} \right)^{n-2} \, dt \geq \int_0^{\pi/2} \left( \frac{3\sqrt{3}}{2\pi} \right)^{n-1} \, dt = [\left( \frac{3\sqrt{3}}{2\pi} \right)^{n-1} t^{n-1} / n]_0^{\pi/2} = \frac{\pi}{3} \left( \frac{3\sqrt{3}}{2} \right)^{n-1} \frac{1}{n} = \delta
\]

where the first inequality is valid when \( \arccos \frac{1}{\sqrt{n}} \geq \frac{\pi}{6} \), that is, when \( n \geq 4 \). Now we can estimate \( \beta \):

\[
\beta = \gamma \frac{1}{\delta} = \frac{3n^2}{\pi \sqrt{3} \left( \frac{3n^3/2}{n} \right)^{n-1} \leq \frac{n}{n(3n^3/2)\delta} = \frac{1}{n}
\]

for all \( n \geq 4 \). Summing all this up, we have found that

\[
(LHS) = \frac{1}{\alpha + \beta} \geq \frac{1}{1 - \frac{1}{\sqrt{n}} + \frac{1}{n}}
\]

and if we can show that

\[
1 - \frac{1}{\sqrt{n}} + \frac{1}{n} \geq \frac{n}{n - 1}
\]

for all \( n \geq 4 \) we are finished. But this is the same as showing

\[
n - 1 \geq n - \sqrt{n} + 1 \Leftrightarrow \sqrt{n} \geq 2,
\]

which is obviously valid for all \( n \geq 4 \).

Summing up facts concerning \( \Pi_n \), we now know the first few values, the value when \( n \) tends to infinity and that \( \Pi_n \) is a decreasing serie. We get the following corollary:

**Corollary 2.4** For every \( n \),

\[
\Pi_n \geq \frac{1}{2} (1 - \text{erf}(1/\sqrt{2})).
\]

### 3 New Estimates for Proving the Algorithm by Dunagan & Vempala

Algorithm 1.1 has two main ingredients: the classical perceptron algorithm and a rescaling step. The convergence speed of the perceptron algorithm depends on the so called radius \( \rho \) of the dataset \( \mathcal{A} = \{a_i\}_{i \leq \varepsilon m} \), defined by

\[
\rho = \max_{z \in \mathbb{R}^n} \min_{a \in \mathcal{A}} \frac{a \cdot z}{\|a\|}.
\]

The rescaling step is used to increase \( \rho \), at least its expectation: by choosing a random vector in (4.a), the rescaling step increases the radius with probability at least \( \Pi_n \), but decreases it with probability at most \( 1 - \Pi_n \).

Here we present Lemma 3.1. from [5] with the incorrect values of \( \Pi_n \). Afterwards we show that we can save the proof by making better estimates in the calculations. These new estimates give new bounds on variables in the theorems using this lemma, but these theorems can nevertheless be reformulated in view of our comment. In the notation below: \( \hat{x} \) denotes the unit vector with the same direction as the vector \( x \).

**Lemma 3.1.** (by Dunagan and Vempala)
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 \( \rho' \) and \( \rho \) be the radii of \( A' \) and \( A \) respectively. Then,
(a) \( \rho' \geq (1 - \frac{1}{32n})\rho \).
(b) With probability at least \( \frac{1}{4} \), \( \rho' \geq (1 + \frac{1}{32n})\rho \).

Proof (by Dunagan and Vempala):

Let \( a_i, i = 1, \ldots, m \) be the rows of \( A \) at the beginning of some iteration. Let \( z \) be the unit vector satisfying \( \rho = \min_j a_j \cdot z \), and let \( \sigma_i = a_i \cdot x \). After a perceptron improvement phase, we get a vector \( z \) such that
\[
\sigma_i \cdot \bar{x} = \sigma_i \geq -\sigma.
\]
As in the theorem statement (see [5]), let \( A' \) be the matrix obtained after the rescaling step, i.e.
\[
a' = a_i + (a_i \cdot x)\bar{x}.
\]
Finally, define
\[
z' = z + \alpha(z \cdot \bar{x})\bar{x},
\]
where \( \alpha \) will be specified shortly. Although \( z' \) is not necessarily the center of \( A' \), \( \rho' \) is a maximum over a set, and so considering one element \( (z') \) of the set suffices to lower bound \( \rho' \). We have
\[
\rho' \geq \min_i \bar{a}_i \cdot \bar{z}' = \bar{a}_i \cdot \bar{z}' \frac{|z'|}{|z'|}.
\]
We will first prove that \( \bar{a}_i \cdot \bar{z}' \) cannot be too small.
\[
\begin{align*}
\bar{a}_i \cdot \bar{z}' &= \frac{\bar{a}_i + (\bar{a}_i \cdot \bar{x})\bar{x}}{|\bar{a}_i + (\bar{a}_i \cdot \bar{x})\bar{x}|} \\
&= \frac{[\bar{a}_i + (\bar{a}_i \cdot x)\bar{x}] [z + \alpha(z \cdot \bar{x})\bar{x}]}{\sqrt{1 + 3(a_i \cdot x)^2}} \\
&\geq \frac{\rho + \sigma_i(z \cdot \bar{x})(1 + 2\alpha)}{\sqrt{1 + 3\sigma_i^2}}
\end{align*}
\]
We choose:
\[
\alpha = \frac{1}{2} \left( \frac{\rho}{|x \cdot z|} - 1 \right)
\]
so that \( 2\alpha + 1 = \rho/(|x \cdot z|) \). We have not ensured that \( x \cdot z \neq 0 \), but substituting in for \( \alpha(x \cdot z) \) in the definition of \( z' \) using the value we have chosen for \( \alpha \) would remove this boundary case. We have chosen not to do this to aid the exposition. We proceed to calculate
\[
\begin{align*}
\bar{a}_i \cdot \bar{z}' &\geq \rho \frac{1 + \sigma_i}{\sqrt{1 + 3\sigma_i^2}} \geq \frac{1 - \sigma}{\sqrt{1 + 3\sigma_i^2}}
\end{align*}
\]
where the second inequality follows from \( \sigma_i \in [\sigma, 1] \). Next, observe that
\[
|z'|^2 = 1 + (\alpha^2 + 2\alpha)(x \cdot z)^2 = 1 + \frac{\rho^2}{4} + (z \cdot \bar{x})(\frac{\rho}{2} - \frac{3}{4}(z \cdot \bar{x})).
\]
We consider two cases:
1. \( |z \cdot x| < \frac{1}{\sqrt{n}} \). This happens with probability at most \( 3/4 \).
   Viewing \( |z'|^2 \) as a quadratic polynomial in \( (z' \cdot x) \), we see that it is maximised when \( (z' \cdot x) = \frac{\rho}{2} \). 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 \( \frac{1}{1+\beta} \geq 1 - \frac{\beta}{2} \) for every \( \beta \in (-1, 1) \), we find
   \[
   \rho' \geq \rho \frac{1 - \frac{1}{32n}}{\sqrt{1 + \frac{\rho^2}{2} + \frac{\rho^2}{12}}} \geq \rho(1 - \frac{1}{16n}).
   \]
2. \( |z \cdot x| \geq \frac{1}{\sqrt{n}} \). This happens with probability at least \( 1/4 \). In this case
   \[
   |z'|^2 = 1 + \frac{\rho^2}{4} + (z \cdot x)(\frac{\rho}{2} - \frac{3}{4}(z \cdot x))^2 \leq 1 + \frac{1}{64n^2} + \frac{1}{8n\sqrt{n}} - \frac{3}{4n} \leq 1 - \frac{39}{64n}.
   \]
   Using the same identity as above, we find
   \[
   \rho' \geq \rho \frac{1 - \frac{1}{32n}}{\sqrt{1 + \frac{\rho^2}{2} + \frac{\rho^2}{12}}} \geq \rho(1 - \frac{1}{4n}).
   \]
This proves both parts of the lemma.

Now, these estimates above are incorrect when the correct probabilities are taken in account. But using the second last estimates in the calculations yield the following lemma:

Lemma 3.2 With new estimates taking care of the correct probabilities of \( \Pi_n \), Lemma 3.1 is true.

Proof: When \( \rho \) is increased by rescaling, it is multiplied by a factor \( C^+ \) such that
\[
C^+ \geq (1 - \frac{1}{32n})(1 - \frac{3}{2048n^2})(1 + \frac{39}{128n}),
\]
and when decreased it is multiplied by a factor \( C^- \) which satisfies the inequality
\[
C^- \geq (1 - \frac{1}{32n})(1 - \frac{3}{2048n^2})(1 + \frac{1}{96n^2}).
\]
For $\rho$ to increase in the mean, we want to have
\[
\Pi_n C^+ + (1 - \Pi_n) C^- > 1
\]
for all $n$. In [5], the authors assume $\Pi_n > \frac{1}{2}$ and hence they use crude estimates for the constants $C^-$ and $C^+$ to prove that the last inequality is satisfied. If we instead use the given expressions, we would like to show that
\[
\Pi_n \geq \frac{4(196008n^4 + 74752n^3 - 23360n^2 - 96n + 3)}{(2048n^2 - 3)(117n + 4)(32n - 1)}.
\]
By differentiating the right hand side and finding its zeros (which we have done with the aid of MAPLE 8), we find that it is strictly decreasing for $n \geq 1$, and we can check that for $n = 1$ the expression equals approximately 0.1402. But in the previous chapter we found that $\Pi_n \geq \frac{1}{2}(1 - \text{erf} \frac{1}{\sqrt{2}}) \approx 0.1587$, so the proof for Algorithm 1.1 is still valid. Although, the theorems using this lemma to calculate the complexity of the algorithm must be adjusted, fortunately only with a constant, leaving the same order of complexity as before.

\section{Improving the Rescaling Step}

\subsection{Finding an Initial Vector}

In Algorithm 1.1, the steps (4.a-c) are intended to produce a direction for rescaling of the data. This is done by choosing an initial vector $w_0$ at random, and then “wiggle” it a number of times until it, hopefully, points in the right direction. This improves the perceptron phase, at least in the mean, but an actual improvement is only reached with a probability of at least $\Pi_{n+1}$. To increase this probability of success, we suggest to let the $w_0$ depend on data in the following way:

"Pick a random rotation $Q$ of $\mathbb{R}^n$ from a uniform distribution and consider the rotation $A Q$ of all $n$ unit vectors in $A$. Look for the element in $A Q$ with largest absolute value. The column decides which of the $n$ dimensions we are aiming at and the sign of the element decides the direction. Retruncate $v$ to get the vector $w_0$, that is $w_0 = v Q^T$."

There are of course numerous of possible ways of using the dataset to pick an initial vector, but we will concentrate on this method and present some arguments for its performance.

If the initial vector $w_0$ is such that its scalar product with the optimal normal $w$ satisfies $w_0 \cdot w \geq 1/\sqrt{n}$, the rescaling will be successful. Now, we can reduce the number of vectors to choose from to $2n$ different vectors in $\mathbb{R}^n$. This set of vectors is
\[
V = \bigcup_{i=1}^{n} \{e_i - e_i \}
\]
that is $(1,0,\ldots,0), (-1,0,\ldots,0), (0,1,0,\ldots,0)$ and so on. At least one of these vectors have an inner product of at least $\frac{1}{\sqrt{n}}$ with the optimal vector $w$, according to the following proposition:

\begin{proposition}
For any unit vector $w \in \mathbb{R}^n$,
\[
\max_{v \in V} w \cdot v \geq \frac{1}{\sqrt{n}}.
\]
\end{proposition}

\begin{proof}
Let $w = (w_1, \ldots, w_n)$. At least one $w_i$ must have an absolute value of at least $\frac{1}{\sqrt{n}}$, otherwise $\|w\| < 1$. This yields that there exists at least one vector $v \in V$ such that $w \cdot v \geq \frac{1}{\sqrt{n}}$ as stated above. \end{proof}

In practice this means that it is sufficient for our purposes to find a good initial vector $w_0 \in V$ with a probability greater than choosing $w_0$ at random. Since we do not know anything about the optimal normal $w$ and we are going to perform our calculations a number of times, we would like to have the possibility of having the normal $w$ situated anywhere on the unit sphere with uniform distribution. Therefore we choose to, before every choice of a stretching vector, rotate the data set with a randomly chosen (from a uniform distribution) rotation. We don’t have to bother if the data set also is mirrored at the same time. In view of these thoughts we also like to conclude the following:

\begin{proposition}
The probability that a randomly chosen $n$-dimensional vector $v \in V$, after a randomly chosen rotation of data, fulfills $w \cdot v \geq \frac{1}{\sqrt{n}}$ is $\Pi_n$.
\end{proposition}

\begin{proof}
The chosen vector is, due to the random rotation, chosen with uniform distribution among all $n$-dimensional unit vectors. \end{proof}

By the following theorem, our method finds $\tilde{\theta} \in V$ which has maximal scalar product with $w$, and hence a sufficiently good initial vector $w_0 = \tilde{\theta}$, if the set of support vectors is the whole of a circle.

\begin{theorem}
Let $Y = \{y \in \mathbb{R}^n : \|y\| = 1, y \cdot w = \varepsilon > 0\}$. If we define $\tilde{\theta} \in V$ by
\[
\tilde{\theta} = \{v \in V \text{ that maximises } \min_{y \in Y} v \cdot y\}
\]
and $\tilde{\theta}$ by
\[
\tilde{\theta} = \{v \in V \text{ that maximises } v \cdot w\},
\]
then $\tilde{\theta} = \tilde{\theta}$. \end{theorem}

\begin{proof}
Define $Y' = \{y' : y' = y - \varepsilon w, y \in Y\}$. Then $Y' \perp w$. For every $v \in V$, it holds that
\[
(v \cdot w)^2 + (v \cdot \frac{y'}{\|y'\|})^2 \leq 1
\]

with equality exactly when the product \( v \cdot \frac{y'}{||y'||} \) is maximised or minimised, since \( y' \in Y' \Rightarrow -y' \in Y' \). (This maximum or minimum occurs when \( v \) is a linear combination of \( y' \) and \( w \). There must be a minimising \( y' \), since \( Y' \) is compact.) But minimising \( v \cdot \frac{y'}{||y'||} \) is equivalent to minimising \( v \cdot y' \), since \( ||y'|| \) is constant (and equal to \( \sqrt{1-\varepsilon^2} \)), and this in turn is equivalent to minimising \( v \cdot y \) since

\[
v \cdot y = v \cdot (\varepsilon w + y') = \varepsilon v \cdot w + v \cdot y'.
\]

Thus, we have

\[
(v \cdot w)^2 + (\min_{y'} v \cdot \frac{y'}{||y'||})^2 = 1.
\]

Now, it is clear that \( \hat{v} \) maximises \( v \cdot w \) over all \( v \in V \Rightarrow \) it maximises \( (v \cdot w)^2 \) over all \( v \in V \Rightarrow \) it minimises \( (\min_{y'} v \cdot \frac{y'}{||y'||})^2 \) over all \( v \in V \Rightarrow \) it minimises \( (\min_{y'} v \cdot \frac{y'}{||y'||}) \) over all \( v \in V \). Thus \( \hat{v} \leq \hat{v} \). By reversing the order of this chain, we see that \( \hat{v} = \hat{v} \).

4.2 Experimental Validation and Further Improvements

Theorem 4.3 is not as useful as one could wish, since the assumptions made are never fulfilled in practice. However, in the following way:

1. Generate a dataset with specified margin \( \varepsilon \), dimension \( n \) and number of points \( m \) and calculate the optimal normal \( w \).
2. Repeat a number of times: Choose a random ON-basis for \( \mathbb{R}^n \), determine the stretching vector \( w_0 \) and calculate the scalar product \( w_0 \cdot w \) in this basis.
3. Determine \( P \), that equals the percentage of trials for which \( \hat{v} \cdot w \geq 1/\sqrt{n} \).

As one could suspect, our experiments indicate that \( P \) for a certain dataset depends mainly on the configuration of data-points, and not very much on \( \varepsilon \), \( n \) or \( m \). This makes it hard to generate ill-conditioned datasets by random generation of datasets. We have, however, not been able to produce a dataset for which our proposed method of choosing a rescaling vector has a probability \( P \) of less than 25%.

Our method seems to work best for datasets that are less sparse. Thus, a possible way of improving the performance of our method is to add points to the data set in the following way: Let \( Y = \{y_1, \ldots, y_n\} \) be the original data set. Since \((y_i + y_j)/\|y_i + y_j\|\) will lie on the right side of the unit sphere, we can construct a larger data set by

\[
\hat{Y} = Y \cup \{y_{ij} = (y_i + y_j)/\|y_i + y_j\|; y_i, y_j \in Y\},
\]

and hopefully create a data set where our algorithm performs even better. We hope to investigate this idea further in the future. Also, we will try to compare the complexity of Algorithm 1.1 in its present form with a version where Step (4.a) has been modified according to our suggestion.

Acknowledgement

We thank the Swedish Research Council for financial support, thank you also to Amirian Ambroladze and Anna Torstensson for helpful discussions and encouragement.

References


