Teaming interior and combinatorial algorithms for linear programming.

Adrien Chan-Hon-Tong

Teaming interior and combinatorial algorithms for linear programming.

Adrien CHAN-HON-TONG

July 17, 2020

Abstract

This paper introduces several algorithms for linear programming which verifies 3 properties: they have complementary times complexity features (one being polynomial times), they can exchange their current points, and, they work with related goal. It seems to be the first portfolio of algorithm verifying such properties.

1 Introduction

Linear programming is the very studied task of optimizing a linear criterion under linear equality and inequality constraints: the canonical form of linear program is thus

$$\max_{x \ / \ Ax=b, x \geq \mathbf{0}} c^T x \tag{1}$$

with A a matrix, b, c two vectors. This problems has been first tackled by exponential algorithms like simplex [3]. The first polynomial time algorithm solving this problem is the ellipsoid method [5, 4]. Today, the state of the art of polynomial time algorithm for linear programming is central-path log-barrier algorithm [7, 6] (despite the recent Chubanov method [2] could be a challenger).

motivation:

Those different algorithms have different time complexity features: for example, simplex is strongly polynomial times for some families of instances (e.g. markov chain [8]) despite being exponential times in general, while log barrier is polynomial times in general but not known strongly polynomial even for relevant families (it is known not to be strongly polynomial times in general [1]). Thus teaming them could be relevant.

However, simplex, ellipsoid and log barrier algorithms can not straightforwardly help each other. Let imagine that one solves simultaneously a common linear program with an ellipsoid, a simplex and a log barrier algorithm. Yet, the internal state of ellipsoid is not just a point x_{elli} but mainly the ellipsoid which bound the feasible space. So, it is impossible to swap x_{elli} with (for example)

$x_{simplex}$ as the related ellipsoid is not known (and may hardely be smaller). Similarly, classical log barrier method consists in minimizing some function $G(x, \rho)$ with x while increasing ρ (because, on one hand, the x-minimum of $G(x, \rho)$ is a solution of the linear program only for high ρ, but, on the other hand, ρ should not be too high to maintain x in the quadratic convergence area of the Newton descent). So, this dependency in ρ makes it hard to know if it is relevant to swap $x_{interior}$ with x_{elli} or $x_{simplex}$ at a given running times.

Now, one could wonder if it is possible to adapt and embed some of these algorithms into a space where they could help each other. This paper offers such framework and 3 algorithms with complementary times complexity features, possibility to exchange their current states, and, similar goals.

Claim:

Precisely, this paper introduces 3 algorithms, each of them takes $AA^T \in \mathbb{Z}^{N \times N}$ as input, and, each of them finds $v > \mathbf{0}$ such that $AA^T v > \mathbf{0}$. Let recall that it is a common knowledge that given an algorithm that finds $v > \mathbf{0}$ such that $AA^T v \geq \mathbf{0}$ (when it exists), one could build an algorithm for generic linear programming (see appendix A).

Importantly, they have a common internal state $v \in \mathbb{Q}^N$ (with simple normalization rules allowing to compare two current points to check if swapping is relevant). And, all their goal are linked to minimize $v^T AA^T v$ (more or less explicitly, with different constraints), making swapping potentially relevant.

Those 3 algorithms are:

1. *Log barrier minimization of* $v^T AA^T v$: this algorithm is an interior-point algorithm, it consists to use Newton descent on $\min\limits_{v} \frac{1}{2} v^T AA^T v - \mathbf{1}^T \log(v)$. It produces $v > \mathbf{0}$ such that $AA^T v > \mathbf{0}$ in a polynomial number of steps (precisely, if $x = \underset{z \ / \ Az \geq \mathbf{1}}{\arg\min}\ z^T z$, then, $\widetilde{O}(N \log(x^T x \times \mathbf{1}^T AA^T \mathbf{1})$ Newton step are required i.e. $\widetilde{O}(N^2 L)$ with N the number of rows of A and 2^L bounding the numbers of A).

2. *Constrained minimization of* $v^T AA^T v$: this algorithm is a combinatorial: current state can be matched with a subset $\{1, ..., N\}$). It consists in solving $\min\limits_{v \geq \mathbf{1}} v^T AA^T v$. Let $\mathcal{R} = \{I \subset \{1, ..., N\} \ / \ \min\limits_{v \geq \mathbf{1}, v_I = \mathbf{1}} v^T AA^T v = \min\limits_{v_I = \mathbf{1}} v^T AA^T v\}\}$, the algorithm does not require more than $\widetilde{O}(N|\mathcal{R}|)$ steps (despite $|\mathcal{R}|$ could be as high as 2^N, this algorithm is strongly polynomial on families with bounded $|\mathcal{R}|$).

3. *Implicit minimization of* $v^T AA^T v$: this algorithm is very different because it uses recursion and seeks to solve $\max\limits_{v \geq \mathbf{0}, \mathbf{1}^T v = 1} \min\limits_{n} A_n A^T v$. Yet, by construction, it maintains $v > 0$ and decreases $v^T AA^T v$ at each iteration. Let $\mathcal{S} = \{I \subset \{1, ..., N\} \ / \ \exists v > \mathbf{0} / I = \underset{n}{\arg\min}\ A_n A^T v\}$, then, this algorithm does not perform more than $|\mathcal{S}|$ recursive calls, thus complexity is linked

to $factorial(|\mathcal{S}|) = |\mathcal{S}| \times (|\mathcal{S}| - 1) \times (|\mathcal{S}| - 2) \times ... \times 1$ (for example, it is strongly polynomial if $|\mathcal{S}|$ is $O(1)$).

2 Log barrier minimization of $v^T AA^T v$

This section proves that if exists x / $Ax > \mathbf{0}$, then approximating the minimum of

$$F(v) = \frac{1}{2} v^T AA^T v - \mathbf{1}^T \log(v) \tag{2}$$

allows to build a solution of

$$find\ v > \mathbf{0}\ such\ that\ AA^T v > \mathbf{0}\ (which\ exists\ when\ x\ exists) \tag{3}$$

2.1 Property of the minimum

Let A a matrix with N rows ($AA^T \in \mathbb{Z}^{N \times N}$), and, the prior that exists x with $Ax \geq \mathbf{1}$.

For all $v \geq \mathbf{0}$, $\mathbf{1}^T v \leq (Ax)^T v = x^T (A^T v) \leq \sqrt{x^T x \times v^T AA^T v}$ (last inequality is Cauchy). Thus, $v^T AA^T v \geq \frac{1}{xx} (\mathbf{1}^T v)^2 \geq \frac{1}{xx} v^T v$ (as $v \geq \mathbf{0}$).

Now, let consider the functions $F(v) = \frac{1}{2} \times v^T AA^T v - \sum_n \log(v_n)$ (range is always $\{1, ..., N\}$ when omitted) and $f(t) = \frac{1}{2x^T x} t^2 - \log(t)$. Using previous inequality, it stands that $F(v) \geq \sum_n f(v_n)$.

Now, $f(t) \underset{t \to 0\ or\ \infty}{\to} \infty$ because $-\log(t)$ is dominated by t^2 in ∞ and the opposite in 0. Thus, f is lower bounded i.e. has a minimum f^*. As f is smooth, this minimum is a zero of the derivative: $f'(t) = \frac{1}{x^T x} t - \frac{1}{t}$ i.e. $t^* = \sqrt{x^T x}$ and $|f^*| \leq \frac{1}{2} - \frac{1}{2} \log(x^T x)$. So, $|f^*| \leq \log(x^T x)$ (assuming it is higher than 1). Let also stress that $f''(t) = \frac{1}{x^T x} + \frac{1}{t^2} > 0$ i.e. f is convex.

Now as f has a minimum, F has one too with $|F^*| \leq N \log(x^T x)$, and, as F is also smooth, this minimum is a zero of the gradient: $\nabla_{v^*} F = AA^T v^* - \frac{1}{v^*} = \mathbf{0}$. So, the minimum of F verifies $AA^T v^* = \frac{1}{v^*} > \mathbf{0}$ i.e. this is a solution of eq.(3).

Minimizing F eq.(2) allows to solve $AA^T v > \mathbf{0}, v > \mathbf{0}$ i.e. eq.(3).

2.2 Property of approximate minimums

Now, how close one should be to have a solution of eq.(3) ? Let before state 4 trivial lemmas:

1. $\phi(t) = \frac{1}{2}\alpha t^2 - \log(1 + t) \leq \frac{1}{2}(\alpha + 1)t^2 - t = \psi(t)$
2. $\psi(\frac{1}{\alpha+1}) \leq -\frac{1}{2\alpha+2}$ (thus $\phi(\frac{1}{\alpha+1}) \leq -\frac{1}{2\alpha+2}$ from lemma 1)
3. $\varphi(t) = \frac{t}{2} - \log(t)$ is positive for $t \geq 2$
4. $f(t) \leq \Omega \Rightarrow t \leq x^T x + 2\Omega + 2$

First, $\psi'(t) - \phi'(t) = (\alpha+1)t - 1 - \alpha t + \frac{1}{1+t} = t - 1 + \frac{1}{1+t} = \frac{t^2}{1+t} > 0$, so $\psi(t) - \phi(t)$ always increase. But, $\psi(0) = \phi(0) = 0$ so $\psi(t) \geq \phi(t)$. Second, $\psi(\frac{1}{\alpha+1}) = \frac{1}{2}(\alpha+1)\frac{1}{(\alpha+1)^2} - \frac{1}{\alpha+1} = -\frac{1}{2\alpha+2}$. Three, $\varphi'(t) = \frac{1}{2} - \frac{1}{t}$ so φ increases for $t > 2$ but $\varphi(2) = 1 - \log(2)$ as $\log(2) < 1$, $\varphi(2) > 0$ and thus $\varphi(t) > 0$ for $t > 2$. Four, f is convex so $f(t + x^T x + 2) \geq f(x^T x + 2) + tf'(x^T x + 2)$, but, $f(x^T x + 2) \geq \frac{1}{2}(x^T x + 2) - \log(x^T x + 2) \geq 0$ (because $\frac{x^T x + 2}{x^T x} > 1$ and lemma 3). And, $f'(x^T x + 2) \geq 1 - \frac{1}{x^T x + 2} \geq \frac{1}{2}$. So $f(t + x^T x + 2) \geq \frac{t}{2}$. Thus, $f(2\Omega + x^T x + 2) \geq \Omega$, so, lemma 4.

Then, the key point of the minimization of F is that: $F(v) \leq F(v_0) \Rightarrow f(v_n) \leq F(v_0) + N|f^*|$ because $f(v_n) + (N-1)f^* \leq \sum_n f(v_n) \leq F(v)$. Let $\Gamma = \max_n A_n A_n^T$ and $v_0 = \frac{1}{\sqrt{\Gamma}}\mathbf{1}$, then $F(v_0) \leq N^2 + N\log(\Gamma)$. And, using lemma 4, $F(v) \leq F(v_0)$ implies $f(v_n) \leq F(v_0) + N|f^*|$ which bounds all v_n by $\Theta = 2N^2 + 2N\log(\Gamma) + 2N\log(x^T x) + x^T x + 2$.

Now, if $F(v) \leq F(v_0)$ and $A_k A^T v \leq 0$, let define $\delta = \frac{1}{4\Gamma\Theta^2+2}$ and w such that $\forall n \neq k, w_n = v_n$ and $w_k = (1+2\delta)v_k$, then $F(w) \leq F(v) + 2A_k A^T v \times \delta + 2v_k A_k A_k^T v_k \times \delta^2 - \log(1+2\delta)$. Then, using the fact that $A_k A^T v \leq 0$, it stands that $F(w) \leq F(v) + \frac{1}{2}v_k A_k A_k^T v_k (2\delta)^2 - \log(1+2\delta)$. Using the bound on v, it stands that $F(w) \leq F(v) + \frac{1}{2}\Gamma\Theta^2(2\delta)^2 - \log(1+2\delta)$. Then, as expected 2δ has been selected such that lemma 2 can be applied, thus $F(w) < F(v) - \delta$. This implies that $F(v) > F^* + \delta$ because $F(w)$ can not be lower.

$$\forall v \in \mathbb{Q}^N,\ F(v) \leq F(v_0) \text{ and } F(v) \leq F^* + \delta \Rightarrow AA^T v > \mathbf{0}$$
$$\text{(with } \delta = \tfrac{1}{4\Gamma\Theta^2+2},\ \Theta = 2N^2 + 2N\log(\Gamma) + 2N\log(x^T x) + x^T x + 2,$$
$$\Gamma = \max_n A_n A_n^T,\ v_0 = \tfrac{1}{\sqrt{\Gamma}}\mathbf{1} \text{ and } x \text{ such that } Ax \geq \mathbf{1})$$

2.3 Complexity of Newton descent on F

Let now assume that $\forall i, j,\ A_{i,j} \leq 2^L$. Then, $\Gamma \leq D2^{2L}$ (D is the number of variables of the original problem which has only logarithmic impact here), so $N\log(\Gamma) = \widetilde{O}(NL)$. Also, there is x such that $Ax \geq \mathbf{1}$ and $x^T x \leq N2^{2NL}N^{2N}$ (see appendix B for precision). So $\log(x^T x) = \widetilde{O}(NL)$ Finally, $\log(\Theta) = \widetilde{O}(NL)$ ($x^T x$ dominates all other term in $N^2 + N\log(\Gamma) + N\log(x^T x) + x^T x$), and, thus $\log(\frac{1}{\delta}) = \widetilde{O}(NL)$.

Now, F is self concordant as sum of self concordant function ($-\log$ is and semi definite quadratic term is). So, Damped Newton descent [6]

$$v = v - \frac{1}{1 + \sqrt{(\nabla_v F)^T (\nabla_v^2 F)^{-1} (\nabla_v F)}} (\nabla_v^2 F)^{-1} (\nabla_v F) \tag{4}$$

starting from v_0 allows to reach v such that $F(v) \leq F^* + \delta$ in $O(F(v_0) - F^* + \log\log(\frac{1}{\delta}))$ Newton steps. Here $\log\log(\frac{1}{\delta})$ is completely negligible, only count $F(v_0) + |F^*| = \widetilde{O}(N\log(\mathbf{1}^T AA^T \mathbf{1} \times x^T x)) = \widetilde{O}(N^2 L)$. Thus starting from v_0, reaching v such that $F(v) \leq F^* + \delta$ can be done in $\widetilde{O}(N^2 L)$ Newton steps.

Importantly, preprocessing A (e.g. normalizing rows) has no impact on the complexity as $\mathbf{1}^T AA^T \mathbf{1} \times x^T x$ is constant when scaling A.

Damped Newton descent eq.(4) allows to minimize sufficiently F eq.(2) and thus to find $v > \mathbf{0}$ such that $AA^T v > \mathbf{0}$ eq.(3) in a polynomial number of Newton steps.

Importantly, during a step, using infinite precision representation of numbers (e.g. fraction with infinite length integer) is possible allowing to avoid any numerical issue. But using infinite precision in all algorithm leads the binary size of the number to explode. Yet, using infinite precision in a step, but, ceiling all v_n on a common denominator after each step allows both to avoid numerical issue and to keep a frozen binary size. Indeed, as v is bounded (as $F(v) \leq F(v_0)$), ceiling can be done such that this increases $v^T AA^T v$ no more than the half of Newton decrease (log terms are not affected). So it just double the number of Newton steps. Also, it is required to approximate $\frac{1}{1+\sqrt{(\nabla F)^T (\nabla^2 F)^{-1} (\nabla F)}}$ as this quantity is not in $\mathbb{Q}$. Hopefully, as F is convex, approximating it by a factor 2 just doubles the number of Newton steps (importantly, here approximating $\sqrt{\rho}$ can be computed easily in $\widetilde{O}(\log(L))$ because one does not need μ such that $\mu - 1 < \sqrt{\rho} < \mu + 1$, but, $\mu < \sqrt{\rho} < 2\mu$). Thus, each step requires at most $\widetilde{O}(N^3 \times NL)$ binary operations using controlled arithmetic: precisely it is the complexity of solving $(\nabla^2 F)u = \nabla F$ which is slightly less than N^3 (for the NL it is because binary size of number may increase by a factor N). It also assumes L-binary-size number multiplication is done in $\widetilde{O}(L)$. So, not only the number of step is polynomial, but also, the binary complexity of each step.

Damped Newton descent eq.(4) allows to find $v > \mathbf{0}$ such that $AA^T v > \mathbf{0}$ eq.(3) in a polynomial times.

3 Constrained minimization of $v^T AA^T v$

Let consider v^* the solution of

$$\min_{v \geq \mathbf{1}} v^T AA^T v \tag{5}$$

If $A_n A^T v^* < 0$, then (this is the principle of the Perceptron) one could increase v_n^* a little and produces a better solution. So $A_n A^T v^* \geq 0$.

Yet, $AA^T v^* \neq \mathbf{0}$ because $v^* AA^T v^* \neq 0$ so one could remove the n such that $A_n A^T v^* > 0$ and starts again with the others. This process of removing rows converges in at most N runs (see appendix C).

An algorithm minimizing eq.(5) can be used to solve eq.(3).

Yet, here solving eq.(5) can be done exactly (without exploding binary size) using combinatorial algorithm.

3.1 Algorithm for constrained $v^T AA^T v$ minimization

Let consider the following algorithm:

1. $v = \mathbf{1}$
2. if $AA^T v \geq \mathbf{0}$ exits
3. $I = \{i \ / \ v_I = 1\}$
4. $w = \min\limits_{u_I = \mathbf{1}} u^T AA^T u$
5. if $w^T AA^T w < v^T AA^T v$
   (a) consider $u = v + t(w - v)$
   (b) if $\exists 0 < t \leq 1$ such that $u_j = 1$ and $j \notin I$
      i. $v = v + t(w - v)$ (add at least one j to I)
      ii. goto 2 ($v^T AA^T v$ has decreased and I has increased)
   (c) else $v = w$ goto 2 (5 will false and $v^T AA^T v$ has decreased)
6. select $i \in I$ such that $A_i A^T v < 0$
7. $v_i = v_i + \frac{|A_i A^T v|}{A_i A_i^T}$ goto 2 ($v^T AA^T v$ has decreased)

This algorithm converges:

- Step 5.a can not be seen more than N times without going to step 6 (because I increases strictly)
- When entering step 6, v is entirely defined by I because v is the solution of $\min\limits_{u_I = \mathbf{1}} u^T AA^T u$ (currently this solution may not be unique but at least $v^T AA^T v$ is entirely defined by I)
- But step 7 ensures $v^T AA^T v$ decreases lower than the value when entering step 6. So step 6 can happens only once per I, but, there is no more 2^N such set I
- Thus, the number of steps 6 is lower than 2^N and there is no more than N steps 5.a between 2 steps 6

Obviously, step 6 could be improved by trying to make all i such that $A_i A^T v < 0$ exiting I the same time. Step 1 should start from $\mathbf{2}$ instead of $\mathbf{1}$, and, solving step 4 could probably be done efficiently as such minimization is somehow making free $A_i A^T v$ to be 0, so it may not require expensive matrix inversion. But anyway, this algorithm converges as it (despite complexity may be exponential).

3.2 Synergy with log barrier

Let introduce an important quantity related to this algorithm

$$\mathcal{R} = \{I \subset \{1, ..., N\} \ / \ \min_{v \geq \mathbf{1}, v_I = \mathbf{1}} v^T AA^T v = \min_{v_I = \mathbf{1}} v^T AA^T v\}\} \tag{6}$$

By construction, algorithm enters step 6 with a set I such that $I \in \mathcal{R}$, because one enters step 6 if and only if $w^T AA^T w = v^T AA^T v$ with $w = \min_{u_I = \mathbf{1}} u^T AA^T u$ and $v \geq \mathbf{1}$.

Thus, despite $\mathcal{R}$ may contain a lot of sets in general, this combinatorial algorithm will be very efficient on instances such that $\mathcal{R}$ is small. This is a complementary feature regarding log barrier, because log barrier complexity is not related to $\mathcal{R}$. Even more, there may have instance where $\mathcal{R}$ is small bellow (or higher or between) than a specific $v^T AA^T v$ value. In such situation, starting (or terminating or swapping) with log barrier may be relevant.

Let v_{cons} and v_{log} the current point of constraint and log barrier minimization of $v^T AA^T v$, v_{cons} should be replaced by v_{log} i.i.f.

$$v_{cons}^T AA^T v_{cons} > \frac{v_{log}}{\min_n v_{log,n}}^T AA^T \frac{v_{log}}{\min_n v_{log,n}}$$

(adding $1 - \min_n v_{log,n}) \times \mathbf{1}$ is also a possibility).

And, inversely, v_{log} should be replaced by v_{cons} i.i.f.

$$F(v_{log}) > F\left(\sqrt{\frac{N}{v_{cons}^T AA^T v_{cons}}} v_{cons}\right)$$

(indeed, $F(\lambda v)$ is minimal on λ when $N\lambda^2 v^T AA^T v = 1$).

Currently, both swapping could naturally happens. Log barrier on F leads to increase the product of v_n with constant $v^T AA^T v$. So despite, the minimum of v_n could be low (while the product may be high), log barrier tends to create large v with low $v^T AA^T v$ which may be suitable for constrained minimization. And, inversely, constrained minimization decreases $v^T AA^T v$ while keeping large v producing points which may naturally suitable for F.

4 Implicit minimization of $v^T AA^T v$

The final algorithm is also exponential, but, potentially fast for examples when rank of the matrix is very low. It is quite different from the other as it contains recursive call, and, some memory. Yet, at the highest level of recursion, synergy with log barrier or constrained minimization is still possible (and the memory could benefit to other algorithms). Technically, this algorithm does not focus on $v^T AA^T v$ but on $\min_n A_n A^T v$. Yet, each step consists in increasing $\min_n A_n A^T v$ while making v higher. Thus, $v^T AA^T v$ decrease at each step (even normalizing v for example by $\mathbf{1}^T v$).

4.1 Recursive algorithm

Let consider the following algorithm:

- The algorithm has memories E, B, H: with the following specifications
  - $A_I A_I^T E[I] > \mathbf{0}$, $E[I] \geq \mathbf{0}$ (E for exit)
  - $A_I A_I^T B[I] = \mathbf{1}$, $B[I] \geq \mathbf{0}$ (B for bisection)
  - H is the list of all encountered point - in particular if $v, w \in H$ with $w \geq v$ and $A_I A_I^T (w - v)_I = \rho \mathbf{1}$, then, $\frac{1}{\rho}(w - v)_I$ can be push into $B[I]$.
  - $recurse(I)$ has access to shared A, E, B, H
- highest level of recursion is $recurse(\{1, ..., N\})$
- $recurse(I) =$
  1. $v = \mathbf{1}$ (v has only dimension $|I|$)
  2. if $A_I A_I^T v \geq \mathbf{0}$ exits after adding v to $E[I]$
  3. $\mu = \min_{i \in I} A_i A_I^T v$
  4. $J = \{i \in I \ / \ A_j A_I^T v = \mu\}$
  5. if $J \notin B$
     (a) $addToHandUpdateBandE((I, J, v))$
     (b) if $J \notin B$ and $J \notin E$
        i. $recurse(J)$
  6. if $J \in B$
     (a) consider $v = v + tB[J]$ to maximize $\min_{i \in I} A_i A^T v$
  7. else
     (a) consider $v = v + tE[J]$ to maximize $\min_{i \in I} A_i A^T v$
  8. goto 2

This algorithm converges because

- at any level of recursion, step 5.a can happen only twice per set J (the second time J enters in B)
- step 6.a can not happen more than N successive times because J strictly increases each step 6.a
- J can not be equal to I (even more $A_I A_I^T v \leq \mathbf{0}$ is impossible otherwise $v A_I A_I^T v \leq 0$) so recursive call does not loop. And, recursion can only happens once per J as otherwise it is in B
- so number of steps is no more than $N 2^{2N}$ (with no more than 2^N recursive calls).

Obviously, this algorithm could benefit for directly checking if $A_I A_I^T v = \mathbf{1}$ admits a positive solution (it admits a single one when rank of A_I is maximal). Yet, convergence does not require this improvement. Let remark that looking for $v \ / \ A_I A_I^T v = \mathbf{1},\ v \geq \mathbf{0}$ is as hard as the problem tackled in this paper, so it is not an option.

4.2 Synergies with log barrier

This recursive algorithm works by improving $\min_n A_n A^T v$. Yet, a trivial way to improve $\min_n A_n A^T v$ is just to decrease the norm of v. Thus, it is required to consider normalized vector. Typically, v_{rec} should be replaced by v_{log} i.i.f.

$$\min_n \frac{A_n A^T v_{rec}}{\mathbf{1}^T v_{rec}} < \min_n \frac{A_n A^T v_{log}}{\mathbf{1}^T v_{log}}$$

Inversely, the condition to replace v_{log} by v_{rec} is the same than for v_{cons} (let stress that at the higher level of recursion v_{rec} is initialized by $\mathbf{1}$ and only increase so $v_{rec} \geq \mathbf{1}$ like v_{cons}): i.i.f.

$$F(v_{log}) > F\left(\sqrt{\frac{N}{v_{rec}^T A A^T v_{rec}}} v_{rec}\right)$$

Again, such swapping is possible: at the end of Newton descent, $AA^T v_{log} > \mathbf{0}$, so a fortiori, at some point v_{log} may help implicit minimization. And, inversely, implicit minimization decreases $v^T AA^T v$ while increasing v so each of these steps decrease F, and, thus swapping is possible.

Finally, let consider $\mathcal{S} = \{I \subset \{1, ..., N\} \ / \ \exists v > \mathbf{0} / I = \arg\min_n \ A_n A^T v\}$, on some instances, $|S|$ may be small (it seems it could happen when normalized rows of A have very low rank as only support vectors could be included in I). The recursive algorithm only performs $|\mathcal{S}|$ recursive calls (from top level). As, recursive calls are done on sets from $|\mathcal{S}|$, they will also lead to recursive calls in $|\mathcal{S}|$. So, the algorithm converges after at most $N(factorial(|\mathcal{S}|))^2$ steps (currently complexity of a single step is not that clear depending on the implementation of H). Obviously, this complexity is bad in general, but, if $|\mathcal{S}|$ is a $O(1)$, then, this algorithm is strongly polynomial.

References

[1] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Log-barrier interior point methods are not strongly polynomial. *SIAM Journal on Applied Algebra and Geometry*, 2(1):140–178, 2018.

[2] Sergei Chubanov. A polynomial projection algorithm for linear feasibility problems. *Mathematical Programming*, 153(2):687–713, 2015.

[3] George B et. al. Dantzig. The generalized simplex method for minimizing a linear form under linear inequality restraints. In *Pacific Journal of MathematicsAmerican Journal of Operations Research*, 1955.

[4] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[5] Leonid Khachiyan. A polynomial algorithm for linear programming. *Doklady Akademii Nauk SSSR*, 1979.

[6] Arkadi Nemirovski. Interior point polynomial time methods in convex programming. *Lecture notes*, 2004.

[7] Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13. Siam, 1994.

[8] Ian Post and Yinyu Ye. The simplex method is strongly polynomial for deterministic markov decision processes. *Mathematics of Operations Research*, 40(4):859–868, 2015.

[9] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

Appendix A: Equivalence on linear programming

Let consider all those algorithm specifications:

1. $algo_5$ takes $A \in \mathbb{Z}^{N \times D}, b \in \mathbb{Z}^N, c \in \mathbb{Z}^D$, and, produces x such that $x = \min_{z/Az \geq b} c^T z$ or a certificate that no such x exists in less than $C_5(N, D, L)$ binary operations.

2. $algo_4$ takes $A \in \mathbb{Z}^{N \times D}, b \in \mathbb{Z}^N$, and, produces x such that $Ax \geq b$ or a certificate that no such x exists in less than $C_4(N, D, L)$ binary operations.

3. $algo_3$ takes $A \in \mathbb{Z}^{N \times D}, b \in \mathbb{Z}^N$, and, produces x such that $Ax \geq b$ if such x exists in less than $C_3(N, D, L)$ binary operations - or has an undefined behavior otherwise.

4. $algo_2$ takes $A \in \mathbb{Z}^{N \times D}, b \in \mathbb{Z}^N$, and, produces x such that $Ax > b$ if such x exists in less than $C_2(N, D, L)$ binary operations - or has an undefined behavior otherwise.

5. $algo_1$ takes $A \in \mathbb{Z}^{N \times D}$, and, produces x such that $Ax > \mathbf{0}$ if such x exists in less than $C_1(N, D, L)$ binary operations - or has an undefined behavior otherwise.

6. $algo_2$ takes $A \in \mathbb{Z}^{N \times D}$, and, produces $v \geq \mathbf{0}$ such that $AA^T v > \mathbf{0}$ if such v exists in less than $C_0(N, D, L)$ binary operations - or has an undefined behavior otherwise.

Obviously if one has an implementation of $algo_5$ one could implement the others: $algo_5(A, b, \mathbf{0})$ is a correct implementation of $algo_4(A, b)$, and $algo_4(A, b)$ is a correct implementation of $algo_3(A, b)$ which is a correct implementation of $algo_2(A, b)$. $algo_2(A, \mathbf{0})$ is a correct implementation of $algo_1(A)$, and, finally, by considering the matrix $\mathcal{A} = AA^T$ concatenate with $\mathbf{Id}$ (the identity), then, $algo_1(\mathcal{A})$ is a correct implementation of $algo_0(A)$.

Inversely, if one has an implementation of $algo_0$ one could also implement the others

- let A such that $\exists x / Ax > \mathbf{0}$, the core of Perceptron proof [9] guarantees that one such x could be find in the form of $A^T v$ with $v > \mathbf{0}$. So, $algo_1(A)$ can be implemented by computing $v = algo_0(A)$ and returning $A^T v$. With this implementation $C_1(N, D, L) \leq C_0(N, D, L)$.

- let A and b such that $\exists x / Ax > b$, finding such x is equivalent to find a pair x, t such that $Ax - t \times b > \mathbf{0}$, $t > 0$ because $\frac{x}{t}$ is a solution. Let $\mathcal{A}$ the matrix with A plus $\mathbf{1}$ as additional column and $(\mathbf{0} \;\; 1)$ as additional row. Thus, $algo_2(A, b)$ can be implemented by computing $(x \;\; t) = algo_1(\mathcal{A})$ and returning $\frac{x}{t}$. With this implementation $C_2(N, D, L) \leq C_1(N+1, D+1, L)$.

- let A and b such that $\exists x / Ax \geq b$, finding such x is equivalent to find a pair x, t such that $Ax + t\mathbf{1} \times > b$, $0 < t < \frac{1}{\max\limits_{\mathcal{S} \subset A} Det(\mathcal{S})}$. Indeed, from such pair, one could project x on $\{z \;/\; Az \geq b\}$ while minimizing t, this leads to $\hat{x}, \hat{t}$ with $\hat{t} \leq t < \frac{1}{\max\limits_{\mathcal{S} \subset A} Det(\mathcal{S})}$ but $\hat{x}, \hat{t}$ is a vertex of A. So Cramer rule applies, and so there exists $\mathcal{S}$ such that $\hat{t} = \frac{Det(\mathcal{S}_{partiel})}{Det(\mathcal{S})}$ but seeing constraint on $\hat{t}$ it forces $\hat{t} = 0$ i.e. $A\hat{x} \geq b$. So, $algo_3(A, b)$ can be implemented by

  - computing $(x \;\; t) = algo_2(\mathcal{A}, \beta)$ with $\mathcal{A}$ being A plus $\mathbf{1}$ as additional column, and $(\mathbf{0} \;\; 1)$, $(\mathbf{0} \;\; -1)$ as additional rows and $\beta = b$ plus 0 and $-\frac{1}{\max\limits_{\mathcal{S} \subset A} Det(\mathcal{S})}$
  - projecting $(x \;\; t)$ on $\{(z \;\; \tau) \;/\; Az + \tau\mathbf{1} \geq b\}$ while minimizing t
  - returning x

  Importantly, if highest number in A, b is bounded by 2^L, then, maximal determinant is lower than $2^{NL + N\log(N)}$ (Hadamard bound when $N > D$ otherwise $2^{DL + D\log(D)}$), so with such implementation $C_3(N, D, L) \leq C_2(N, D, NL)$

- let A, b, solving $Ax \geq b$ is equivalent to solve $\min\limits_{z \;/ Az + t\mathbf{1} \geq b, t \geq 0} t$ (there is a solution if the minimum is 0). Yet, this last linear program is structurally feasible and bounded, thus, primal dual theory gives a system $A_{primal-dual}(x \;\; y) \geq b_{primal-dual}$ whose solution contains solution of the linear program $\min\limits_{z \;/ Az + t\mathbf{1} \geq b, t \geq 0} t$ and thus of x such that $Ax \geq b$ or a certificate that no solution exists. So a possible implementation of $algo_4(A, b)$

is to compute $(x \quad y) = algo_3(A_{primal-dual}, b_{primal-dual})$ returning only x or the certificate. Importantly, this primal dual has only $N + D + 6$ variables and constraints from an original problems with D variables and N constraints, so $C_4(N, D, L) \leq C_3(N + D + 6, N + D + 6, L)$.

- finally, $algo_5(A, b, c)$ can be computed using 3 calls to $algo_4$: one to know if the problem is feasible $algo_4(A, b)$, one on the dual to known if it is bounded $algo_4(A_{dual}, b_{dual})$ and one on the primal dual to get the optimal solution if it is feasible and bounded. So $C_5(N, D, L) \leq 3C_4(N + D + 6, N + D + 6, L)$.

All these equivalences shows that if one produce a polynomial times implementation of $algo_0$, it leads to a polynomial times implementation of $algo_5$ just adding a factor N.

In addition, these equivalences consider the production of certificate of impossibility. Currently, if $weakAlgo(A)$ produces $v > \mathbf{0}$ such that $AA^T v > \mathbf{0}$ in less than $\widetilde{O}(\mathcal{F}(N, L))$ binary operations when it exists, than, it is sufficient to wait $\widetilde{O}(\mathcal{F}(N, L)) + 1$ operations to known that no solution exists, but, in this case there is no certificate.

Appendix B: Perceptron and support vector machine

If $\exists v > \mathbf{0} \ / \ AA^T v > \mathbf{0}$, then, $A(A^T v) > \mathbf{0}$ so $\exists x \ / \ Ax > \mathbf{0}$ and by scaling $\exists x \ / \ Ax \geq \mathbf{1}$.

Inversely, if $\exists x \ / \ Ax \geq \mathbf{1}$ with rows of A being normalized, one could consider the algorithm $x_{t+1} = x_t + a_t^T$ with $a_t x_t \leq 0$ (which stops if $Ax > \mathbf{0}$) with $v_0 = \mathbf{1}$.

Then, $(x^T x_t)^2 \leq x^T x x_t^T x_t$ (Cauchy)

But $x_{t+1}^T x_{t+1} \leq x_t^T x_t + 1$ because $a_t x_t \leq 0$ and $a_t a_t^T = 1$

While $x^T x_{t+1} \geq t$ because $Ax \geq \mathbf{1}$

So $(x^T x_t)^2 \leq x^T x x_t^T x_t$ becomes $t^2 \leq x^T x_t$ i.e. $t \leq x^T x$.

So, this serie can not last more than $x^T x$ step, and, thus the algorithm converges and returns some x.

Now, at the end, the solution is a positive linear combination of rows of A. So, by remembering the steps of the algorithm, one could recover $v > \mathbf{0}$ such that $AA^T v > \mathbf{0}$.

So, Perceptron converges in less than $\mathbf{1}^T AA^T \mathbf{1} \times x^T x$ steps with x such that $Ax \geq \mathbf{1}$ while Newton descent on F terminate in less than $N \log(\mathbf{1}^T AA^T \mathbf{1} \times x^T x)$ (coarsely). But, how much $x^T x$ could be large ?

The minimal value of such x is

$$\min_{x \ / Ax \geq \mathbf{1}} x^T x$$

i.e. the solution of the support vector machine problem linked to A. Currently, from any x such that $Ax \geq \mathbf{1}$, one could build a larger x but this paper proof is true with the minimal x.

Now, it is not trivial to show directly that this $x^T x$ is small. But the classical trick is to consider the following theoretical problem (that nobody is going to solve) $\min_{\chi \ / A\chi \geq \mathbf{1}} \Upsilon((\chi \quad A\chi - \mathbf{1})$ where $\Upsilon(p) > \Upsilon(q)$ if and only if $\upsilon(p) > \upsilon(q)$ or $\upsilon(p) = \upsilon(q)$ and $||p||_1 > ||q||_1$ and where $\upsilon(p)$ is the lexicographic rank of not-zero index vector p i.e. $\upsilon(p) > \upsilon(q)$ if and only if $\exists i \ /(\forall j < i, \ p_j = 0 \Leftrightarrow q_i = 0) \wedge p_i \neq 0 \wedge q_i = 0$.

The solution of this problem is unique (because if there is 2, one could explore the linear combination of the two either to saturated some $A_j\chi \geq 1$ or to make 0 some χ_i). And, this solution is naturally a vertex because it maximizes the size of a set of equality. Thus, the solution of this theoretical problem is also the only solution of a set of equality $A_J\chi = \mathbf{1}$, $\chi_J = \mathbf{0}$. Thus, this system is full rank, so one could extract a square not singular matrix $\mathcal{A}$ from it, and, $\mathcal{A}\chi = \mathcal{B}$ with $\mathcal{A}$ being a sub matrix of A (plus some Dirac vector) and $\mathcal{B}$ being 0-1 vectors. Then Cramer rules hold, and, ensure that each component of χ is bounded by a sub determinant of A.

So, there exists χ such that $A\chi \geq \mathbf{1}$ and such that $\chi^T\chi \leq NN^{2N}2^{2NL}$ i.e. $\log(\chi^T\chi) = \widetilde{O}(NL)$ (a fortiori $x^T x \leq \chi^T\chi = \widetilde{O}(NL)$).

Appendix C: recursive row elimination

If $v > \mathbf{0}$ verifies $AA^T v > \mathbf{0}$, then, $AA^T v \geq \mathbf{0}$.

Now, assume one is able to solve $v > \mathbf{0}$ and $AA^T v \geq \mathbf{0}$ i.e. it takes A as input such that $\exists x / Ax > \mathbf{0}$ and it produces $\mathcal{F}(A) > \mathbf{0}$ / $AA^T\mathcal{F}(A) \geq \mathbf{0}$.

First, $AA^T\mathcal{F}(A) = \mathbf{0}$ is impossible otherwise $\mathcal{F}(A)^T AA^T\mathcal{F}(A) = 0$ i.e. $A^T\mathcal{F}(A) = \mathbf{0}$ and $0 = x^T\mathbf{0} = x^T A^T\mathcal{F}(A) \geq \mathbf{1}^T\mathcal{F}(A) > 0$ contradiction.

So, let consider the following algorithm given A:

- $A_0 = A$, $v_0 = \mathcal{F}(A)$
- given v_t, $I_t = \{i \ / \ A_{t,i}A_t^T v_t = 0\}$, $J_t = \{i \ / \ A_{t,i}A_t^T v_t > 0\}$, $A_{t+1} = A_{t,I_t}$
- $v_{t+1} = \mathcal{F}(A_{t+1})$

Using previous lemma, it stands that I_t does not contain all indexes, thus, this serie can not be infinite. At the end, one has $v_1, ..., v_K$ and $I_1, J_1, ..., I_K, J_K$ with $A_{I_k}v_k = 0$ and $A_{J_k}v_k > 0$ and $J_{k+1} \cup I_{k+1} = I_k$, $I_k \neq I_{k+1}$

Thus, it is than possible to consider:

- $w_K = v_K$
- $w_{k-1} = (1 + \min_{j \in J_{k-1}} \frac{A_j A^T w_k}{v_j})v_{k-1} + w_k$

This way, $\forall k$, $\forall j \in \bigcup_{t \geq j} J_t$, $A_j A^T w_k > 0$ For $k = 0$, $AA^T w_0 > \mathbf{0}$.

So, N calls to the routine finding $v > \mathbf{0}$ and $AA^T v \geq \mathbf{0}$ are sufficient to find $v > \mathbf{0}$ and $AA^T v > \mathbf{0}$.

Independently, if $\exists \phi \ / \ A\phi = \mathbf{0}, \phi \neq \mathbf{0}$, then, for all solutions x such that $Ax > \mathbf{0}$, $x + \lambda\phi$ is also a solution. Thus, if there is a solution, there is one such that $Ax > \mathbf{0}, \phi^T x = 0$. But, the constraint $\phi^T x = 0$ can be used to remove one variable while leading to a new problem in the same framework (i.e. still a *find x such that $Ax > \mathbf{0}$ problem)*.

So, one could even assume that $\forall \phi \neq \mathbf{0}, \ A\phi \neq \mathbf{0}$. Unfortunately, it does not make AA^T not singular (duplicate rows for example create singularity in AA^T). Log barrier minimization does not require this feature as the diagonal $\frac{1}{v^2}$ is added to AA^T. But, constrained minimization algorithm is less deterministic considering that $\min_{u_I=1} u^T AA^T u$ has multiple solutions. However, algorithm is still valid: if $v^T AA^T v = \min_{u_I=1} u^T AA^T u$ then u has no importance, and, if $v^T AA^T v > \min_{u_I=1} u^T AA^T u$, any such u makes the algorithm to converge.