The simplest polynomial times algorithm for linear programming.

Adrien Chan-Hon-Tong

The simplest polynomial times algorithm for linear programming.

Adrien CHAN-HON-TONG

August 5, 2020

Abstract

This paper introduces a simple and straightforward algorithm for linear programming. Only self concordant theory and basic linear algebra are required to prove the convergence and the polynomial times complexity.

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 [2]. The first polynomial time algorithm solving this problem is the ellipsoid method [4, 3]. Today, the state of the art of polynomial time algorithm for linear programming is central-path log-barrier algorithm [6, 5] (despite the recent Chubanov method [1] could be a challenger).

Yet, classical method are not trivial to explain. Ellipsoid method requires not trivial algebra (like the computation of the minimal enclosing ellipsoid of an ellipsoid cut by a plane). And, classical log barrier algorithm are not straightforward because it consists to apply Newton descent of a function which changes during runtimes, and, it requires post processing to purify the resulting solution. Inversely, the offered algorithm apply Newton descent of a static function whose minimum is easily characterized.

For this reason (and because this algorithm is theoretically not that slower than current state of the art, and, because it deals structurally with linear separation problems which is common in machine learning), this algorithm could be useful for education.

So, let state the required common knowledge to build the proof and the main claims of this paper:

Required common knowledge:

1. If an algorithm is able to decide if $\exists x \ / \ Ax > \mathbf{0}$ for any given matrix A in polynomial times (in the binary size of A), then, it can be adapted to solve any linear program instances eq.(1) in polynomial times (in the binary size of the matrix A and vectors b, c) (see appendix A).

2. The set of single variable function ϕ verifying $|\phi'''(t)| \leq 2(\phi''(t))^{\frac{3}{2}}$ is the set of self concordant single variable function, a multi variable function $\Phi(v)$ is self concordant if $\forall v, w$, $\phi_{v,w}(t) = \Phi(v + tw)$ is self concordant.

3. If Φ is self concordant with a minimum, damped Newton descent allows to produces ω such that $\Phi(\omega) - \Phi^* \leq \varepsilon$ in $O(\Phi(\omega_0) - \Phi^* + \log\log(\frac{1}{\varepsilon}))$ steps.

Theorem 1:

Let $A \in \mathbb{Q}^{N \times M}$ a matrix with normalized rows[1] (i.e. $A_n A_n^T = 1$) such that exists $x \in \mathbb{Q}^M$ verifying $Ax \geq \mathbf{1}$, then the function

$$F(v) = \frac{v^T A A^T v}{2} - \mathbf{1}^T \log(v) = \sum_{i,j=1}^{N} v_i v_j \times A_i A_j^T - \sum_{n=1}^{N} \log(v_n) \tag{2}$$

has a minimum F^*, and, $\forall v \in \mathbb{Q}^N$ such that $v^T A A^T v = N$:

$$F(v) - F^* \leq \frac{1}{2\sqrt{N x^T x} + 2} \Rightarrow A A^T v > \mathbf{0} \tag{3}$$

Theorem 2:

As F is self concordant, and, $F(\frac{1}{N}\mathbf{1}) - F^* \leq N \log(N x^T x) + N$, applying damped Newton descent from $v_0 = \frac{1}{N}\mathbf{1}$ allows to build a solution of $A A^T v > \mathbf{0}$ in less than $\widetilde{O}(N \log(x^T x)) = \widetilde{O}(N^2 L)$ steps where L is the binary size of the largest value in A (when rescaling the matrix in $\mathbb{Z}$).

As computation can be done with controlled precision, this algorithm has a polynomial times of $\widetilde{O}(N^2 L)$ steps, each step being mainly the inversion of a $N \times N$ matrix having value with binary size $NL \log(NM)$ when scaled on $\mathbb{Z}$ (number of initial variables M only has the minimal impact of increasing $A_i A_j^T$ i.e. it adds a $\log(M)$ factor in the complexity - yet $N \geq M$ in typical instances).

2 Proof

2.1 Minimum of F

Let recall that Cauchy states that $\forall u, v$, $u^T v \leq \sqrt{u^T u \times v^T v}$.

[1] The algorithm works with not normalized matrix but normalizing is possible - see appendix C

So, $v^T Ax = (A^T v)^T x \leq \sqrt{v^T AA^T v \times x^T x}$. But, by definition $Ax \geq \mathbf{1}$, so $\forall v \geq \mathbf{0}$, $v^T \mathbf{1} \leq v^T Ax \leq \sqrt{v^T AA^T v \times x^T x}$. So, $\forall v \geq \mathbf{0}$, $\frac{(v^T \mathbf{1})^2}{x^T x} \leq v^T AA^T v$, and, $(v^T \mathbf{1})^2 > v^T v$ $(v \geq \mathbf{0})$. So $\forall v \geq \mathbf{0}$, $\frac{v^T v}{x^T x} \leq v^T AA^T v$.

Let introduce $f(t) = \frac{t^2}{2x^T x} - \log(t)$, from previous inequality it stands that $F(v) \geq \sum_n f(v_n)$.

Now, f is a single variable function which goes to infinity when t goes to 0 or to infinity, so, f has a minimum and so F too.

As F is smooth the minimum is characterized by a null gradient so the minimum of F verifies

$$\nabla_v^* F = 0 \Leftrightarrow AA^T v^* = \frac{1}{v^*}$$

So, the minimum is a solution to the problem, but, bellow, it is proven than any low values is ones.

2.2 Normalization, linearization and lemmas

Independently, let remark that $\theta(t) = F(tv) = \frac{v^T AA^T v}{2} t^2 - \mathbf{1}^T \log(v) - N \log(t)$ is minimal when $v^T AA^T v = N$, so any iterative algorithm could normalize v between each step[2] so that $v^T AA^T v = N$ without increasing F.

So, let consider $v \geq \mathbf{0}$ such that $v^T AA^T v = N$. As, $v^T AA^T v \geq \frac{(\mathbf{1}v)^2}{x^T x}$, no v_n could be higher than $\sqrt{Nx^T x}$ i.e. $\mathbf{0} \leq v \leq \sqrt{Nx^T x}\mathbf{1}$.

Let also remark that $F(v+w) = \frac{v^T AA^T v}{2} + \frac{w^T AA^T w}{2} + w^T AA^T v - \mathbf{1}^T \log(v) - \mathbf{1}^T \log(1 + \frac{w}{v}) = F(v) + \frac{w^T AA^T w}{2} + w^T AA^T v - \mathbf{1}^T \log(1 + \frac{w}{v})$

Finally, let consider the following lemmas from basic analysis:

1. $\varphi(t) = \frac{1}{2}\alpha t^2 - \log(1+t) \leq \frac{1}{2}(\alpha+1)t^2 - t = \psi(t)$
2. $\psi(\frac{1}{\alpha+1}) \leq -\frac{1}{2\alpha+2}$
3. $\varphi(\frac{1}{\alpha+1}) \leq -\frac{1}{2\alpha+2}$ i.e. $\forall \rho \geq 0$, $\frac{1}{2}\frac{\rho}{\rho+1} - \log(1 + \frac{1}{\rho+1}) \leq -\frac{1}{2\rho+2}$

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 is just 1+2.

2.3 Low values of F

Now, if $A_k A^T v \leq 0$ and $v^T AA^T v = N$, let introduce $w = v + \frac{vk}{v_k+1}\mathbf{1}_k$. Then $F(w) = F(v + \frac{vk}{v_k+1}\mathbf{1}_k) = F(v) + \frac{A_k A_k^T}{2}(\frac{vk}{v_k+1})^2 + A_k A^T v \times \frac{vk}{v_k+1} - \log(1 + \frac{1}{v_k+1})$
But, $A_k A^T v \leq 0$ and $A_k A_k^T = 1$, so $F(w) \leq F(v) + \frac{1}{2}\frac{vk}{v_k+1} - \log(1 + \frac{1}{v_k+1})$.

[2]The algorithm hopefully works without this normalization as square root computation is not possible on $\mathbb{Q}$ - see arithmetic considerations

And, from lemma 3 $F(w) \leq F(v) - \frac{1}{2v_k+2}$. But, $v_k \leq \sqrt{Nx^Tx}$, so, $F(w) \leq F(v) - \frac{1}{2\sqrt{Nx^Tx}+2}$ which is impossible if $F(v) - F^* < \frac{1}{2\sqrt{Nx^Tx}+2}$

So, the theorem 1 is proven: $\forall v > \mathbf{0}$ such that $v^TAA^Tv = N$, $F(v) - F^* \leq \frac{1}{2\sqrt{Nx^Tx}+2} \Rightarrow AA^Tv > \mathbf{0}$.

2.4 Complexity

Self concordant function are linear, positive quadratic and -log function. So, F is self concordant.

So, Damped Newton descent [5]

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

plus renormalization of v, starting from v_0 allows to reach $v > 0$ with $v^TAA^Tv = N$ such that $F(v) \leq F^* + \frac{1}{2\sqrt{Nx^Tx}+2}$ in $O(F(v_0) - F^* + \log\log(\frac{1}{Nx^Tx}))$ Newton steps.

But, as A is normalized $F(v_0) \leq N\log(N) + 1$.

Finally, let assume that $\forall i, j,\ A_{i,j} \leq 2^L$ (when projecting A in $\mathbb{Z}$), then, there is x such that $Ax \geq \mathbf{1}$ with $x^Tx \leq N2^{2NL}N^{2N}$ due to Hadarmard bound + Cramer rules. Indeed, x can be seen as the solution of a linear system built from A so Cramer rules for linear system allows to characterize each x_m with sub determinant of A, but, Hadarmard bound states that no sub determinant could be higher than $2^{NL}N^N$ (this is common knowledge, see appendix B for precision). So $\log(x^Tx) = \widetilde{O}(NL)$.

Currently, here the duration of the convergence in the quadratic part is completely negligible, only count $F(v_0) + |F^*| = \widetilde{O}(N\log(x^Tx)) = \widetilde{O}(N^2L)$ (F can not be lower than $N - N\log(Nx^Tx)$ as $v^TAA^Tv = N$ and $v \leq \sqrt{Nx^Tx}$).

So, the first part of theorem 2 is proven: damped Newton descent allows to produces v such that $AA^Tv > 0$ in less than $\widetilde{O}(N^2L)$ steps.

3 Arithmetic considerations

This section deals with the fact that operations done by a computer can only be in $\mathbb{Z}$ and eventually in $\mathbb{Q}$ using using infinite precision representation of numbers (e.g. fraction with infinite length numerator and denominator).

So, this raise 2 issues : first the cost of an elementary operation depends on the integer sizes, and any not rational operations can not be done *exactly*.

First, the only not rational operations are normalization and computation of the damped step. The solution is in both case to approximate with a fixed ratio the correct value. This way, as computing $\frac{1}{1+\sqrt{(\nabla_v F)^T (\nabla_v^2 F)^{-1} (\nabla_v F)}}$ is impossible one will just compute $q \leq \frac{1}{1+\sqrt{(\nabla_v F)^T (\nabla_v^2 F)^{-1} (\nabla_v F)}} \leq 2q$. Computing such q is possible even using bisection in $\widetilde{O}(\log\log(L))$ because one

look for q such that $q \leq \frac{1}{1+\sqrt{(\nabla_v F)^T(\nabla_v^2 F)^{-1}(\nabla_v F)}} \leq 2q$ and **not** such that $q \leq \frac{1}{1+\sqrt{(\nabla_v F)^T(\nabla_v^2 F)^{-1}(\nabla_v F)}} \leq q+1$. So the bisection can be done on the power - not on the value. This requires initial bounds which are $\frac{1}{1+(\nabla_v F)^T(\nabla_v^2 F)^{-1}(\nabla_v F)}$ and 1.

This way, the approximate damped Newton step may lead to only half decreasing, but, hopefully not less because the function is convex. So, two approximate damped Newton steps are at least as good as one exact damped step (during first phase - second phase is negligible with this algorithm).

For the normalization, computing $\sqrt{v^T AA^T v}$ is impossible so exact normalization can not be done. But making $\sqrt{v^T AA^T v}$ not higher than $2N$ or not lower than $\frac{N}{2}$ is possible with bisection on power in $\widetilde{O}(\log\log(L))$.

Importantly, consideration on variable bound still hold if $v^T AA^T v \in [\frac{N}{2}, N]$ (adding a factor two on the maximal size, and, thus on the required proximity to optimal value - but it does not change anything from theoretical point of view).

Second point, during a step, it is possible allowing to avoid any numerical issue using infinite representation. But using infinite precision in all algorithm leads the binary size of the number to explode. So, the idea is to use infinite representation in a step, and, infinite representation with bounded denominator in all algorithm by ceiling v between each step - importantly, ceiling and not flooring should be used to avoid any issue with the log. Precisely, after each step, $v_n = \textbf{fraction}(Q \times v_n + 1, Q)$ (for example in python).

The good point is that $v \leq \sqrt{Nx^T x}$, so the ceiling operation can not increase F by more than $\frac{N^2\sqrt{Nx^T x}}{Q}$. As Newton step guarantee a decrease of at least $\frac{1}{4} - \log(\frac{5}{4}) \approx \frac{1}{50}$ (during first phase but there is not second phase here), it is sufficient to select Q such that $\frac{N^2\sqrt{Nx^T x}}{Q} \leq \frac{1}{100}$ (currently less because all the numerical consideration should be taken simultaneously, so it is $\frac{1}{1600}$ in fact). So, it just that using $Q = 2^{\log(1600N^{\frac{3}{2}}x^T x)+1}$ is acceptable. So binary size of Q is just $\widetilde{O}(NL)$.

So, using this ceiling strategy binary size of denominator is $\widetilde{O}(NL)$ and as $v \leq \sqrt{Nx^T x}$ binary size of the numerator is at most two times higher i.e. still $\widetilde{O}(NL)$.

All those arithmetic considerations[3] prove the second part of the theorem 2: the number of binary operation of each step is lower than $\widetilde{O}(N^\kappa \times NL)$ (under assumption that multiplication of numbers is done efficiently in $\widetilde{O}(L)$) where N^κ is the complexity of the computation of $(\nabla_v^2 F)^{-1}(\nabla_v F)$ - let say 3 for naive implementation or 2.8 with Strassen or so on.

[3]These considerations are true from theoretical point of view - from a practical point of view, this algorithm is absolutely not able to solve even small with $N = 500$ instances using such exact arithmetic scheme with python fraction while solving instance with $N = 10^6$ has been achieved using numpy float - taking the risk of failing on ill conditioned instances - exact version will never fail.

4 Conclusion

The offered algorithm may not be competitive with classical log barrier as it tackle $\exists ?x \ / \ Ax > \mathbf{0}$ problems, and, not $\min_{Ax \geq b} c^T x$ problems + complexity of state of the art log barrier is $\widetilde{O}(\sqrt{N}L)$ step against $\widetilde{O}(N^2 L)$ here[4].

Yet, the offered algorithm is very simple to explain, and directly tackle linear separation. So, it could be used for education and machine learning.

Also - this will be the topic of futur works - Newton descent can be hybridized for example $F(v+w) = F(v) + \frac{w^T A A^T w}{2} + w^T A A^T v - \mathbf{1}^T \log(1 + \frac{w}{v}) \leq F(v) + \frac{N}{2} w^T w + w^T A A^T v - \mathbf{1}^T \log(1 + \frac{w}{v})$ as A is normalized. But, $\frac{N}{2} w^T w + w^T A A^T v - \mathbf{1}^T \log(1 + \frac{w}{v})$ can be optimized exactly in $O(N)$ operations because it is N independent simple single variable optimization. So, some Newton step could be avoided when the bound is tight... In the same way, using an exponential-simplex-like algorithm (in parallel) on $\min_{v \geq 1} v^T A A^T v$ could lead to some synergies...

References

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

[2] 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.

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

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

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

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

Appendix A: Equivalence on linear programming

This paper provides an algorithm $\mathbf{algo}_0$ such that for any A such that $\exists x \ / \ Ax \geq \mathbf{1}$, the algorithm returns v such that $AA^T v > \mathbf{0}$ in $\widetilde{O}(N^2 L)$ operations.

[4] Currently, it is not clear that L is similar, but, the L selected in this paper allows to use Hadamard bound...

- Let consider any A such that $\exists x \ / \ Ax \geq \mathbf{1}$, producing χ such that $A\chi > \mathbf{0}$ can be done easily by applying $\mathbf{algo}_0$ and returning $A^T v$ instead of just v. So, one could build $\mathbf{algo}_1$ which returns χ with the same complexity.

- Let consider any A, 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, $\mathbf{algo}_2(A, b)$ can be implemented by computing $(x \ \ t) = \mathbf{algo}_1(\mathcal{A})$ and returning $\frac{x}{t}$ (complexity still unchanged)

- Let any 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, $\mathbf{algo}_3(A, b)$ can be implemented by

  - computing $(x \ \ t) = \mathbf{algo}_2(\mathcal{A}, \beta)$ 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), so, the complexity of $\mathbf{algo}_3$ is increased by a factor N compared to the complexity of $\mathbf{algo}_2$.

- Let any A, b - **without assumption** - 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 ($x = 0$ and a sufficiently large t) and bounded because $t \geq 0$. 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 $\mathbf{algo}_4(A, b)$ is to compute $(x \ \ y) = \mathbf{algo}_3(A_{primal-dual}, b_{primal-dual})$ returning only x or the certificate.

  Importantly, the number of variables-constraints is scaled two folds but from theoretical point of view, it does not change the complexity...

  Let stress that from $\mathbf{algo}_4$ there is no assumption on the input: for any A, b $\mathbf{algo}_4(A, b)$ either returns x such that $Ax \geq b$, or, a certificate that no such x exists.

- Finally, for any A, b, c without any assumption solving eq.(1) can be done with $\mathbf{algo}_5(A, b, c)$: $\mathbf{algo}_5$ does 2 $\mathbf{algo}_4$ calls and one $\mathbf{algo}_3$ call
  - one to know if the problem is feasible i.e. $\mathbf{algo}_4(A, b)$
  - one on the dual to known if it is bounded $\mathbf{algo}_4(A_{dual}, b_{dual})$
  - and one call to $\mathbf{algo}_3$ on the primal dual to get the optimal solution (if one passes the two first step than the problem is feasible and bounded so the primal dual has a solution).

  Again, from theoretical point of view, the complexity does not change: it only does 3 calls on instances only scaled 2 times. At the end, it returns the optimal solution or a certificate that the problem is not feasible or not bounded

So, this recall the common knowledge that the ability to solve linear separation v / $AA^Tv > 0$ (assuming a solution exists) allows to solve linear programming in general just adding a factor N when going from solving $Ax \geq b$ using $Ax > b$ algorithm (the opposite is trivial $\mathbf{algo}_5(AA^T, \mathbf{1}, \mathbf{0})$ is a correct implementation of $\mathbf{algo}_0(A)$ for any A).

Appendix B: Perceptron and support vector machine

If $\exists v > \mathbf{0}$ / $AA^Tv > \mathbf{0}$, then, $A(A^Tv) > \mathbf{0}$ 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 $AA^Tv > \mathbf{0}$.

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: Normalizing linear separation

Let A such that $\exists x \ / \ Ax \geq \mathbf{1}$, then let consider $\mathcal{A}$ which has 4 times more constraint and 2 additional variables (so complexity does not change) built as follow: for all constraint A_n, $\mathcal{A}$ gets the 4 constraints $2A_n :: \pm 1 :: \pm 2A_nA_n^T$.

The point is that each such new constraints has a norm $4A_nA_n^T + 1 + 4(A_nA_n^T)^2 = (2A_nA_n^T + 1)^2$. So, each new constraint can be normalized on $\mathbb{Q}$ as the root of the norm is an integer (while binary size is only scaled twice).

Importantly, $(x \quad 0 \quad 0)$ is a solution for $\mathcal{A}$, and inversely, any solution χ for $\mathcal{A}$ is a solution for A when removing the last two coordinates (if not zeros, it always make one from the 4 constraints worse).

However, normalizing is mainly for understanding - the algorithm works with the raw matrix but proof is a bit harder. Also, despite normalizing is not required, it makes sure that all constraints are considered independently from the norm, so it may have an influence on the geometry underlying the problem.