# Solving linear feasibility in linear number of steps.

Adrien Chan-Hon-Tong

# Solving linear feasibility in linear number of steps.

Adrien CHAN-HON-TONG

December 14, 2020

**Abstract**

This paper introduces an algorithms for linear feasibility which is deterministic, exact, and, with a linear complexity (in step - each step corresponding to solve a linear system).

In addition, this algorithm has several features which should be explored. First, both the numbers of steps with high or low Newton increment are negligible. Then, this algorithm does not seem to be related to central path when extended to linear programming.

## Introduction

This paper is about an algorithm for solving linear feasibility (also called linear separability depending on the community). Such task is central in machine learning, but, marginal in optimization because it requires strong assumption on the input (in fact even in machine learning community lower assumption are usually considered). However, appendix A recalls that any linear programming (bounded or not, feasible or not) instance can be encoded into a linear feasibility instance. So the scope of this paper may be larger than linear feasibility only.

**Definition 1** A normalized linear feasibility instance is given by a matrix $A \in \mathbb{Z}^{M \times N}$ such that

- $\exists x \in \mathbb{Q}^N$ such that $Ax > \mathbf{0}$
- $\forall m \in \{1, ..., M\}, \sqrt{\mathcal{A}_m \mathcal{A}_m^T} \in \mathbb{Z}$

solving the instance consists in finding such $x$ verifying $Ax > \mathbf{0}$.

$M_A$ is the number of constrains, $N_A$ is the number of variable, $B_A$ is the maximal binary size of any entry of $A$, and, $L_A$ the total binary size with $L_A \lessapprox M_A N_A B_A$.

For all linear feasibility instances, $\Omega_A^2 = \min_{\omega \ / \ \forall m, A_m \omega \geq \mathbf{1}} \omega^T \omega$.

Importantly, any linear feasibility instance can be trivially encoded into a normalized one (see appendix A). Inversely, the $x$ hypothesis may sound not

*real-life* (because it seems improbable to have assumption that a solution exists without having solving the problem). Yet, this assumption is central, otherwise, the problem should be processed like a regular linear program (by encoding it as a feasible linear feasibility problem as presented in appendix A) but with loss of performance.

Importantly, for an exact algorithm, the output should be $x$ such that $Ax > \mathbf{0}$ (or equivalently $Ax \geq \mathbf{1}$), and not *an approximation* like $x$ such that $Ax \geq -\epsilon\mathbf{1}$ with very small $\epsilon$, or, such that $A_I x > \mathbf{0}$ with $I$ being almost-but-not-all rows. So an algorithm solving exactly linear feasibility is not comparable with common solvers (e.g. [7, 2, 8, 13, 14]) for which $Ax > \mathbf{0}$ may not be true even when a solution exists.

Also, if the algorithm is deterministic, then, it may never fail which is not the case for example of [6] which is independently much faster than the algorithm offered in this paper but with the drawback of having a probability of failure (which is probably not a problem in real life as this probability is mastered, yet it changes something in theory).

Finally, for complexity analysis, the paper relies on classical $O$ and $\widetilde{O}$ notation, and, a first important common knowledge result is that $\log(\Omega_A) = O(NB)$ (see appendix C).

Now, the introduced algorithm does not directly try to find $x$ such that $Ax > \mathbf{0}$. Instead it solves an auxiliary problem, but, this results in such $x$.

**Definition 2** $\forall v \in \mathbb{Q}^M, A \in \mathbb{Q}^{M \times N}$,

$$F_A(v) = \frac{v^T A A^T v}{2} - \sum_{m=1}^{M} \log(v_m) = ||A^T v||_2^2 - \mathbf{1}^T \log(v)$$

Importantly $F_A$ is self concordant (see appendix D), and, the offered algorithm is somehow just a Newton descent on $F_A$. Yet, the contribution is that, minimizing $F_A$ is not relevant by itself, but will result in some $v$ such that $AA^T v > \mathbf{0}$ i.e. a solution to the linear feasibility instance $A$. Indeed, theorem 1 claims:

**Theorem 1**: let $A \in \mathbb{Z}^{M \times N}$ be a normalized instance of linear feasibility, and $\widehat{A}$ be $A$ when each row are normalized on $\mathbb{Q}$, then,

- $F_{\widehat{A}}(\frac{1}{M_A}\mathbf{1}) = \widetilde{O}(M_A)$
- $F_{\widehat{A}}(v)$ is lower bounded
- let write $F^*_{\widehat{A}}$ for the minimum, then $F^*_{\widehat{A}} = \widetilde{O}(M_A \log(\Omega_A))$
- and for all $v$, $F_{\widehat{A}}(v) - F^*_{\widehat{A}} \leq \frac{1}{2M_A\Omega_A+2} \Rightarrow \widehat{A}\widehat{A}^T v > \mathbf{0}$

The link with linear feasibility is a trivial corollary of theorem 1 as minimizing sufficiently $F_{\widehat{A}}$ allows to find $x = \widehat{A}^T v$ such that $Ax > \mathbf{0}$. This leads to theorem 2:

**Theorem 2**: Damped Newton descent on $F_{\widehat{A}}$ requires at most $\widetilde{O}(L_A)$ steps to produce a solution of the linear feasibility instance $A$.

Currently, the only contribution of the paper is theorem 1. Theorem 2 is a trivial consequence of self concordant theory which is briefly recalled in appendix D but could be found for example in [10]. Globally, many results are required but are common knowledge. So, they are only presented in appendix for completeness: conversions between linear feasibility and linear programming (appendix A), Perceptron [12] (in appendix B) which share some idea with the offered algorithm, bound on $\Omega_A$ (appendix C) which is based on Cramer rules and Hadamard bound

Normally, all notations of the paper are standards. The index $_A$ or $_{\widehat{A}}$ of $F_A$ and $\Omega_A$ would be omitted when no ambiguity is possible.

# Proof of theorem 1

$A \in \mathbb{Q}^{M\times N}$ is a linear separability instance with normalized rows (so, $\exists x \ / \ Ax > \mathbf{0}$) with $\Omega^2 = \min\limits_{\omega \ / \ \forall m, A_m\omega \geq \mathbf{1}} \omega^T\omega$, and, $F(v) = \frac{v^TAA^Tv}{2} - \sum\limits_{m=1}^{M} \log(v_m)$.

## Existence of a minimum

First, as $A$ is normalized $A_iA_j^T \leq 1$ due to Cauchy, so it trivially holds that $F(\frac{1}{M}\mathbf{1}) = \frac{1}{M^2}\sum\limits_{i,j} A_iA_j^T + \sum\limits_{m=1} \log(M) \leq M\log(M) + 1 = \widetilde{O}(M)$.

then, $v^TA\omega = (A^Tv)^T\omega \leq \sqrt{v^TAA^Tv \times \omega^T\omega}$. But, by definition $A\omega \geq \mathbf{1}$, so $\forall v \geq \mathbf{0}$, $v^T\mathbf{1} \leq v^TA\omega \leq \sqrt{v^TAA^Tv \times \Omega^2}$. So, $\forall v \geq \mathbf{0}$, $\frac{(v^T\mathbf{1})^2}{\Omega} \leq v^TAA^Tv$, and, $(v^T\mathbf{1})^2 > v^Tv$ $(v \geq \mathbf{0})$. So $\forall v \geq \mathbf{0}$, $\frac{v^Tv}{\Omega} \leq v^TAA^Tv$. This is the same idea than in Perceptron, if $v$ is large then $||x|| = ||A^Tv||$ can not be too low.

Let introduce $f(t) = \frac{t^2}{2\Omega} - \log(t)$, from previous inequality it stands that $F(v) \geq \sum\limits_m f(v_m)$.

Now, $f$ is a single variable function which goes to infinity when $t$ goes to 0 ($t^2 \to 0$ but $-\log(t) \to \infty$) or to infinity ($t^2$ growths faster than $\log(t)$). So, $f$ has a minimum and so $F$ too (more precisely, as $f$ goes to $\infty$ on 0 or $\infty$, one could define a compact on which $F(v) \leq F(v_{start})$, but on this compact it has a minimum which is also the minimum of the function as the value if higher outside the compact). Let call them $f^*$ and $F^*$.

Importantly, if there is no $x$ such that $Ax \geq \mathbf{1}$ then, there exists $y$ such that $Ay = \mathbf{0}$ and $y > \mathbf{0}$, and, $F$ is not bounded (can go to $-\infty$). But, the existence of $x$ (assumption of linear separability) forces the existence of $F^*$ (thank to Cauchy because $\frac{v^Tv}{\Omega} \leq v^TAA^Tv$ for positive $v$).

As $f$, $F$ are smooth the minimums are characterized by a null derivative or gradient. $f'(t) = \frac{t}{\Omega} - \frac{1}{t}$, so, $f'(\sqrt{\Omega}) = 0$, so $f^* = f(\sqrt{\Omega}) = \frac{1}{2} - \frac{1}{2}\log(\Omega) \geq -\log(\Omega)$. Thus, the minimum of $F$ verifies $F^* \geq Mf^* \geq -M\log(\Omega)$.

So $F$ is bounded and $F(v_{start}) - F^* \leq \widetilde{O}(M\log(\Omega))$ for $v_{start} = \frac{1}{M}\mathbf{1}$.

### Normalization, linearization and lemmas

Independently, let remark that $\theta(t) = F(tv) = \frac{v^T AA^T v}{2}t^2 - \mathbf{1}^T\log(v) - M\log(t)$ is minimal when $v^T AA^T v = M$. So for any $w$, one could build a $v = \mu w$ such that $v^T AA^T v = M$ and $F(v) \leq F(w)$. In other words, it stands that $F\left(\sqrt{\frac{M}{v^T AA^T v}}v\right) \leq F(v)$.

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

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

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 $\psi(t) \geq \phi(t)$ for $t \geq 0$. 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.

### Convergence

Now, if $A_k A^T v \leq 0$ and $v^T AA^T v = M$, let introduce $w = v + \frac{vk}{v_k^2+1}\mathbf{1}_k$: this idea is again Perceptron based, one could increase $v$ and decrease $||A^T v||$ (precisely $F$ here) in the same times. Then $F(w) = F(v + \frac{vk}{v_k^2+1}\mathbf{1}_k) = F(v) + \frac{A_k A_k^T}{2}(\frac{vk}{v_k^2+1})^2 + A_k A^T v \times \frac{vk}{v_k^2+1} - \log(1+\frac{1}{v_k^2+1})$. But, $A_k A^T v \leq 0$ and $A_k A_k^T = 1$, so $F(w) \leq F(v) + \frac{1}{2}(\frac{vk}{v_k^2+1})^2 - \log(1+\frac{1}{v_k^2+1})$.

And, from lemmas just above, $F(w) \leq F(v) - \frac{1}{2v_k^2+2}$. But, $v_k \leq \sqrt{M\Omega}$, so, $F(w) \leq F(v) - \frac{1}{2M\Omega+2}$ which is impossible if $F(v) - F^* < \frac{1}{2M\Omega+2}$. So, $\forall v > \mathbf{0}$ such that $v^T AA^T v = M$, $F(v) - F^* \leq \frac{1}{2M\Omega+2} \Rightarrow AA^T v > \mathbf{0}$.

Finally, $\forall v > \mathbf{0}$, $F(v) - F^* \leq \frac{1}{2M\Omega+2} \Rightarrow F(\sqrt{\frac{M}{v^T AA^T v}}v) - F^* \leq \frac{1}{2M\Omega+2}$

$\Rightarrow \sqrt{\frac{M}{v^T AA^T v}}AA^T v > \mathbf{0} \Rightarrow AA^T v > \mathbf{0}$. This finishes the proof of the main theorem.

## Theorem 2 and binary complexity

This section deals with theorem 2 i.e. minimization of $F_A$.

It also deal with binary complexity i.e. complexity when elementary operations are not rational but binary. Currently, when representing variable as fraction of two infinite-size integer, using a specific rounding process allows to use an common denominator, plus, bounding the binary size of all numerators after each steps of the algorithm. The bound being $\widetilde{O}(M\log(\Omega))$, it results that the algorithm is polynomial times, with strongly polynomial steps having binary complexity $\widetilde{O}(LM^\gamma)$ with $M^\gamma$ the number of integer multiplication required to solve a $M \times M$ linear system. So, the global algorithm has time complexity of $\widetilde{O}(L)$ (in term of step) which is state of the art for linear separability and binary time complexity of $\widetilde{O}(L^2M^\gamma)$.

## Self concordance

Despite that all this paper relies on self concordance theory, there is nothing to prove to apply results from [10] recalled in appendix D to $F$ ($AA^T$ is positive because $v^TAA^Tv = ||A^Tv||_2^2 \geq 0$ and $F$ has a minimum see theorem 1).

So results recalled in in appendix D directly leads to state that damped Newton descent starting from any $v_{start}$ builds $v$ such that $F(v) - F^* \leq \frac{1}{2M\Omega+2}$ (i.e. a solution of the linear feasibility) in less than $\widetilde{O}(F(v_{start}) - F^* + \log\log(2M\Omega))$ steps.

But as proven in theorem 1, $F(v_{start}) - F^* = M\log(\Omega)$ for $v_{start} = \frac{1}{M}\mathbf{1}$. So, damped Newton descent builds a solution of the linear feasibility problem in less than $\widetilde{O}(M\log(\Omega) + \log\log(M\Omega))$ steps. Precisely, almost all the optimization is done in the so called phase 1 which may last $M\log(\Omega)$ because the phase 2 only requires $\log\log(M\Omega)$ steps i.e. it is negligible.

Using Cramer rules and Hadarmard bound (see appendix C), one can then proof that $\Omega \leq N2^{2NB\log(N)}$ with $2^B$ being a bound on each entry of $A \in \mathbb{Z}^{M\times N}$ before normalization - i.e. maximal binary size of all entries. This finally gives a $\widetilde{O}(NMB)$ number of steps, which becomes $\widetilde{O}(L)$ by reintroducing the total binary size.

This proves theorem 2.

## Binary complexity

For practical floating point implementation, the previous claim that complexity is $\widetilde{O}(L)$ Newton steps is sufficient. Yet, floating point implementation is impossible with too large numbers (and may lead to numerical issue even with not too large numbers).

So, from theoretical point of view, it is required to have also a bound on the binary complexity of each step when representing rational number as fraction of two infinite size integers - let recall that naive Gaussian elimination is exponential when not taking care about intermediary values.

For establishing binary complexity of each step, one should use approximation instead of square root and round the number after each damped Newton step without breaking the convergence. This is possible as presented bellow.

Computing $\lambda$ is impossible on $\mathbb{Q}$, but, as $F$ is convex, it holds that $F(v - \theta(\nabla_v^2 F)^{-1}(\nabla_v F)) \leq \frac{1}{2}(F(v) + F(v - \frac{1}{1+\lambda(v)}(\nabla_v^2 F)^{-1}(\nabla_v F)))$ if $\frac{1}{2}\frac{1}{1+\lambda(v)} \leq \theta \leq \frac{1}{1+\lambda(v)}$. So approximating $\lambda$ by a factor 2 approximation is sufficient and possible on $\mathbb{Q}$ (only twice number of steps are required compared to infeasible exact computation version). Then, with the same idea, normalizing $v^T AA^T v = M$ is not possible exactly, but, $\frac{M}{2} \leq v^T AA^T v \leq 2M$ is possible and still guarantees that $v \leq \widetilde{O}(M\Omega)$.

Importantly, finding $\mu$ such that $\mu \leq \sqrt{\rho} \leq \mu + 1$ is in $\log(\rho)$ (with bisection) which is too much here. But, finding $\mu$ such that $\mu \leq \sqrt{\rho} \leq 2\mu$ is in $\log(\log(\rho))$ because bisection can be done on power. So, computing approximation of $\lambda$ or normalization should be done using this fast bisection, and this way, binary complexity of this operation will be $\widetilde{O}(B)$ and not $O(B^2)$ (currently there are dedicated algorithms to compute square root, but, here, there are not even required - this may allows to find an 2 approximation of the optimal value instead of a fixed $\frac{1}{\lambda+1}$ for the same cost).

Finally, flooring variable could be dramatic due to the log but ceiling is acceptable. So at each step, one should ceil $v$: $v_{round} = int(Q * v + 1)/Q$ (after Newton and normalization). So, one has $F(v_{round}) \leq F(v) + 2|v - v_{round}|\mathbf{1}^T AA^T v$ (logs are not affected by ceiling) so $F(v_{round}) \leq F(v) + \frac{2}{Q}\mathbf{1}^T AA^T v$ but as $v$ is bounded by $O(M\Omega)$ and $A$ is normalized this leads to $F(v_{round}) \leq F(v) + O(\frac{M^2\Omega}{Q})$. By setting $Q$ such that this remainder is - at most - half the minimal decreasing during a damped Newton descent ($\frac{1}{4} - \log(\frac{5}{4})$ in phase 1), one could guarantee that the only twice the number of damped Newton steps are required to reach a solution while keeping a mastered infinite integer representation (with a common denominator $Q$ having a binary size limited by $\widetilde{O}(\log(M\Omega))$ and numerators with binary size also bounded by $\widetilde{O}(\log(M\Omega))$ as $v \leq O(M\Omega)$.

So, binary size of all numbers can be round to $\widetilde{O}(NB)$ at the end of each damped Newton step. Within a step, binary size will not exceed $\widetilde{O}(NMB)$ (because there is $O(M)$ multiplications). This leads to an algorithm without any numerical issue, and, steps being the resolution of a $M \times M$ linear system with entries with size $\widetilde{O}(NMB)$.

So, finally, for **floating point implementation** (i.e. with possible numerical instabilities and not arbitrary large numbers), the algorithm converges after solving less than $\widetilde{O}(M\Omega)$ linear systems of size $M \times M$.

And, using **exact infinite integer representation**, the algorithm converges after solving less than $\widetilde{O}(L)$ linear systems of size $M \times M$ with fraction coded by two integers with maximal binary value being $\widetilde{O}(L)$ using the offered rounding strategy. This leads to a $\widetilde{O}(N^\omega L^2)$ binary complexity (assuming fast multiplication of number of binary size $B$ is done in $\widetilde{O}(B)$ binary operations and solving a $M \times M$ linear system requires $\widetilde{O}(N^\omega)$ multiplications) for $A \in \mathbb{Z}^{M \times N}$ with maximal entry bounded (in absolute value) by $2^B$ and total binary size $L \approx NMB$.

# Perspective

## High Newton increments are good

As described in appendix D, during the first phase, $F(\mathcal{N}(v)) \leq F(v) - \lambda(v) + \log(1 + \lambda(v))$ (with $\mathcal{N}(v)$ the point after a damped newton step and $\lambda(v)$ the Newton increment).

As the Newton increment is **at least** of $O(1)$ and $F(v) \geq F^*$, then, it is impossible that phase 1 lasts more than $O(F(v_{start}) - F^*)$ otherwise one would have $F(v) - F^* \leq F(v_{start}) - F^* - O(F(v_{start}) - F^*)O(1) < 0$. Then, second phase which starts when $\lambda(v)$ is small ($\leq 0.25$) and is negligible (see complexity analysis).

So small $\lambda(v)$ is a good thing. But, interestingly, high $\lambda(v)$ is a good thing too because damped Newton step guarantee a decrease of at least $\lambda(v) - \log(1 + \lambda(v)) \approx \lambda(v)$ (when $\lambda(v) \geq 1$).

Indeed, if one consider the number of steps with $\lambda(v) \geq \sqrt{F(v_{start}) - F^*}$, each of these step decreases $F(v) - F^*$ by $\sqrt{F(v_{start}) - F^*}$. So, the first phase can not contain more than $\sqrt{F(v_{start}) - F^*}$ steps with $\lambda(v) \geq \sqrt{F(v_{start}) - F^*}$.

But, the question about whether $F(v) - F^*$ can be arbitrary large with $\lambda$ neither high or low (i.e. $\frac{1}{4} < \lambda(v) \ll \sqrt{F(v_{start}) - F^*}$) is not opened by this paper.

Yet if it was true that $\lambda(v) \geq \sqrt{F(v_{start}) - F^*}$ as long as $F(v) - F^* \geq \sqrt{F(v_{start}) - F^*}$, then, complexity would be $\widetilde{O}(\sqrt{L})$ steps instead of $\widetilde{O}(L)$ (because there could not exist more than $\sqrt{F(v_{start}) - F^*}$ steps after reaching $v$ such that $F(v) - F^* \leq \sqrt{F(v_{start}) - F^*}$).

## Linear feasibility for linear programming

Using common conversion process from linear separability to linear programming (appendix A), it holds that this algorithm can be used to solve general linear programming (without any assumption on input contrary to linear separability).

These conversion leads to an algorithm solving linear programming in $\widetilde{O}(NL)$ damped Newton steps.

This complexity is slightly higher than state of the art which is $\widetilde{O}(\sqrt{N}L)$ for central path log barrier [11, 10] and even less for [4, 3, 9] (but higher than just $\widetilde{O}(L)$).

So at first glance this algorithm is only good for linear feasibility (with complexity $\widetilde{O}(L)$) without processing linear feasibility instances like regular linear programming ones.

Yet, this algorithm may bypass negative result from [1]. Indeed, [1] states that any algorithm whose current point is locked in the neighborhood of the central path is not strongly polynomial.

Now, despite relying on self concordant theory, this algorithm seems not related to central path log barrier. Indeed, in this algorithm there is not tradeoff

between constraint and goal contrary to classical log barrier optimization leading to the concept of central path. But this is not the case for the offered algorithm: if the goal was to minimize $v^T AA^T v$ under constraint $v \geq \mathbf{0}$, then, the result would have just trivially been 0.

Thus, it seems possible that if one consider a linear program mapped into a linear feasibility, then, the point during optimization using the offered algorithm is not locked in the neighborhood of the central path.

# References


[1] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig. Log-barrier interior point methods are not strongly polynomial. *Journal on Applied Algebra and Geometry*, 2018.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2011.

[3] Sergei Chubanov. A polynomial projection algorithm for linear feasibility problems. *Mathematical Programming*, 2015.

[4] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, 2019.

[5] Corinna Cortes and Vladimir Vapnik. Support vector machine. *Machine learning*, 20(3):273–297, 1995.

[6] John Dunagan and Santosh Vempala. A simple polynomial-time rescaling algorithm for solving linear programs. *Mathematical Programming*, 114(1):101–114, 2008.

[7] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 2008.

[8] Thorsten Joachims. Svmlight: Support vector machine. *SVM-Light Support Vector Machine http://svmlight. joachims. org/, University of Dortmund*, 1999.

[9] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in o (vrank) iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*. IEEE, 2014.

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

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

[12] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 1958.

[13] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 2011.

[14] Shai Shalev-Shwartz and Nathan Srebro. Svm optimization: inverse dependence on training set size. In *Proceedings of the 25th international conference on Machine learning*, 2008.

# Appendix A: Common conversions

## Normalization

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 $\chi$ 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).

Let stress that normalizing is mainly for understanding - the offered algorithm works with the raw matrix but proof is a bit harder.

## 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 $AA^Tv > \mathbf{0}$ in $\widetilde{O}(L)$ Newton ($v$ is positive but this does not matter).

- Trivially, if $AA^Tv > \mathbf{0}$ then $A(A^Tv) > \mathbf{0}$. So $\mathbf{algo}_1(A) = A^T\mathbf{algo}_0(A)$ returns $\chi$ such that $A\chi > \mathbf{0}$ (when exists) 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 $(x \quad t) = \mathbf{algo}_1(\mathcal{A})$ 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 $2^B$, then, maximal determinant is lower than $2^{NB+N\log(N)}$ (Hadamard bound), so, the complexity of $\mathbf{algo}_3$ is just the one of $\mathbf{algo}_2$ an un input on which maximal binary size $B$ is increased by a factor $N$.

- Let any $A, b$ - **without assumption** - solving $Ax \geq b$ (or producing a certificate) 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 again 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 $\min\limits_{x \ / Ax \geq b} c^T x$ can be done with 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.

Let stress that the opposite way is trivial $\mathbf{algo}_5(AA^T, \mathbf{1}, \mathbf{0})$ is a correct implementation of $\mathbf{algo}_1(A)$ for any $A$.

## Appendix B: Perceptron

The goal of the Perceptron algorithm [12] is to find $x$ such that $Ax > \mathbf{0}$ (assuming such $x$ exists). Perceptron also assumes that $A$ has normalized rows and could consider $\Omega_A = \min_{\omega \ / \ A\omega \geq \mathbf{1}} \omega^T\omega$. Let write $\omega_A$ for the solution of this SVM [5] problem.

The key idea of Perceptron is to look for $x$ as $x = A^Tv$ with positive $v$, and, to remark that either $Ax > \mathbf{0}$ or it is possible both to increase $v$ and decrease $x^Tx$. But then, a Cauchy inequality linking $v^Tv$ (which increases) and $x^Tx$ (which decreases) would forces this process to converge.

Precisely, one could consider the algorithm $x_{t+1} = x_t + A_{i_t}^T$ with $A_{i_t}x_t \leq 0$ (possible if the algorithm has not reached $Ax > \mathbf{0}$) with $x_0 = \mathbf{0}$.

Let recall the Cauchy inequality: $\forall u, v, u^Tv \leq \sqrt{u^Tu \times v^Tv}$. Then, $(\omega_A^Tx_t)^2 \leq x_t^Tx_t\Omega_A$.

But $x_{t+1}^Tx_{t+1} = x_t^Tx_t + 2A_{i_t}x_t + A_{i_t}^TA_{i_t} \leq x_t^Tx_t + 1$ because $2A_{i_t}x_t \leq 0$ and $A_{i_t}^TA_{i_t} = 1$.

So, $x_{t+1}^Tx_{t+1} \leq t$.

While $\omega_A^Tx_{t+1} = \omega_A^Tx_t + \omega_A^TA_{i_t}^T \geq \omega_A^Tx_t + 1$ because $A\omega_A \geq \mathbf{1}$.

So, $\omega_A^Tx_t \geq t$.

So $(\omega_A^Tx_t)^2 \leq x_t^Tx_t\Omega_A$ becomes:

$$t^2 \leq (\omega_A^Tx_t)^2 \leq x_t^Tx_t\Omega_A \leq \Omega_A \times t$$

But, $t^2$ can not be lower than $\Omega_A \times t$ for $t \geq \Omega_A$, so the algorithm converges in less than $\Omega_A$ steps and returns some $x = A^Tv$ with $v \geq \mathbf{0}$.

Now, $\Omega_A$ can be large: classical tricks combining Cramer rules and Hadarmard bound allow to state that if the maximal value of all entries of $A \in \mathbf{Z}^{M\times N}$ (before normalization) is $2^B$, then, $\Omega_A \leq N2^{2BN\log(N)}$ (see appendix C for completeness), so this algorithm is exponential because it may require $\widetilde{O}(N2^{2BN\log(N)})$ steps. Inversely, the offered algorithm is *just* an updated version of this Perceptron such that the number of steps goes from $\Omega_A$ to $M\log(\Omega_A) = \widetilde{O}(NMB) = \widetilde{O}(L)$ thank to the encoding of this Perceptron idea into a self concordant function (a self concordant Perceptron).

Let stress that Perceptron steps are much faster than Newton ones: Perceptron step is just checking if $Ax_t > \mathbf{0}$ and if not $x_{t+1} = x_t + A_{i_t}^T$, so binary step complexity is just $\widetilde{O}(NMB)$ while damped Newton steps require to solve a $M \times M$ linear system. More generally, as *step* may have different complexity, it is important to distinguish *binary complexity* (total number of binary operations - unbiased) and *complexity* (number of steps - a step being a strongly polynomial times sub algorithm - like solving a linear system - biased).

# Appendix C: SVM

Assuming $\exists x \in \mathbb{Q}^N \ / Ax \geq \mathbf{1}$ with $A \in \mathbb{Q}^{M \times N}$ a normalized matrix, then, there exists a solution to the SVM problem [5] $\Omega_A = \min_{\omega \ / \ A\omega \geq \mathbf{1}} \omega^T \omega$, and, Perceptron will converge in $\widetilde{O}(\Omega_A)$ steps while the self concordant Perceptron in $\widetilde{O}(M \log(\Omega_A))$

Now, this value $\Omega_A$ should be characterized. Currently, it is not trivial to show directly that this $\Omega_A$ is small. But the classical trick is to consider the following theoretical problem (that nobody is going to solve) $\min_{\chi \ / A\chi \geq \mathbf{1}} \Upsilon((\chi \ \ A\chi - \mathbf{1})$ 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 $\chi_i$). 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 $A_I \chi = \mathbf{1}$, $\chi_J = \mathbf{0}$ with some set $I, J$. Thus, this system is full rank, and one could extract a square not singular matrix $\mathcal{A}$ from it, and, $\chi$ is the only solution of $\mathcal{A}x = \mathcal{B}$ with $\mathcal{A}$ being a sub matrix of $A$ (plus some Dirac vector) and $\mathcal{B}$ being 0-1 vectors. So Cramer rules hold, thus, each component of $\chi$ is a fraction of sub determinants of $A$. But, Hadamard bound (again) gives that if maximal entry of this matrix is $2^B$, then $\chi^T \chi \leq 2^{2NB \log(N)}$.

So, there exists $\chi$ such that $A\chi \geq \mathbf{1}$ and such that $\log(\Omega_A) \leq \log(\chi^T \chi) = \widetilde{O}(NB)$.

# Appendix D: Self concordant theory

This appendix only reports few results from [10] which could be considered for completeness.

A smooth single variable function $\phi$ is self concordant if and only if $|\phi'''(t)| \leq 2(\phi''(t))^{\frac{3}{2}}$. And, a smooth multi variable function $\Phi$ is self concordant if each ray $t \to \Phi(at + b)$ is self concordant (i.e. for all vectors $a, b$).

Mainly, self concordant functions are quadratic, linear, constant and $-\log$ ones. For example, the function $x \to x^T Q x + q^T x - \log(\alpha x - \beta)$ is a self concordant function for any vector $q, \alpha$, bias $\beta$ and any matrix $Q$ as soon as $Q$ is positive (not necessarily strictly positive, $\mathbf{0}$ is possible).

Now, if $G$ is a smooth self concordant function, let introduce $\lambda$ and the damped Newton descent:

$$\lambda_G(x) = \sqrt{(\nabla_x G)^T (\nabla_x^2 G)^{-1} (\nabla_x G)}$$

$$x = x - \frac{1}{1 + \lambda_G(x)} (\nabla_x^2 G)^{-1} (\nabla_x G)$$

The important property of Newton descent on self concordant functions is that: **if** $G$ has a minimum $G^*$, then, damped Newton descent starting from any $x_{start}$ allows to build $x$ such that $G(x) - G^* \leq \varepsilon$ in less than $\widetilde{O}(G(x_{start}) - G^* + \log\log(\frac{1}{\varepsilon}))$ damped Newton steps.

Precisely, [10] describes this optimization in two phases:

- while $\lambda(x) \geq \frac{1}{4}$, each damped Newton step decreases $G$ of at least $\frac{1}{4} - \log(\frac{5}{4})$, this is the so called phase 1.
- as soon as $\lambda(x) \leq \frac{1}{4}$, then, the two next statements hold

  $$G(x) - G^* \leq \frac{\lambda_G(x)^2}{2(1-\lambda_G(x))} \quad ; \quad \lambda(x - \frac{1}{1+\lambda_G(x)}(\nabla_x^2 G)^{-1}(\nabla_x G)) \leq 2\lambda(x)^2$$

  this is the so called phase 2 (with $G(x) - G^*$ and/or $\lambda_G(x)$ having a quadratic convergence).

Classically, damped Newton descent is considered in the phase 2 while it is considered in phase 1 in this paper (phase 2 being negligible as performed in $\widetilde{O}(\log\log(M\Omega_A))$.