Fun and Games with Sums of Squares

Boaz Barak

February 13, 2014

This blog post is an introduction to the "Sum of Squares" (SOS) algorithm from my biased perspective. This post is rather long - I apologize. If you'd rather "see the movie", I'll be giving a TCS+ seminar about this topic on Wednesday, February 26th 1pm EST. If you prefer the live version, I'll be giving a summer course in Stockholm from June 30 till July 4th. I've been told that there are worse places to be in than Stockholm in June, and there are definitely worse things to do than taking a course on analysis of Boolean functions from Ryan O'Donnell, which will also be part of the same summer school.

This post starts with an 1888 existential proof by Hilbert, given a constructive version by Motzkin in 1965. We will then go through proof complexity and semidefinite programming to describe the SOS algorithm and how it can be analyzed. Most of this is based on my recent paper with Kelner and Steuer and a (yet unpublished) followup work of ours, but we'll also discuss notions that can be found in our previous work with Brandao, Harrow and Zhou. While our original motivation to study this algorithm came from the Unique Games Conjecture, our methods turn out to be useful to problems from other domains. In particular, we will see an application for the *Sparse Coding* problem (also known as dictionary learning) that arises in machine learning, computer vision and image processing, and computational neuroscience. In fact, we will close a full circle as we will see how polynomials related to Motzkin's end up playing a role in our analysis of this algorithm.

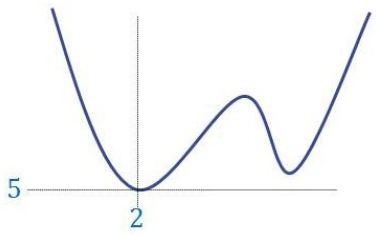
I am still a novice myself in this area, and so this post is likely to contain inaccuracies, misattributions, and just plain misunderstandings. Comments are welcome! For deeper coverage of this topic, see Pablo Parrilo's lecture notes, and Monique Laurent's monograph.

1 Sum of squares and the Positivstellensatz

One of the tedious exercises in high school mathematics involves finding the minimal value of a polynomial such as

$$P(x) = x^4 - 24x^3 + 185x^2 - 484x + 409 \; . \tag{1}$$

To solve this, we were taught to go over all critical points (where the derivative vanishes) and check their values, so eventually we get a picture like this



and can deduce that the minimum is achieved at the point $P(2) = 5$.

As theoretical Computer Scientists, we can make this feeling of tedium precise— this method involves exhaustive search over all local optima and hence can take $\exp(n)$ time for an n-variate polynomial, even if it is only of degree 4. Unfortunately, in general, we do not know of a better way. Indeed, in practice people use generalizations of the high school method such as Gradient Descent to solve such problems, and unsurprisingly often these algorithms do not return the global minimum but get stuck at some local optima. In fact, we know one cannot do better in the worst case, as one can easily transform an n-variable 3SAT formula φ into an n-variable degree-4 polynomial $P_\varphi = Q_\varphi(x) + M\sum(x_i^2 - x_i)^2$ which will have 0 as its global minimum if and only if φ is satisfiable. (For every $x \in \{0,1\}^n$, the polynomial $Q_\varphi(x)$ will equal 0 iff x satisfies φ, while for a large enough value of M, the second term will guarantee that the minimum can only be achieved at a point $x \in \{0,1\}^n$.)

However, some times there is a nicer, more global way to argue about the extremal points of polynomials. For example, one can see that the minimum of the polynomial $P(x)$ in (1) is equal to 5 by writing it as

$$P(x) = 5 + (x-2)^2(1 + (x-10)^2) \; ,$$

using the simple (but deep!) fact that a square of a number is never negative.

Bruce Reznick paraphrased Jane Austen to say

> It is a truth universally acknowledged, that a mathematical object whose orderings are non-negative must be in want of a representation as a sum of squares.

Despite this saying, the 3SAT example above implies that (assuming NP≠coNP) there exists a non-negative polynomial that is not equal to a sum of squares (SOS) of other polynomials, as otherwise we could have had a short proof of unsatisfiability for every 3SAT formula. However proving the existence of such a polynomial unconditionally is not so trivial. Hilbert first gave a non-constructive proof in 1888, and it took almost 80 years until Motzkin gave an explicit example: the Arithmetic-Mean Geometric-Mean (AMGM) inequality implies that the polynomial

$$M(x,y) = (x^4y^2 + x^2y^4 + 1)/3 - x^2y^2 \tag{2}$$

is always non-negative since the last term is the geometric mean of x^4y^2, x^2y^4, and 1, but it turns out that it does not equal a sum of squares. (See Reznick's or Schmudgen's surveys for the proof and more on the history of this problem.)

However, we cam still prove Motzkin's polynomial is non-negative via a simple and short "SOS proof" since it is the sum of squares of four *rational* functions:

$$M(x,y) = \frac{x^4y^2(x^2+y^2-2)^2}{3(x^2+y^2)^2} + \frac{x^2y^4(x^2+y^2-2)^2}{3(x^2+y^2)^2} + \frac{(x^2+y^2-2)^2}{3(x^2+y^2)^2} + \frac{(x^2-y^2)^2}{3(x^2+y^2)^2} \ .$$

Already Hilbert was aware of this notion of SOS proofs, and asked as his 17th problem whether we can always certify the non-negativity of any polynomial by such a proof. Through works of Artin (1927), Krivine (1964), and Stengle (1974), we know the answer is a resounding *yes!*. These results, which serve as the foundation of real algebraic geometry, are known as the *Positivstellensatz* and hold for even more general polynomial equalities and inequalities. In particular, given some polynomials $P_1, \ldots, P_m : \mathbb{R}^n \to \mathbb{R}$, the set of equations

$$P_i(x) = 0 \quad i = 1 \ldots m \tag{3}$$

is unsatisfiable if and only if this can be certified by finding m polynomials $Q_1, \ldots, Q_m$ and a sum of squares polynomial S such that

$$-1 = S(x) + \sum_{i=1}^{m} P_i(x)Q_i(x) \ . \tag{4}$$

(The Positivstellensatz applies to polynomials *inequalities* as well, but in the context of optimization, one can always transform an inequality $P(x) \geq 0$ into the equality $P(x) = y^2$ by introducing an auxiliary variable y. Even without using inequalities, the Positivstellensatz fundamentally relies on the fact that the real numbers, as opposed to complex numbers, are well ordered.)

2 Proof systems and algorithms

As Theoretical Computer Scientists, we typically try to make *qualitative* questions into *quantitative* ones, and indeed in 1999 Gregoriev and Vorobjov defined the *Positivstellensatz proof complexity* of a set of equations as the minimal *degree* needed for

the polynomials S and $Q_1, \ldots, Q_m$ in (4). Note that a degree d SOS proof can always be written down using $n^{O(d)}$ coefficients, and hence the 3SAT example suggests that at least some equations require a large (i.e., $\Omega(n)$) degree, and such a result was indeed shown by Grigoriev in 2001.

Bounds on the degree turn out to be very important for optimization as well. Several researchers, including N. Shor, Y. Nesterov, P. Parillo, and J. Lasserre, realized independently that it is possible to search for a degree d SOS proof in time $n^{O(d)}$. This follows from a correspondence between polynomials that are sums of squares and positive semidefinite (p.s.d.) matrices; combined with the fact that the latter convex set has an efficient separation oracle, and hence can be efficiently optimized over. (This is known as semidefinite programming.) Indeed, for any n-variable degree-$2d$ polynomial S we can define an $n^d \times n^d$ matrix $B = B(S)$ such that for every $x \in \mathbb{R}^n$, $B \cdot x^{\otimes 2d} = \sum_{i_1,\ldots,i_{2d} \in [n]} B_{i_1,\ldots,i_{2d}} x_{i_1} \cdots x_{i_{2d}} = S(x)$. (As written this assumes S is a homogenous polynomial, but this can be suitably generalized to the non-homogenous case as well.) By definition, the matrix B is p.s.d. if and only if it equals $\sum_{i=1}^{r} A_i^{\otimes 2}$ for some vectors $A_1, \ldots, A_r \in \mathbb{R}^{n^d}$. Therefore one can see that if B is p.s.d. then each one of those A_i's defines a degree d polynomial R_i such that $S(x) = \sum R_i^2(x)$ for every x. The other direction works in a similar way.

So, we can efficiently certify the unsatisfiability of a set of polynomial equations if it has a low degree SOS proof. But under what conditions will it have such a proof? and what about finding solutions for *satisfiable* polynomial equations?

Progress on these questions has been quite slow. For computational problems of interest, degree upper and lower bounds have both been very hard to come by. We essentially knew only one degree lower bound— Gregoriev's result mentioned above (later rediscovered by Schoenebeck and expanded upon by Tulsiani). As for upper bounds, until very recently we essentially had none, in the sense of having no significant algorithmic results using the SOS algorithm that did not already follow from weaker algorithms. However, some recent results, including the quasipolynomial time quantum separability algorithm of Brandao, Christiandl and Yard, and our results on solving "hard" unique games instances, gave some signs that the SOS framework does have the potential to solve problems beyond the reach of other techniques. In a very recent work with Kelner and Steurer, we show a general way to exploit the power of SOS, which I will now describe.

3 Pseudoexpectations and Combining algorithms

Our approach for using SOS to solve computational problems is based on the following wise proverb

It is easier to solve a problem if you already have a solution.

To make this more concrete, fix the set of polynomial equations (3). A *combining algorithm* is an algorithm A that takes as input a (multi) set X of solutions for (3), and outputs a single solution x^*. Based on the proverb above, I guess most readers of this blog would be able to come up with such an algorithm. Now, to make things more challenging, imagine that A does not get X represented as a list of all its elements (which, after all, could be exponentially large), but rather only gets some *low order statistics* of X. That is, for some smallish d (say $d = O(1)$ or $d = polylog(n)$), A gets a vector $v \in \mathbb{R}^{n^d}$ such that for every $i_1, \ldots, i_d \in [n]$, $v_{i_1,\ldots,i_d} = \mathbb{E} X_{i_1} \cdots X_{i_d}$ (where we identify X with the distribution over its elements). This is indeed a harder task, but it still seems much easier than solving the problem from scratch. Surprisingly, it turns out that you can often reduce solving the equations to constructing such a combining algorithm. Our approach works as follows— given a combining algorithm A, instead of giving it moments of an actual distribution X over solutions (which of course we don't have), we feed it with a "fake moment" vector v. If we're lucky, A won't notice the difference and will still output a good solution x^*.

How do we construct those "fake moments"? and when will we be lucky? Those "fake moments" are more formally known as *pseudoexpectations*, and can be found using a semidefinite program which is the dual to the SOS program. In particular, they will obey some of the consistency properties of actual moments, most importantly that for every degree $\leq d/2$ polynomial R, if we combine the fake moments given by v linearly to compute the presumed value of $\mathbb{E} R^2(X)$ then it will be non-negative. Those properties imply that if we have a *proof* that A works as combining algorithm, and that proof can be encapsulated in the SOS framework with not too high a degree, then that proof in fact shows that A will still output a good solution even when it is fed the fake moments.

4 Machine learning applications: the sparse coding and sparse vector problems

To make things more concrete, we now sketch a concrete example, taken from an upcoming paper with Kelner and Steurer. One of the challenging tasks for machine learning is to find good *representations* for data. For example, representing a picture as a bitmap of pixel works great for projecting it on a screen, but not is not as well suited for trying to decide if it's a picture of a cat or a dog. Finding the "right" representation for, say, images, is often a first task not just in learning applications, but also for other tasks such as edge detection, image denoising, image completion and more. *Sparsity* is one way to capture "rightness" of representation. For example, it is often the case that the data is sparse when represented in the "right" linear basis

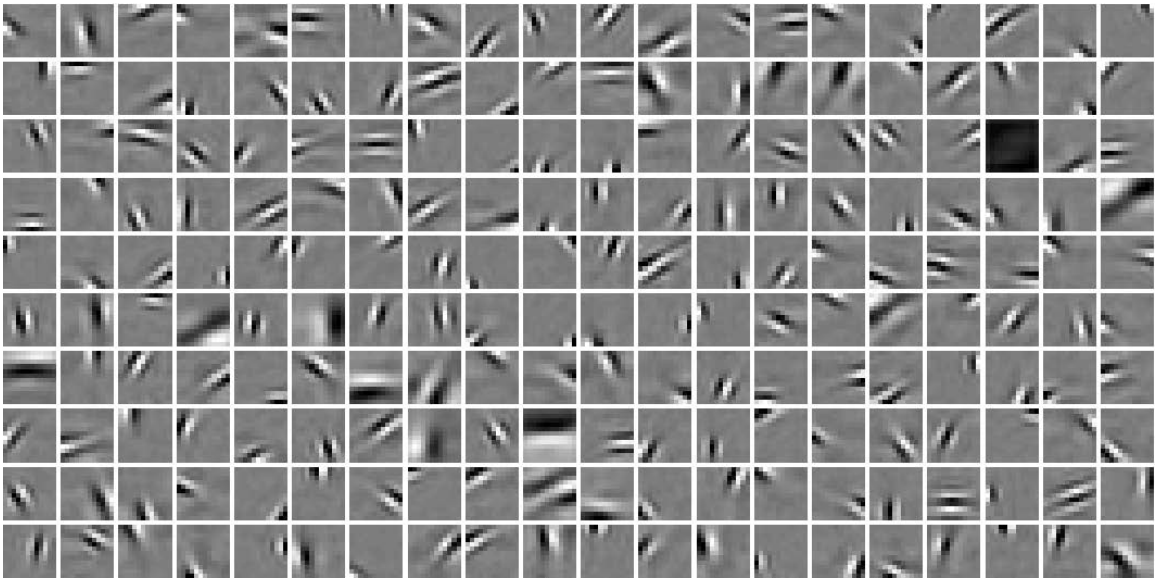


Figure 1: The basis/dictionary recovered by Olshausen and Field that makes natural images sparse. We abuse the word "basis" slightly since it is overcomplete- has more basis elements than dimensions - and so contains some linear dependencies.

(such as the Fourier or wavelet bases). Olshausen and Field (1997) argued that this may be the way that images are represented by in visual cortex, and they used a heuristic alternating-minimization based algorithm to recover a basis (also known as a *dictionary*) for natural images (see Figure 2).

But of course as theorists we are not satisfied with a heuristic that works fast and achieves good results. We want a slower algorithm with worse performance that we can prove something about. This is where Sum of Squares comes to the rescue. (At least on some of the counts: it is definitely slower, and we can prove something about it; however "unfortunately" beyond just amenability to analysis there is an (unverified) chance that it actually outperforms the heuristics on some instances, since at least in principle it can avoid getting stuck in local minima.) Recently there have been several works giving rigorous guarantees for sparse coding (e.g., [SWW12 , AGM13, AAJNT13]) but all of them require the representations to be "super-sparse"- have less than $\sqrt{n}$ significant non-zero coordinates. (There are some works handling the denser case such as [FJK96, GVX13 , ABGM14] , but they make rather strong assumptions on the dictionary and/or distribution of samples.) In our new work, we use the SOS algorithm to recover the dictionary even as long as the number of nonzero coefficients is less than δn for some sufficiently small constant $\delta > 0$. (For these parameters our algorithms takes quasipolynomial time; it takes polynomial time if the number of nonzero coefficients is at most $n^{1-\epsilon}$ for an arbitrarily small $\epsilon > 0$.)

We now sketch the ideas behind the solution. For simplicity of notation, lets assume

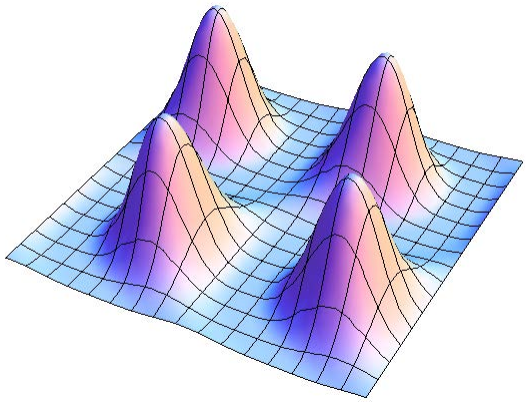


Figure 2: We construct an optimization problem whose optima correspond to the dictionary vectors. We then show that, despite its non convexity, the SOS algorithm can still find a global optima of the problem.

that the "right basis" is in fact an orthonormal basis $a^1, \ldots, a^n$, and that we obtain examples of the form $y = \sum W_i a^i$, where the W_i's are independently and identically distributed random variables with $\Pr[W_i = 0] = 1 - p$ and $\Pr[W_i = +1] = \Pr[W_i = -1] = p/2$ for some $p \ll 1$. Like the works [AGM13, AAJNT13], our methods apply to much more general settings, including non-independent W_i's and overcomplete dictionaries, but this is a good case to keep in mind.

We focus on the task of approximately recovering a single basis vector: once you have such a subroutine, recovering all vectors is not that hard. We do so in three steps:

1. Using the examples observed, we construct a system $\mathcal{E}$ of polynomial equations. We argue that any solution x to $\mathcal{E}$ is a good solution for us (namely close to one of the basis vectors).

2. On its own, phrasing the question as a polynomial system is not necessarily helpful, since $\mathcal{E}$ is not convex, and so in general we don't know how to solve it. However, we do show how to solve the easier problem of coming up with a *combining algorithm* A, that given the low order moments of a distribution over solutions of $\mathcal{E}$, manages to recover a single solution.

3. We then verify that all our arguments in Steps 1 and 2 can be encapsulated in the SOS proof system with a low degree, and hence A will still work even when fed "fake moments", thus establishing the result.

We now describe how to implement those three steps.

Step 1: The system of equations. Given examples $y^1, \ldots, y^s$ each one of the form $\sum W_i a^i$, we construct the polynomial $P(x) = \frac{1}{s}\sum_{i=1}^{s}\langle y^i, x\rangle^4$, and consider the equations $\mathcal{E}$ defined as follows:

$$P(x) \geq p$$
$$\|x\|_2^2 = 1$$

(As mentioned before, we can always translate the inequality into an equality by adding an auxiliary variable.)

As s grows, P converges to the polynomial

$$\mathbb{E}\left(\sum W_i \langle a^i, x\rangle\right)^4 . \tag{5}$$

It suffices to take $s = poly(n)$ to get good enough convergence for our purposes, and so we will just assume that P is equal to (5). Opening the parenthesis, and using the fact that $\mathbb{E}W_i = 0$ and $\mathbb{E}W_i^2 = \mathbb{E}W_i^4 = p$, we see that

$$P(x) = p\sum_{i=1}^{n}\langle a^i, x\rangle^4 + 2p^2\left(\sum\rangle a^i, x\langle^2\right)^2$$

which equals

$$p\sum_{i=1}^{n}\langle a^i, x\rangle^4 + 2p^2\|x\|^4$$

since the a^i's are an orthonormal basis.

Note that $P(a^i) \geq p$ for all i, and hence the system $\mathcal{E}$ is feasible. Moreover, if $P(x) \geq p$ and $\|x\|^2 = 1$ then

$$p \leq p\sum_{i=1}^{n}\langle a^i, x\rangle^4 + 2p^2 \leq p\max_i\langle a^i, x\rangle^2\sum\langle a^i, x\rangle^2 + 2p^2\|x\|^2 = p\max_i\langle a^i, x\rangle^2 + 2p^2 \tag{6}$$

which means that

$$\langle a^i, x\rangle^2 \geq 1 - 2p$$

for some i.

Step 2: Combining algorithm I will describe a particularly simple combining algorithm that will involve moments up to logarithmic order, and hence result in a quasipolynomial time algorithm. Under some conditions (namely $p \leq n^{-\epsilon}$) we are able to give a polynomial-time algorithm.

The combining algorithm gets access to a distribution X over solutions of $\mathcal{E}$, and needs to output a single solution. We do so by choosing a set $U = \{u^1, \ldots, u^\ell\}$ of random gaussian vectors for $\ell = O(\log n)$, and consider the matrix M^U defined as

$$M^U_{i,j} = \mathbb{E}f_U(X)X_iX_j \tag{7}$$

where $f_U(x) = \prod_{u \in U} \langle u, x \rangle^2$. We output the top eigenvector of M^U.

We claim that with probability $\exp(-O(\ell))$, this vector will be highly correlated with one of the a^i's. To see this, note that every element in the support of X is very close to one of the a^i's. In particular, there is an i such that X is close to a^i with probability at least $1/n$. We can assume $i = 1$ without loss of generality, and so X is a convex combination of two distributions X' and X'' such that every element in the support of X' is close to a^1.

Note that for any unit vector x, $\mathbb{E}\langle u, x \rangle^2 = 1$ for a standard Gaussian u and hence $\mathbb{E} f_U(x) = 1$. Now let us condition on the event that all of the u^i vectors satisfy $\langle u^i, a^1 \rangle^2 \geq 2$; this will happen with $\exp(-O(\ell))$ probability. This would mean that $f_U(x)$ is roughly 2^ℓ for x that is close to a^1, while it has a much lower value for x's that are not close to it. Thus, almost all the weight in (7) is on the x's close to a^1, meaning that M^U is roughly equal to a constant times $(a^1)^{\otimes 2}$, and in particular its top eigenvector will be close to a^1.

Step 3: Lifting to pseudoexpectations. Step 3 is in some sense the heart of the proof, moving from a combining algorithm, that requires an actual distribution (which we don't have), to a rounding algorithm, that only needs a pseudo-distribution (which we can find using semidefinite programming). However, it is also the most tedious, since it involves going over the steps of the proof one by one, verifying that each one only used SOS arguments. However, occasionally "lifting" the proofs for pseudoexpectations requires more care. Rather than giving the full proof here, we show just one example of how we deal with one of those more subtle cases.

Recall that while deriving (6), we used the following simple inequality:

$$\sum_{i=1}^{n} \alpha_i^4 \leq \max \alpha_i^2 \sum_{j=1}^{n} \alpha_j^2 \tag{8}$$

(substituting $\langle a^i, x \rangle$ for α_i). One challenge in lifting this inequality to pseudo-expectations is that the max operation is not a low degree polynomial. Since the L_∞ norm can be approximated by the L_t norm for large t, it turns out it suffices to prove the statement

$$\sum_i \alpha_i^4 \leq \left(\sum_i \alpha_i^{2t} \right)^{1/t} \sum_j \alpha_j^2 \ ,$$

for some large t. Of course to make this a polynomial statement, we need to raise this to the t^{th} power and get

$$\left(\sum_i \alpha_i^4 \right)^t \leq \left(\sum_i \alpha_i^{2t} \right) \left(\sum_j \alpha_j^2 \right)^t \ . \tag{9}$$

(9) becomes non-trivial to prove in SOS for $t \geq 3$ so let us focus on the $t = 3$ case. The LHS has the form

$$\sum_{i,j,k} \alpha_i^4 \alpha_j^4 \alpha_k^4$$

The RHS has the form

$$\sum_{i,j,k,\ell} \alpha_i^2 \alpha_j^2 \alpha_k^2 \alpha_\ell^6 \ .$$

Fixing some particular i, j, k, and dividing both sides by $\alpha_i^2 \alpha_j^2 \alpha_k^2$ we see we need to show that

$$\alpha_i^2 \alpha_j^2 \alpha_k^2 \leq \sum_\ell \alpha_\ell^6 \ . \tag{10}$$

(10) follows from

$$\alpha_i^2 \alpha_j^2 \alpha_k^2 \leq \tfrac{1}{3} \left(\alpha_i^6 + \alpha_j^6 + \alpha_k^6 \right) \tag{11}$$

but this is just our old friend the AMGM inequality $(abc)^{1/3} \leq (|a| + |b| + |c|)/3$. We might worry that this is very related to Motzkin's polynomial, whose claim to fame is exactly the fact that despite being non-negative, it is not a sum of squares. There are two answers to this concern. The first one is that we don't care, since Motzkin's polynomial does have a low degree SOS proof (using its representations as a sum of squares of rational functions) and this is good enough for us to show that it holds for pseudo-expectations as well. The second one is that, at least in my experience, if you stare at (11) long enough, then eventually you get tired of staring, look up and realize you are in a workshop on semidefinite programming, where you can easily find an expert or two that will show you why the RHS minus the LHS is actually a sum of squares. (Apparently, this was proven by Motzkin; (11) was probably his failed first attempt at an explicit example for Hilbert's theorem.) Thus (11) holds if the α_i's come from a pseudo-expectation, which is what we wanted to prove. (Going over the proof, one can see that it can extend into a much more general settings of both the dictionary and the distribution.)

Conclusion

While until recently we had very little tools to take advantage of the SOS algorithm (at least in the sense of having rigorous analysis), we now have some indications that, when applied to the right problems it can be a powerful tool, that may may be able to goals that resisted previous attempts. We have seen some examples of this phenomenon, but I hope (and believe) the best is yet to come, and am looking forward to seeing how this research area develops in the near future.

Acknowledgements. Many thanks to David Steurer for great suggestions, corrections, and insights that greatly improved this blog post, as well as for patiently explaining to me for the nth time the difference between the Positivstellensatz and Nullstellensatz. Thanks to Amir Ali Ahmadi, Jon Kelner, and Pablo Parillo for showing me the SOS proof for (11) in the ICERM workshop on semidefinite programming and graph partitioning.