Princeton CS talks

Optimization over Nonnegative Polynomials November 12, 2015

Lecturer: Amir Ali Ahmadi Scribe: Kiran Vodrahalli

Contents

1 Overview

We introduce the importance of optimizing over nonnegative polynomials instead of simply sum-of-squares-expressible polynomials. We show that we can use LPs and second-order conic programming to search over a smaller search space than SOS-class of polynomials, which can be solved with SDPs (which have higher time complexity). Optimizing over these classes of non-negative polynomials efficiently gives useful certificates of optimality in a wide variety of situations, particularly in controls (parametrize the Lyapunov function by polynomials). We also introduce a new paradigm of solving optimization problems equipped with a dynamical system, where you want to optimize over the set of points which never leaves the system.

2 Introduction

We want to optimize over non-negative polynomials: The coefficients are decision variables of problem. We have to pick them in some convex set so that the polynomial is non-negative for all inputs.

2.1 Why do this?

The most obvious reason is polynomial optimization:

$$\min_x p(x)$$

with $f_i(x) \leq 0, h_i(x) = 0$ are linear constraints.

We can trivially formulate as finding the maximum constant γ such that $p(x) - \gamma \geq 0$. γ is the largest number we can subtract and still be nonnegative. We are looking for γ such that the polynomial becomes nonnegative on a basic semialgebraic set.

We would also like to give optimality certificates in discrete optimization, for instance finding the independent set number of a graph: the largest number of nodes you can pick in a graph such that no two nodes are connected. For instance, solving this problem solves course scheduling optimization of final exams. How do you prove you cannot do better? If you solve any optimization problem, you need to find high-quality solution, and you need to prove you cannot do better. What is the connection to nonnegative polynomials? It turns out proofs of nonnegativity of polynomials give you certificates of optimality.

We have $a(G) \leq k$ iff $-2k \sum_{i,j} x_i x_j y_i y_j - (1-k)(\sum_i x_i^2)(\sum_i y_i^2)$ is nonnegative (this is also a direct proof of NP-hardness of the problem).

We also are interested in regression problems in stats and machine learning. Suppose we want to fit polynomials that minimizes least square error that has shape considerations (monotone in a given direction, convex). We want a regressor to also satisfy these constraints. For instance we want the Hessian $\nabla^2 p(x)$ is nonnegative (PSD). By optimizing over non-negative polynomials that satisfy this, you get this convexity for free, which is nice!

We can also do clustering with semialgebraic sets. For instance, we may want to find an ellipsoid with minimal volume.

The last class of applications is to dynamical systems and control. The idea is you have a dynamical system and you want to know collective behavior of the trajectories as a whole. I may want to know whether something attracts all nearby trajectories, whether I will remain in a certain trajectory. Can I guarantee collision avoidance? It turns out you can convert these questions into optimization questions by using Lyapunov theory; searching functions for non-negativity constraints. If we parametrize these Lyapunov functions as polynomials, we can search over these directly with non-negativity constraints.

2.2 How to Prove Nonnegativity?

If you can write your polynomial as a sum of squares of polynomials, this gives you non-negativity (of course). If there is at least one, we say that it is SOS. We are immediately convinced it is non-negative if we can write it in this form. It extends to the constrained case. We try to write p as a sum of squares + summation sum of squares of nonnegative polynomials: $p(x) = \sigma_0(x) + \sum \sigma_i(x) g_i(x)$ with $\sigma_i(x)$ is SOS and we have this implies $p(x) \geq 0$ on $\{x | g_i(x) \geq 0\}$, a semialgebraic set of constraints. This is really fundamental work in algebra. The search for algebraic certificates gives you SDPs!

Theorem 2.1. *Polynomial $p(x)$ of degree $2d$ is SOS iff there exists a Gram matrix Q which is PSD, with $p(x) = z(x)^T Q z(x)$ where $z = [1 x_1, \cdots x_n, x_1 x_n, \cdots, \cdots, x_n^d]$.*

3 Optimization → Controls

In dynamical systems, we are given a differential equation $dx/dt = f(x)$ where $f : \mathbb{R}^n \to \mathbb{R}^n$, $f(0) = 0$. In local stability analysis, you typically Taylor expand so these are polynomial approximations. In ecology, these are directly polynomial, in robotics you Taylor expand since you are interested in local asymptotic stability in that domain.

Definition 3.1. Local stability.
Is there a ball around the origin such that you will eventually go to the origin?

Definition 3.2. Global stability.
Will you go to origin from anywhere?

We are interested in knowing the complexity of answering these decision questions for a given dynamical system.

Theorem 3.3. *If you define a Lyapunov function such that $V(x) : \mathbb{R}^n \to \mathbb{R}$ such that $V(x) > 0$, $V(x) \leq \beta \to dV/dt < 0$ implies that $\{x : V(x) \leq \beta\}$ is in the region of attraction. The amazing thing is that the converse of this statement is also true. If the system is locally stable, there exists some Lyapunov function that satisfies this. We would like to parameterize these guys as polynomials and automatically search for them via semidefinite programming.*

3.1 Complexity of deciding asymptotic stability?

For linear systems, this is decideable in polynomial time: The answer is it is stable iff A is Hurwitz. If degree(f) > 1, this is still open. However, if it is stable, there exists a polynomial Lyapunov function, together with a computable bound would imply that this problem is decidable. However there is a simple counterexample, which is only slightly non-linear. If you go to degree 2 vector field you lose. It is $dx/dt = -x + xY, dy/dt = -y$. This system is globally asymptotically stable. By proof, take $V(x, y) = \ln(1+x^2)+y^2$. Then we claim no polynomial Lyapunov function of any degree exists. Thus, decidability still remains open. A few years ago, I proved a much weaker result: Deciding asymptotic stability of cubic vector fields is strongly NP-hard. Deciding asymptotic stability of cubic homogenous vector fields is strongly NP-hard. This just means that $dx/dt = f(x), f(\lambda x) = \lambda^d f(x)$. Then all monomials in f have the same degree, and you can reduce from one-in-three 3SAT. You have to assign $0, 1$ values such that each clause has exactly one one and two zeros, such that the whole formula evaluates to 1. The goal is to design a cubic differential equation such that choosing its variables maps to one-in-three 3SAT. Instaed of relating to a solution of an ODE, we relate to a candidate Lyapunov function, which proves things for us as an intermediary. The reduction is: One in Three 3SAT is satisfiable only if there is a zero. Otherwise it will be positive everywhere. A standard trick is to homogenize this polynomial: every monomial will have the same degree. Then the final vector field is just a gradient flow for this polynomial. If the gradient is 0 at some point (if there exists a zero), you will never go to the origin! Thus if the polynomial has a 0, the system will never be globally stable.

We change the conditions on $V(x)$ to sum of squares conditions instead of non-negativity conditions, and then SDPs make the search over polynomial-parametrizations of $V(x)$ automatic.

Applying this method to find a control on an inverted pendulum. Finding a cubic approximation does better than standard LQR (linear approximation) and improves upon the robustness, just by using SDP to find the Lyapunov function.

However, scalability is often a real challenge. (Most people don'ot have more than $6 - 8$ states unless you are exploiting additional structure in the problem).

However, our approach is not work with SOS to being with, let us optimize over an inner approximation, the sum of squares polynomials - but let's go even further to find even more tractable convex sets to optimize over. It is not immediately clear what sets you can pick. All polynomials are sums of 4^{th} powers of polynomials, and all polynomials that are sums of 3 quares of polynomials. These are both inside the SDP cone. We define these as DSOS and SDOS polynomials.

Definition 3.4. Diagonally dominant sum of squares.
$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} \beta_{ij}^+ (m_i + m_j)^2 + \beta_{ij}^- (m_i - m_j)^2$ where m_i is a monomial for nonegative coefficients α, β.

Definition 3.5. Scaled diagonally dominant sum of squares.
$p = \sum_i \alpha_i m_i^2 + \sum_{i,j} (m_i \beta_{ij}^+ + m_j \gamma_{ij}^+)^2 + (m_i \beta_{ij}^- - m_j \gamma_{ij}^-)^2$ where m_i is a monomial for nonegative coefficients $\alpha > 0, \beta, \gamma$.

Definition 3.6. Diagonally and Scaled Diagonally Dominant.
A symmetric matrix is diagonally dominant if $a_i i \geq \sum_{j \neq i} |a_{ij}|$, these are PSD by Gershgorin circle theorem. A symmetric matrix A is scaled diagonally dominant if there exists a diagonal matrix such that DAD is diagonally dominant.

You can do optimization over DD with **linear programming**, and SDD with **second order cone programming**. These are where the savings come from, we get better more tractable problems.

Definition 3.7. A polynomial p is DSOS iff $p(x) = z^t(x)Qz(x)$ where Q is diagonally dominant. Analagously for SDOS, Q must be scaled diagonally dominant.

The Motzkin polynomial is the simplest polynomial that is non-negative but is not expressed as a sum of squares. This polynomial is 2-DSOS. This just means that we take a sum of **squares** of each of the monomials (multiplied by constants). In general r-DSOS means you take monomial to the power of r and write as a sum of coefficients. Thus you can use LP to determine non-negativity.

Theorem 3.8. *For any positive definite form p, there exists an integer r and a polynomial q of degree r such that q is DSOS and pq is DSOS.*

Thus you can always do polynomial optimization using hierarchies of LP and second-order conic programming. They use a powerful result from algebraic geometry: If a basic semialgebraic set is empty, there is always a proof of a certain type. You can write -1 as a sum of polynomials times equality constraints and SOS polynomials times inequality constraints (think of this as a generalization of Lagrange multipliers with polynomials where equality constraints and inequality constraints get different "coefficients" where "coefficients" are polynomials.

As for empirical results, you approximate the SDP solution with SDOS and get 80% of the search space, which is more than good enough for the time complexity improvement. We show applications to robotics (ATLAS stabilization) - this is much better than what could be done before. Perhaps you could make even simpler relaxations (to linear for instance) if you are only interested in trajectories. There are also interesting applications to formal proof verification.

4 Controls → Optimization

I will talk about a new direction of research I'm excited about. Here we discuss robust-to-dynamics optimization (RDO) problems. Here is a problem very much motivated by controls. We solve a constrained optimization problem today, but there is some external dynamics which make your decisions move your optimal point out of the feasible set. You want your initial decision to be safe enough to disallow this from happening. For instance, drug design: Concentrations and stuff may change in your body after you put it in your mouth. This reaction may change your decision: You want to find the cheapest drug today that for **all time** satisfies a certain concentration (chemical) property that makes it safe for you.

Definition 4.1. RDO.

1. $\min_x \{f(x) : x \in \Omega\}$
2. $x_{k+1} = g(x_k)$ or $dx/dt = g(x)$.

The RDO is then $\min_{x_0} \{f(x_0) : x_k \in \Omega, \forall k; x_{k+1} = g(x_k)\}$. You select a dynamics and an optimization and analyze.

For instance, robust to linear dynamics linear programming (R-LD-LP). We want our LP optimization to be satisfied as our system updates. In other words: $\min_x\{c^T x : Ax \leq b, AGx \leq b, AG^2x_1 \leq b, \cdots, AG^r x \leq b\}$ where G is your linear dynamics update. Solutions cannot exist in your ooriginal set. Does it always happen that you can stop at a finite point?

Lemma 4.2. *The final feasible set of R-LD-LP is closed, convex, and invariant. However it may not be polyhedral. This is obvious: Check* $\cap_i^r\{x : AG^i x \leq b\}$.

As you intersect these sets, if it turns out that $S_r = S_{r+1}$, then you can finish: For proof, see $x \in S_r \to x \in S_{r+1}$, then $Gx \in S_r$, and you can repeat this. Note that this conddition is checkable with an LP.

Theorem 4.3. *If spectral radius* $\lambda_{max}(G)$ *is* < 1*, then convergence is finite and the number of steps is polynomial.*

4.1 Optimization with Dynamical Systems constraints

Here's a broader picture of what you want to study. Usually optimization is given in a functional form. If you want to optimize over the set of points that never leave some Ω is not a closed form! But if we want to optimize over this, then we have a different class of problems to solve. These problems have not been systematically studied algorithmically, and that is what I am currently doing.