First-order methods for Sum of Squares Optimization

Presented by Chenyang Yuan
Joint work with Benoît Legat and Pablo Parrilo

Introduction

Semidefinite programming (SDP) is a very powerful and expressive convex optimization method

Positive semidefinite variable $X \succcurlyeq 0$ + linear constraints

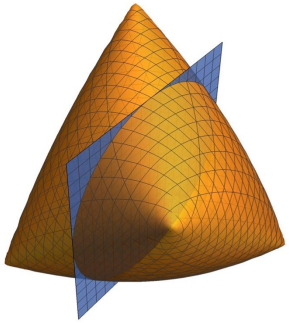
Applications: Optimal control, convex relaxations of combinatorial optimization, rank minimization and nuclear norm, ...

Typically solved with expensive interior point methods ($n \sim 10^3$)

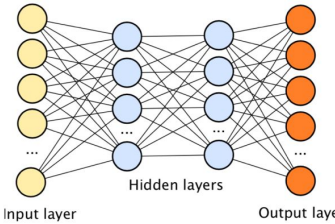
Deep Learning uses gradient-based solvers on large scale problems

Very successful on various classification and inference tasks

Solved with highly parallelized first-order methods ($n > 10^8$)



mosek



PyTorch

TensorFlow

Can we apply these techniques to solving SDPs?

Introduction

Burer-Monteiro methods factor PSD constraint $X = UU^T$, then perform local optimization on resulting non-convex unconstrained problem

$$\langle A_i, X \rangle = b_i \quad \forall i$$
$$X \succeq 0$$

Feasible

$\longleftrightarrow$

$$\min_U \sum_i (\langle A_i, UU^T \rangle - b_i)^2$$

Optimum = 0

May get stuck in local optimum (explicit counterexamples where second-order critical point ≠ global minimum)

When is non-convexity benign?

Related work

For general SDP feasibility with m linear constraints, with the factorization $X = UU^T$, where U is a nxr matrix.

Second-order critical point ⇒ Global minimum (non-convexity benign) when:

- $r > n$ [Burer and Monteiro]
- $r = \Omega(\sqrt{m})$, but with smoothed analysis [Cifuentes and Moitra], generic constraints [Bhojanapalli, Boumal, Jain, Netrapalli], or determinant regularization [Burer and Monteiro], (necessary because of counterexamples)

Can we do better if the SDP has special structure?

Samuel Burer and Renato Monteiro. (2005) Local Minima and Convergence in Low-Rank Semidefinite Programming, Mathematical Programming.
Diego Cifuentes and Ankur Moitra. (2019) Polynomial time guarantees for the Burer-Monteiro method, arXiv:1912.01745.
Srinadh Bhojanapalli, Nicolas Boumal, Prateek Jain, Praneeth Netrapalli (2018) Smoothed analysis for low-rank solutions to semidefinite programs in quadratic penalty form, Conference on Learning Theory.

Sum of Squares Optimization

Given p(x), can we write it as a sum of squares?

$$p(x) = \sum_{i=1}^{r} u_i(x)^2$$

Certifies that p(x) ≥ 0, and can be formulated as SDP

Focus on univariate trigonometric polynomials in this talk (methods can be generalized to multivariate case)

$$p(x) = a_0 + \sum_{k=1}^{2n} a_k \cos(kx) \quad x \in [0, \pi]$$

Applications in signal processing, filter design and control

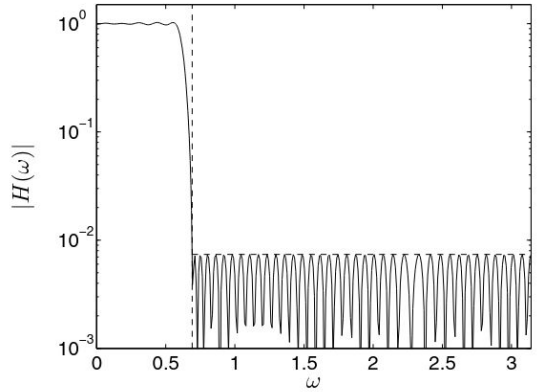


Image credit: Tae Roh and Lieven Vandenberghe. (2006) Discrete transforms, semidefinite programming and sum-of-squares representations of nonnegative polynomials. SIAM J. on Optimization.

Contributions

Find sum of squares decomposition of p(x) by solving

$$\min_u f(u) = \left\| \sum_{j=1}^{r} u_j(x_i)^2 - p(x_i) \right\|^2$$

For any norm on polynomials, if f(u) = 0, sum of squares decomposition agrees with p(x)

Theorem: when **r ≥ 2** (vs r = Ω(√m)) first-order methods find sum of squares decomposition for univariate polynomials (non-convexity benign)

If we choose right norm, ∇f(u) can be computed in O(n log n) time using fast fourier transforms (FFTs)

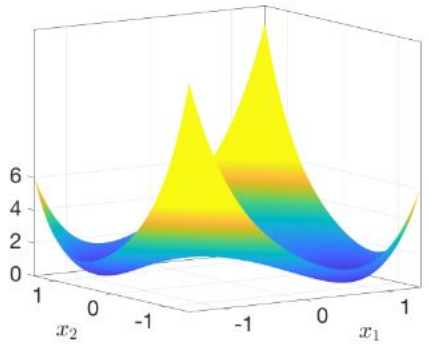


Image credit: Chi Yuejie, Lu Yue, Chen Yuxin. (2019) Nonconvex Optimization Meets Low-Rank Matrix Factorization: An Overview. IEEE Transactions on Signal Processing.

Proof Sketch

Assume that p(x) is a univariate polynomial and r = 2

$$f(u) = \left\| u_1(x)^2 + u_2(x)^2 - p(x) \right\|^2 = \|s(x) - p(x)\|^2$$

Given u such that $\nabla f(u)(v) = 0$ and $\nabla^2 f(u)(v,v) \geq 0$ for all v, show that f(u) = 0

We have inner product <p(x),q(x)> on polynomials with associated norm ||.||:

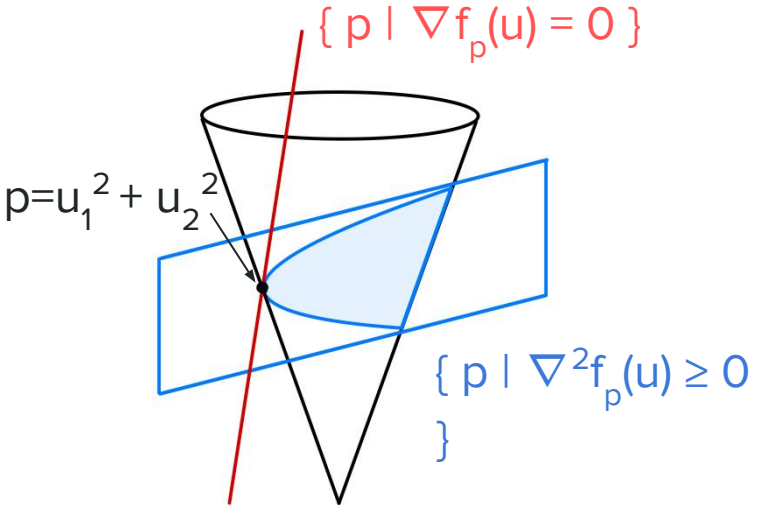
$$\nabla f(u)(v) \sim \left\langle \sum_{j=1}^{r} u_j(x) v_j(x), s(x) - p(x) \right\rangle = 0$$

$$\nabla^2 f(u)(v,v) \sim \langle \sum_{j=1}^{r} v_j(x)^2, s(x) - p(x) \rangle + 2 \left\| \sum_{j=1}^{r} u_j(x) v_j(x) \right\|^2 \geq 0$$

Proof Sketch

Geometrically, we want to show that the only intersection between set with zero gradient and PSD Hessian is when f(u) = 0.

For fixed u, these sets are convex!



Our proof can be interpreted as finding a certificate of this condition for every u and p.

Proof Sketch

$$\nabla f(u)(v) \sim \langle u_1(x)v_1(x) + u_2(x)v_2(x), s(x) - p(x)\rangle = 0$$
$$\nabla^2 f(u)(v,v) \sim \langle v_1(x)^2 + v_2(x)^2, s(x) - p(x)\rangle + 2\left\|u_1(x)v_1(x) + u_2(x)v_2(x)\right\|^2 \geq 0$$

Suppose u_1, u_2 coprime (true generically)

Bézout's lemma + gradient condition ⇒ exist v_1, v_2 s.t.

$$u_1(x)v_1(x)+u_2(x)v_2(x) = s(x)-p(x) \implies \|s(x)-p(x)\|^2 = 0$$

Suppose $u_1 = u_2$, choose $v_1 = v$ and $v_2 = -v$ in Hessian condition so for all v,

$$\langle v(x)^2, s(x) - p(x)\rangle \geq 0 \implies \langle p(x), s(x) - p(x)\rangle \geq 0$$

However, $\langle s(x), s(x) - p(x)\rangle = 0$ (gradient condition), so $\|s(x) - p(x)\|^2 = 0$

Interpolate between these two cases with the Positivstellensatz

Sampled basis

Which inner product <p(x),q(x)> on polynomials to choose?

Given p(x), q(x) degree d, choose d+1 points x_k

$$\langle p(x), q(x) \rangle = \sum_{k=1}^{d+1} p(x_k) q(x_k), \quad \|p(x)\|^2 = \sum_{k=1}^{d+1} p(x_k)^2$$

Valid inner product: when x_k are distinct points, if $\|p(x)\|^2 = 0$ then $p(x) = 0$.

Sum of squares using a sampled/interpolation basis studied by [Löfberg and Parrilo] and [Cifuentes and Parrilo]

How should we choose x_k?

Diego Cifuentes and Pablo Parrilo. (2017) Sampling algebraic varieties for sum of squares programs, SIAM J. Optimization.
Johan Lofberg and Pablo Parrilo. (2004) From coefficients to samples: a new approach to SOS optimization, 43rd IEEE Conference on Decision and Control.

Numerical Implementation

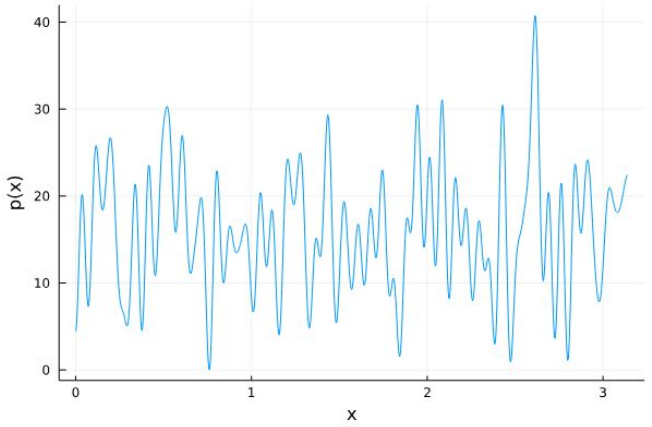
Compute sum of squares decomposition of degree 4n trigonometric polynomial

$$p(x) = a_0 + \sum_{k=1}^{2n} a_k \cos(kx) \quad x \in [0, \pi]$$

Using basis vectors evaluated at 4n + 1 points

$$B_k = \left[1, \cos(x_k), \ldots, \cos(nx_k)\right]$$

$$x_k = \frac{k\pi}{n}, \quad k = 1, \ldots, 4n+1$$



Matrix-vector products in ∇f(U) can be computed by FFT

$$\nabla f(U) = U^T B \,\mathrm{diag}(\|U^T B_k\|^2 - p(x_k)) B^T$$

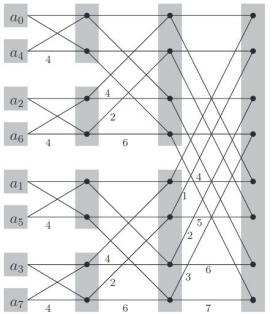


Image credit: Christos Papadimitriou, Sanjoy Dasgupta, and Umesh Vazirani. (2006) Algorithms

Numerical Implementation

TrigPolys.jl: a new package for fast manipulation of trigonometric polynomials

```
function Base.:*(p1::TrigPoly, p2::TrigPoly)
    n = p1.n + p2.n
    interpolate(evaluate(pad_to(p1, n)) .* evaluate(pad_to(p2, n)))
end
```

```
p1 = random_trig_poly(10^6)
p2 = random_trig_poly(10^6)
@btime p1 * p2;
```

```
1.737 s (160 allocations: 778.21 MiB)
```

`evaluate, evaluateT` and `interpolate` uses FFTW.jl, enables fast computation of f(U):

```
f(u) = sum((evaluate(pad_to(u, p.n)).^2 - evaluate(p)).^2)
```

AutoGrad.jl enables automatic computation of $\nabla f(U)$

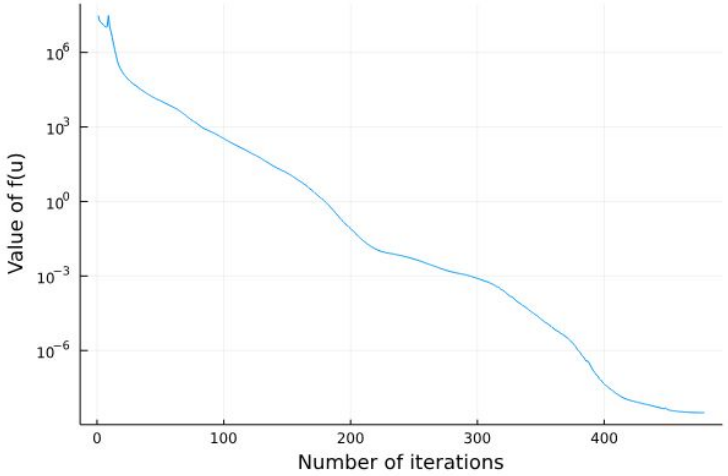
```
AutoGrad.@primitive evaluate(u::AbstractArray),dy,y evaluateT(dy)
fgrad = AutoGrad.grad(f)
```

Pass f(U), $\nabla f(U)$ to NLopt.jl to minimize f(U) with first-order optimization algorithms

Results

Sum of squares decomposition for random trigonometric polynomial

Convergence rate for LBFGS with random initialization:



Running times (stop at 10^{-7} relative error in U):

Degree of p(x)	1000	5000	10,000	50,000	100,000
Time (s)	1	5	11	45	112

Conclusion

When does it make sense to solve non-convex formulations of convex problems?

In our setting we can prove non-convexity does not hurt us

Also enables fast implementation in Julia