Counterexamples to the Low-Degree Conjecture

Alex Wein
Courant Institute, New York University

Joint work with:

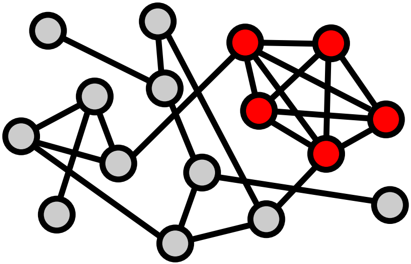
Justin Holmgren
NTT Research

High-Dimensional Testing Problems

Example: planted clique

High-Dimensional Testing Problems

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$



High-Dimensional Testing Problems

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

Other examples:

- Sparse PCA
- Planted CSPs
- Community detection in graphs
- Tensor PCA
- Spiked matrix models
- ...

Statistical-to-Computational Gaps

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

Statistical-to-Computational Gaps

Example: planted clique

Distinguish between

$\mathbb{Q}$: random graph $G(n, 1/2)$

$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique

with error probability $o(1)$

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

We don't know how to prove the "hard" regime is hard

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

We don't know how to prove the "hard" regime is hard

What kind of formal evidence can we give for hardness?

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

We don't know how to prove the "hard" regime is hard

What kind of formal evidence can we give for hardness?

- Average-case reductions

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

We don't know how to prove the "hard" regime is hard

What kind of formal evidence can we give for hardness?

- Average-case reductions
- Sum-of-squares lower bounds

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

We don't know how to prove the "hard" regime is hard

What kind of formal evidence can we give for hardness?

- Average-case reductions
- Sum-of-squares lower bounds
- ...

Statistical-to-Computational Gaps

Example: planted clique
Distinguish between
$\mathbb{Q}$: random graph $G(n, 1/2)$
$\mathbb{P}$: random graph $G(n, 1/2)$ with planted k-clique
with error probability $o(1)$

We don't know how to prove the "hard" regime is hard

What kind of formal evidence can we give for hardness?

- Average-case reductions
- Sum-of-squares lower bounds
- ...
- Low-degree polynomials

Low-Degree Algorithms

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares:

[Barak, Hopkins, Kelner, Kothari, Moitra, Potechin '16]

[Hopkins, Steurer '17]

[Hopkins, Kothari, Potechin, Raghavendra, Schramm, Steurer '17]

[Hopkins '18] (PhD thesis)

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares

Low-degree algorithm: multivariate polynomial
$f : \mathbb{R}^N \to \mathbb{R}$ e.g. $N = \binom{n}{2}$
"Low-degree": degree $O(\log n)$

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares

Low-degree algorithm: multivariate polynomial
$f : \mathbb{R}^N \to \mathbb{R}$ e.g. $N = \binom{n}{2}$
"Low-degree": degree $O(\log n)$

What does it mean for a polynomial to succeed at testing?

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares

Low-degree algorithm: multivariate polynomial
$f : \mathbb{R}^N \to \mathbb{R}$ e.g. $N = \binom{n}{2}$
"Low-degree": degree $O(\log n)$

What does it mean for a polynomial to succeed at testing?

$$\mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] = 0 \quad \mathop{\mathbb{E}}_{Y \sim \mathbb{P}}[f(Y)] = 1$$

$$\mathop{\mathrm{Var}}_{Y \sim \mathbb{Q}}[f(Y)] = o(1) \quad \mathop{\mathrm{Var}}_{Y \sim \mathbb{P}}[f(Y)] = o(1)$$

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares

Low-degree algorithm: multivariate polynomial
$f : \mathbb{R}^N \to \mathbb{R}$ e.g. $N = \binom{n}{2}$
"Low-degree": degree $O(\log n)$

What does it mean for a polynomial to succeed at testing?

$$\mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] = 0 \qquad \mathop{\mathbb{E}}_{Y \sim \mathbb{P}}[f(Y)] = 1$$

$$\mathop{\mathrm{Var}}_{Y \sim \mathbb{Q}}[f(Y)] = o(1) \qquad \mathop{\mathrm{Var}}_{Y \sim \mathbb{P}}[f(Y)] = o(1)$$

Tractable to analyze! Both upper and lower bounds

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares

Low-degree algorithm: multivariate polynomial
$f : \mathbb{R}^N \to \mathbb{R}$ e.g. $N = \binom{n}{2}$
"Low-degree": degree $O(\log n)$

What does it mean for a polynomial to succeed at testing?

$$\mathop{\mathbb{E}}_{Y \sim \mathbb{Q}}[f(Y)] = 0 \qquad \mathop{\mathbb{E}}_{Y \sim \mathbb{P}}[f(Y)] = 1$$

$$\mathop{\mathrm{Var}}_{Y \sim \mathbb{Q}}[f(Y)] = o(1) \qquad \mathop{\mathrm{Var}}_{Y \sim \mathbb{P}}[f(Y)] = o(1)$$

Tractable to analyze! Both upper and lower bounds

Low-Degree Algorithms

Inspired by a line of work on sum-of-squares

Low-degree algorithm: multivariate polynomial
$f : \mathbb{R}^N \to \mathbb{R}$ e.g. $N = \binom{n}{2}$
"Low-degree": degree $O(\log n)$

What does it mean for a polynomial to succeed at testing?

$$\mathop{\mathbb{E}}_{Y\sim\mathbb{Q}}[f(Y)] = 0 \qquad \mathop{\mathbb{E}}_{Y\sim\mathbb{P}}[f(Y)] = 1$$

$$\mathop{\mathrm{Var}}_{Y\sim\mathbb{Q}}[f(Y)] = o(1) \qquad \mathop{\mathrm{Var}}_{Y\sim\mathbb{P}}[f(Y)] = o(1)$$

Tractable to analyze! Both upper and lower bounds

- E.g. planted clique: low-degree algorithms succeed when $k \gg \sqrt{n}$ and fail when $k \ll \sqrt{n}$

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

Why: spectral methods (power iteration)

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

Why: spectral methods (power iteration)

Lets us predict when many problems are easy or hard

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

Why: spectral methods (power iteration)

Lets us predict when many problems are easy or hard

Counterexample: planted 3-XOR

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

Why: spectral methods (power iteration)

Lets us predict when many problems are easy or hard

Counterexample: planted 3-XOR

- Low-degree polynomials cannot implement Gaussian elimination

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

Why: spectral methods (power iteration)

Lets us predict when many problems are easy or hard

Counterexample: planted 3-XOR

- Low-degree polynomials cannot implement Gaussian elimination
- But low-degree prediction seems correct for noisy XOR

The Low-Degree Conjecture (Informally)

Conjecture (informal): for "natural" high-dimensional testing problems, degree-$O(\log n)$ polynomials are as powerful as all polynomial-time algorithms

Verified for long list of problems: planted clique, sparse PCA, ...

Why: spectral methods (power iteration)

Lets us predict when many problems are easy or hard

Counterexample: planted 3-XOR

- Low-degree polynomials cannot implement Gaussian elimination
- But low-degree prediction seems correct for noisy XOR

How to formalize "natural"?

The Low-Degree Conjecture (Formally)

Conjecture (Hopkins '18): Suppose that

- $\mathbb{Q}$ has i.i.d. entries,
- $\mathbb{P}$ is "symmetric",
- $\log^{1+\epsilon}(n)$-degree polynomials fail to distinguish $\mathbb{Q}, \mathbb{P}$.

Then for any fixed $\delta > 0$, no poly-time algorithm can distinguish $\mathbb{Q}$ from $T_\delta \mathbb{P}$.

The Low-Degree Conjecture (Formally)

Conjecture (Hopkins '18): Suppose that

- $\mathbb{Q}$ has i.i.d. entries,
- $\mathbb{P}$ is "symmetric",
- $\log^{1+\epsilon}(n)$-degree polynomials fail to distinguish $\mathbb{Q}, \mathbb{P}$.

Then for any fixed $\delta > 0$, no poly-time algorithm can distinguish $\mathbb{Q}$ from $T_\delta \mathbb{P}$.

Noise operator T_δ: for each entry, resample from $\mathbb{Q}$ with probability δ

- Rules out XOR counterexample

The Low-Degree Conjecture (Formally)

Conjecture (Hopkins '18): Suppose that

- $\mathbb{Q}$ has i.i.d. entries,
- $\mathbb{P}$ is "symmetric",
- $\log^{1+\epsilon}(n)$-degree polynomials fail to distinguish $\mathbb{Q}, \mathbb{P}$.

Then for any fixed $\delta > 0$, no poly-time algorithm can distinguish $\mathbb{Q}$ from $T_\delta \mathbb{P}$.

Noise operator T_δ: for each entry, resample from $\mathbb{Q}$ with probability δ

- Rules out XOR counterexample

Symmetry: typical for high-dimensional problems, but is this assumption needed?

Our Contributions

- Refute Hopkins '18 conjecture $\mathbb{Q} = \mathcal{N}(0, I_n)$

Our Contributions

- Refute Hopkins '18 conjecture $\mathbb{Q} = \mathcal{N}(0, I_n)$
  - Each entry of $\mathbb{P}$ encodes $\approx \log n$ bits of information in binary expansion

Our Contributions

- Refute Hopkins '18 conjecture $\mathbb{Q} = \mathcal{N}(0, I_n)$
  - Each entry of $\mathbb{P}$ encodes $\approx \log n$ bits of information in binary expansion
  - Fix: noise operator should slightly corrupt every entry

Our Contributions

- Refute Hopkins '18 conjecture $\mathbb{Q} = \mathcal{N}(0, I_n)$
  - Each entry of $\mathbb{P}$ encodes $\approx \log n$ bits of information in binary expansion
  - Fix: noise operator should slightly corrupt every entry
- Symmetry is necessary $\mathbb{Q} = \mathrm{Unif}(\{0,1\}^n)$

Our Contributions

- Refute Hopkins '18 conjecture $\mathbb{Q} = \mathcal{N}(0, I_n)$
  - Each entry of $\mathbb{P}$ encodes $\approx \log n$ bits of information in binary expansion
  - Fix: noise operator should slightly corrupt every entry
- Symmetry is necessary $\mathbb{Q} = \mathrm{Unif}(\{0,1\}^n)$
  - Cannot drop symmetry assumption

Our Contributions

- Refute Hopkins '18 conjecture $\mathbb{Q} = \mathcal{N}(0, I_n)$
  - Each entry of $\mathbb{P}$ encodes $\approx \log n$ bits of information in binary expansion
  - Fix: noise operator should slightly corrupt every entry
- Symmetry is necessary $\mathbb{Q} = \mathrm{Unif}(\{0,1\}^n)$
  - Cannot drop symmetry assumption

This refines our understanding of what types of problems we expect low-degree algorithms to be optimal for

Proof Ideas

First prove second claim: symmetry assumption is necessary

Proof Ideas

First prove second claim: symmetry assumption is necessary

- Goal: construct $\mathbb{Q}, \mathbb{P}$ over $\{0,1\}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Proof Ideas

First prove second claim: symmetry assumption is necessary

- Goal: construct $\mathbb{Q}, \mathbb{P}$ over $\{0,1\}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Construction:

- $\mathbb{Q}$: random bit string $\{0,1\}^n$
- $\mathbb{P}$: random codewode from an error-correcting code $\mathcal{C} \subseteq \{0,1\}^n$

Proof Ideas

First prove second claim: symmetry assumption is necessary

- Goal: construct $\mathbb{Q}, \mathbb{P}$ over $\{0,1\}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Construction:

- $\mathbb{Q}$: random bit string $\{0,1\}^n$
- $\mathbb{P}$: random codewode from an error-correcting code $\mathcal{C} \subseteq \{0,1\}^n$

Key properties:

Proof Ideas

First prove second claim: symmetry assumption is necessary

- Goal: construct $\mathbb{Q}, \mathbb{P}$ over $\{0,1\}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Construction:

- $\mathbb{Q}$: random bit string $\{0,1\}^n$
- $\mathbb{P}$: random codewode from an error-correcting code $\mathcal{C} \subseteq \{0,1\}^n$

Key properties:

- Codewords are D-wise independent (for $D = \Theta(n)$)
  - Degree-D polynomials cannot distinguish $\mathbb{Q}$ from $\mathbb{P}$

Proof Ideas

First prove second claim: symmetry assumption is necessary

- Goal: construct $\mathbb{Q}, \mathbb{P}$ over $\{0,1\}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Construction:

- $\mathbb{Q}$: random bit string $\{0,1\}^n$
- $\mathbb{P}$: random codewode from an error-correcting code $\mathcal{C} \subseteq \{0,1\}^n$

Key properties:

- Codewords are D-wise independent (for $D = \Theta(n)$)
  - Degree-D polynomials cannot distinguish $\mathbb{Q}$ from $\mathbb{P}$
- Poly-time decoding algorithm can recover $c \in \mathcal{C}$ after a random δ fraction of bits are flipped
  - Gives a poly-time algorithm to distinguish $\mathbb{Q}$ from $T_\delta \mathbb{P}$

Proof Ideas (Continued)

Now prove first claim: refute conjecture

Proof Ideas (Continued)

Now prove first claim: refute conjecture

- Goal: construct **permutation-symmetric** $\mathbb{Q}, \mathbb{P}$ over $\mathbb{R}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Proof Ideas (Continued)

Now prove first claim: refute conjecture

- Goal: construct **permutation-symmetric** $\mathbb{Q}, \mathbb{P}$ over $\mathbb{R}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Similar construction as before, but need to make distributions symmetric

Proof Ideas (Continued)

Now prove first claim: refute conjecture

- Goal: construct **permutation-symmetric** $\mathbb{Q}, \mathbb{P}$ over $\mathbb{R}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Similar construction as before, but need to make distributions symmetric

- Instead of observing $x = (x_1, \ldots, x_n)$, observe a list of pairs (i, x_i) with each i chosen randomly

Proof Ideas (Continued)

Now prove first claim: refute conjecture

- Goal: construct **permutation-symmetric** $\mathbb{Q}, \mathbb{P}$ over $\mathbb{R}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Similar construction as before, but need to make distributions symmetric

- Instead of observing $x = (x_1, \ldots, x_n)$, observe a list of pairs (i, x_i) with each i chosen randomly
- Encode each (i, x_i) by a real number

Proof Ideas (Continued)

Now prove first claim: refute conjecture

- Goal: construct **permutation-symmetric** $\mathbb{Q}, \mathbb{P}$ over $\mathbb{R}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Similar construction as before, but need to make distributions symmetric

- Instead of observing $x = (x_1, \ldots, x_n)$, observe a list of pairs (i, x_i) with each i chosen randomly
- Encode each (i, x_i) by a real number
- Throw out conflicting information: (i, a) and (i, b)

Proof Ideas (Continued)

Now prove first claim: refute conjecture

- Goal: construct **permutation-symmetric** $\mathbb{Q}, \mathbb{P}$ over $\mathbb{R}^n$ such that
  - $\mathbb{Q}$ is i.i.d.
  - no low-degree algorithm can distinguish $\mathbb{Q}, \mathbb{P}$
  - some poly-time algorithm can distinguish $\mathbb{Q}$, $T_\delta \mathbb{P}$

Similar construction as before, but need to make distributions symmetric

- Instead of observing $x = (x_1, \ldots, x_n)$, observe a list of pairs (i, x_i) with each i chosen randomly
- Encode each (i, x_i) by a real number
- Throw out conflicting information: (i, a) and (i, b)
- Error-correcting code now needs to handle erasures

Summary

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

- Refuted conjecture of Hopkins '18: noise operator

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

- Refuted conjecture of Hopkins '18: noise operator
- Symmetry is necessary

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

- Refuted conjecture of Hopkins '18: noise operator
- Symmetry is necessary

Future directions?

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

- Refuted conjecture of Hopkins '18: noise operator
- Symmetry is necessary

Future directions?

- How much symmetry is necessary?

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

- Refuted conjecture of Hopkins '18: noise operator
- Symmetry is necessary

Future directions?

- How much symmetry is necessary?
- Can the low-degree conjecture be used in e.g. cryptography?

Summary

Goal: formalize the class of problems for which we believe low-degree polynomials are optimal

Our contributions:

- Refuted conjecture of Hopkins '18: noise operator
- Symmetry is necessary

Future directions?

- How much symmetry is necessary?
- Can the low-degree conjecture be used in e.g. cryptography?

For more on the low-degree method (survey):

- **Notes on Computational Hardness of Hypothesis Testing: Predictions using the Low-Degree Likelihood Ratio**
  Kunisky, W., Bandeira
  arXiv:1907.11636