Understanding Statistical-vs-Computational Tradeoffs via the Low-Degree Likelihood Ratio

Alex Wein
Courant Institute, NYU

Joint work with:

Afonso Bandeira (ETH Zurich)  
Yunzi Ding (NYU)  
Tim Kunisky (NYU)
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden "signal," e.g.,
- determine which combination of genes cause a certain disease
- find "communities" in a social network
- predict which users will click on which ads
- etc.

There are many potential solutions

The naïve algorithm would check all possibilities, too slow!
- "curse of dimensionality"

Is there a "smarter" algorithm that can find the solution efficiently?

Goal: develop a theory to understand which statistical tasks can be solved efficiently (and which ones cannot)
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

- determine which combination of genes cause a certain disease
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

- determine which combination of genes cause a certain disease
- find “communities” in a social network
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

- determine which combination of genes cause a certain disease
- find “communities” in a social network
- predict which users will click on which ads
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

- determine which combination of genes cause a certain disease
- find “communities” in a social network
- predict which users will click on which ads
- etc.
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

- determine which combination of genes cause a certain disease
- find “communities” in a social network
- predict which users will click on which ads
- etc.

There are many potential solutions
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

> determine which combination of genes cause a certain disease
> find “communities” in a social network
> predict which users will click on which ads
> etc.

There are many potential solutions

The naïve algorithm would check all possibilities, too slow!

> “curse of dimensionality”
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,

- determine which combination of genes cause a certain disease
- find “communities” in a social network
- predict which users will click on which ads
- etc.

There are many potential solutions

The naïve algorithm would check all possibilities, too slow!

- “curse of dimensionality”

Is there a “smarter” algorithm that can find the solution efficiently?
Motivation

Imagine we have a large noisy dataset and want to extract some kind of hidden “signal”, e.g.,
  - determine which combination of genes cause a certain disease
  - find “communities” in a social network
  - predict which users will click on which ads
  - etc.

There are many potential solutions

The naïve algorithm would check all possibilities, too slow!
  - “curse of dimensionality”

Is there a “smarter” algorithm that can find the solution efficiently?

Goal: develop a theory to understand which statistical tasks can be solved efficiently (and which ones cannot)
Part I: Statistical-to-Computational Gaps and the “Low-Degree Method”
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$

- $n$ vertices
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$

- $n$ vertices
- Each of the $\binom{n}{2}$ edges occurs with probability $1/2$
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$

$n$ vertices

- Each of the $\binom{n}{2}$ edges occurs with probability $1/2$

- Planted clique on $k$ vertices
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$

- $n$ vertices
- Each of the $\binom{n}{2}$ edges occurs with probability $1/2$
- Planted clique on $k$ vertices
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$

- $n$ vertices
- Each of the $\binom{n}{2}$ edges occurs with probability $1/2$
- Planted clique on $k$ vertices
- Goal: find the clique
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$

- Other examples of stat-comp gaps
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
  - Random constraint satisfaction problems (e.g. 3-SAT)
  - Tensor PCA
  - Tensor decomposition

Different from theory of NP-hardness: average-case
Statistical-to-Computational Gaps

- **Planted clique:** $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

Other examples of stat-comp gaps:

- Sparse PCA
- Community detection in graphs (stochastic block model)
- Random constraint satisfaction problems (e.g. 3-SAT)
- Tensor PCA
- Tensor decomposition

Different from theory of NP-hardness: average-case

Q: What fundamentally makes a problem easy or hard?
Statistical-to-Computational Gaps

- Planted clique: \( G(n, 1/2) \cup \{k\text{-clique}\} \)
  - Statistically, can find planted clique of size \((2 + \varepsilon) \log_2 n\)
  - In polynomial time, we only know how to find clique of size \(\Omega(\sqrt{n})\) [Alon, Krivelevich, Sudakov '98]

---

Different from theory of NP-hardness: average-case Q: What fundamentally makes a problem easy or hard?
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- Other examples of stat-comp gaps

![Diagram showing the difficulty scale for finding cliques of different sizes](diagram.png)
Statistical-to-Computational Gaps

- **Planted clique:** $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- **Other examples of stat-comp gaps**
  - Sparse PCA
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- Other examples of stat-comp gaps
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
Statistical-to-Computational Gaps

- **Planted clique**: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- **Other examples of stat-comp gaps**
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
  - Random constraint satisfaction problems (e.g. 3-SAT)
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- Other examples of stat-comp gaps
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
  - Random constraint satisfaction problems (e.g. 3-SAT)
  - Tensor PCA
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- Other examples of stat-comp gaps
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
  - Random constraint satisfaction problems (e.g. 3-SAT)
  - Tensor PCA
  - Tensor decomposition
Statistical-to-Computational Gaps

- **Planted clique**: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov ’98]

- **Other examples of stat-comp gaps**
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
  - Random constraint satisfaction problems (e.g. 3-SAT)
  - Tensor PCA
  - Tensor decomposition

Different from theory of NP-hardness: average-case
Statistical-to-Computational Gaps

- Planted clique: $G(n, 1/2) \cup \{k\text{-clique}\}$
  - Statistically, can find planted clique of size $(2 + \varepsilon) \log_2 n$
  - In polynomial time, we only know how to find clique of size $\Omega(\sqrt{n})$ [Alon, Krivelevich, Sudakov '98]

- Other examples of stat-comp gaps
  - Sparse PCA
  - Community detection in graphs (stochastic block model)
  - Random constraint satisfaction problems (e.g. 3-SAT)
  - Tensor PCA
  - Tensor decomposition

Different from theory of NP-hardness: average-case

Q: What fundamentally makes a problem easy or hard?
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet '13; Brennan, Bresler,...]
- Failure of MCMC [Jerrum '92]
- Shattering of solution space [Achlioptas, Coja-Oghlan '08]
- Failure of local algorithms [Gamarnik, Sudan '13]
- Statistical physics, belief propagation [Decelle, Krzakala, Moore, Zdeborová '11]
- Optimization landscape, Kac-Rice formula [Auffinger, Ben Arous, Černý '10]
- Statistical query lower bounds [Feldman, Grigorescu, Reyzin, Vempala, Xiao '12]
- Sum-of-squares lower bounds [Barak, Hopkins, Kelner, Kothari, Moitra, Potechin '16]
- This talk: “low-degree method” [Barak, Hopkins, Kelner, Kothari, Moitra, Potechin '16; Hopkins, Steurer '17; Hopkins, Kothari, Potechin, Raghavendra, Schramm, Steurer '17; Hopkins '18 (PhD thesis)]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler,...]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

▶ Reductions (e.g. from planted clique) [Berthet, Rigollet '13; Brennan, Bresler,...]

▶ Failure of MCMC [Jerrum '92]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- **Reductions (e.g. from planted clique)** [Berthet, Rigollet ’13; Brennan, Bresler,…]
- **Failure of MCMC** [Jerrum ’92]
- **Shattering of solution space** [Achlioptas, Coja-Oghlan ’08]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler, ...]
- Failure of MCMC [Jerrum ’92]
- Shattering of solution space [Achlioptas, Coja-Oghlan ’08]
- Failure of local algorithms [Gamarnik, Sudan ’13]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler,...]
- Failure of MCMC [Jerrum ’92]
- Shattering of solution space [Achlioptas, Coja-Oghlan ’08]
- Failure of local algorithms [Gamarnik, Sudan ’13]
- Statistical physics, belief propagation [Decelle, Krzakala, Moore, Zdeborová ’11]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler,...]
- Failure of MCMC [Jerrum ’92]
- Shattering of solution space [Achlioptas, Coja-Oghlan ’08]
- Failure of local algorithms [Gamarnik, Sudan ’13]
- Statistical physics, belief propagation [Decelle, Krzakala, Moore, Zdeborová ’11]
- Optimization landscape, Kac-Rice formula [Auffinger, Ben Arous, Černý ’10]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler,...]
- Failure of MCMC [Jerrum ’92]
- Shattering of solution space [Achlioptas, Coja-Oghlan ’08]
- Failure of local algorithms [Gamarnik, Sudan ’13]
- Statistical physics, belief propagation [Decelle, Krzakala, Moore, Zdeborová ’11]
- Optimization landscape, Kac-Rice formula [Auffinger, Ben Arous, Černý ’10]
- Statistical query lower bounds [Feldman, Grigorescu, Reyzin, Vempala, Xiao ’12]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler,...]
- Failure of MCMC [Jerrum ’92]
- Shattering of solution space [Achlioptas, Coja-Oghlan ’08]
- Failure of local algorithms [Gamarnik, Sudan ’13]
- Statistical physics, belief propagation [Decelle, Krzakala, Moore, Zdeborová ’11]
- Optimization landscape, Kac-Rice formula [Auffinger, Ben Arous, Černý ’10]
- Statistical query lower bounds [Feldman, Grigorescu, Reyzin, Vempala, Xiao ’12]
- Sum-of-squares lower bounds [Barak, Hopkins, Kelner, Kothari, Moitra, Potechin ’16]
How to Show that a Problem is Hard?

We don’t know how to prove that average-case problems are hard, but various forms of “rigorous evidence”:

- Reductions (e.g. from planted clique) [Berthet, Rigollet ’13; Brennan, Bresler,...]
- Failure of MCMC [Jerrum ’92]
- Shattering of solution space [Achlioptas, Coja-Oghlan ’08]
- Failure of local algorithms [Gamarnik, Sudan ’13]
- Statistical physics, belief propagation [Decelle, Krzakala, Moore, Zdeborová ’11]
- Optimization landscape, Kac-Rice formula [Auffinger, Ben Arous, Černý ’10]
- Statistical query lower bounds [Feldman, Grigorescu, Reyzin, Vempala, Xiao ’12]
- Sum-of-squares lower bounds [Barak, Hopkins, Kelner, Kothari, Moitra, Potechin ’16]
- This talk: “low-degree method”
  [Barak, Hopkins, Kelner, Kothari, Moitra, Potechin ’16; Hopkins, Steurer ’17; Hopkins, Kothari, Potechin, Raghavendra, Schramm, Steurer ’17; Hopkins ’18 (PhD thesis)]
The Low-Degree Method (e.g. [Hopkins, Steurer '17])

Suppose we want to hypothesis test with error probability $o(1)$ between two distributions:

- **Null model** $Y \sim Q$, e.g. $G(n, 1/2)$
- **Planted model** $Y \sim P$, e.g. $G(n, 1/2) \cup \{\text{random } k\text{-clique}\}$

Look for a degree-$D$ (multivariate) polynomial $f : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ that distinguishes $P$ from $Q$:

Want $f(Y)$ to be big when $Y \sim P$ and small when $Y \sim Q$.

Compute $\max_{f \text{ deg } D} \mathbb{E}_{Y \sim P} [f(Y)] \sqrt{\mathbb{E}_{Y \sim Q} [f(Y)^2]}$ mean in $P$ fluctuations in $Q$. 

7 / 27
The Low-Degree Method (e.g. [Hopkins, Steurer '17])

Suppose we want to hypothesis test with error probability $o(1)$ between two distributions:

- Null model $Y \sim \mathcal{Q}_n$ e.g. $G(n, 1/2)$
Suppose we want to hypothesis test with error probability $o(1)$ between two distributions:

- Null model $Y \sim \mathcal{Q}_n$ e.g. $G(n, 1/2)$
- Planted model $Y \sim \mathcal{P}_n$ e.g. $G(n, 1/2) \cup \{\text{random } k\text{-clique}\}$
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

Suppose we want to hypothesis test with error probability $o(1)$ between two distributions:

- Null model $Y \sim \mathcal{Q}_n$ e.g. $G(n, 1/2)$
- Planted model $Y \sim \mathcal{P}_n$ e.g. $G(n, 1/2) \cup \{\text{random } k\text{-clique}\}$

Look for a degree-$D$ (multivariate) polynomial $f : \mathbb{R}^{n \times n} \to \mathbb{R}$ that distinguishes $\mathcal{P}$ from $\mathcal{Q}$:
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

Suppose we want to hypothesis test with error probability $o(1)$ between two distributions:

- Null model $Y \sim \mathcal{Q}_n$ e.g. $G(n, 1/2)$
- Planted model $Y \sim \mathcal{P}_n$ e.g. $G(n, 1/2) \cup \{\text{random } k\text{-clique}\}$

Look for a degree-$D$ (multivariate) polynomial $f : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$ that distinguishes $\mathcal{P}$ from $\mathcal{Q}$:

Want $f(Y)$ to be big when $Y \sim \mathcal{P}$ and small when $Y \sim \mathcal{Q}$
The Low-Degree Method (e.g. [Hopkins, Steurer '17])

Suppose we want to hypothesis test with error probability $o(1)$ between two distributions:

- Null model $Y \sim Q_n$ e.g. $G(n, 1/2)$
- Planted model $Y \sim P_n$ e.g. $G(n, 1/2) \cup \{\text{random } k\text{-clique}\}$

Look for a degree-$D$ (multivariate) polynomial $f : \mathbb{R}^{n \times n} \to \mathbb{R}$ that distinguishes $P$ from $Q$:

Want $f(Y)$ to be big when $Y \sim P$ and small when $Y \sim Q$

Compute $\max_{f \ \text{deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}}$ mean in $P$

fluctuations in $Q$
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

\[
\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}}
\]
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

\[
\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim p}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim q}[f(Y)^2]}}
\]

\[
\langle f, g \rangle = \mathbb{E}_{Y \sim q}[f(Y)g(Y)]
\]

\[
\|f\| = \sqrt{\langle f, f \rangle}
\]
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

\[
\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} = \max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim Q}[L(Y)f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}}
\]

\[
\langle f, g \rangle = \mathbb{E}_{Y \sim Q}[f(Y)g(Y)]
\]

\[
\|f\| = \sqrt{\langle f, f \rangle}
\]

Likelihood ratio:
\[L(Y) = \frac{dP}{dQ}(Y)\]
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

$$\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}}$$

$$= \max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim Q}[L(Y)f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}}$$

$$= \max_{f \text{ deg } D} \frac{\langle L, f \rangle}{\|f\|}$$

$$\langle f, g \rangle = \mathbb{E}_{Y \sim Q}[f(Y)g(Y)]$$

$$\|f\| = \sqrt{\langle f, f \rangle}$$

Likelihood ratio:
$$L(Y) = \frac{dP}{dQ}(Y)$$
The Low-Degree Method (e.g. [Hopkins, Steurer ’17])

\[
\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} \\
= \max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim Q}[L(Y)f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} \\
= \max_{f \text{ deg } D} \frac{\langle L, f \rangle}{\|f\|}
\]

\[
\langle f, g \rangle = \mathbb{E}_{Y \sim Q}[f(Y)g(Y)] \\
\|f\| = \sqrt{\langle f, f \rangle}
\]

Likelihood ratio:
\[
L(Y) = \frac{dP}{dQ}(Y)
\]

Maximizer: \( f = L^{\leq D} := \text{projection of } L \text{ onto degree-}D \text{ subspace} \)
The Low-Degree Method (e.g. [Hopkins, Steurer '17])

\[
\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} = \max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim Q}[L(Y)f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} = \max_{f \text{ deg } D} \langle L, f \rangle = \|L\| \leq D
\]

\[
\langle f, g \rangle = \mathbb{E}_{Y \sim Q}[f(Y)g(Y)]
\]

\[
\|f\| = \sqrt{\langle f, f \rangle}
\]

Likelihood ratio:

\[
L(Y) = \frac{dP}{dQ}(Y)
\]

Maximizer: \( f = L^{\leq D} := \text{projection of } L \text{ onto degree-} D \text{ subspace} \)

Norm of low-degree likelihood ratio
The Low-Degree Method

Conclusion: $\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} = \|L^{\leq D}\|_*$
The Low-Degree Method

Conclusion: \( \max_{f \text{ degree } D} \frac{\mathbb{E}_{Y \sim P}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim Q}[f(Y)^2]}} = \|L^{\leq D}\| \)

Heuristically,

\[ \|L^{\leq D}\| = \begin{cases} \omega(1) & \text{degree-}D \text{ polynomial can distinguish } Q, P \\ O(1) & \text{degree-}D \text{ polynomials fail} \end{cases} \]
The Low-Degree Method

Conclusion: \[
\max_{f \text{ deg } D} \frac{\mathbb{E}_{Y \sim \mathbb{P}}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim \mathbb{Q}}[f(Y)^2]}} = \|L^{\leq D}\|
\]

Heuristically,

\[
\|L^{\leq D}\| = \begin{cases} 
\omega(1) & \text{degree-} D \text{ polynomial can distinguish } \mathbb{Q}, \mathbb{P} \\
O(1) & \text{degree-} D \text{ polynomials fail}
\end{cases}
\]

Conjecture (informal variant of [Hopkins '18])

For “nice” \(\mathbb{Q}, \mathbb{P}\), if \(\|L^{\leq D}\| = O(1)\) for some \(D = \omega(\log n)\) then no polynomial-time algorithm can distinguish \(\mathbb{Q}, \mathbb{P}\) with success probability \(1 - o(1)\).
The Low-Degree Method

Conclusion: \[
\max_{f \text{ degree } D} \frac{\mathbb{E}_{Y \sim \mathbb{P}}[f(Y)]}{\sqrt{\mathbb{E}_{Y \sim \mathbb{Q}}[f(Y)^2]}} = \|L^{\leq D}\|
\]

Heuristically,

\[
\|L^{\leq D}\| = \begin{cases} \omega(1) & \text{degree-}D \text{ polynomial can distinguish } \mathbb{Q}, \mathbb{P} \\ O(1) & \text{degree-}D \text{ polynomials fail} \end{cases}
\]

Conjecture (informal variant of [Hopkins '18])

*For “nice” \( \mathbb{Q}, \mathbb{P} \), if \( \|L^{\leq D}\| = O(1) \) for some \( D = \omega(\log n) \) then no polynomial-time algorithm can distinguish \( \mathbb{Q}, \mathbb{P} \) with success probability \( 1 - o(1) \).*

Degree-\( O(\log n) \) polynomials \( \iff \) Polynomial-time algorithms
Formal Consequences of the Low-Degree Method

The case $D = \infty$: If $\|L\| = O(1)$ (as $n \to \infty$) then no test can distinguish $\mathbb{Q}$ from $\mathbb{P}$ (with success probability $1 - o(1)$)

- Classical second moment method
Formal Consequences of the Low-Degree Method

The case $D = \infty$: If $\|L\| = O(1)$ (as $n \to \infty$) then no test can distinguish $Q$ from $P$ (with success probability $1 - o(1)$)

- Classical second moment method

If $\|L_{\leq D}\| = O(1)$ for some $D = \omega(\log n)$ then no spectral method can distinguish $Q$ from $P$ (in a particular sense) [Kunisky, W, Bandeira '19]

- Spectral method: threshold top eigenvalue of poly-size matrix $M = M(Y)$ whose entries are $O(1)$-degree polynomials in $Y$
Formal Consequences of the Low-Degree Method

The case $D = \infty$: If $\|L\| = O(1)$ (as $n \to \infty$) then no test can distinguish $Q$ from $P$ (with success probability $1 - o(1)$)

- Classical second moment method

If $\|L^{\leq D}\| = O(1)$ for some $D = \omega(\log n)$ then no spectral method can distinguish $Q$ from $P$ (in a particular sense) [Kunisky, W, Bandeira '19]

- Spectral method: threshold top eigenvalue of poly-size matrix $M = M(Y)$ whose entries are $O(1)$-degree polynomials in $Y$

- Proof: consider polynomial $f(Y) = \text{Tr}(M^q)$ with $q = \Theta(\log n)$
Formal Consequences of the Low-Degree Method

The case $D = \infty$: If $\|L\| = O(1)$ (as $n \to \infty$) then no test can distinguish $Q$ from $P$ (with success probability $1 - o(1)$)

- Classical second moment method

If $\|L^{\leq D}\| = O(1)$ for some $D = \omega(\log n)$ then no spectral method can distinguish $Q$ from $P$ (in a particular sense) [Kunisky, W, Bandeira '19]

- Spectral method: threshold top eigenvalue of poly-size matrix $M = M(Y)$ whose entries are $O(1)$-degree polynomials in $Y$

- Proof: consider polynomial $f(Y) = \text{Tr}(M^q)$ with $q = \Theta(\log n)$

- Spectral methods are believed to be as powerful as sum-of-squares for average-case problems [HKPRSS '17]
Low-Degree Method: Recap

Given a hypothesis testing question $Q_n$ vs $P_n$
Low-Degree Method: Recap

Given a hypothesis testing question $Q_n$ vs $P_n$

Take $D \approx \log n$
Low-Degree Method: Recap

Given a hypothesis testing question $Q_n$ vs $P_n$

Take $D \approx \log n$

Compute/bound $\|L^{\leq D}\|$ in the limit $n \rightarrow \infty$
Low-Degree Method: Recap

Given a hypothesis testing question $Q_n$ vs $P_n$

Take $D \approx \log n$

Compute/bound $\|L^{\leq D}\|$ in the limit $n \to \infty$

- If $\|L^{\leq D}\| = \omega(1)$, suggests that the problem is poly-time solvable
Given a hypothesis testing question $Q_n$ vs $P_n$

Take $D \approx \log n$

Compute/bound $\|L^{\leq D}\|$ in the limit $n \to \infty$

- If $\|L^{\leq D}\| = \omega(1)$, suggests that the problem is poly-time solvable

- If $\|L^{\leq D}\| = O(1)$, suggests that the problem is NOT poly-time solvable (and gives rigorous evidence: spectral methods fail)
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
Advantages of the Low-Degree Method

➢ Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
➢ Predictions seem “correct”!
   ➢ Planted clique, sparse PCA, stochastic block model, ...

By varying degree $D$, can explore runtimes other than polynomial

Conjecture (Hopkins '18): degree-$D$ polynomials $\Leftrightarrow$ time-$n\tilde{\Theta}(D)$ algorithms

No ingenuity required

Interpretable
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
- Detection vs certification
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^\leq D\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
- Detection vs certification
- General: no assumptions on $\mathbb{Q}, \mathbb{P}$
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
- Detection vs certification
- General: no assumptions on $\mathbb{Q}, \mathbb{P}$
- Captures sharp thresholds [Hopkins, Steurer ’17]
Advantages of the Low-Degree Method

- Possible to calculate/bound $\| L^{\leq D} \|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
- Detection vs certification
- General: no assumptions on $\mathbb{Q}, \mathbb{P}$
- Captures sharp thresholds [Hopkins, Steurer ’17]
- By varying degree $D$, can explore runtimes other than polynomial
  - Conjecture (Hopkins ’18): degree-$D$ polynomials $\Leftrightarrow$ time-$n^{\tilde{O}(D)}$ algorithms
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
- Detection vs certification
- General: no assumptions on $Q$, $P$
- Captures sharp thresholds [Hopkins, Steurer ’17]
- By varying degree $D$, can explore runtimes other than polynomial
  - Conjecture (Hopkins ’18): degree-$D$ polynomials $\leftrightarrow$ time-$n^{\tilde{O}(D)}$ algorithms
- No ingenuity required
Advantages of the Low-Degree Method

- Possible to calculate/bound $\|L^{\leq D}\|$ for many problems
- Predictions seem “correct”!
  - Planted clique, sparse PCA, stochastic block model, ...
- (Relatively) simple
  - Much simpler than sum-of-squares lower bounds
- Detection vs certification
- General: no assumptions on $\mathbb{Q}, \mathbb{P}$
- Captures sharp thresholds [Hopkins, Steurer ’17]
- By varying degree $D$, can explore runtimes other than polynomial
  - Conjecture (Hopkins ’18): degree-$D$ polynomials $\iff$ time-$n^{\tilde{O}(D)}$ algorithms
- No ingenuity required
- Interpretable
How to Compute $\|L^{\leq D}\|$ 

Additive Gaussian noise: $\mathcal{P}: Y = X + Z$ vs $\mathcal{Q}: Y = Z$
where $X \sim \mathcal{P}$, any distribution over $\mathbb{R}^N$
and $Z$ is i.i.d. $\mathcal{N}(0, 1)$
How to Compute $\| L^{\leq D} \|

Additive Gaussian noise: $P: Y = X + Z$ vs $Q: Y = Z$
where $X \sim P$, any distribution over $\mathbb{R}^N$
and $Z$ is i.i.d. $\mathcal{N}(0, 1)$

$$L(Y) = \frac{dP}{dQ}(Y) = \frac{\mathbb{E}_X \exp(-\frac{1}{2}\| Y - X \|^2)}{\exp(-\frac{1}{2}\| Y \|^2)} = \mathbb{E}_X \exp(\langle Y, X \rangle - \frac{1}{2}\| X \|^2)$$
How to Compute $\| L^{\leq D} \|

Additive Gaussian noise: $\mathbb{P} : Y = X + Z$ vs $\mathbb{Q} : Y = Z$
where $X \sim \mathbb{P}$, any distribution over $\mathbb{R}^N$ and $Z$ is i.i.d. $\mathcal{N}(0, 1)$

$$L(Y) = \frac{d\mathbb{P}}{d\mathbb{Q}}(Y) = \frac{\mathbb{E}_X \exp\left(-\frac{1}{2} \| Y - X \|^2\right)}{\exp\left(-\frac{1}{2} \| Y \|^2\right)} = \mathbb{E}_X \exp\left(\langle Y, X \rangle - \frac{1}{2} \| X \|^2\right)$$

Expand $L = \sum_\alpha c_\alpha h_\alpha$ where $\{ h_\alpha \}$ are Hermite polynomials (orthonormal basis w.r.t. $\mathbb{Q}$)
How to Compute $\|L^{\leq D}\|$  

Additive Gaussian noise: $\mathbb{P}: Y = X + Z$ \text{ vs } $\mathbb{Q}: Y = Z$ 
where $X \sim \mathbb{P}$, any distribution over $\mathbb{R}^N$ 
and $Z$ is i.i.d. $\mathcal{N}(0, 1)$

$$L(Y) = \frac{d\mathbb{P}}{d\mathbb{Q}}(Y) = \frac{\mathbb{E}_X \exp\left(-\frac{1}{2}\|Y - X\|^2\right)}{\exp\left(-\frac{1}{2}\|Y\|^2\right)} = \mathbb{E}_X \exp(\langle Y, X \rangle - \frac{1}{2}\|X\|^2)$$

Expand $L = \sum_\alpha c_\alpha h_\alpha$ where $\{h_\alpha\}$ are Hermite polynomials
(orthonormal basis w.r.t. $\mathbb{Q}$)

$$\|L^{\leq D}\|^2 = \sum_{|\alpha| \leq D} c_\alpha^2 \text{ where } c_\alpha = \langle L, h_\alpha \rangle = \mathbb{E}_{Y \sim \mathbb{Q}}[L(Y)h_\alpha(Y)]$$

\[ \]
How to Compute $\|L^{\leq D}\|$ 

Additive Gaussian noise: $\mathbb{P} : Y = X + Z$ vs $\mathbb{Q} : Y = Z$
where $X \sim \mathbb{P}$, any distribution over $\mathbb{R}^N$
and $Z$ is i.i.d. $\mathcal{N}(0, 1)$

$$L(Y) = \frac{d\mathbb{P}}{d\mathbb{Q}}(Y) = \frac{\mathbb{E}_X \exp\left(-\frac{1}{2}\|Y - X\|^2\right)}{\exp\left(-\frac{1}{2}\|Y\|^2\right)} = \mathbb{E}_X \exp(\langle Y, X \rangle - \frac{1}{2}\|X\|^2)$$

Expand $L = \sum_\alpha c_\alpha h_\alpha$ where $\{h_\alpha\}$ are Hermite polynomials
(orthonormal basis w.r.t. $\mathbb{Q}$)

$$\|L^{\leq D}\|^2 = \sum_{|\alpha| \leq D} c_\alpha^2$$
where $c_\alpha = \langle L, h_\alpha \rangle = \mathbb{E}_{Y \sim \mathbb{Q}}[L(Y)h_\alpha(Y)]$

\ldots
How to Compute $\| L_{\leq D} \|

Additive Gaussian noise: $\mathbb{P}: Y = X + Z$ vs $\mathbb{Q}: Y = Z$
where $X \sim \mathbb{P}$, any distribution over $\mathbb{R}^N$
and $Z$ is i.i.d. $\mathcal{N}(0, 1)$

$L(Y) = \frac{d\mathbb{P}}{d\mathbb{Q}}(Y) = \frac{\mathbb{E}_X \exp(-\frac{1}{2} \| Y - X \|^2)}{\exp(-\frac{1}{2} \| Y \|^2)} = \mathbb{E}_X \exp(\langle Y, X \rangle - \frac{1}{2} \| X \|^2)$

Expand $L = \sum_{\alpha} c_\alpha h_\alpha$ where $\{ h_\alpha \}$ are Hermite polynomials
(orthonormal basis w.r.t. $\mathbb{Q}$)

$\| L_{\leq D} \|^2 = \sum_{|\alpha| \leq D} c_\alpha^2$ where $c_\alpha = \langle L, h_\alpha \rangle = \mathbb{E}_{Y \sim \mathbb{Q}}[L(Y)h_\alpha(Y)]$

\[
\frac{1}{d!} \mathbb{E}_{X, X'}[\langle X, X' \rangle^d]
\]

Result: $\| L_{\leq D} \|^2 = \sum_{d=0}^{D} \frac{1}{d!} \mathbb{E}_{X, X'}[\langle X, X' \rangle^d]$
For more on the low-degree method...

- Samuel B. Hopkins, PhD thesis ’18: “Statistical Inference and the Sum of Squares Method”
  - Connection to SoS

- Survey article: Kunisky, W, Bandeira, “Notes on Computational Hardness of Hypothesis Testing: Predictions using the Low-Degree Likelihood Ratio”, arxiv:1907.11636
Part II: Sparse PCA

Based on: Ding, Kunisky, W., Bandeira, “Subexponential-Time Algorithms for Sparse PCA”, arxiv:1907.11635
Spiked Wigner Model

Observe $n \times n$ matrix $Y = \lambda xx^T + W$

**Signal:** $x \in \mathbb{R}^n$, $\|x\| = 1$

**Noise:** $W \in \mathbb{R}^{n \times n}$ with entries $W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n)$ i.i.d.

$\lambda > 0$: signal-to-noise ratio
Spiked Wigner Model

Observe $n \times n$ matrix $Y = \lambda xx^T + W$

Signal: $x \in \mathbb{R}^n$, $\|x\| = 1$

Noise: $W \in \mathbb{R}^{n \times n}$ with entries $W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n)$ i.i.d.

$\lambda > 0$: signal-to-noise ratio

Goal: given $Y$, estimate the signal $x$
Spiked Wigner Model

Observe $n \times n$ matrix $Y = \lambda x x^T + W$

**Signal:** $x \in \mathbb{R}^n$, $\|x\| = 1$

**Noise:** $W \in \mathbb{R}^{n \times n}$ with entries $W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n)$ i.i.d.

$\lambda > 0$: signal-to-noise ratio

Goal: given $Y$, estimate the signal $x$

Or, even simpler: distinguish (w.h.p.) $Y$ from pure noise $W$
Spiked Wigner Model

Observe \( n \times n \) matrix \( Y = \lambda xx^T + W \)

- **Signal**: \( x \in \mathbb{R}^n, \|x\| = 1 \)
- **Noise**: \( W \in \mathbb{R}^{n \times n} \) with entries \( W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n) \) i.i.d.
- \( \lambda > 0 \): signal-to-noise ratio

**Goal**: given \( Y \), estimate the signal \( x \)

Or, even simpler: distinguish (w.h.p.) \( Y \) from pure noise \( W \)

**Structure**: suppose \( x \) is drawn from some prior, e.g.
Spiked Wigner Model

Observe $n \times n$ matrix $Y = \lambda xx^T + W$

**Signal:** $x \in \mathbb{R}^n$, $\|x\| = 1$

**Noise:** $W \in \mathbb{R}^{n \times n}$ with entries $W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n)$ i.i.d.

$\lambda > 0$: signal-to-noise ratio

Goal: given $Y$, estimate the signal $x$

Or, even simpler: distinguish (w.h.p.) $Y$ from pure noise $W$

Structure: suppose $x$ is drawn from some prior, e.g.

- spherical (uniform on unit sphere)
Spiked Wigner Model

Observe $n \times n$ matrix $Y = \lambda xx^T + W$

**Signal:** $x \in \mathbb{R}^n$, $\|x\| = 1$

**Noise:** $W \in \mathbb{R}^{n \times n}$ with entries $W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n)$ i.i.d.

$\lambda > 0$: signal-to-noise ratio

Goal: given $Y$, estimate the signal $x$

Or, even simpler: distinguish (w.h.p.) $Y$ from pure noise $W$

Structure: suppose $x$ is drawn from some prior, e.g.

- spherical (uniform on unit sphere)
- Rademacher (i.i.d. $\pm 1/\sqrt{n}$)
Spiked Wigner Model

Observe \( n \times n \) matrix \( Y = \lambda xx^T + W \)

Signal: \( x \in \mathbb{R}^n, \|x\| = 1 \)
Noise: \( W \in \mathbb{R}^{n \times n} \) with entries \( W_{ij} = W_{ji} \sim \mathcal{N}(0, 1/n) \) i.i.d.
\( \lambda > 0 \): signal-to-noise ratio

Goal: given \( Y \), estimate the signal \( x \)
Or, even simpler: distinguish (w.h.p.) \( Y \) from pure noise \( W \)

Structure: suppose \( x \) is drawn from some prior, e.g.
- spherical (uniform on unit sphere)
- Rademacher (i.i.d. \( \pm 1/\sqrt{n} \))
- sparse
PCA (Principal Component Analysis)

\[ Y = \lambda xx^T + W \]

Theorem (BBP'05, FP'06)

Almost surely, as \( n \to \infty \),

\[ \lambda_1(Y) \] and (unit-norm) eigenvector \( v_1 \)

If \( \lambda \leq 1 \):
\[ \lambda_1(Y) \to 2 \] and \( \langle x, v_1 \rangle \to 0 \)

If \( \lambda > 1 \):
\[ \lambda_1(Y) \to \lambda + 1 \] and \( \langle x, v_1 \rangle^2 \to 1 - 1/\lambda^2 > 0 \)

Sharp threshold: PCA can detect and recover the signal iff \( \lambda > 1 \)

---

D. Feral, S. Peche, CMP 2006.
PCA (Principal Component Analysis)

\[ Y = \lambda xx^T + W \]

PCA: top eigenvalue \( \lambda_1(Y) \) and (unit-norm) eigenvector \( v_1 \)

\[ \text{Theorem (BBP'05, FP'06)} \]

Almost surely, as \( n \to \infty \),

\[ \begin{align*}
\text{If } \lambda \leq 1: & \\
\lambda_1(Y) & \to 2 \\
\langle x, v_1 \rangle & \to 0 \\
\text{If } \lambda > 1: & \\
\lambda_1(Y) & \to \lambda + 1 \\
\langle x, v_1 \rangle & \to 1 - 1/\lambda^2 > 0
\end{align*} \]

Sharp threshold: PCA can detect and recover the signal iff \( \lambda > 1 \)

---

D. Feral, S. Peche, CMP 2006.
PCA (Principal Component Analysis)

\[ Y = \lambda x x^T + W \]
PCA: top eigenvalue \( \lambda_1(Y) \) and (unit-norm) eigenvector \( v_1 \)

Theorem (BBP’05, FP’06)

Almost surely, as \( n \to \infty \),

\[ \lambda_1(Y) \to \lambda + 1 \quad \text{and} \quad \langle x, v_1 \rangle \to 1 - \frac{1}{\lambda^2} > 0 \]

---

D. Feral, S. Peche, CMP 2006.
PCA (Principal Component Analysis)

\[ Y = \lambda x x^T + W \]

PCA: top eigenvalue \( \lambda_1(Y) \) and (unit-norm) eigenvector \( v_1 \)

**Theorem (BBP’05, FP’06)**

Almost surely, as \( n \to \infty \),

- If \( \lambda \leq 1 \): \( \lambda_1(Y) \to 2 \) and \( \langle x, v_1 \rangle \to 0 \)

---

D. Feral, S. Peche, CMP 2006.
PCA (Principal Component Analysis)

\[ Y = \lambda xx^T + W \]

PCA: top eigenvalue \( \lambda_1(Y) \) and (unit-norm) eigenvector \( v_1 \)

**Theorem (BBP’05, FP’06)**

*Almost surely, as \( n \to \infty \),*

- If \( \lambda \leq 1 \): \( \lambda_1(Y) \to 2 \) and \( \langle x, v_1 \rangle \to 0 \)
- If \( \lambda > 1 \): \( \lambda_1(Y) \to \lambda + \frac{1}{\lambda} > 2 \) and \( \langle x, v_1 \rangle^2 \to 1 - 1/\lambda^2 > 0 \)

---

D. Feral, S. Peche, CMP 2006.
PCA (Principal Component Analysis)

\[ Y = \lambda xx^T + W \]

PCA: top eigenvalue \( \lambda_1(Y) \) and (unit-norm) eigenvector \( v_1 \)

**Theorem (BBP’05, FP’06)**

Almost surely, as \( n \to \infty \),

- If \( \lambda \leq 1 \): \( \lambda_1(Y) \to 2 \) and \( \langle x, v_1 \rangle \to 0 \)
- If \( \lambda > 1 \): \( \lambda_1(Y) \to \lambda + \frac{1}{\lambda} > 2 \) and \( \langle x, v_1 \rangle^2 \to 1 - 1/\lambda^2 > 0 \)

Sharp threshold: PCA can detect and recover the signal iff \( \lambda > 1 \)

---

D. Feral, S. Peche, CMP 2006.
Is PCA Optimal?

PCA does not exploit structure of signal $x$.

Is the PCA threshold ($\lambda = 1$) optimal?

▶ Is it statistically possible to detect/recover when $\lambda < 1$?

Answer: it depends on the prior for $x$.

For some priors (e.g. spherical, Rademacher), detection and recovery are statistically impossible when $\lambda < 1$ [MRZ'14, DAM'15, PWBM'18].

But what if $x$ is sparse?
Is PCA Optimal?

PCA does not exploit structure of signal $x$
Is PCA Optimal?

PCA does not exploit structure of signal $x$

Is the PCA threshold ($\lambda = 1$) optimal?

- Is it statistically possible to detect/recover when $\lambda < 1$?
Is PCA Optimal?

PCA does not exploit structure of signal $x$

Is the PCA threshold ($\lambda = 1$) optimal?

- Is it statistically possible to detect/recover when $\lambda < 1$?

Answer: it depends on the prior for $x$
Is PCA Optimal?

PCA does not exploit structure of signal $x$

Is the PCA threshold ($\lambda = 1$) optimal?
  ▶ Is it statistically possible to detect/recover when $\lambda < 1$?

Answer: it depends on the prior for $x$

For some priors (e.g. spherical, Rademacher), detection and recovery are statistically impossible when $\lambda < 1$ [MRZ’14, DAM’15, PWBM’18]
Is PCA Optimal?

PCA does not exploit structure of signal $x$

Is the PCA threshold ($\lambda = 1$) optimal?
  - Is it statistically possible to detect/recover when $\lambda < 1$?

Answer: it depends on the prior for $x$

For some priors (e.g. spherical, Rademacher), detection and recovery are statistically impossible when $\lambda < 1$ \cite{MRZ14, DAM15, PWBM18}

But what if $x$ is sparse?
Sparse PCA

Suppose $x \in \mathbb{R}^n$ is drawn from the $k$-sparse Rademacher prior:

- $k$ random entries of $x$ are nonzero
- the nonzero entries are drawn uniformly from $\{\pm 1/\sqrt{k}\}$
Suppose $x \in \mathbb{R}^n$ is drawn from the $k$-sparse Rademacher prior:

- $k$ random entries of $x$ are nonzero
- the nonzero entries are drawn uniformly from $\{\pm 1/\sqrt{k}\}$

Normalization: $\|x\| = 1$
Sparse PCA

Suppose $x \in \mathbb{R}^n$ is drawn from the $k$-sparse Rademacher prior:

- $k$ random entries of $x$ are nonzero
- the nonzero entries are drawn uniformly from $\{ \pm 1/\sqrt{k} \}$

Normalization: $\|x\| = 1$

As before, $Y = \lambda xx^T + W$
Sparse PCA

Suppose \( x \in \mathbb{R}^n \) is drawn from the \( k \)-sparse Rademacher prior:

- \( k \) random entries of \( x \) are nonzero
- the nonzero entries are drawn uniformly from \( \{\pm 1/\sqrt{k}\} \)

Normalization: \( \|x\| = 1 \)

As before, \( Y = \lambda xx^T + W \)

Assume \( \lambda < 1 \) is a constant

- PCA fails

Johnstone, Lu '04, '09
Maximum Likelihood Estimator

Let $S_k := \{v \in \{0, \pm 1/\sqrt{k}\}^n : \|v\|_0 = k\}$
(set of $k$-sparse Rademacher vectors)
Maximum Likelihood Estimator

Let $S_k := \{v \in \{0, \pm 1/\sqrt{k}\}^n : \|v\|_0 = k\}$
(set of $k$-sparse Rademacher vectors)

MLE: $\hat{x} = \arg\max_{v \in S_k} v^T Y v$
Maximum Likelihood Estimator

Let \( S_k := \{ v \in \{0, \pm 1/\sqrt{k}\}^n : \|v\|_0 = k \} \)
(set of \( k \)-sparse Rademacher vectors)

\[
\text{MLE: } \hat{x} = \arg\max_{v \in S_k} v^\top Yv
\]

Succeeds (\( \hat{x} = x \) with high probability) provided \( k \lesssim n/\log n \)

[PJ'12, VL'12, CMW'13]
Maximum Likelihood Estimator

Let $S_k := \{v \in \{0, \pm 1/\sqrt{k}\}^n : \|v\|_0 = k\}$
(set of $k$-sparse Rademacher vectors)

MLE: $\hat{x} = \arg\max_{v \in S_k} v^\top Yv$

Succeeds ($\hat{x} = x$ with high probability) provided $k \lesssim n / \log n$

[PJ’12, VL’12, CMW’13]

- For weak recovery, $k < \rho^* n \approx 0.09n$

[LKZ’15, KXZ’16, DMK’16, LM’19, EKJ’17]
Maximum Likelihood Estimator

Let $S_k := \{v \in \{0, \pm 1/\sqrt{k}\}^n : \|v\|_0 = k\}$ (set of $k$-sparse Rademacher vectors)

MLE: $\hat{x} = \arg\max_{v \in S_k} v^\top Yv$

Succeeds ($\hat{x} = x$ with high probability) provided $k \lesssim n/\log n$ [PJ’12, VL’12, CMW’13]

- For weak recovery, $k < \rho^* n \approx 0.09n$ [LKZ’15, KXZ’16, DMK⁺’16, LM’19, EKJ’17]

Runtime: $\binom{n}{k} \approx n^k \approx \exp(k)$
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu ’09]:

- Identify the largest $k$ diagonal entries $Y_{ii}$
- Report these indices $i$ as the support of $x$
- (Easy to then recover $x$ once you know the support)

Succeeds (exact recovery) provided $k \lesssim \sqrt{n / \log n}$ [Amini, Wainwright ’08]

Runtime: polynomial

Variant: covariance thresholding is poly-time and succeeds when $k \lesssim \sqrt{n}$ (removes log factor) [Krauthgamer, Nadler, Vilenchik ’15, Deshpande, Montanari ’14]
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu ’09]:

- Identify the largest $k$ diagonal entries $Y_{ii}$
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu '09]:

- Identify the largest $k$ diagonal entries $Y_{ii}$
- Report these indices $i$ as the support of $x$
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu ’09]:

- Identify the largest $k$ diagonal entries $Y_{ii}$
- Report these indices $i$ as the support of $x$
- (Easy to then recover $x$ once you know the support)
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu '09]:

- Identify the largest $k$ diagonal entries $Y_{ii}$
- Report these indices $i$ as the support of $x$
- (Easy to then recover $x$ once you know the support)

Succeeds (exact recovery) provided $k \lesssim \sqrt{n / \log n}$ [Amini, Wainwright '08]
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu '09]:

- Identify the largest \( k \) diagonal entries \( Y_{ii} \)
- Report these indices \( i \) as the support of \( x \)
- (Easy to then recover \( x \) once you know the support)

Succeeds (exact recovery) provided \( k \lesssim \sqrt{n/\log n} \) [Amini, Wainwright '08]

Runtime: polynomial
Diagonal Thresholding

Diagonal thresholding algorithm [Johnstone, Lu '09]:

- Identify the largest $k$ diagonal entries $Y_{ii}$
- Report these indices $i$ as the support of $x$
- (Easy to then recover $x$ once you know the support)

Succeeds (exact recovery) provided $k \lesssim \sqrt{n / \log n}$ [Amini, Wainwright '08]

Runtime: polynomial

Variant: covariance thresholding is poly-time and succeeds when $k \lesssim \sqrt{n}$ (removes log factor) [Krauthgamer, Nadler, Vilenchik '15, Deshpande, Montanari '14]
Hard Regime

To summarize:

Statistically possible when \( k \ll n \) ▶ Runtime \( \exp(k) \)

Poly-time solvable when \( k \ll \sqrt{n} \)

Believed “hard” when \( \sqrt{n} \ll k \ll n \) ▶ Reduction from planted clique [BR’13, WBS’16, BBH’18, BB’19]

▶ Sum-of-squares lower bounds [MW’15, HKP’17]

Question: exactly how hard is the “hard” regime? ▶ Can you do better than \( \exp(k) \)? ▶ Reduction from planted clique doesn’t rule out quasipolynomial time \( n^{O(\log n)} \)
**Hard Regime**

To summarize:

Statistically possible when $k \ll n$

- Runtime $\exp(k)$

Believed “hard” when $\sqrt{n} \ll k \ll n$

- Reduction from planted clique \cite{BR'13, WBS'16, BBH'18, BB'19}

- Sum-of-squares lower bounds \cite{MW'15, HKP'17}

Question: exactly how hard is the “hard” regime?

- Can you do better than $\exp(k)$?

- Reduction from planted clique doesn’t rule out quasipolynomial time $n^{O(\log n)}$
Hard Regime

To summarize:

Statistically possible when $k \ll n$
  - Runtime $\exp(k)$

Poly-time solvable when $k \ll \sqrt{n}$
Hard Regime

To summarize:

Statistically possible when $k \ll n$
- Runtime $\exp(k)$

Poly-time solvable when $k \ll \sqrt{n}$

Believed “hard” when $\sqrt{n} \ll k \ll n$
Hard Regime

To summarize:

Statistically possible when \( k \ll n \)
  - Runtime \( \exp(k) \)

Poly-time solvable when \( k \ll \sqrt{n} \)

Believed “hard” when \( \sqrt{n} \ll k \ll n \)
  - Reduction from planted clique [BR’13, WBS’16, BBH’18, BB’19]
  - Sum-of-squares lower bounds [MW’15, HKP’17]
Hard Regime

To summarize:

Statistically possible when $k \ll n$
  - Runtime $\exp(k)$

Poly-time solvable when $k \ll \sqrt{n}$

Believed “hard” when $\sqrt{n} \ll k \ll n$
  - Reduction from planted clique [BR’13, WBS’16, BBH’18, BB’19]
  - Sum-of-squares lower bounds [MW’15, HKP’17]

Question: exactly how hard is the “hard” regime?
Hard Regime

To summarize:

Statistically possible when \( k \ll n \)
  - Runtime \( \exp(k) \)

Poly-time solvable when \( k \ll \sqrt{n} \)

Believed “hard” when \( \sqrt{n} \ll k \ll n \)
  - Reduction from planted clique [BR’13, WBS’16, BBH’18, BB’19]
  - Sum-of-squares lower bounds [MW’15, HKP’17]

Question: exactly how hard is the “hard” regime?
  - Can you do better than \( \exp(k) \)?
Hard Regime

To summarize:

Statistically possible when $k \ll n$
- Runtime $\exp(k)$

Poly-time solvable when $k \ll \sqrt{n}$

Believed “hard” when $\sqrt{n} \ll k \ll n$
- Reduction from planted clique $[\text{BR}'13, \text{WBS}'16, \text{BBH}'18, \text{BB}'19]$
- Sum-of-squares lower bounds $[\text{MW}'15, \text{HKP}^+ '17]$

Question: exactly how hard is the “hard” regime?
- Can you do better than $\exp(k)$?
- Reduction from planted clique doesn’t rule out quasipolynomial time $n^{O(\log n)}$
Hard Regime

To summarize:

Statistically possible when $k \ll n$
  - Runtime $\exp(k)$

Poly-time solvable when $k \ll \sqrt{n}$

Believed “hard” when $\sqrt{n} \ll k \ll n$
  - Reduction from planted clique [BR’13, WBS’16, BBH’18, BB’19]
  - Sum-of-squares lower bounds [MW’15, HKP+’17]

Question: exactly how hard is the “hard” regime?
  - Can you do better than $\exp(k)$? Yes: $\exp(k^2/n)$
  - Reduction from planted clique doesn’t rule out quasipolynomial time $n^{O(\log n)}$
Low-Degree Prediction

Hypothesis testing between:

- $\mathcal{P} : Y = \lambda xx^\top + W$ with $x$ drawn from $k$-sparse prior
- $\mathcal{Q} : Y = W$

Theorem (Ding, Kunisky, W., Bandeira '19)

Suppose $\lambda = \Theta(1)$.

- If $\lambda < 1$ and $D \ll k^2/n$ then $\|L\| = O(1)$ ("hard")
- If $\lambda > 1$ or $D \gg k^2/n$ then $\|L\| = \omega(1)$ ("easy")

So degree-$D$ polynomials can distinguish iff $\lambda > 1$ or $D \gg k^2/n$

Suggests an algorithm of runtime $n k^2/n \approx \exp(k^2/n)$ (and no better)

Subexponential time: $\exp(n \delta)$ with $\delta \in (0,1)$

And indeed we will find such an algorithm...
Low-Degree Prediction

Hypothesis testing between:

- **P**: \( Y = \lambda x x^\top + W \) with \( x \) drawn from \( k \)-sparse prior
- **Q**: \( Y = W \)

**Theorem (Ding, Kunisky, W., Bandeira ’19)**

Suppose \( \lambda = \Theta(1) \).

- If \( \lambda < 1 \) and \( D \ll k^2/n \) then \( \| L^{\leq D} \| = O(1) \) (“hard”)
- If \( \lambda > 1 \) or \( D \gg k^2/n \) then \( \| L^{\leq D} \| = \omega(1) \) (“easy”)

So degree-\( D \) polynomials can distinguish iff \( \lambda > 1 \) or \( D \gg k^2/n \)

Suggests an algorithm of runtime \( n k^2/n \approx \exp(k^2/n) \) (and no better)

- Subexponential time: \( \exp(n \delta) \) with \( \delta \in (0, 1) \)

And indeed we will find such an algorithm...
Low-Degree Prediction

Hypothesis testing between:

- \( \mathbb{P} : Y = \lambda xx^\top + W \) with \( x \) drawn from \( k \)-sparse prior
- \( \mathbb{Q} : Y = W \)

Theorem (Ding, Kunisky, W., Bandeira ’19)

Suppose \( \lambda = \Theta(1) \).

- If \( \lambda < 1 \) and \( D \ll k^2/n \) then \( \|L^{\leq D}\| = O(1) \) ("hard")
- If \( \lambda > 1 \) or \( D \gg k^2/n \) then \( \|L^{\leq D}\| = \omega(1) \) ("easy")

So degree-\( D \) polynomials can distinguish iff \( \lambda > 1 \) or \( D \gg k^2/n \)
Low-Degree Prediction

Hypothesis testing between:
- $\mathcal{P} : Y = \lambda x x^\top + W$ with $x$ drawn from $k$-sparse prior
- $\mathcal{Q} : Y = W$

Theorem (Ding, Kunisky, W., Bandeira ’19)

Suppose $\lambda = \Theta(1)$.
- If $\lambda < 1$ and $D \ll k^2/n$ then $\|L^{\leq D}\| = O(1)$ (“hard”)
- If $\lambda > 1$ or $D \gg k^2/n$ then $\|L^{\leq D}\| = \omega(1)$ (“easy”)

So degree-$D$ polynomials can distinguish iff $\lambda > 1$ or $D \gg k^2/n$

Suggests an algorithm of runtime $n^{k^2/n} \approx \exp(k^2/n)$ (and no better)
**Low-Degree Prediction**

Hypothesis testing between:
- $\mathbb{P} : Y = \lambda x x^\top + W$ with $x$ drawn from $k$-sparse prior
- $\mathbb{Q} : Y = W$

**Theorem (Ding, Kunisky, W., Bandeira ’19)**

*Suppose $\lambda = \Theta(1)$.)*

- If $\lambda < 1$ and $D \ll k^2/n$ then $\|L^{\leq D}\| = O(1)$ (“hard”)
- If $\lambda > 1$ or $D \gg k^2/n$ then $\|L^{\leq D}\| = o(1)$ (“easy”)

So degree-$D$ polynomials can distinguish iff $\lambda > 1$ or $D \gg k^2/n$

Suggests an algorithm of runtime $n^{k^2/n} \approx \exp(k^2/n)$ (and no better)

- Subexponential time: $\exp(n^\delta)$ with $\delta \in (0, 1)$
Low-Degree Prediction

Hypothesis testing between:

- \( \mathbb{P} : Y = \lambda xx^\top + W \) with \( x \) drawn from \( k \)-sparse prior
- \( \mathbb{Q} : Y = W \)

Theorem (Ding, Kunisky, W., Bandeira ’19)

Suppose \( \lambda = \Theta(1) \).

- If \( \lambda < 1 \) and \( D \ll k^2/n \) then \( \|L^{\leq D}\| = O(1) \) (“hard”)
- If \( \lambda > 1 \) or \( D \gg k^2/n \) then \( \|L^{\leq D}\| = \omega(1) \) (“easy”)

So degree-\( D \) polynomials can distinguish iff \( \lambda > 1 \) or \( D \gg k^2/n \)

Suggests an algorithm of runtime \( n^{k^2/n} \approx \exp(k^2/n) \) (and no better)

- Subexponential time: \( \exp(n^\delta) \) with \( \delta \in (0, 1) \)

And indeed we will find such an algorithm...
The Algorithm

For now, consider the detection problem ($\mathbb{P}$ vs $\mathbb{Q}$)
The Algorithm

For now, consider the detection problem ($\mathbb{P}$ vs $\mathbb{Q}$)

Choose a parameter $1 \leq \ell \leq k$
The Algorithm

For now, consider the detection problem \((\mathbb{P} \text{ vs } \mathbb{Q})\)

Choose a parameter \(1 \leq \ell \leq k\)

Let \(S_\ell := \{v \in \{\pm 1\}^n : \|v\|_0 = \ell\}\)
The Algorithm

For now, consider the detection problem ($P$ vs $Q$)

Choose a parameter $1 \leq \ell \leq k$

Let $S_{\ell} := \{ v \in \{\pm 1\}^n : \|v\|_0 = \ell \}$

Let $T := \max_{v \in S_{\ell}} v^\top Y v$

Algorithm: compute $T$ and threshold it (large $\Rightarrow P$)

$\triangleright \ell = k \Rightarrow$ exhaustive search (MLE)

$\triangleright \ell = 1 \Rightarrow$ diagonal thresholding $\max_i Y_{ii}$

Runtime: $(n \ell) \approx n \ell \approx \exp(\ell)$
The Algorithm

For now, consider the detection problem ($\mathbb{P}$ vs $\mathbb{Q}$)

Choose a parameter $1 \leq \ell \leq k$

Let $S_\ell := \{ v \in \{\pm 1\}^n : \|v\|_0 = \ell \}$

Let $T := \max_{v \in S_\ell} v^\top Yv$

Algorithm: compute $T$ and threshold it (large $\Rightarrow \mathbb{P}$)
The Algorithm

For now, consider the detection problem (\(P\ vs \ Q\))

Choose a parameter \(1 \leq \ell \leq k\)

Let \(S_\ell := \{ v \in \{ \pm 1 \}^n : \|v\|_0 = \ell \}\)

Let \(T := \max_{v \in S_\ell} v^\top Yv\)

Algorithm: compute \(T\) and threshold it (large \(\Rightarrow P\))

- \(\ell = k\ \Rightarrow\) exhaustive search (MLE)
- \(\ell = 1\ \Rightarrow\) diagonal thresholding \(\max_i Y_{ii}\)
The Algorithm

For now, consider the detection problem (\(\mathbb{P}\) vs \(\mathbb{Q}\))

Choose a parameter \(1 \leq \ell \leq k\)

Let \(S_\ell := \{v \in \{\pm 1\}^n : \|v\|_0 = \ell\}\)

Let \(T := \max_{v \in S_\ell} v^\top Yv\)

Algorithm: compute \(T\) and threshold it (large \(\Rightarrow \mathbb{P}\))

\(\bullet\) \(\ell = k\) \(\Rightarrow\) exhaustive search (MLE)

\(\bullet\) \(\ell = 1\) \(\Rightarrow\) diagonal thresholding \(\max_i Y_{ii}\)

Runtime: \(\binom{n}{\ell} \approx n^\ell \approx \exp(\ell)\)
Analysis of the Algorithm

Recall: algorithm thresholds  

\[ T := \max_{v \in S_\ell} v^\top Yv \]

Under \( P \), \( Y = \lambda x x^\top + W \), show \( T \) is large by considering a 'good' \( v \) (contained in \( x \)).

Under \( Q \), \( Y = W \), show \( T \) is small by Chernoff bound + union bound over \( S_\ell \).

Theorem (Ding, Kunisky, W., Bandeira '19): algorithm succeeds if \( \ell \gg k^2/n \) For any given \( k \), choose \( \ell \approx k^2/n \), get runtime \( \exp(k^2/n) \)
Analysis of the Algorithm

Recall: algorithm thresholds \( T := \max_{v \in S_\ell} v^\top Yv \)

Analysis:

- Under \( \mathbb{P} \), \( Y = \lambda xx^\top + W \), show \( T \) is large by considering a ‘good’ \( v \) (contained in \( x \))
Analysis of the Algorithm

Recall: algorithm thresholds $T := \max_{v \in S_\ell} v^\top Y v$

Analysis:

- **Under $\mathbb{P}$**, $Y = \lambda xx^\top + W$, show $T$ is large by considering a ‘good’ $v$ (contained in $x$)
- **Under $\mathbb{Q}$**, $Y = W$, show $T$ is small by Chernoff bound + union bound over $S_\ell$
Analysis of the Algorithm

Recall: algorithm thresholds $T := \max_{v \in S_\ell} v^\top Y v$

Analysis:

- Under $\mathbb{P}$, $Y = \lambda xx^\top + W$, show $T$ is large by considering a ‘good’ $v$ (contained in $x$)
- Under $\mathbb{Q}$, $Y = W$, show $T$ is small by Chernoff bound + union bound over $S_\ell$

Theorem (Ding, Kunisky, W., Bandeira ’19): algorithm succeeds if $\ell \gg k^2/n$
Analysis of the Algorithm

Recall: algorithm thresholds $T := \max_{v \in S_\ell} v^\top Y v$

Analysis:
- Under $\mathbb{P}$, $Y = \lambda xx^\top + W$, show $T$ is large by considering a ‘good’ $v$ (contained in $x$)
- Under $\mathbb{Q}$, $Y = W$, show $T$ is small by Chernoff bound + union bound bound over $S_\ell$

Theorem (Ding, Kunisky, W., Bandeira ’19): algorithm succeeds if $\ell \gg k^2/n$

For any given $k$, choose $\ell \approx k^2/n$, get runtime $\exp(k^2/n)$
From Detection to Recovery

Algorithm for recovering $x$ from $Y = \lambda x x^T + W$:
Algorithm for recovering $x$ from $Y = \lambda x x^T + W$:

1. Compute initial guess: $u = \arg\max_{v \in S} v^T Y v$

Theorem (Ding, Kunisky, W., Bandeira '19): $\hat{x} = x$ with high probability, provided $\ell \gg k^2/n$ (same as detection). Technically, need independent copies of $Y$ for steps 1 & 2.
From Detection to Recovery

Algorithm for recovering $x$ from $Y = \lambda xx^\top + W$:

1. Compute initial guess: $u = \arg\max_{v \in S_\ell} v^\top Yv$

But $u$ is too sparse...
From Detection to Recovery

Algorithm for recovering $x$ from $Y = \lambda xx^\top + W$:

1. Compute initial guess: $u = \arg\max_{v \in S^\ell} v^\top Yv$

   But $u$ is too sparse...

2. Let $w = Yu$
From Detection to Recovery

Algorithm for recovering $x$ from $Y = \lambda xx^\top + W$:

1. Compute initial guess: $u = \arg\max_{v \in S_\ell} v^\top Y v$

But $u$ is too sparse...

2. Let $w = Yu$

3. Construct $\hat{x} \in \{0, \pm 1/\sqrt{k}\}^n$ by thresholding entries of $w$
From Detection to Recovery

Algorithm for recovering $x$ from $Y = \lambda xx^\top + W$:

1. Compute initial guess: $u = \arg\max_{v \in S_\ell} v^\top Y v$

   But $u$ is too sparse...

2. Let $w = Y u$

3. Construct $\hat{x} \in \{0, \pm 1/\sqrt{k}\}^n$ by thresholding entries of $w$

Theorem (Ding, Kunisky, W., Bandeira '19): $\hat{x} = x$ with high probability, provided $\ell \gg k^2/n$ (same as detection)
From Detection to Recovery

Algorithm for recovering $x$ from $Y = \lambda xx^T + W$:

1. Compute initial guess: $u = \arg\max_{v \in S_\ell} v^T Y v$

But $u$ is too sparse...

2. Let $w = Yu$

3. Construct $\hat{x} \in \{0, \pm 1/\sqrt{k}\}^n$ by thresholding entries of $w$

Theorem (Ding, Kunisky, W., Bandeira ’19): $\hat{x} = x$ with high probability, provided $\ell \gg k^2/n$ (same as detection)

Technically, need independent copies of $Y$ for steps 1 & 2

$Y + W'$ and $Y - W'$ where $W'$ is independent copy of $W$
Summary

- Continuum of subexponential-time algorithms for sparse PCA

- Smooth interpolation between diagonal thresholding and exhaustive search

- Smooth tradeoff between sparsity and runtime: $\exp\left(\frac{k^2}{n}\right)$

- Extensions:
  - Allow $\lambda \ll 1$; runtime $\exp\left(\frac{k^2}{\lambda^2 n}\right)$
  - Spiked Wishart model
  - More general assumptions on $x$

- Optimal: for a given $k$, the low-degree likelihood ratio suggests that no better runtime is possible

Thanks!
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: $\exp(k^2/n)$
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: $\exp(k^2/n)$
- Extensions:
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: $\exp(k^2/n)$
- Extensions:
  - Allow $\lambda \ll 1$; runtime $\exp(k^2/\lambda^2 n)$
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: \( \exp\left(\frac{k^2}{n}\right) \)
- Extensions:
  - Allow \( \lambda \ll 1 \); runtime \( \exp\left(\frac{k^2}{(\lambda^2 n)}\right) \)
  - Spiked Wishart model
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: $\exp\left(\frac{k^2}{n}\right)$
- Extensions:
  - Allow $\lambda \ll 1$; runtime $\exp\left(\frac{k^2}{\lambda^2 n}\right)$
  - Spiked Wishart model
  - More general assumptions on $x$
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: \( \exp(k^2/n) \)
- Extensions:
  - Allow \( \lambda \ll 1 \); runtime \( \exp(k^2/(\lambda^2 n)) \)
  - Spiked Wishart model
  - More general assumptions on \( x \)
- Optimal: for a given \( k \), the low-degree likelihood ratio suggests that no better runtime is possible
Summary

- Continuum of subexponential-time algorithms for sparse PCA
- Smooth interpolation between diagonal thresholding and exhaustive search
- Smooth tradeoff between sparsity and runtime: $\exp(k^2/n)$
- Extensions:
  - Allow $\lambda \ll 1$; runtime $\exp(k^2/(\lambda^2 n))$
  - Spiked Wishart model
  - More general assumptions on $x$

- Optimal: for a given $k$, the low-degree likelihood ratio suggests that no better runtime is possible

Thanks!