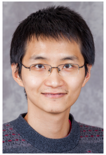
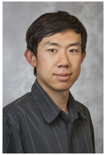
High-Dimensional Robust Mean Estimation in Nearly-Linear Time



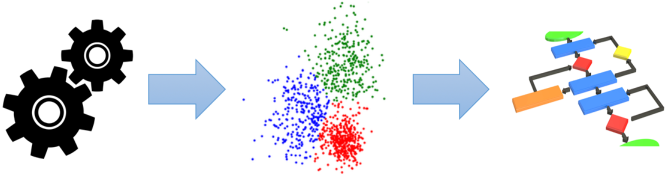


Yu Cheng [1] Ilias Diakonikolas [2] Rong Ge [1]

[1] Duke University

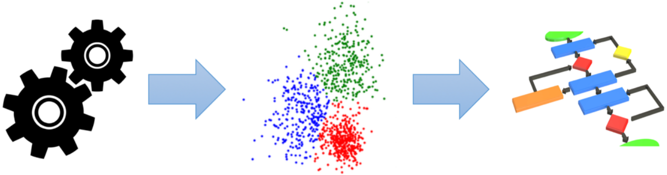
[2] University of Southern California

Statistical Learning



(Unknown) Parameters Samples Algorithms

Statistical Learning

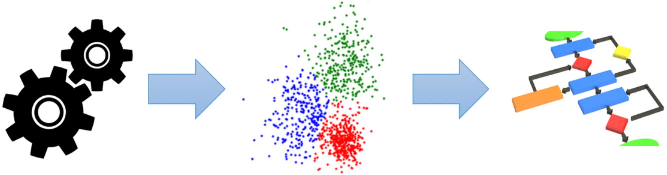


(Unknown) Parameters Samples Algorithms

Performance criteria:

- Sample complexity

Statistical Learning

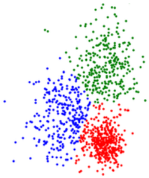


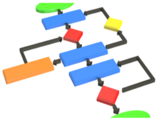
(Unknown) Parameters | Samples | Algorithms

Performance criteria:

- Sample complexity
- Running time

Statistical Learning





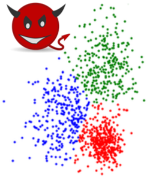
(Unknown) Parameters | Samples | Algorithms

Performance criteria:

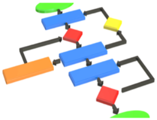
- Sample complexity
- Running time
- **Robustness**

Robust Statistical Learning

(Unknown) Parameters

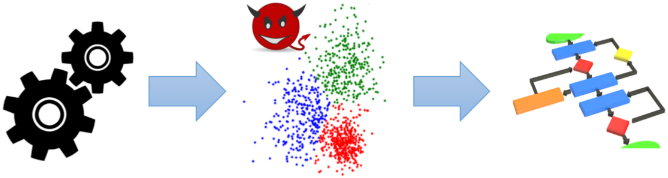


Corrupted samples



Algorithms

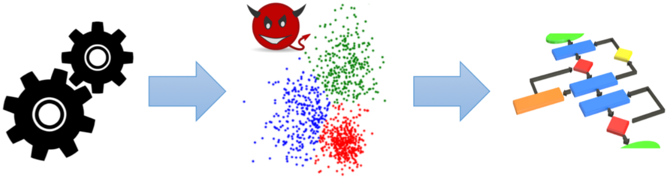
Robust Statistical Learning



(Unknown) Parameters Corrupted samples Algorithms

Q: Can we design provably robust and computational efficient learning algorithms when a small fraction of the data is corrupted?

Robust Statistical Learning



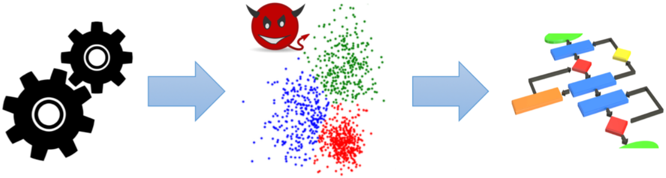
(Unknown) Parameters Corrupted samples Algorithms

Q: Can we design provably robust and computational efficient learning algorithms when a small fraction of the data is corrupted?

Motivation:

- Model misspecification / Robust statistics [Huber 1960s, Tukey 1960s, ...]

Robust Statistical Learning



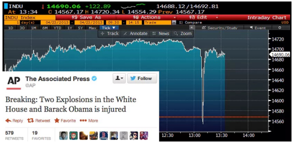
(Unknown) Parameters Corrupted samples Algorithms

Q: Can we design provably robust and computational efficient learning algorithms when a small fraction of the data is corrupted?

Motivation:

- Model misspecification / Robust statistics [Huber 1960s, Tukey 1960s, ...]
- Data poisoning attacks, Reliable / Adversarial / Secure ML

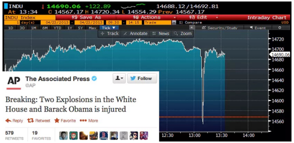
Motivation



Data Poisoning: High-frequency trading algorithms.

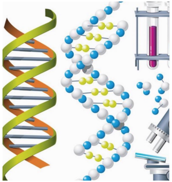
Twitter account of the Associated Press was hacked in April 2013 (\$136 billion in 3 minutes).

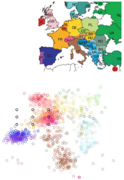
Motivation



Data Poisoning: High-frequency trading algorithms.

Twitter account of the Associated Press was hacked in April 2013 ($136 billion in 3 minutes).



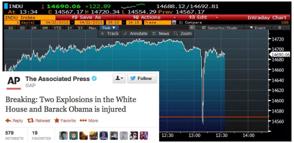


Biological Datasets: POPRES project, HGDP datasets.

High-dimensional datasets tend to be inherently noisy.

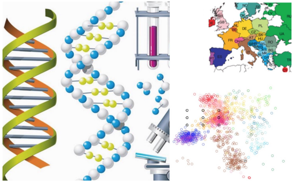
Hard to detect in several cases [Rosenberg et al., Science'02; Li et al., Science'08; Paschou et al., Medical Genetics'10]

Motivation



Data Poisoning: High-frequency trading algorithms.

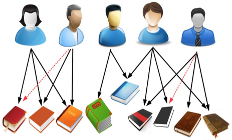
Twitter account of the Associated Press was hacked in April 2013 ($136 billion in 3 minutes).



Biological Datasets: POPRES project, HGDP datasets.

High-dimensional datasets tend to be inherently noisy.

Hard to detect in several cases [Rosenberg et al., Science'02; Li et al., Science'08; Paschou et al., Medical Genetics'10]



Reliable/Adversarial/Secure ML:

Recommendation Systems, Crowdsourcing, ...

Attacker can generate malicious data to maximize his objectives. [Mayzlin et al. '14] [Wang et al. '14] [Li et al. '16]

Mean Estimation

Mean Estimation

- *Input:* N samples $\{X_1, \ldots, X_N\}$ drawn from $\mathcal{N}(\mu^\star, I)$ on $\mathbb{R}^d$.
- *Goal:* Learn $\mu^\star$.

Mean Estimation

Mean Estimation

- *Input:* N samples $\{X_1, \ldots, X_N\}$ drawn from $\mathcal{N}(\mu^\star, I)$ on $\mathbb{R}^d$.
- *Goal:* Learn $\mu^\star$.
- Empirical mean $\widehat{\mu} = \frac{1}{N} \sum_{i=1}^N X_i$ works:

$$\|\widehat{\mu} - \mu^\star\|_2 \le \epsilon \text{ when } N = \Omega(d/\epsilon^2).$$

Mean Estimation

Mean Estimation

- *Input:* N samples $\{X_1, \ldots, X_N\}$ drawn from $\mathcal{N}(\mu^\star, I)$ on $\mathbb{R}^d$.
- *Goal:* Learn $\mu^\star$.

- Empirical mean $\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} X_i$ works:

  $$\|\widehat{\mu} - \mu^\star\|_2 \le \epsilon \text{ when } N = \Omega(d/\epsilon^2).$$

- Running time: $O(Nd)$.

Robust Mean Estimation

Definition (ϵ-Corruption)

- N samples are drawn i.i.d. from the ground-truth distribution D.

Robust Mean Estimation

Definition (ϵ-Corruption)

- N samples are drawn i.i.d. from the ground-truth distribution D.
- Adversary replaces ϵN samples with arbitrary points (after inspecting D, the samples, and the algorithm).

Robust Mean Estimation

Definition (ϵ-Corruption)

- N samples are drawn i.i.d. from the ground-truth distribution D.
- Adversary replaces ϵN samples with arbitrary points (after inspecting D, the samples, and the algorithm).

Robust Mean Estimation

- *Input:* an ϵ-corrupted set of N samples $\{X_1, \ldots, X_N\}$ drawn from an unknown distribution D on $\mathbb{R}^d$ with mean $\mu^\star$.
- *Goal:* Learn $\mu^\star$ in ℓ_2-norm.

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error Guarantee	Poly-Time?

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error Guarantee	Poly-Time?
Tukey Median	$O(\epsilon)$	No
Geometric Median	$O(\epsilon\sqrt{d})$	Yes
Tournament	$O(\epsilon)$	No
Pruning	$O(\epsilon\sqrt{d})$	Yes
RANSAC	∞	Yes

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error Guarantee	Poly-Time?
Tukey Median	$O(\epsilon)$	No
Geometric Median	$O(\epsilon\sqrt{d})$	Yes
Tournament	$O(\epsilon)$	No
Pruning	$O(\epsilon\sqrt{d})$	Yes
RANSAC	∞	Yes
[LRV'16]	$O(\epsilon\sqrt{\log d})$	Yes
[DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	Yes

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error Guarantee	Poly-Time?
Tukey Median	$O(\epsilon)$	No
Geometric Median	$O(\epsilon\sqrt{d})$	Yes
Tournament	$O(\epsilon)$	No
Pruning	$O(\epsilon\sqrt{d})$	Yes
RANSAC	∞	Yes
[LRV'16]	$O(\epsilon\sqrt{\log d})$	Yes
[DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	Yes

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error (δ)	Runtime

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error (δ)	Runtime
Dimension Halving [LRV'16]	$O(\epsilon\sqrt{\log d})$	$\Omega(Nd^2)$ + SVD
Convex Programming [DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	Ellipsoid Algorithm
Filtering [DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$\Omega(Nd^2)$

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error (δ)	Runtime
Dimension Halving [LRV'16]	$O(\epsilon\sqrt{\log d})$	$\Omega(Nd^2)$ + SVD
Convex Programming [DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	Ellipsoid Algorithm
Filtering [DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$\Omega(Nd^2)$
This paper	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$\widetilde{O}(Nd/\epsilon^6)$

Previous Work

Robustly learn $\mu^\star$ given ϵ-corrupted samples from $\mathcal{N}(\mu^\star, I)$:

Algorithm	Error (δ)	Runtime
Dimension Halving [LRV'16]	$O(\epsilon\sqrt{\log d})$	$\Omega(Nd^2)$ + SVD
Convex Programming [DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	Ellipsoid Algorithm
Filtering [DKKLMS'16]	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$\Omega(Nd^2)$
This paper	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$\widetilde{O}(Nd/\epsilon^6)$

All these algorithms have the right sample complexity $N = O(d/\delta^2)$.

Our Results

Robustly learn $\mu^\star$ given ϵ-corrupted samples from D on $\mathbb{R}^d$.

Distribution	Error (δ)	# of Samples (N)	Runtime

Our Results

Robustly learn $\mu^\star$ given ϵ-corrupted samples from D on $\mathbb{R}^d$.

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance ($\Sigma \preceq I$)	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

Our Results

Robustly learn $\mu^\star$ given ϵ-corrupted samples from D on $\mathbb{R}^d$.

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance ($\Sigma \preceq I$)	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

When ϵ is constant, our algorithm has the best possible error guarantee, sample complexity, and running time (up to polylogarithmic factors).

Our Results

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance ($\Sigma \preceq I$)	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

Our Results

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance ($\Sigma \preceq I$)	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

Robust mean estimation under bounded covariance assumptions has been used as a subroutine to obtain robust learners for a wide range of supervised learning problems that can be phrased as stochastic convex programs.

Our Results

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance ($\Sigma \preceq I$)	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

Robust mean estimation under bounded covariance assumptions has been used as a subroutine to obtain robust learners for a wide range of supervised learning problems that can be phrased as stochastic convex programs.

Our result provides a faster implementation of such a subroutine, hence yields faster robust algorithms for all these problems.

Reweight the Samples

[DKKLMS'16]: To shift the empirical mean far from $\mu^\star$, the corrupted samples must introduce a large eigenvalue in the second-moment matrix.

Reweight the Samples

[DKKLMS'16]: To shift the empirical mean far from $\mu^\star$, the corrupted samples must introduce a large eigenvalue in the second-moment matrix.

- For $X \sim \mathcal{N}(\mu^\star, I)$, $\mathbb{E}[(X - \mu^\star)(X - \mu^\star)^\top] = I$.

Reweight the Samples

[DKKLMS'16]: To shift the empirical mean far from $\mu^\star$, the corrupted samples must introduce a large eigenvalue in the second-moment matrix.

- For $X \sim \mathcal{N}(\mu^\star, I)$, $\mathbb{E}[(X - \mu^\star)(X - \mu^\star)^\top] = I$.

Good Weights

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^N w_i (X_i - \mu^\star)(X_i - \mu^\star)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \qquad \left(\sum_i w_i = 1 \text{ and } 0 \le w_i \le \frac{1}{(1-\epsilon)N}\right) \end{array}$$

Reweight the Samples

[DKKLMS'16]: To shift the empirical mean far from $\mu^\star$, the corrupted samples must introduce a large eigenvalue in the second-moment matrix.

- For $X \sim \mathcal{N}(\mu^\star, I)$, $\mathbb{E}[(X - \mu^\star)(X - \mu^\star)^\top] = I$.

Good Weights

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^N w_i (X_i - \mu^\star)(X_i - \mu^\star)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \qquad \left(\sum_i w_i = 1 \text{ and } 0 \le w_i \le \frac{1}{(1-\epsilon)N}\right) \end{array}$$

Lemma ([DKKLMS'16])

If we can find a near-optimal solution w, we can output $\widehat{\mu}_w = \sum_i w_i X_i$.

Reweight the Samples

[DKKLMS'16]: To shift the empirical mean far from $\mu^\star$, the corrupted samples must introduce a large eigenvalue in the second-moment matrix.

- For $X \sim \mathcal{N}(\mu^\star, I)$, $\mathbb{E}[(X - \mu^\star)(X - \mu^\star)^\top] = I$.

Good Weights

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^N w_i (X_i - \mu^\star)(X_i - \mu^\star)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \qquad \left(\sum_i w_i = 1 \text{ and } 0 \le w_i \le \frac{1}{(1-\epsilon)N}\right) \end{array}$$

Lemma ([DKKLMS'16])

If we can find a near-optimal solution w, we can output $\widehat{\mu}_w = \sum_i w_i X_i$.

This looks like a packing SDP in w (which we can solve in nearly-linear time).

Except that ...

Reweight the Samples

[DKKLMS'16]: To shift the empirical mean far from $\mu^\star$, the corrupted samples must introduce a large eigenvalue in the second-moment matrix.

- For $X \sim \mathcal{N}(\mu^\star, I)$, $\mathbb{E}[(X-\mu^\star)(X-\mu^\star)^\top] = I$.

Good Weights

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^N w_i (X_i - \mu^\star)(X_i - \mu^\star)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \qquad \left(\sum_i w_i = 1 \text{ and } 0 \le w_i \le \frac{1}{(1-\epsilon)N}\right) \end{array}$$

Lemma ([DKKLMS'16])

If we can find a near-optimal solution w, we can output $\widehat{\mu}_w = \sum_i w_i X_i$.

This looks like a packing SDP in w (which we can solve in nearly-linear time).

Except that ... we do not know $\mu^\star$.

Our Approach

Idea: guess the mean ν and solve the SDP with parameter ν.

Our Approach

Idea: guess the mean ν and solve the SDP with parameter ν.

Primal SDP (with parameter ν)

$$\begin{aligned} &\text{minimize} \quad && \lambda_{\max}\left(\textstyle\sum_{i=1}^{N} w_i (X_i - \nu)(X_i - \nu)^\top\right) \\ &\text{subject to} \quad && w \in \Delta_{N,\epsilon} \end{aligned}$$

Our Approach

Idea: guess the mean ν and solve the SDP with parameter ν.

Primal SDP (with parameter ν)

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^{N} w_i (X_i - \nu)(X_i - \nu)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \end{array}$$

We give a win-win analysis: either

- a near-optimal solution w to the primal SDP give a good answer $\widehat{\mu}_w$, or

Our Approach

Idea: guess the mean ν and solve the SDP with parameter ν.

Primal SDP (with parameter ν)

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^{N} w_i (X_i - \nu)(X_i - \nu)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \end{array}$$

We give a win-win analysis: either

- a near-optimal solution w to the primal SDP give a good answer $\widehat{\mu}_w$, or
- a near-optimal solution to the dual SDP yields a new guess ν' that is closer to $\mu^\star$ by a constant factor.

Our Approach

Iteratively move ν closer to $\mu^\star$ using the dual SDP,

until primal SDP has a good solution and we can output $\widehat{\mu}_w$.

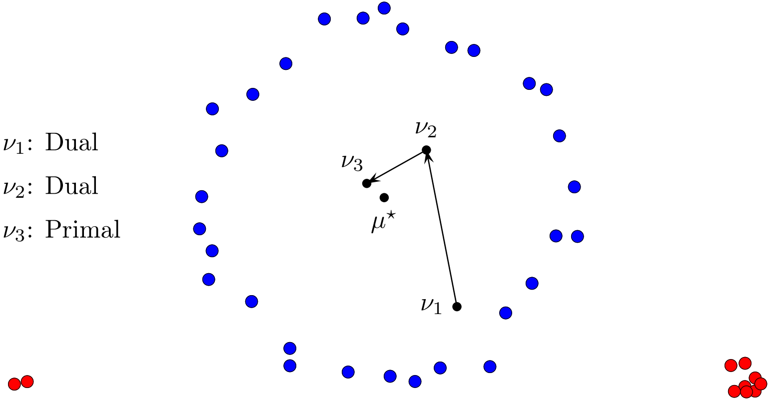
Our Approach

Iteratively move ν closer to $\mu^\star$ using the dual SDP,
until primal SDP has a good solution and we can output $\widehat{\mu}_w$.

ν_1: Dual

ν_2: Dual

ν_3: Primal



Our Approach

Iteratively move ν closer to $\mu^\star$ using the dual SDP:

Our Approach

Iteratively move ν closer to $\mu^\star$ using the dual SDP:

- Which direction is $\mu^\star$?

Our Approach

Iteratively move ν closer to $\mu^\star$ using the dual SDP:

- Which direction is $\mu^\star$?
- How far is $\mu^\star$?

Direction of $\mu^\star$: Dual SDP

Primal SDP (with parameter ν)

$$\begin{aligned} \text{minimize} \quad & \lambda_{\max}\left(\textstyle\sum_{i=1}^{N} w_i (X_i - \nu)(X_i - \nu)^\top\right) \\ \text{subject to} \quad & w \in \Delta_{N,\epsilon} \end{aligned}$$

Direction of $\mu^\star$: Dual SDP

Primal SDP (with parameter ν)

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^N w_i (X_i - \nu)(X_i - \nu)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \end{array}$$

SDP Duality

$$\min_{w \in \Delta_{N,\epsilon}} \max_{M \succeq 0, \operatorname{tr}(M) \le 1} \left\langle M, \sum_i w_i (X_i - \nu)(X_i - \nu)^\top \right\rangle$$

Direction of $\mu^\star$: Dual SDP

Primal SDP (with parameter ν)

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}\left(\sum_{i=1}^N w_i (X_i - \nu)(X_i - \nu)^\top\right) \\ \text{subject to} & w \in \Delta_{N,\epsilon} \end{array}$$

SDP Duality

$$\min_{w \in \Delta_{N,\epsilon}} \max_{M \succeq 0, \mathrm{tr}(M) \le 1} \left\langle M, \sum_i w_i (X_i - \nu)(X_i - \nu)^\top \right\rangle$$

$$\max_{M \succeq 0, \mathrm{tr}(M) \le 1} \min_{w \in \Delta_{N,\epsilon}} \left\langle M, \sum_i w_i (X_i - \nu)(X_i - \nu)^\top \right\rangle$$

Direction of $\mu^\star$: Dual SDP

Dual SDP (with parameter ν)

$$\begin{array}{ll} \text{maximize} & \text{Mean of the smallest } (1-\epsilon)\text{-fraction of } \left((X_i - \nu)^\top M (X_i - \nu)\right)_{i=1}^N \\ \text{subject to} & M \succeq 0, \mathrm{tr}(M) \le 1 \end{array}$$

Direction of $\mu^\star$: Dual SDP

Dual SDP (with parameter ν)

$$\begin{array}{ll} \text{maximize} & \text{Mean of the smallest } (1-\epsilon)\text{-fraction of } \left((X_i - \nu)^\top M (X_i - \nu)\right)_{i=1}^N \\ \text{subject to} & M \succeq 0, \operatorname{tr}(M) \le 1 \end{array}$$

- The dual SDP certifies that there are no good weights that can make the spectral norm small.

Direction of $\mu^\star$: Dual SDP

Dual SDP (with parameter ν)

$$\begin{aligned} &\text{maximize} && \text{Mean of the smallest } (1-\epsilon)\text{-fraction of } \left((X_i-\nu)^\top M (X_i-\nu)\right)_{i=1}^N \\ &\text{subject to} && M \succeq 0, \operatorname{tr}(M) \le 1 \end{aligned}$$

- The dual SDP certifies that there are no good weights that can make the spectral norm small.
- If the solution is rank-one: $M = yy^\top$, then in the direction of y, the variance is large no matter how we reweight the samples.

Direction of $\mu^\star$: Dual SDP

Dual SDP (with parameter ν)

$$\begin{aligned} \text{maximize} \quad & \text{Mean of the smallest } (1-\epsilon)\text{-fraction of } \left((X_i-\nu)^\top M (X_i-\nu)\right)_{i=1}^N \\ \text{subject to} \quad & M \succeq 0, \operatorname{tr}(M) \le 1 \end{aligned}$$

- The dual SDP certifies that there are no good weights that can make the spectral norm small.
- If the solution is rank-one: $M = yy^\top$, then in the direction of y, the variance is large no matter how we reweight the samples.
- Intuition: When ν is far from $\mu^\star$, y should align with $(\nu - \mu^\star)$.

Direction of $\mu^\star$: Dual SDP

Why would the dual SDP pick the direction $(\nu - \mu^\star)$?

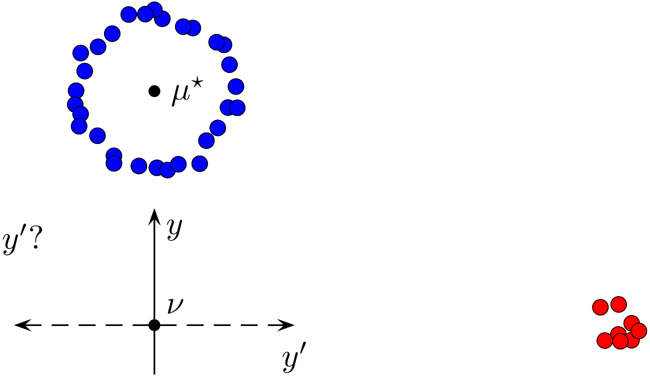
Direction of $\mu^\star$: Dual SDP

Why would the dual SDP pick the direction $(\nu - \mu^\star)$?

$$M = yy^\top$$

$$y \approx (\nu - \mu^\star)$$

Why is y better than y'?



How Far is $\mu^\star$: Optimal Value of the SDPs

Lemma

When $\|\nu - \mu^\star\|_2 \geq \ldots$,

$$1 + 0.99 \|\nu - \mu^\star\|_2^2 \leq OPT_\nu \leq 1 + 1.01 \|\nu - \mu^\star\|_2^2 .$$

Putting it Together: Moving ν Closer to $\mu^\star$

We show that despite the error from

Putting it Together: Moving ν Closer to $\mu^\star$

We show that despite the error from

- the errors in the concentration bounds, and
- we are only solving the SDP approximately,

Putting it Together: Moving ν Closer to $\mu^\star$

We show that despite the error from

- the errors in the concentration bounds, and
- we are only solving the SDP approximately,

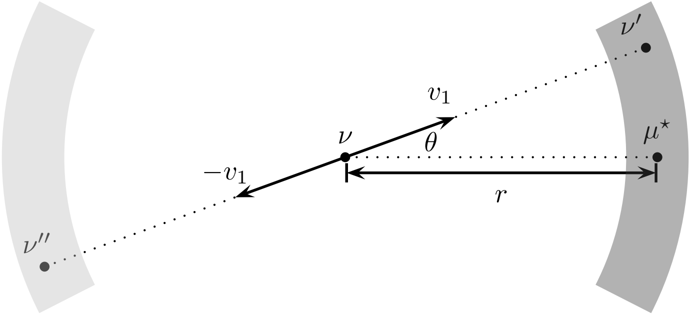
$\mathrm{OPT}_\nu \approx 1 + \|\nu - \mu^\star\|^2$, and the top eigenvector of M still aligns approximately with $(\nu - \mu^\star)$.

Putting it Together: Moving ν Closer to $\mu^\star$

We show that despite the error from

- the errors in the concentration bounds, and
- we are only solving the SDP approximately,

$\mathrm{OPT}_\nu \approx 1 + \|\nu - \mu^\star\|^2$, and the top eigenvector of M still aligns approximately with $(\nu - \mu^\star)$.



Full Algorithm: Sub-Gaussians

Algorithm 1: Robust Mean Estimation for Known Covariance Sub-Gaussian

Let ν be the coordinate-wise median of $\{X_i\}_{i=1}^N$;

for $i = 1$ **to** $O(\log d)$ **do**

Compute either

(i) a good solution $w \in \mathbb{R}^N$ for the primal SDP with parameters $(\nu, 2\epsilon)$; or

(ii) a good solution $M \in \mathbb{R}^{d \times d}$ for the dual SDP with parameters (ν, ϵ);

if *the objective value of w in primal SDP* $\leq 1 + c_0 \epsilon \ln(1/\epsilon)$ **then**

return *the weighted empirical mean* $\widehat{\mu}_w = \sum_{i=1}^N w_i X_i$;

else

Move ν closer to $\mu^\star$ using the top eigenvector of M.

Full Algorithm: Bounded Covariance

Algorithm 2: Robust Mean Estimation for Bounded Covariance Distributions

Let ν be the coordinate-wise median of $\{X_i\}_{i=1}^N$;

for $i = 1$ **to** $O(\log d)$ **do**

Compute either

(i) a good solution $w \in \mathbb{R}^N$ for the primal SDP with parameters $(\nu, 2\epsilon)$; or

(ii) a good solution $M \in \mathbb{R}^{d \times d}$ for the dual SDP with parameters (ν, ϵ);

if *the objective value of* w *in primal SDP is at most* c_1 **then**

return *the weighted empirical mean* $\widehat{\mu}_w = \sum_{i=1}^N w_i X_i$;

else

Move ν closer to $\mu^\star$ using the top eigenvector of M .

Summary: Robust Mean Estimation

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

Summary: Robust Mean Estimation

Distribution	Error (δ)	# of Samples (N)	Runtime
Sub-Gaussian	$O(\epsilon\sqrt{\log(1/\epsilon)})$	$O(d/\delta^2)$	$\widetilde{O}(Nd/\epsilon^6)$
Bounded Covariance	$O(\sqrt{\epsilon})$	$\widetilde{O}(d/\delta^2)$	

We hope our work will serve as a starting point for the design of faster algorithms for high-dimensional robust estimation.

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Input: ϵ-corrupted set of N samples drawn from $\mathcal{N}(0, \Sigma)$.

Goal: Estimate Σ.

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Input: ϵ-corrupted set of N samples drawn from $\mathcal{N}(0, \Sigma)$.

Goal: Estimate Σ.

Distribution	Error (δ)	# of Samples (N)	Runtime
Gaussian	$\left\|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\right\|_F = O(\epsilon \log(1/\epsilon))$ $\left\|\widehat{\Sigma} - \Sigma\right\|_F = O(\epsilon \log(1/\epsilon))$	$\widetilde{O}(d^2/\delta^2)$	$\widetilde{O}(d^{3.26} \log \kappa/\epsilon^8)$ $\widetilde{O}(d^{3.26}/\epsilon^8)$

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Input: ϵ-corrupted set of N samples drawn from $\mathcal{N}(0, \Sigma)$.

Goal: Estimate Σ.

Distribution	Error (δ)	# of Samples (N)	Runtime
Gaussian	$\left\\|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\right\\|_F = O(\epsilon \log(1/\epsilon))$ $\left\\|\widehat{\Sigma} - \Sigma\right\\|_F = O(\epsilon \log(1/\epsilon))$	$\widetilde{O}(d^2/\delta^2)$	$\widetilde{O}(d^{3.26} \log \kappa/\epsilon^8)$ $\widetilde{O}(d^{3.26}/\epsilon^8)$

All previous algorithms with similar error guarantee run in time $\Omega(d^{2\omega}) = \Omega(d^{4.74})$.

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Distribution	Error (δ)	# of Samples (N)	Runtime
Gaussian	$\left\|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\right\|_F = O(\epsilon \log(1/\epsilon))$ $\left\|\widehat{\Sigma} - \Sigma\right\|_F = O(\epsilon \log(1/\epsilon))$	$\widetilde{O}(d^2/\delta^2)$	$\widetilde{O}(d^{3.26} \log \kappa/\epsilon^8)$ $\widetilde{O}(d^{3.26}/\epsilon^8)$

Fast rectangular multiplication: $d \times d^2 \times d$ matrix multiplication can be done in time $O(d^{3.26})$.

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Distribution	Error (δ)	# of Samples (N)	Runtime
Gaussian	$\left\|\Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I\right\|_F = O(\epsilon \log(1/\epsilon))$ $\left\|\widehat{\Sigma} - \Sigma\right\|_F = O(\epsilon \log(1/\epsilon))$	$\widetilde{O}(d^2/\delta^2)$	$\widetilde{O}(d^{3.26} \log \kappa/\epsilon^8)$ $\widetilde{O}(d^{3.26}/\epsilon^8)$

Fast rectangular multiplication: $d \times d^2 \times d$ matrix multiplication can be done in time $O(d^{3.26})$.

Our runtime almost matches that of the best non-robust covariance estimation algorithm.

Computing the empirical covariance matrix $\frac{1}{N}\sum_{i=1}^{N} X_i X_i^\top$ takes $O(d^{3.26}/\epsilon^2)$ time.

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Distribution	Error (δ)	# of Samples (N)	Runtime
Gaussian	$\left\lVert \Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I \right\rVert_F = O(\epsilon \log(1/\epsilon))$ $\left\lVert \widehat{\Sigma} - \Sigma \right\rVert_F = O(\epsilon \log(1/\epsilon))$	$\widetilde{O}(d^2/\delta^2)$	$\widetilde{O}(d^{3.26} \log \kappa/\epsilon^8)$ $\widetilde{O}(d^{3.26}/\epsilon^8)$

Fast rectangular multiplication: $d \times d^2 \times d$ matrix multiplication can be done in time $O(d^{3.26})$.

Our runtime almost matches that of the best non-robust covariance estimation algorithm.
Computing the empirical covariance matrix $\frac{1}{N}\sum_{i=1}^{N} X_i X_i^\top$ takes $O(d^{3.26}/\epsilon^2)$ time.

$\mathbb{E}[XX^\top] = \Sigma$. Reduce to robust mean estimation with input $X \otimes X \in \mathbb{R}^{d^2}$.

Follow up: Robust Covariance Estimation [C Diakonikolas Ge Woodruff '19]

Distribution	Error (δ)	# of Samples (N)	Runtime
Gaussian	$\left\lVert \Sigma^{-1/2}\widehat{\Sigma}\Sigma^{-1/2} - I \right\rVert_F = O(\epsilon \log(1/\epsilon))$ $\left\lVert \widehat{\Sigma} - \Sigma \right\rVert_F = O(\epsilon \log(1/\epsilon))$	$\widetilde{O}(d^2/\delta^2)$	$\widetilde{O}(d^{3.26} \log \kappa/\epsilon^8)$ $\widetilde{O}(d^{3.26}/\epsilon^8)$

Fast rectangular multiplication: $d \times d^2 \times d$ matrix multiplication can be done in time $O(d^{3.26})$.

Our runtime almost matches that of the best non-robust covariance estimation algorithm.
Computing the empirical covariance matrix $\frac{1}{N}\sum_{i=1}^{N} X_i X_i^\top$ takes $O(d^{3.26}/\epsilon^2)$ time.

$\mathbb{E}[XX^\top] = \Sigma$. Reduce to robust mean estimation with input $X \otimes X \in \mathbb{R}^{d^2}$.

We use the primal-dual framework presented in this talk.

Naive implementation takes $\Omega(Nd^2) = \Omega(d^4)$ time. We need to open up the positive SDP solvers.

Open Problems

- Faster algorithms for other high-dimensional robust learning problems (e.g., sparse mean estimation / sparse PCA)?

Open Problems

- Faster algorithms for other high-dimensional robust learning problems (e.g., sparse mean estimation / sparse PCA)?
- Can we avoid the $\mathrm{poly}(1/\epsilon)$ in the runtime?