Robust List Decoding of Spherical Gaussians

Ilias Diakonikolas [1] **Daniel M. Kane** [2] Alistair Stewart [3]

[1]Department of Computer Science
University of Southern California
diakonik@usc.edu

[2]Departments of CS/Math
University of California, San Diego
dakane@ucsd.edu

[3]Department of Computer Science
University of Southern California
stewart.al@gmail.com

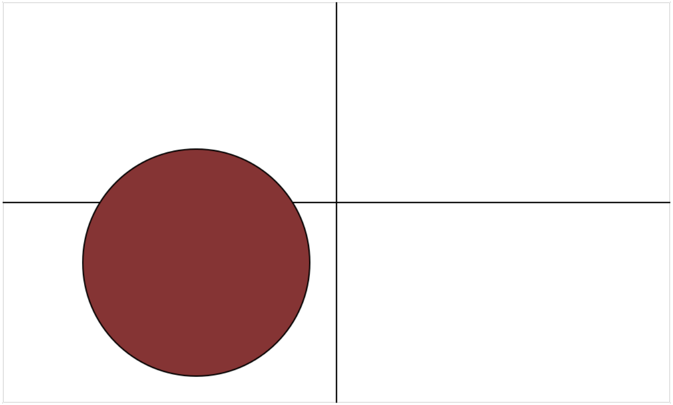
October 29th, 2018

Outline

- Problem Setup
- Information Theoretic Bounds
- Basic Multifilters
- Higher Degree Tests
- Learning Mixtures

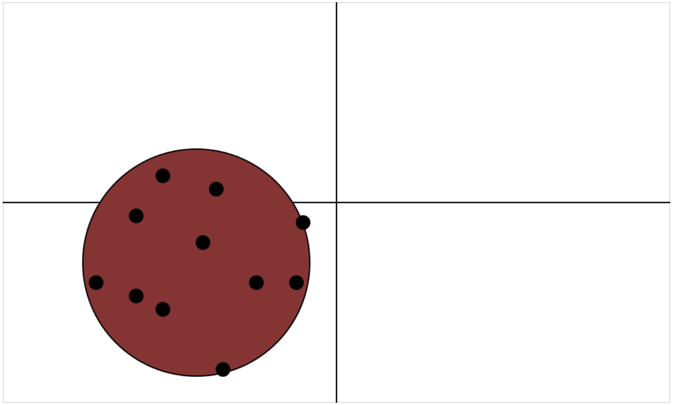
Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$



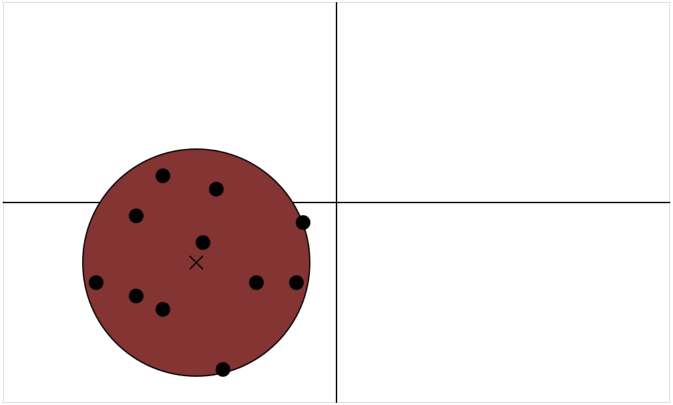
Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- Given m independent samples x_i from G



Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- Given m independent samples x_i from G
- Learn approximation to μ



Mean Estimation

- Classic statistics problem

Mean Estimation

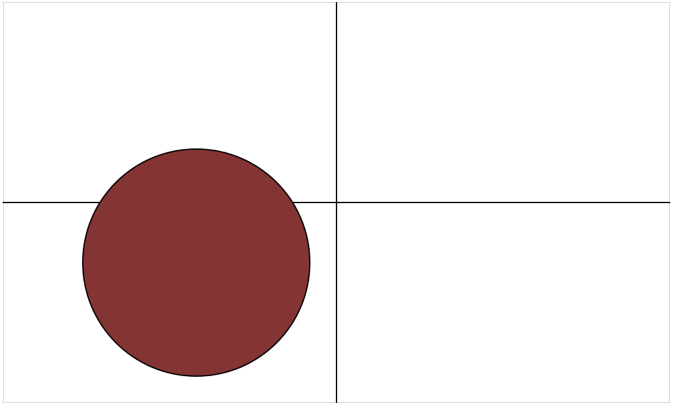
- Classic statistics problem
- Use $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$

Mean Estimation

- Classic statistics problem
- Use $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Error $O(\sqrt{n/m}) \to 0$

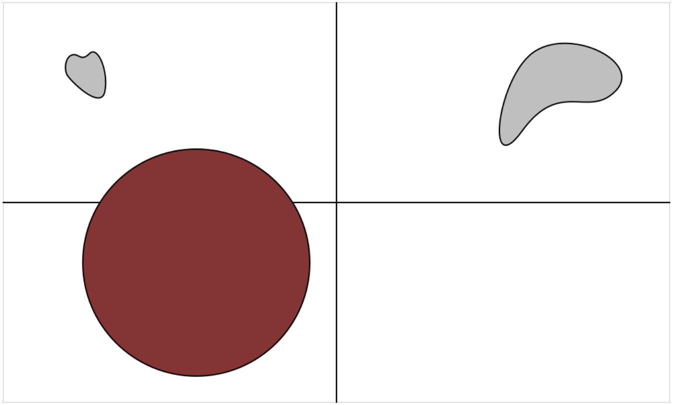
Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$



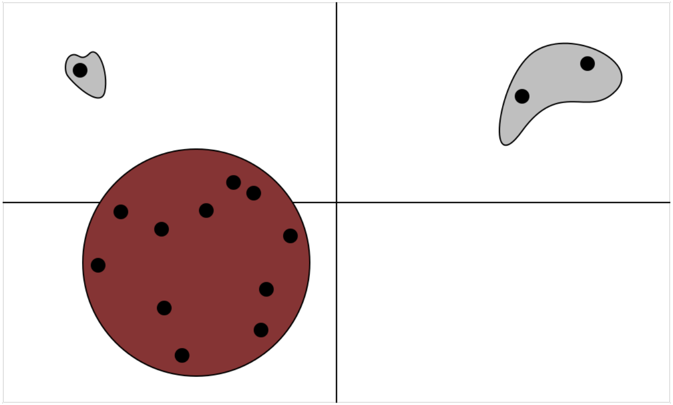
Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- $X = (1-\epsilon)G + \epsilon E$ for small ϵ



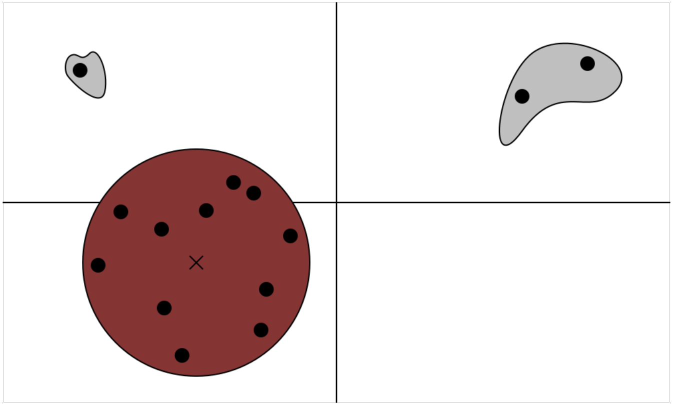
Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- $X = (1-\epsilon)G + \epsilon E$ for small ϵ
- Given m independent samples x_i of X



Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- $X = (1-\epsilon)G + \epsilon E$ for small ϵ
- Given m independent samples x_i of X
- Learn Approximation to μ



Robust Mean Estimation

- [Tukey] gave *exponential time* algorithm to attain $O(\epsilon)$ error (information theoretically optimal).

Robust Mean Estimation

- [Tukey] gave *exponential time* algorithm to attain $O(\epsilon)$ error (information theoretically optimal).
- Various polynomial time algorithms giving error $O(\epsilon\sqrt{n})$.

Robust Mean Estimation

- [Tukey] gave *exponential time* algorithm to attain $O(\epsilon)$ error (information theoretically optimal).
- Various polynomial time algorithms giving error $O(\epsilon\sqrt{n})$.
- [D-Kamath-K-Li-Moitra-S '16] gave polynomial time algorithm with $O(\epsilon\sqrt{\log(1/\epsilon)})$ error (against stronger error model).

Robust Mean Estimation

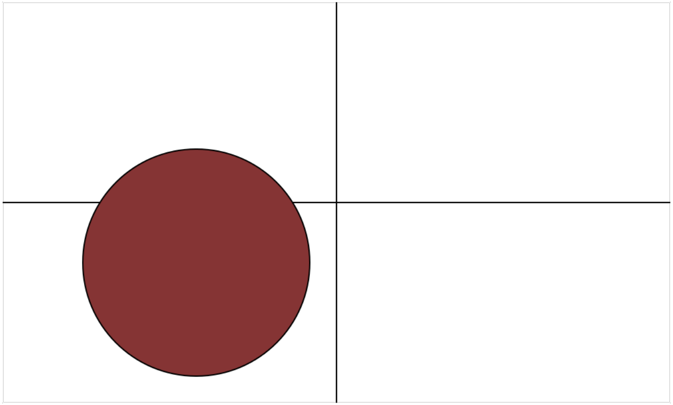
- [Tukey] gave *exponential time* algorithm to attain $O(\epsilon)$ error (information theoretically optimal).
- Various polynomial time algorithms giving error $O(\epsilon\sqrt{n})$.
- [D-Kamath-K-Li-Moitra-S '16] gave polynomial time algorithm with $O(\epsilon\sqrt{\log(1/\epsilon)})$ error (against stronger error model).
- [D-Kamath-K-Li-Moitra-S '18] gave polynomial time algorithm for $O(\epsilon)$ error

Robust Mean Estimation

- [Tukey] gave *exponential time* algorithm to attain $O(\epsilon)$ error (information theoretically optimal).
- Various polynomial time algorithms giving error $O(\epsilon\sqrt{n})$.
- [D-Kamath-K-Li-Moitra-S '16] gave polynomial time algorithm with $O(\epsilon\sqrt{\log(1/\epsilon)})$ error (against stronger error model).
- [D-Kamath-K-Li-Moitra-S '18] gave polynomial time algorithm for $O(\epsilon)$ error
- Substantial recent work on similar robust statistics problems

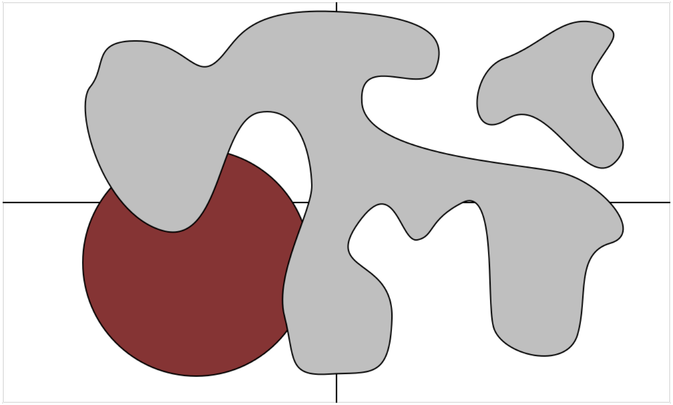
Very Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$



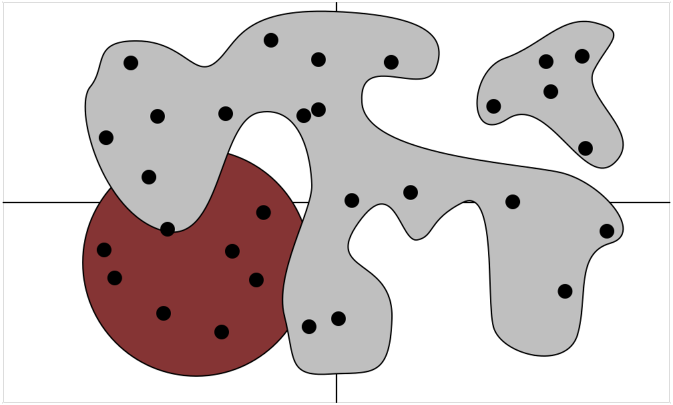
Very Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- $X = \alpha G + (1-\alpha)E$ for small α



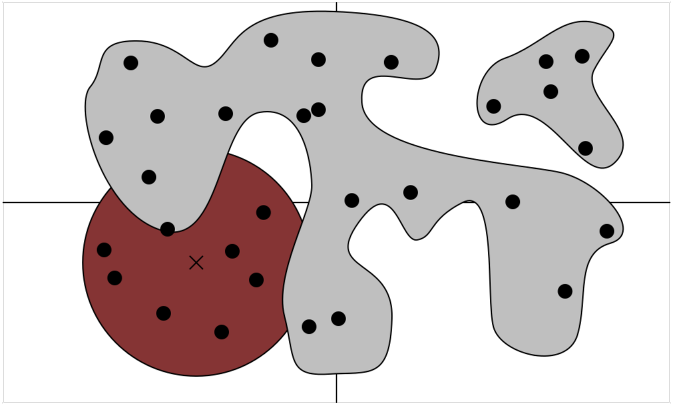
Very Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- $X = \alpha G + (1-\alpha)E$ for small α
- Given m independent samples x_i of X



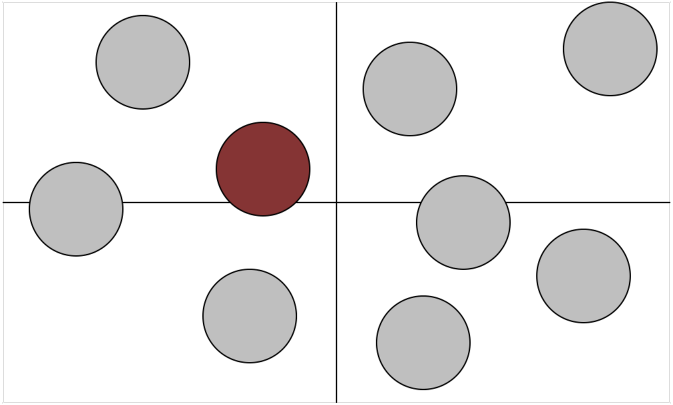
Very Robust Mean Estimation

- Gaussian $G = N(\mu, I) \subset \mathbb{R}^n$
- $X = \alpha G + (1-\alpha)E$ for small α
- Given m independent samples x_i of X
- Learn Approximation to μ



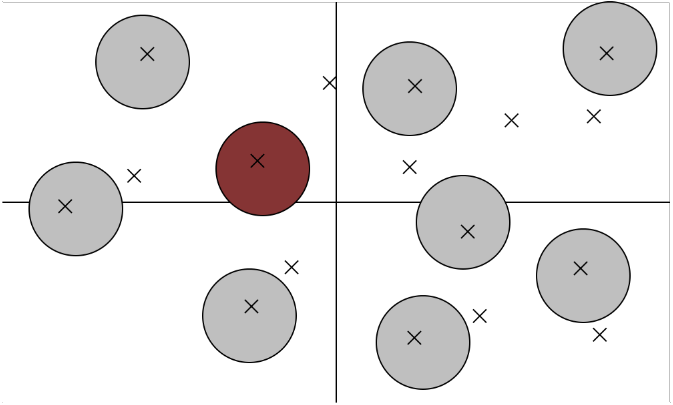
Problem

What if $X = \sum_i \alpha_i G_i$? Which is the "real" G?



Problem

What if $X = \sum_i \alpha_i G_i$? Which is the "real" G?



List decoding: return several hypotheses h_i with guarantee that at least one is close.

Robust List Decoding

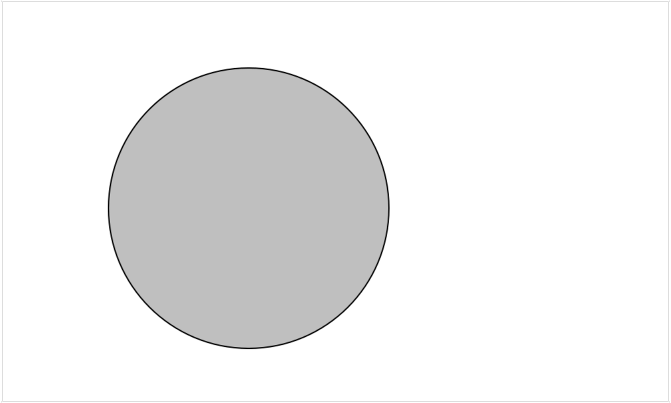
- [Steinhardt-Charikar-Valiant '17] first to study problem
  - Polynomial time (convex programming)
  - $O(1/\alpha)$ hypotheses
  - $\tilde{O}(\alpha^{-1/2})$ error

Information Theoretic Bounds

Before we begin, we should determine what errors are information-theoretically possible.

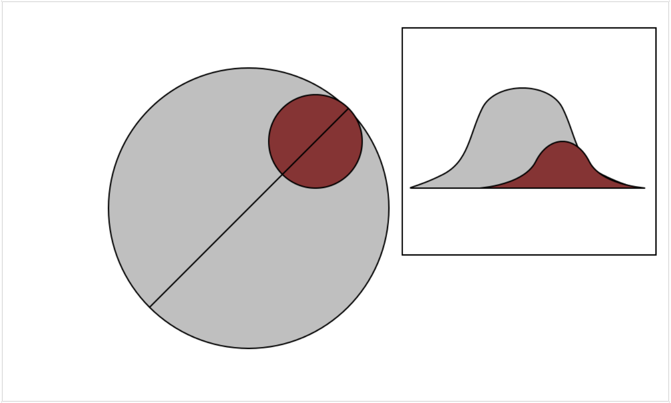
Lower Bounds

- Suppose $X = N(0, I)$.



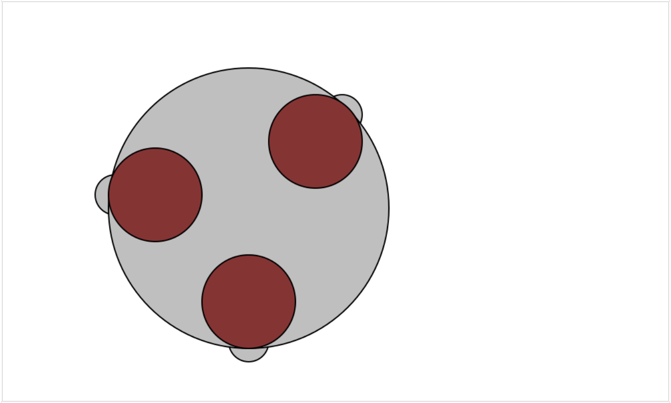
Lower Bounds

- Suppose $X = N(0, I)$.
- Any $\alpha N(\mu, I)$ with $|\mu| \leq \sqrt{\log(1/\alpha)}/C$ nearly hides under X (up to $\alpha^{\Omega(C)}$ error).



Lower Bounds

- Suppose $X = N(0, I)$.
- Any $\alpha N(\mu, I)$ with $|\mu| \leq \sqrt{\log(1/\alpha)}/C$ nearly hides under X (up to $\alpha^{\Omega(C)}$ error).
- Adding a bit to X, can hide $\alpha^{-\Omega(C)}$ such Gaussians.



Lower Bounds

Proposition

There is no algorithm that returns $poly(1/\alpha)$ many hypothesis so that with at least $2/3$ probability, at least one is within $o(\sqrt{\log(1/\alpha)})$ of the true mean.

- Let X be the slightly modified Gaussian.
- There are $\alpha^{-\Omega(C)}$ possibilities, no two within $\sqrt{\log(1/\alpha)}/C$.
- Algorithm cannot tell which possibility is correct, and must return a hypothesis for each.

Upper Bounds

Proposition

There is an (inefficient) algorithm that returns $O(1/\alpha)$ hypotheses so that with at least $2/3$ probability, at least one of the hypotheses is within $O(\sqrt{\log(1/\alpha)})$ of the true mean.

Hypotheses

Let H be the set of points x for which there is a set S_x of samples so that:

- S_x is large: it contains at least an $\alpha/2$-fraction of the samples.
- S_x is concentrated about x: in any direction, at most a $\alpha/10$-fraction of the points S_x are further than $2\sqrt{\log(1/\alpha)}$ from x in that direction.

Hypotheses

Let H be the set of points x for which there is a set S_x of samples so that:

- S_x is large: it contains at least an $\alpha/2$-fraction of the samples.
- S_x is concentrated about x: in any direction, at most a $\alpha/10$-fraction of the points S_x are further than $2\sqrt{\log(1/\alpha)}$ from x in that direction.

Note that with high probability $\mu \in H$ with $S_\mu = \text{the good samples}$.

Hypotheses

Let H be the set of points x for which there is a set S_x of samples so that:

- S_x is large: it contains at least an $\alpha/2$-fraction of the samples.
- S_x is concentrated about x: in any direction, at most a $\alpha/10$-fraction of the points S_x are further than $2\sqrt{\log(1/\alpha)}$ from x in that direction.

Note that with high probability $\mu \in H$ with $S_\mu = \text{the good samples.}$

Problem: Too many hypotheses.

Idea

Cover H with a small number of balls.

Lemma

There is no set of $5/\alpha$ *elements of H that are pairwise separated by at least* $4\sqrt{\log(1/\alpha)}$.

Idea

Cover H with a small number of balls.

Lemma

There is no set of $5/\alpha$ elements of H that are pairwise separated by at least $4\sqrt{\log(1/\alpha)}$.

Take a maximal set of $4\sqrt{\log(1/\alpha)}$-separated hypotheses.

- Size at most $5/\alpha$.
- Every element of H (including μ) within $4\sqrt{\log(1/\alpha)}$ of one.

Overlaps

Idea: If x and y far away, then S_x and S_y have little overlap. If many separated x's, then too many points.

Overlaps

Idea: If x and y far away, then S_x and S_y have little overlap. If many separated x's, then too many points.

Lemma

If $x, y \in H$ *with* $|x-y| \geq 4\sqrt{\log(1/\epsilon)}$, *then* $|S_x \cap S_y| \leq \alpha/10(|S_x| + |S_y|)$.

Overlaps

Idea: If x and y far away, then S_x and S_y have little overlap. If many separated x's, then too many points.

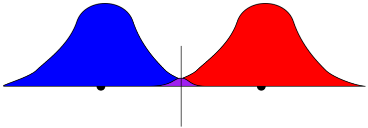
Lemma

If $x, y \in H$ *with* $|x - y| \geq 4\sqrt{\log(1/\epsilon)}$, *then* $|S_x \cap S_y| \leq \alpha/10(|S_x| + |S_y|)$.

Proof.

- Project onto the line between x and y.
- At most $\alpha|S_x|/10$ items from S_x closer to y than x.
- At most $\alpha|S_y|/10$ items from S_y closer to x than y.

□



Counting

If $x_1, x_2, \ldots, x_m \in H$ pairwise far, then

$$|S_{x_1} \cup S_{x_2} \cup \ldots \cup S_{x_m}| \geq \sum_{i=1}^{m} |S_{x_i}| - \sum_{1 \leq i < j \leq m} \alpha/10(|S_{x_i}| + |S_{x_j}|)$$
$$= \sum_{i=1}^{m} |S_{x_i}|(1 - m\alpha/10)$$
$$\geq m\alpha/2|S|(1 - m\alpha/10).$$

Counting

If $x_1, x_2, \ldots, x_m \in H$ pairwise far, then

$$\begin{aligned}
|S_{x_1} \cup S_{x_2} \cup \ldots \cup S_{x_m}| &\geq \sum_{i=1}^{m} |S_{x_i}| - \sum_{1 \leq i < j \leq m} \alpha/10(|S_{x_i}| + |S_{x_j}|) \\
&= \sum_{i=1}^{m} |S_{x_i}|(1 - m\alpha/10) \\
&\geq m\alpha/2|S|(1 - m\alpha/10).
\end{aligned}$$

If $m = 5/\alpha$, this is more than the total number of samples.

Notes

- If the good samples have all but $\alpha/10$-fraction within t of the mean in any direction, can get $O(1/\alpha)$ hypotheses with error $O(t)$.

Notes

- If the good samples have all but $\alpha/10$-fraction within t of the mean in any direction, can get $O(1/\alpha)$ hypotheses with error $O(t)$.
- Given a set H of hypotheses at least one within r of true mean, can in poly-time reduce to a set of $O(1/\alpha)$ with error $O(r + \sqrt{\log(1/\alpha)})$.

Notes

- If the good samples have all but $\alpha/10$-fraction within t of the mean in any direction, can get $O(1/\alpha)$ hypotheses with error $O(t)$.
- Given a set H of hypotheses at least one within r of true mean, can in poly-time reduce to a set of $O(1/\alpha)$ with error $O(r + \sqrt{\log(1/\alpha)})$.
  - Use LP to determine if there is a set S_x with concentration about x in the directions $x - y$.
  - Cover remaining x's with balls.

Summary

- [Steinhardt-Charikar-Valiant '17] gives an algorithm that attains $\tilde{O}(\alpha^{-1/2})$ error.
- Information-theoretically can achieve $O(\sqrt{\log(1/\alpha)})$ error.

Summary

- [Steinhardt-Charikar-Valiant '17] gives an algorithm that attains $\tilde{O}(\alpha^{-1/2})$ error.
- Information-theoretically can achieve $O(\sqrt{\log(1/\alpha)})$ error.

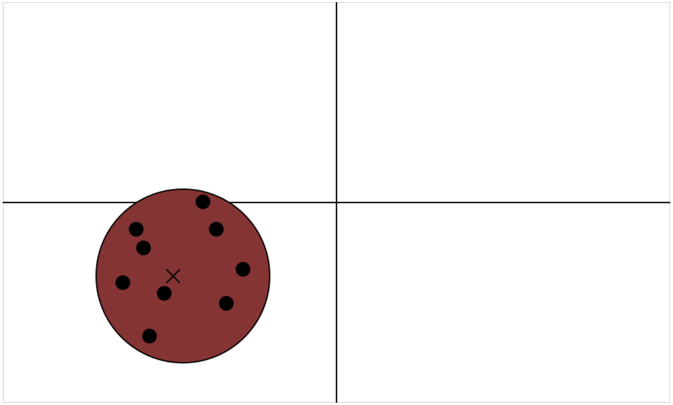
Question: What is achievable efficiently?

Algorithms

- Filters and Multifilters
- Obstacle at $\alpha^{-1/2}$.
- Higher Degree Idea
- Variance Control

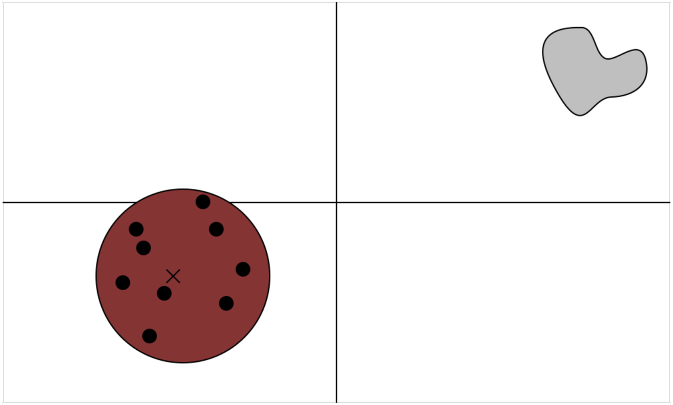
Sample Mean

- For non-robust algorithm use sample mean $\hat{\mu}$.



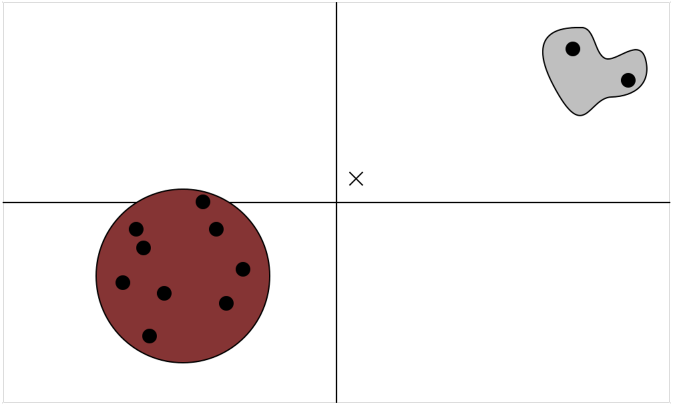
Sample Mean

- For non-robust algorithm use sample mean $\hat{\mu}$.
- For moderately-robust problem would like to use $\hat{\mu}$.



Sample Mean

- For non-robust algorithm use sample mean $\hat{\mu}$.
- For moderately-robust problem would like to use $\hat{\mu}$.
- **Problem:** A few bad samples can seriously change the sample mean.



Identifying Errors

Want to certify $\mu_X \approx \mu$.

Identifying Errors

Want to certify $\mu_X \approx \mu$.

- Otherwise, some unit vector v so that $v \cdot (\mu_X - \mu)$ is large.

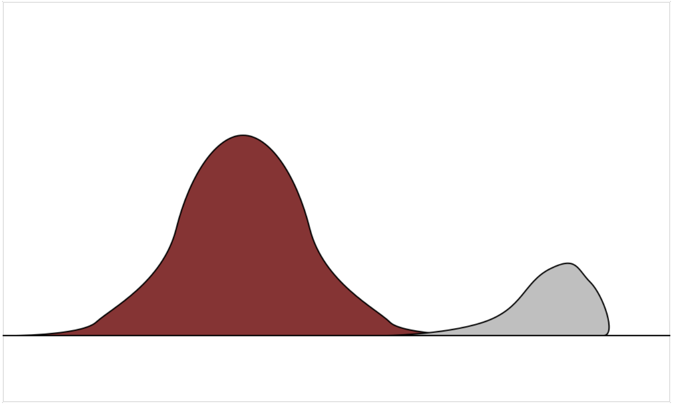
Identifying Errors

Want to certify $\mu_X \approx \mu$.

- Otherwise, some unit vector v so that $v \cdot (\mu_X - \mu)$ is large.
- Requires $\mathrm{Var}(v \cdot X)$ is large.
- Can detect this.

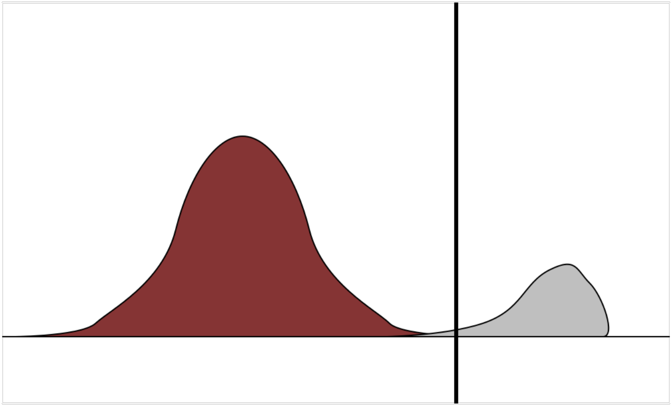
Filters

If $\mathrm{Var}(v \cdot X)$ large, must be some outliers for $v \cdot X$.



Filters

If $\mathrm{Var}(v \cdot X)$ large, must be some outliers for $v \cdot X$.
Can create a filter that throws away mostly bad samples.



Moderately Robust Algorithm

1. Take set S of samples
2. Compute empirical covariance matrix $\hat{\Sigma}$
3. If largest eigenvalue is small
   - Return sample mean μ_S
4. Else
   - Create filter
   - Apply to S
   - Go to step 2.

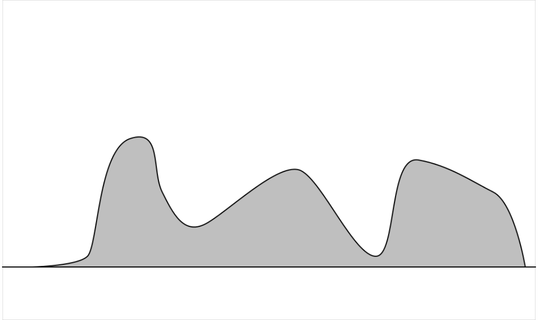
Moderately Robust Algorithm

1. Take set S of samples
2. Compute empirical covariance matrix $\hat{\Sigma}$
3. If largest eigenvalue is small
    - Return sample mean μ_S
4. Else
    - Create filter
    - Apply to S
    - Go to step 2.

Each iteration either returns an answer or produces a cleaner sample.

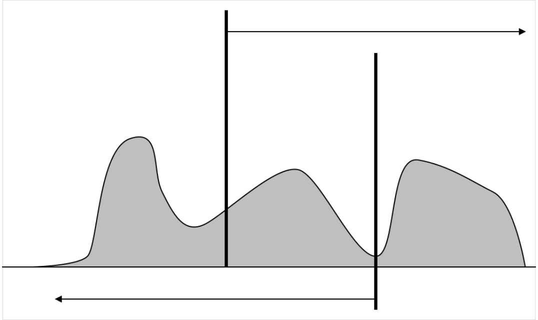
Multifilters

If $\alpha < 1/2$, might not be able to tell where the real samples are.



Multifilters

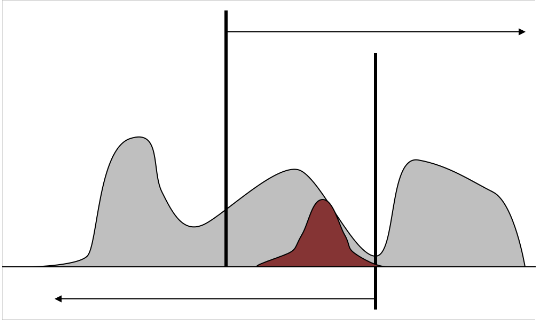
If $\alpha < 1/2$, might not be able to tell where the real samples are.



Split into several overlapping sets of samples S_i

Multifilters

If $\alpha < 1/2$, might not be able to tell where the real samples are.



Split into several overlapping sets of samples S_i so that:

- At least one S_i has higher fraction of good samples than S
- $\sum |S_i|^2 \leq |S|^2$

Analysis

Split into cases

- **Case 1:** Almost all of the samples are in the same small interval.
- **Case 2:** There are clusters of samples far apart from each other.

Filter Case

Suppose that there is an interval I containing all but an $\alpha/3$-fraction of samples.

Filter Case

Suppose that there is an interval I containing all but an $\alpha/3$-fraction of samples.

- With high probability, true mean in I.

Filter Case

Suppose that there is an interval I containing all but an $\alpha/3$-fraction of samples.

- With high probability, true mean in I.
- All but a tiny fraction of good samples within $O(\sqrt{\log(1/\alpha)})$ of I.

Filter Case

Suppose that there is an interval I containing all but an $\alpha/3$-fraction of samples.

- With high probability, true mean in I.
- All but a tiny fraction of good samples within $O(\sqrt{\log(1/\alpha)})$ of I.
- Unless variance is $O(|I|^2 + \log(1/\alpha))$, so that at most an α^2-fraction of removed samples were good.

Multifilter Case

Suppose that there is an interval I with at least an $\alpha/6$-fraction of samples on either side of it.

Multifilter Case

Suppose that there is an interval I with at least an $\alpha/6$-fraction of samples on either side of it.

- Find some x, let $S_1 = \{\text{samples} \leq x + 10\sqrt{\log(1/\alpha)}\}$, $S_2 = \{\text{samples} \geq x - 10\sqrt{\log(1/\alpha)}\}$.

Multifilter Case

Suppose that there is an interval I with at least an $\alpha/6$-fraction of samples on either side of it.

- Find some x, let $S_1 = \{\text{samples} \leq x + 10\sqrt{\log(1/\alpha)}\}$, $S_2 = \{\text{samples} \geq x - 10\sqrt{\log(1/\alpha)}\}$.
- All but an α^2-fraction of removed samples (on the correct side) are bad:
  - If $\mu \geq x$, all but α^3-fraction of good samples in S_2.
  - If $\mu \leq x$, all but α^3-fraction in S_1.
  - Always throw away at least $\alpha/6$ samples.

Multifilter Case

Suppose that there is an interval I with at least an $\alpha/6$-fraction of samples on either side of it.

- Find some x, let $S_1 = \{\text{samples} \leq x + 10\sqrt{\log(1/\alpha)}\}$, $S_2 = \{\text{samples} \geq x - 10\sqrt{\log(1/\alpha)}\}$.
- All but an α^2-fraction of removed samples (on the correct side) are bad:
  - If $\mu \geq x$, all but α^3-fraction of good samples in S_2.
  - If $\mu \leq x$, all but α^3-fraction in S_1.
  - Always throw away at least $\alpha/6$ samples.
- **Need:** $|S_1|^2 + |S_2|^2 \leq |S|^2$.

Analysis

- Let $f(x)$ be the fraction of samples less than x.

Analysis

- Let $f(x)$ be the fraction of samples less than x.
- Need $x \in I$ so that $(1 - f(x))^2 + f(x + 20\sqrt{\log(1/\alpha)})^2 \leq 1$.

Analysis

- Let $f(x)$ be the fraction of samples less than x.
- Need $x \in I$ so that $(1 - f(x))^2 + f(x + 20\sqrt{\log(1/\alpha)})^2 \leq 1$.
- Happens unless $f(x + 20\sqrt{\log(1/\alpha)}) \gg f(x)^{1/2}$.

Analysis

- Let $f(x)$ be the fraction of samples less than x.
- Need $x \in I$ so that $(1 - f(x))^2 + f(x + 20\sqrt{\log(1/\alpha)})^2 \leq 1$.
- Happens unless $f(x + 20\sqrt{\log(1/\alpha)}) \gg f(x)^{1/2}$.
- Good unless $f(x + 20t\sqrt{\log(1/\alpha)}) \gg \alpha^{1/2^t}$, only works for $t \ll \log\log(1/\alpha)$.

Analysis

- Let $f(x)$ be the fraction of samples less than x.
- Need $x \in I$ so that $(1 - f(x))^2 + f(x + 20\sqrt{\log(1/\alpha)})^2 \leq 1$.
- Happens unless $f(x + 20\sqrt{\log(1/\alpha)}) \gg f(x)^{1/2}$.
- Good unless $f(x + 20t\sqrt{\log(1/\alpha)}) \gg \alpha^{1/2^t}$, only works for $t \ll \log\log(1/\alpha)$.

Can find such sets unless $|I| = O(\sqrt{\log(1/\alpha)}\log\log(1/\alpha))$.

General Situation

Can create a filter or multifilter if either:

- No interval I of length $O(\sqrt{\log(1/\alpha)}\log\log(1/\alpha))$ contains all but an $\alpha/3$-fraction of samples.
- An interval I of length $O(\sqrt{\log(1/\alpha)}\log\log(1/\alpha))$ contains all but an $\alpha/3$-fraction of samples, and the variance is $\Omega(|I|^2)$.

General Situation

Can create a filter or multifilter if either:

- No interval I of length $O(\sqrt{\log(1/\alpha)} \log\log(1/\alpha))$ contains all but an $\alpha/3$-fraction of samples.
- An interval I of length $O(\sqrt{\log(1/\alpha)} \log\log(1/\alpha))$ contains all but an $\alpha/3$-fraction of samples, and the variance is $\Omega(|I|^2)$.

Proposition

If the variance in some direction is more than a sufficient multiple of $\log(1/\alpha)$ (with a slight refinement of the argument) then we can find at most two sets of samples S_i so that

1. *For some i, at most an α^2-fraction of $S \setminus S_i$ is good samples.*
2. $\sum_i |S_i|^2 \leq |S|^2$.

Basic Multifilter Algorithm

1. Maintain several sets S_i of samples
2. For each i, compute empirical covariance matrix $\hat{\Sigma}_i$
3. If some $\hat{\Sigma}_i$ has a large eigenvalue
   - Create multifilter
   - Apply to S_i
   - Replace S_i by resulting sets in list
   - Go to step 2.
4. Return list of all μ_{S_i}

Analysis

At each step:

- At least one S_i has an α-fraction of good samples (in fact at least half of the total good samples)
- $\sum |S_i|^2 \leq |S|^2$

Analysis

At each step:

- At least one S_i has an α-fraction of good samples (in fact at least half of the total good samples)
- $\sum |S_i|^2 \leq |S|^2$

When return if:

- S_i has α-fraction of good samples AND
- $\hat{\Sigma}_i$ has no large eigenvalues

Analysis

At each step:

- At least one S_i has an α-fraction of good samples (in fact at least half of the total good samples)
- $\sum |S_i|^2 \leq |S|^2$

When return if:

- S_i has α-fraction of good samples AND
- $\hat{\Sigma}_i$ has no large eigenvalues

Then for all $|v| = 1$,

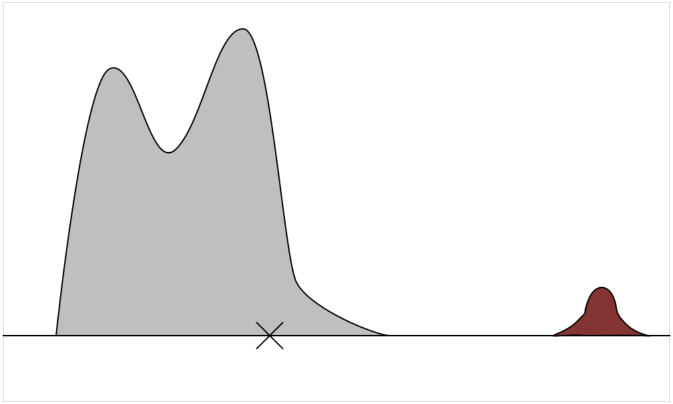
$$\log(1/\alpha) \gg \mathrm{Var}(v \cdot S_i) \geq \alpha[v \cdot (\mu_{S_i} - \mu)]^2,$$

so

$$|\mu_{S_i} - \mu| = O(\alpha^{-1/2}\sqrt{\log(1/\alpha)}).$$

Obstacle at $\alpha^{-1/2}$

Unfortunately, the error *can* be as much as $\alpha^{-1/2}$.



Idea

Bounds on the second moments are not enough to ensure concentration.

Idea

Bounds on the second moments are not enough to ensure concentration.
Fix: use higher moments.

Analysis

If for all unit vectors v,

$$\mathbb{E}[|v \cdot (X - \mu_X)|^{2d}] = O(1),$$

then

$$1 \gg \alpha |v \cdot (\mu - \mu_X)|^{2d},$$

so

$$|\mu - \mu_X| = O(\alpha^{-1/2d}).$$

Computational Difficulty

It is computationally intractable to determine whether or not there is a unit vector v for which $\mathbb{E}[(v \cdot X)^{2d}]$ is large when $d > 1$.

Computational Difficulty

It is computationally intractable to determine whether or not there is a unit vector v for which $\mathbb{E}[(v \cdot X)^{2d}]$ is large when $d > 1$.
Idea: Look at a relaxation of this problem.

- [Hopkins-Li,Kothari-Steinhardt,Kothari-Steurer]: Look for SoS proof that $\mathbb{E}[(v \cdot X)^{2d}] \ll |v|_2^{2d}$ for all v.

Computational Difficulty

It is computationally intractable to determine whether or not there is a unit vector v for which $\mathbb{E}[(v \cdot X)^{2d}]$ is large when $d > 1$.
Idea: Look at a relaxation of this problem.

- [Hopkins-Li,Kothari-Steinhardt,Kothari-Steurer]: Look for SoS proof that $\mathbb{E}[(v \cdot X)^{2d}] \ll |v|_2^{2d}$ for all v.
- This talk: See if there is any degree-d polynomial p with $\mathbb{E}[p(X)^2]$ too big.

Basic Idea

Determine whether or not there is a degree-d polynomial p with $\mathbb{E}[p(S)^2]$ substantially larger than $\mathbb{E}[p(G_{\mu_S})^2]$.

Basic Idea

Determine whether or not there is a degree-d polynomial p with $\mathbb{E}[p(S)^2]$ substantially larger than $\mathbb{E}[p(G_{\mu_S})^2]$.

- Eigenvalue computation.
- If not, implies $|\mu - \mu_S| = \tilde{O}(\alpha^{-1/2d})$.
- If yes, create a (multi-)filter.

A Failed Attempt

If $\mathrm{Var}(p(X))$ is too large, create a (multi-)filter based on the values of p.

A Failed Attempt

If $\mathrm{Var}(p(X))$ is too large, create a (multi-)filter based on the values of p.

- Compute values of $p(x)$ for $x \in S$.
- Fairly spread out.
- Values of $p(G)$ are clustered.
- Use same multifilter ideas as before.

A Failed Attempt

If $\mathrm{Var}(p(X))$ is too large, create a (multi-)filter based on the values of p.

- Compute values of $p(x)$ for $x \in S$.
- Fairly spread out.
- Values of $p(G)$ are clustered.
- Use same multifilter ideas as before.

Problem: $\mathrm{Var}(p(G))$ might also be large!

A Failed Attempt

If $\mathrm{Var}(p(X))$ is too large, create a (multi-)filter based on the values of p.

- Compute values of $p(x)$ for $x \in S$.
- Fairly spread out.
- Values of $p(G)$ are clustered.
- Use same multifilter ideas as before.

Problem: $\mathrm{Var}(p(G))$ might also be large!

- Unlike degree-1 polynomials, for degree-d, $\mathrm{Var}(p(G))$ depends on μ.
- Want a way to verify that $\mathrm{Var}(p(G))$ is small.

The Strategy

Given a p with $\mathbb{E}[p(S)^2] \gg \mathbb{E}[p(G_{\mu_S})^2]$ try to either:

- Verify that $\mathbb{E}[p(G)^2] \approx \mathbb{E}[p(G_{\mu_S})^2]$
  - Can then filter out points with $p(x)^2$ too large.

The Strategy

Given a p with $\mathbb{E}[p(S)^2] \gg \mathbb{E}[p(G_{\mu_S})^2]$ try to either:

- Verify that $\mathbb{E}[p(G)^2] \approx \mathbb{E}[p(G_{\mu_S})^2]$
  - Can then filter out points with $p(x)^2$ too large.
- OR produce a (multi-)filter in failing to verify this.

Bounding $\mathbb{E}[p(G)^2]$

- For any degree-d polynomial p, $\mathbb{E}[p(G)^2] = q(\mu)$ for some degree-$2d$ polynomial q.

Bounding $\mathbb{E}[p(G)^2]$

- For any degree-d polynomial p, $\mathbb{E}[p(G)^2] = q(\mu)$ for some degree-$2d$ polynomial q.
- This in turn equals $\mathbb{E}[r(G_1, G_2, \ldots, G_{2d})]$ for some multilinear r with $|r| \approx |p|$ and G_i i.i.d. copies of G.

Bounding $\mathbb{E}[p(G)^2]$

- For any degree-d polynomial p, $\mathbb{E}[p(G)^2] = q(\mu)$ for some degree-$2d$ polynomial q.
- This in turn equals $\mathbb{E}[r(G_1, G_2, \ldots, G_{2d})]$ for some multilinear r with $|r| \approx |p|$ and G_i i.i.d. copies of G.

Point: If $\mathbb{E}[p(G)^2]$ is too big, then $r(x_1, x_2, \ldots, x_{2d})$ $(x_i \in S)$, has an α^{2d} chance of being large.

Large Values

Suppose that $r(x_1, x_2, \ldots, x_{2d})$ is much larger than expected.

Large Values

Suppose that $r(x_1, x_2, \dots, x_{2d})$ is much larger than expected.

- Assign x_i's one at a time.
- At some stage the size of the polynomial must jump.
- In particular,

$$\mathbb{E}[|r(x_1, x_2, \dots, x_{i+1}, G'_{i+2}, \dots, G'_{2d})|^2] \gg \mathbb{E}[|r(x_1, x_2, \dots, x_i, G'_{i+1}, \dots, G'_{2d})|^2]$$

where G'_j are i.i.d. copies of G_{μ_S}.

Quadratic

- Note that

$$s(y) = \mathbb{E}[|r(x_1, x_2, \ldots, x_i, y, G'_{i+2}, \ldots, G'_{2d})|^2]$$

is a quadratic polynomial in y with $s(x_{i+1}) \gg \mathbb{E}[s(G_{\mu_S})]$.

Quadratic

- Note that

$$s(y) = \mathbb{E}[|r(x_1, x_2, \ldots, x_i, y, G'_{i+2}, \ldots, G'_{2d})|^2]$$

is a quadratic polynomial in y with $s(x_{i+1}) \gg \mathbb{E}[s(G_{\mu_S})]$.

- Can diagonalize s as

$$s(y) = \sum L_j(y)^2$$

for linear polynomials L_j.

Quadratic

- Note that

$$s(y) = \mathbb{E}[|r(x_1, x_2, \ldots, x_i, y, G'_{i+2}, \ldots, G'_{2d})|^2]$$

  is a quadratic polynomial in y with $s(x_{i+1}) \gg \mathbb{E}[s(G_{\mu_S})]$.
- Can diagonalize s as

$$s(y) = \sum L_j(y)^2$$

  for linear polynomials L_j.
- So there must be some j for which $L_j(x_{i+1})$ is much larger than expected. This will let us create a (multi-)filter.

Algorithm

1. Try to find polynomial p with $\mathbb{E}[p(S)^2] \gg \log^{4d}(1/\alpha)\mathbb{E}[p(G_{\mu_S})^2]$.
   - If none exist, return μ_S.
2. Compute corresponding multilinear r. See if $|r(x_1, \ldots, x_{2d})|^2 \gg \log^{2d}(1/\alpha)\mathbb{E}[p(G_{\mu_S})^2]$ with probability at least α^{2d}.
   - If not, $\mathbb{E}[p(G)^2]$ is small, filter out x with $p(x)^2$ more than average, and return to step 1.
3. Find $x_1, x_2, \ldots, x_i$ so that with α probability over $y \in S$, $|r(x_1, \ldots, x_i, y)|^2 \gg \log(1/\alpha)|r(x_1, \ldots, x_i)|^2$.
4. Compute the corresponding quadratic $s(y) = \sum L_j(y)^2$.
5. Find an j so that $L_j(y)$ is likely larger than expected. Use to create a (multi-)filter. Apply and return to step 1.

Requirements

Samples:

- S needs to be representative of G with respect to polynomials of degree $2d$.
- $|S| = \mathrm{poly}(n^d/\alpha)$.

Requirements

Samples:

- S needs to be representative of G with respect to polynomials of degree $2d$.
- $|S| = \mathrm{poly}(n^d/\alpha)$.

Runtime:

- Need to check for events with probability α^{2d}.
- Runtime is $\mathrm{poly}(|S|/\alpha^d)$.

Final Results

Theorem

There exists an algorithm that given $O(d^{2d})n^{O(d)}/poly(\alpha)$ i.i.d. samples from X, there is an $(nd/\alpha)^{O(d)}$ time algorithm which with high probability returns a list of $O(1/\alpha)$ hypotheses so that at least one hypothesis is within $\tilde{O}_d(\alpha^{-1/2d})$ of μ.

Final Results

Theorem

There exists an algorithm that given $O(d^{2d})n^{O(d)}/poly(\alpha)$ i.i.d. samples from X, there is an $(nd/\alpha)^{O(d)}$ time algorithm which with high probability returns a list of $O(1/\alpha)$ hypotheses so that at least one hypothesis is within $\tilde{O}_d(\alpha^{-1/2d})$ of μ.

Note: in quasi-polynomial time/samples can achieve polylog error. We think we can improve to $O(\sqrt{\log(1/\alpha)})$.

SQ Lower Bounds

In fact, this list decoding result is qualitatively tight for SQ algorithms (though note that our algorithm is not *quite* SQ).

Theorem

Any SQ list decoding algorithm that with 2/3 probability returns a list of hypotheses at least one of which is closer than $\alpha^{-1/d}$ from the mean must do one of the following:

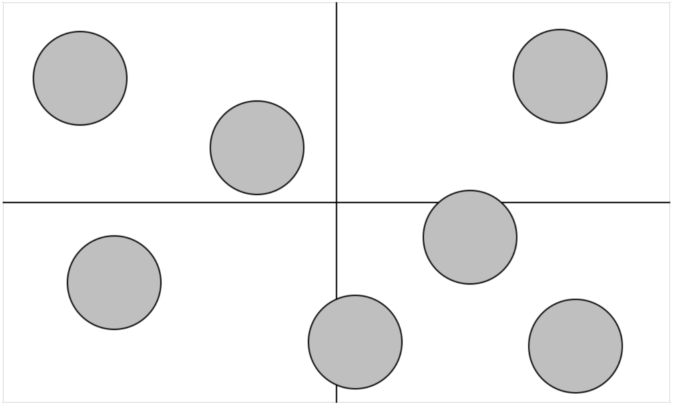
- *Return exponentially many hypotheses.*
- *Perform exponentially many queries.*
- *Perform queries with accuracy $n^{-\Omega(d)}$.*

Learning Mixtures of Spherical Gaussians

Application: Let $X = 1/k \sum_{i=1}^{k} G_i$ with each $G_i \sim N(\mu_i, I)$.

Learning Mixtures of Spherical Gaussians

Application: Let $X = 1/k \sum_{i=1}^{k} G_i$ with each $G_i \sim N(\mu_i, I)$.
Want to learn the μ_i.



History

- [Regev-Vijjayraghavan '17] show information-theoretically impossible to learn the means unless have separation $\Omega(\sqrt{\log(k)})$.

History

- [Regev-Vijjayraghavan '17] show information-theoretically impossible to learn the means unless have separation $\Omega(\sqrt{\log(k)})$.
- [Regev-Vijjayraghavan '17] show how to improve a rough approximation to μ_i to a precise one.

History

- [Regev-Vijjayraghavan '17] show information-theoretically impossible to learn the means unless have separation $\Omega(\sqrt{\log(k)})$.
- [Regev-Vijjayraghavan '17] show how to improve a rough approximation to μ_i to a precise one.
- [Vempala-Wang '02] Give algorithm with separation $\Omega(k^{1/4})$.

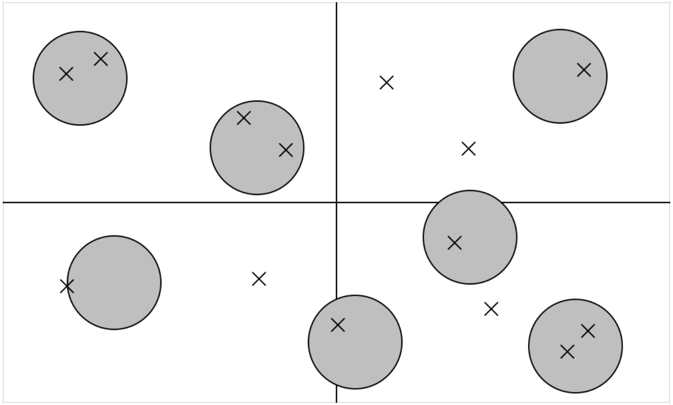
History

- [Regev-Vijjayraghavan '17] show information-theoretically impossible to learn the means unless have separation $\Omega(\sqrt{\log(k)})$.
- [Regev-Vijjayraghavan '17] show how to improve a rough approximation to μ_i to a precise one.
- [Vempala-Wang '02] Give algorithm with separation $\Omega(k^{1/4})$.

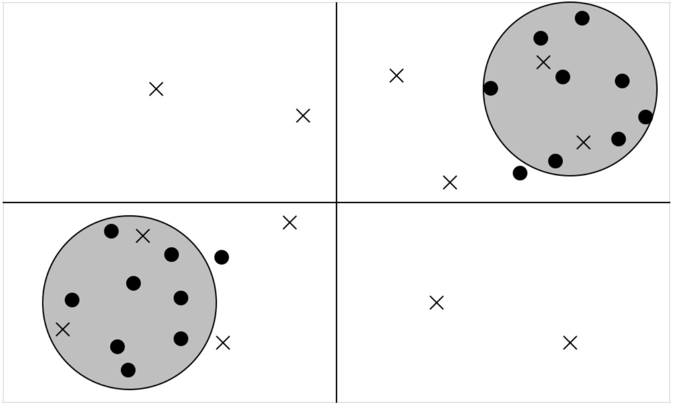
Question: How much separation is actually needed?

List Decoding

Run list decoding algorithm. Since X is a noisy version of *each* G_i, our list contains approximations to all means with error D.

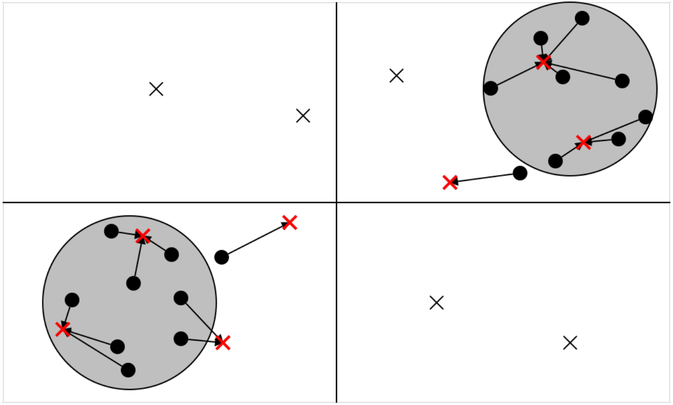


Clustering



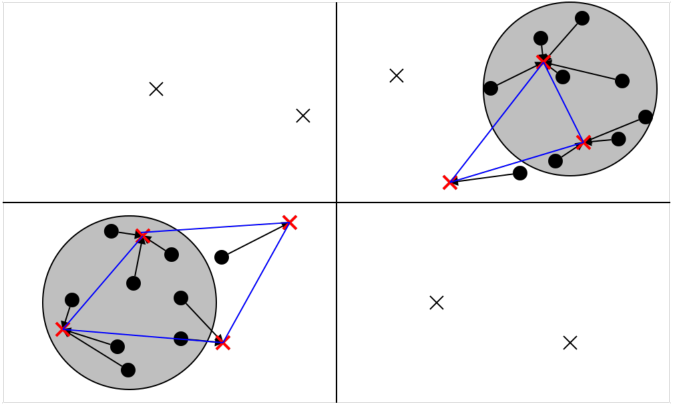
Clustering

Round samples to nearest hypothesis. With high probability samples round to one of hypotheses within $O(D)$ of the mean.



Clustering

Round samples to nearest hypothesis. With high probability samples round to one of hypotheses within $O(D)$ of the mean.
Cluster used hypotheses.

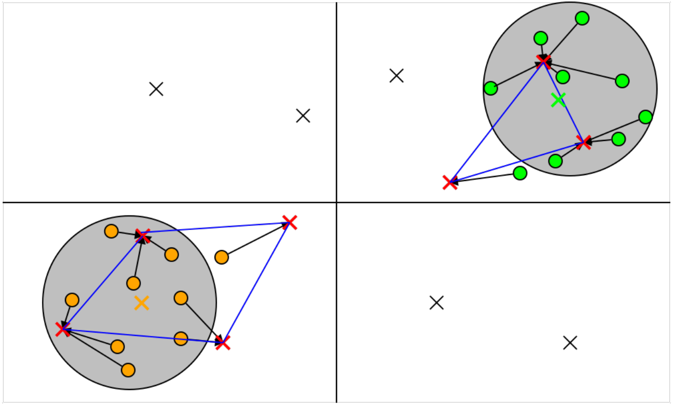


Clustering

Round samples to nearest hypothesis. With high probability samples round to one of hypotheses within $O(D)$ of the mean.

Cluster used hypotheses.

Recover original Gaussians to estimate means.



Results

Theorem

If the means have separation $\Omega(k^{1/2d})$, there is an algorithm that takes $poly(n, (dk)^d)$ samples, runs in sample polynomial time and returns accurate approximations to the μ_i.

Results

Theorem

If the means have separation $\Omega(k^{1/2d})$, there is an algorithm that takes $poly(n, (dk)^d)$ samples, runs in sample polynomial time and returns accurate approximations to the μ_i.

Can be improved to polylogarithmic separation in quasi-polynomial time/samples. We think we can improve this to $O(\sqrt{\log(k)})$ separation.

Results

Theorem

If the means have separation $\Omega(k^{1/2d})$, there is an algorithm that takes $poly(n,(dk)^d)$ samples, runs in sample polynomial time and returns accurate approximations to the μ_i.

Can be improved to polylogarithmic separation in quasi-polynomial time/samples. We think we can improve this to $O(\sqrt{\log(k)})$ separation. Can be generalized to unequal mixtures or to Gaussians with different radii (though still spherical).

Conclusion

Have a robust list decoding algorithm with much better error.
Can use to learn mixtures of spherical Gaussians with k^{δ} separation.

Conclusion

Have a robust list decoding algorithm with much better error.
Can use to learn mixtures of spherical Gaussians with k^δ separation.
Open problems:

1. How much can the Gaussian assumption be relaxed?
2. Can you do better for learning mixtures than for list decoding?
3. Are there better algorithms for density estimation?

- Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, Alistair Stewart, *Robust Estimators in High Dimensions, without the Computational Intractability*, Foundations Of Computer Science, (FOCS) 2016.
- Ilias Diakonikolas, Gautam Kamath, Daniel M. Kane, Jerry Li, Ankur Moitra, Alistair Stewart *Robustly Learning a Gaussian: Getting Optimal Error Efficiently*, Symposium On Discrete Algorithms (SODA) 2018.
- Jacob Steinhardt, Moses Charikar, Gregory Valiant *Learning from Untrusted Data* STOC, 2017.
- O. Regev, A. Vijjayraghavan *On learning mixtures of well-separated gaussians* Proceedings of FOCS, 2017.
- J.W. Tukey, *Mathematics and picturing of data* Proceedings of ICM, volume 6, pp. 523-531, 1975.

S. Vempala, G. Wang *A spectral algorithm for learning mixtures of distributions*, Proceedings of the 43rd Annual Symposium on Foundations of Computer Science, pp. 113–122, 2002.