Robustness and Space

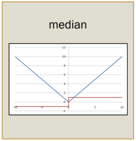
Steve Chien, Katrina Ligett, Andrew McGregor

MSR-SVC, Cornell University, UMass-Amherst

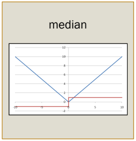
January 2010

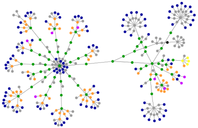
Estimation on large stream of i.i.d. samples

Estimation on large stream of i.i.d. samples

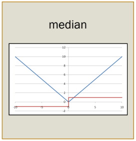


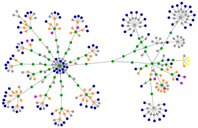
Estimation on large stream of i.i.d. samples





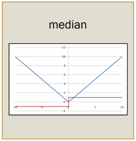
Estimation on large stream of i.i.d. samples





How much space do you need in order to learn properties of the underlying distribution?

When can we learn statistics in small space?



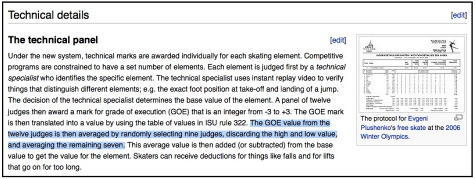
- Would help if statistics are unlikely to skew from a few outliers
- A natural and well-studied problem in statistics

Robust statistics

Are usually more resilient to outliers and errors.

Robust statistics

Are usually more resilient to outliers and errors.



Technical details [edit]

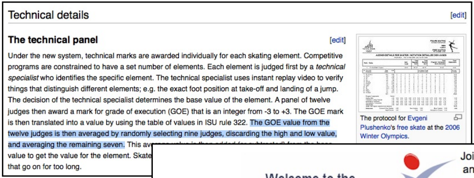
The technical panel [edit]

Under the new system, technical marks are awarded individually for each skating element. Competitive programs are constrained to have a set number of elements. Each element is judged first by a *technical specialist* who identifies the specific element. The technical specialist uses instant replay video to verify things that distinguish different elements; e.g. the exact foot position at take-off and landing of a jump. The decision of the technical specialist determines the base value of the element. A panel of twelve judges then award a mark for grade of execution (GOE) that is an integer from -3 to +3. The GOE mark is then translated into a value by using the table of values in ISU rule 322. The GOE value from the twelve judges is then averaged by randomly selecting nine judges, discarding the high and low value, and averaging the remaining seven. This average value is then added (or subtracted) from the base value to get the value for the element. Skaters can receive deductions for things like falls and for lifts that go on for too long.

The protocol for Evgeni Plushenko's free skate at the 2006 Winter Olympics.

Robust statistics

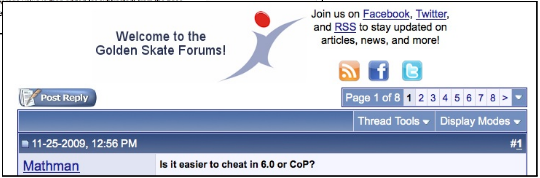
Are usually more resilient to outliers and errors.


Technical details [edit]

The technical panel [edit]

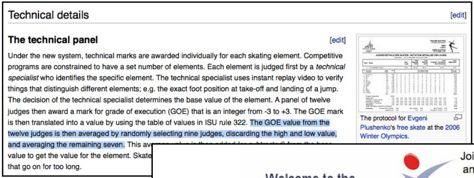
Under the new system, technical marks are awarded individually for each skating element. Competitive programs are constrained to have a set number of elements. Each element is judged first by a *technical specialist* who identifies the specific element. The technical specialist uses instant replay video to verify things that distinguish different elements; e.g. the exact foot position at take-off and landing of a jump. The decision of the technical specialist determines the base value of the element. A panel of twelve judges then award a mark for grade of execution (GOE) that is an integer from -3 to +3. The GOE mark is then translated into a value by using the table of values in ISU rule 322. The GOE value from the twelve judges is then averaged by randomly selecting nine judges, discarding the high and low value, and averaging the remaining seven. This av
value to get the value for the element. Skate
that go on for too long.

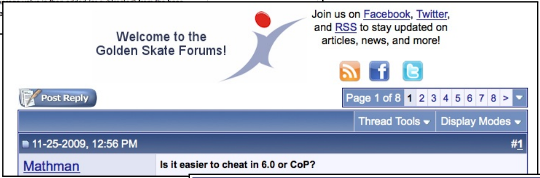
The protocol for Evgeni Plushenko's free skate at the 2006 Winter Olympics.

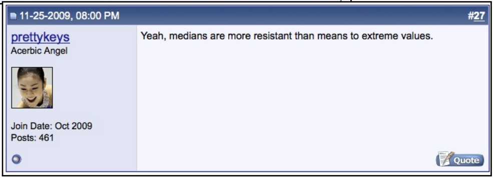


Robust statistics

Are usually more resilient to outliers and errors.







Robust statistics

Established field within theoretical statistics

- The study of when statistics/estimators (median, standard deviation, etc.) are resilient to noise and small perturbations

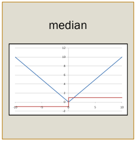
Why are robust estimators useful?

- Resilient way of analyzing data
- Non-robust answers arguably less meaningful

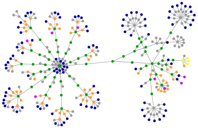
What are the computational properties of robust statistics?

Robust statistics: Our contribution

- Understanding space complexity of approximating robust estimator T.
  - Samples drawn independently from unknown distribution F over the real line
  - Estimator T promised to be robust at F
- Generally, can approximate T(F) in a very small amount of space.



When can we learn properties in small space?



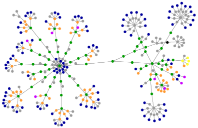
- Would help if the property were robust to changes in the underlying distribution.
- Do we need more samples when we use less space?

Property testing

- Established field within theoretical CS
  - The study of when a distribution satisfies a certain property or is “far” from all distributions that satisfy that property
- “Weak Continuity”, an analogue of robustness, requires that nearby distributions are also close under the property.
- What are the space-related issues in property testing?

Property testing: Our contribution

- Understanding space-sample tradeoff for testing weakly continuous properties.
  - Properties defined on discrete distributions over $[n]$
  - Property promised to be weakly-continuous
- There is a general, direct tradeoff between space complexity and sample complexity and a corresponding space-limited property testing algorithm.



This talk

1. Robust statistics
   - Preliminaries
   - Introductory results
   - Location M-estimators
   - L-estimators
2. Property Testing

This talk

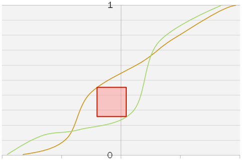
1. Robust statistics
   - Preliminaries
   - Introductory results
   - Location M-estimators
   - L-estimators
2. Property Testing

Preliminaries

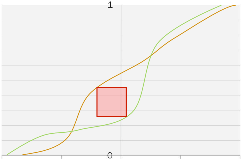
- probability distributions
  - F is cumulative distribution function
  - f is probability density function
  - distance measure is the Lévy distance

Preliminaries

- probability distributions
  - F is cumulative distribution function
  - f is probability density function
  - distance measure is the Lévy distance

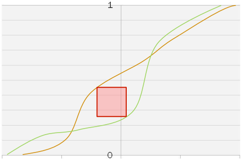


Preliminaries



- probability distributions
  - F is cumulative distribution function
  - f is probability density function
  - distance measure is the Lévy distance
- estimators
  - Have the form $T(F) : D_{\mathbb{R}} \to \mathbb{R}$
  - Mean: $T(F) = \int x \, dF(X)$ or $\frac{1}{m} \sum_i x_i$

Preliminaries



- probability distributions
  - F is cumulative distribution function
  - f is probability density function
  - distance measure is the Lévy distance
- estimators
  - Have the form $T(F) : D_{\mathbb{R}} \to \mathbb{R}$
  - Mean: $T(F) = \int x \, dF(X)$ or $\frac{1}{m} \sum_i x_i$
- approximation
  - additive: an ϵ-approx of $T(F)$ is a value in $[T(F) - \epsilon, T(F) + \epsilon]$

What is robustness?

- Intuitively, a small change to distribution cannot change estimator much
- Defined for a $(T = \text{ estimator}, F = \text{ distribution})$ pair, not the estimator alone
- Key concept: the influence function

$$\mathrm{IF}(x; T, F) = \lim_{t \to 0} \frac{T((1-t)F + t\Delta_x) - T(F)}{t}$$

Robustness: definition and application

Definition

An estimator T is (σ, τ)-robust at F if for all distributions G s.t. $d(F, G) \leq \sigma$,

$$T(F) - T(G) \leq \tau\, d(F, G).$$

Desired result

Let T be an estimator of class C. Then if T is (σ, τ)-robust at F, there is an algorithm that produces an ϵ-approx of $T(F)$ with probab at least $1 - \delta$ and using small space.

This talk

1. Robust statistics
   - Preliminaries
   - Introductory results
   - Location M-estimators
   - L-estimators
2. Property Testing

An easy general application of robustness

Simple algorithm for general robust statistics

Take m samples; output answer from those.

Space: $m +$ space needed to compute $T(F)$

An easy general application of robustness

Simple algorithm for general robust statistics

Take m samples; output answer from those.
Space: $m +$ space needed to compute $T(F)$

How reliable is the statistic on a subsample?

An easy general application of robustness

Simple algorithm for general robust statistics

Take m samples; output answer from those.

Space: $m +$ space needed to compute $T(F)$

How reliable is the statistic on a subsample?

Dvoretzky-Kiefer-Wolfowitz inequality

Let $x_1, \ldots, x_m$ be m samples drawn independently with respect to F, and let $F_m = \frac{1}{m} \sum_{i=1}^{m} \Delta_{x_i}$. Then

$$\Pr[\sup_x |F_m(x) - F(x)| > \epsilon] \leq \exp(-2m\epsilon^2).$$

An easy general application of robustness

Simple algorithm for general robust statistics

Take m samples; output answer from those.
Space: $m +$ space needed to compute $T(F)$

How reliable is the statistic on a subsample?

Theorem

If any estimator T is (σ, τ)-robust at F, there is an algorithm that produces an ϵ-approximation of $T(F)$ with probability at least $1 - \delta$ using poly$(\frac{1}{\epsilon^2}, \ln \frac{1}{\delta})$ space.

An easy general application of robustness

Simple algorithm for general robust statistics

Take m samples; output answer from those.
Space: $m +$ space needed to compute $T(F)$

How reliable is the statistic on a subsample?

Theorem

If any estimator T is (σ, τ)-robust at F, there is an algorithm that produces an ϵ-approximation of $T(F)$ with probability at least $1 - \delta$ using poly$(\frac{1}{\epsilon^2}, \ln \frac{1}{\delta})$ space.

Can we use less space?

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

A building block [Guha-McGregor]

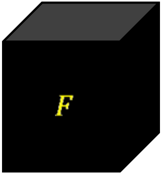
Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample

range $(-\infty, \infty)$

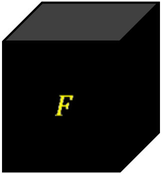
current $u =$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample 44.734

range $(-\infty, \infty)$

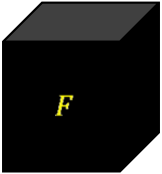
current $u =$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample 44.734

range $(-\infty, \infty)$

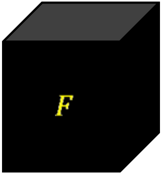
current $u = 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0,1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample 68.336

range $(-\infty, \infty)$

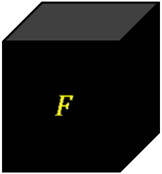
current $u = 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0,1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample 12.950

range $(-\infty, \infty)$

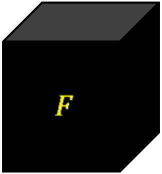
current $u = 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample . . .

range $(-\infty, \infty)$

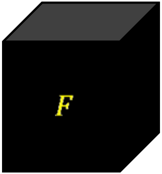
current $u = 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample ...

range $(-\infty, 44.734)$

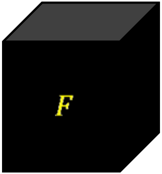
current $u = 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample 50.182

range $(-\infty, 44.734)$

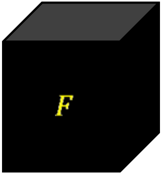
current $u =$ ...

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample 19.233

range $(-\infty, 44.734)$

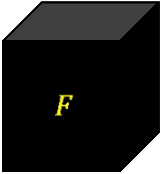
current $u =\ 19.233$

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



sample ...

range $(\ldots, \ldots)$

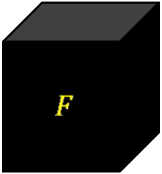
current $u =$ ...

1. Sample repeatedly to get $u \in (a, b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0,1]$, we want to find an element whose rank is within ϵ of t.

$\epsilon = 0.05$; target $t = 0.6$



range $(37.384, 37.401)$

sample $\dots$

current $u =$ $\dots$

1. Sample repeatedly to get $u \in (a,b)$
2. Estimate rank of u with sufficient samples and update range
3. Repeat until u has rank close enough to t

A building block [Guha-McGregor]

Given a distribution F and input t in $[0, 1]$, we want to find an element whose rank is within ϵ of t.

Theorem

Given a distribution F and a value t, the Guha-McGregor algorithm returns a value whose rank is within ϵ of t with probability at least $1 - \delta$, using space at most poly$(\log 1/\epsilon \log\log 1/\delta)$.

Estimator classes we handle

- M-estimators: generalized **m**aximum likelihood estimators
- L-estimators: **l**inear combination of order statistics
- R-estimators: based on **r**ank statistics

This talk

1. Robust statistics
   - Preliminaries
   - Introductory results
   - Location M-estimators
   - L-estimators

2. Property Testing

Location M-estimators

- ρ-type: Given a function $\rho : \mathbb{R} \to \mathbb{R}$ and a distribution F, we can define

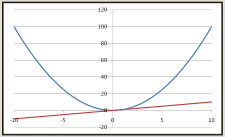
$$T(F) = \text{argmin}_\theta \int \rho(x - \theta)\ dF(x)$$

- ψ-type: Given a function $\psi : \mathbb{R} \to \mathbb{R}$ and a distribution F, we can define $T(F) = \theta$ where

$$\int \psi(x - \theta)\ dF(x) = 0$$

Examples of M-estimators

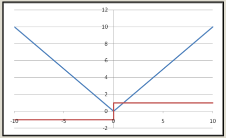
mean



$$\rho(x-\theta) = (x-\theta)^2$$
$$\psi(x-\theta) = (x-\theta)$$

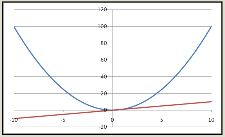
Examples of M-estimators

median



$$\rho(x-\theta) = |x-\theta|$$
$$\psi(x-\theta) = \mathrm{sign}(x-\theta)$$

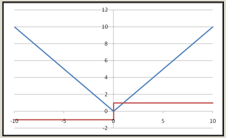
mean



$$\rho(x-\theta) = (x-\theta)^2$$
$$\psi(x-\theta) = (x-\theta)$$

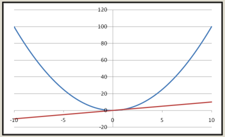
Examples of M-estimators

median



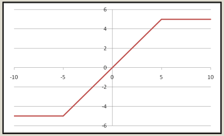
$$\rho(x-\theta) = |x-\theta|$$
$$\psi(x-\theta) = \text{sign}(x-\theta)$$

mean



$$\rho(x-\theta) = (x-\theta)^2$$
$$\psi(x-\theta) = (x-\theta)$$

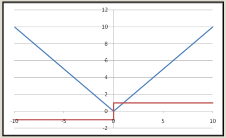
Huber estimator



$$\psi(x-\theta) = \begin{cases} -b \text{ if } x-\theta < -b \\ x \text{ if } -b \le x-\theta \le b \\ b \text{ if } x-\theta > b \end{cases}$$

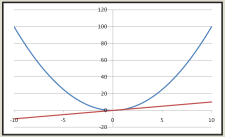
Examples of M-estimators

median



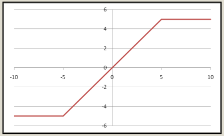
$\rho(x-\theta) = |x-\theta|$
$\psi(x-\theta) = \text{sign}(x-\theta)$

mean



$\rho(x-\theta) = (x-\theta)^2$
$\psi(x-\theta) = (x-\theta)$

Huber estimator



$$\psi(x-\theta) = \begin{cases} -b \text{ if } x-\theta < -b \\ x \text{ if } -b \leq x-\theta \leq b \\ b \text{ if } x-\theta > b \end{cases}$$

Useful distinction: Redescending and Non-Redescending

- Redescending estimators have finite rejection point
- value r s.t. $\psi(y) = 0$ when $|y| > r$

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x - u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

Space-efficient algorithm for M-estimators

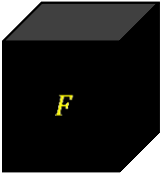
Define $\Psi(u) = \int \psi(x - u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m} \sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x - u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\boxed{\epsilon = 0.05}$



sample

range $(-\infty, \infty)$

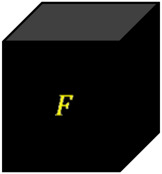
current $u =$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m} \sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\epsilon = 0.05$



sample 44.734

range $(-\infty, \infty)$

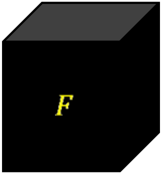
current $u =$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\epsilon = 0.05$



sample 44.734

range $(-\infty, \infty)$

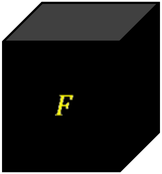
current $u = \ 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\epsilon = 0.05$



sample 68.336

range $(-\infty, \infty)$

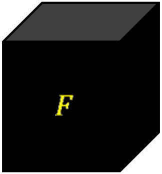
current $u =$ 44.734

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\epsilon = 0.05$



sample 12.950

range $(-\infty, \infty)$

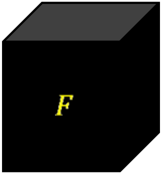
current $u =$ 44.734

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\, dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\epsilon = 0.05$



sample ...

range $(-\infty, \infty)$

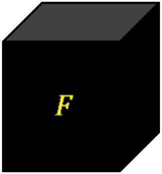
current $u = 44.734$

1. Sample repeatedly to get $u \in (a,b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\boxed{\epsilon = 0.05}$



sample ...

range $(-\infty, 44.734)$

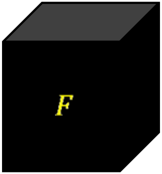
current $u =\ 44.734$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\, dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\epsilon = 0.05$



sample 50.182

range $(-\infty, 44.734)$

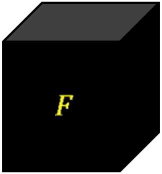
current $u = \ \ldots$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\boxed{\epsilon = 0.05}$



sample 19.233

range $(-\infty, 44.734)$

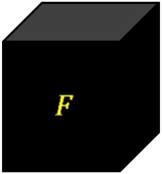
current $u =$ 19.233

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\, dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\boxed{\epsilon = 0.05}$



sample $\ldots$

range $(\ldots, \ldots)$

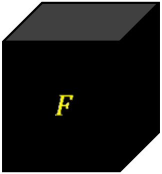
current $u =$ $\ldots$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Space-efficient algorithm for M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

$\boxed{\epsilon = 0.05}$



sample ...

range $(37.384, 37.401)$

current $u = \ \ldots$

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m}\sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

Result for Location M-estimators

Define $\Psi(u) = \int \psi(x-u)\ dF(x)$. Given a distribution F, we want to find an element s.t. $\Psi(u) = 0$.

Theorem

If a Location M-estimator T is (σ, τ)-robust at F, there is an algorithm that produces an ϵ-approximation of $T(F)$ with probability at least $1-\delta$ using poly$(\log \frac{\tau}{\epsilon}, \log\log \frac{1}{\delta})$ space.

Robustness and M-estimators

1. Sample repeatedly to get $u \in (a, b)$
2. Sample to estimate $\hat{\Psi}(u) = \frac{1}{m} \sum_i \psi(x_i - u)$ and update range
3. Repeat until $\hat{\Psi}(u)$ is close enough to 0
   If sampling fails before this happens, need small cleanup phase.

What did robustness give us?

- Guarantee that when $\Psi(u)$ is close to 0, u is close to right answer
- Guarantee that $\hat{\Psi}(u)$ is actually close to $\Psi(u)$
- Guarantee that cleanup phase terminates quickly

Redescending M-estimators

- The problem: we're trying to find θ the global min of $R(u) = \int \rho(x,u)dF(x)$—how can we tell the local minima apart?

Redescending M-estimators

- The problem: we're trying to find θ the global min of $R(u) = \int \rho(x, u) dF(x)$—how can we tell the local minima apart?
- How can robustness help?
  - For any point u sufficiently far from θ, there is a Δ gap between $R(u)$ and $R(\theta)$.
  - We pick $\xi_1 < \xi_2 < \dots$ an increasing sequence of reals with ρ values that differ by at most $\Delta/4$, so for any pair of points $a < b$ s.t. $|\rho(b) - \rho(a)| > \Delta/4$, there must exist $\xi_j \in [a, b]$.
  - Bounds the average derivative of $R(\cdot)$ around θ so that a random sample x from the distribution has reasonably high probability that $R(x + \xi_j)$ is close to $R(\theta)$ for some ξ_j.

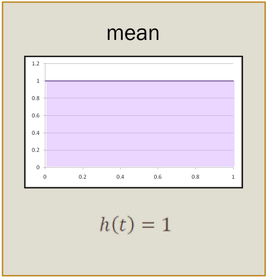
This talk

1. Robust statistics
   - Preliminaries
   - Introductory results
   - Location M-estimators
   - L-estimators
2. Property Testing

L-estimator definition

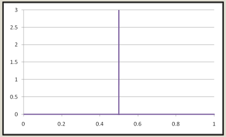
- Given a function $h : [0,1] \to \mathbb{R}_{\geq 0}$ s.t. $\int_0^1 h(t)dt = 1$, we can define
$$T(F) = \int_0^1 F^{-1}(t)h(t)dt$$
- That is, an L-estimator is a weighted average of the distribution, with weights based on rank

L-estimator examples



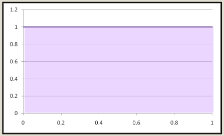
L-estimator examples

median



$$h(t) = \delta_{1/2}$$

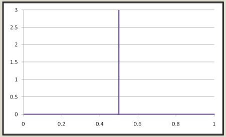
mean



$$h(t) = 1$$

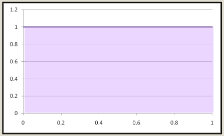
L-estimator examples

median



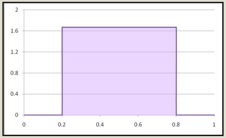
$$h(t) = \delta_{1/2}$$

mean



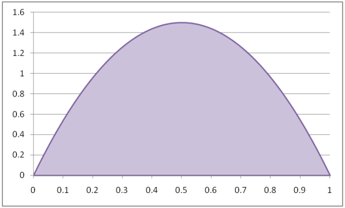
$$h(t) = 1$$

α-trimmed mean



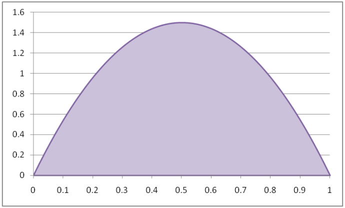
$$h(t) = \begin{cases} 0 \text{ if } x < \alpha \text{ or } x > 1 - \alpha \\ \frac{1}{1-2\alpha} \text{ otherwise} \end{cases}$$

L-estimator algorithm sketch

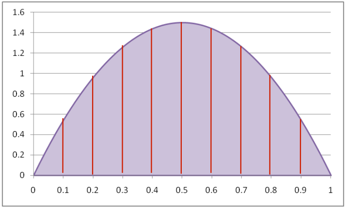


Weighting function $h(t)$.

L-estimator algorithm sketch

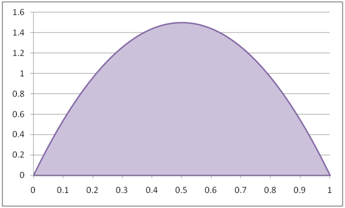


Weighting function $h(t)$.

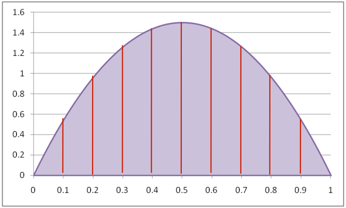


Slice into intervals.

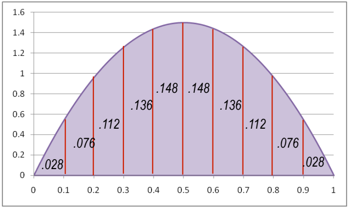
L-estimator algorithm sketch



Weighting function $h(t)$.

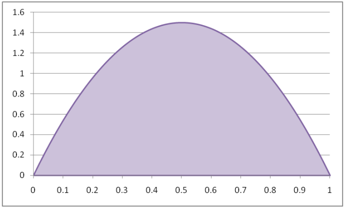


Slice into intervals.

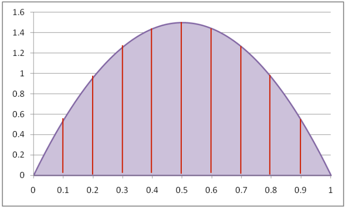


Compute area of each slice.

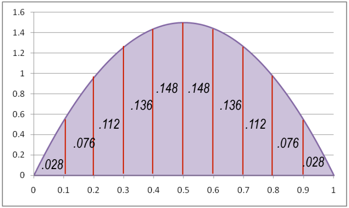
L-estimator algorithm sketch



Weighting function $h(t)$.



Slice into intervals.



Compute area of each slice.

Estimate F^{-1} of each slice midpoint [Guha-McGregor], and keep running total.

Robustness and L-estimators

Theorem

If an L-estimator T is (σ, τ)-robust at F, there is an algorithm that produces an ϵ-approximation of $T(F)$ with probability at least $1 - \delta$ using poly$(\log \frac{\tau}{\epsilon}, \log\log \frac{1}{\delta})$ *space.*

What did robustness give us?

- Guarantee that discrete approximation of weighting function is sufficient
- Guarantee that error introduced by Guha-McGregor subroutine can be contained

This talk

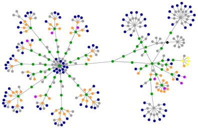
1. Robust statistics
   - Preliminaries
   - Introductory results
   - Location M-estimators
   - L-estimators
2. Property Testing

Preliminaries

- probability distributions
  - Discrete distributions over $[n]$
  - Distance metric is variation distance

$$L_1(p,q) = \sum_{i \in [n]} |p(i) - q(i)|$$

- properties
  - real-valued function π on pdf
  - want to distinguish $\pi(p) < a$ from $\pi(p) > b$



Canonical testing: definitions

Definition

A property π is (ϵ, δ)-*weakly-continuous* if for all distributions p^+, p^- satisfying $|p^+ - p^-| \leq \delta$ we have $|\pi(p^+) - \pi(p^-)| \leq \epsilon$.

Definition

We say π is *symmetric* if

$$\pi(p(1), \ldots, p(n)) = \pi(p(\sigma(1)), \ldots, p(\sigma(n)))$$

for any permutation σ on $[n]$.

Canonical testing [Val08]

Canonical Tester

1. Draw k samples.
2. Consider all distributions that exactly match the fraction of observed heavy elements and have relatively low weight on any observed light element.
3. If all such distributions satisfy $\pi > b$ output "yes", otherwise output "no".

Theorem (Val08)

If $f(n, a, b, \epsilon)$ is the sample complexity to distinguish between $\pi > b - \epsilon$ and $\pi < a + \epsilon$, then the canonical algorithm can distinguish $\pi > b + \epsilon$ and $\pi < a - \epsilon$ using $O(f(n, a, b, \epsilon) 16^{\sqrt{\log n}} / \delta)$ samples.

Canonical testing theorem reframed

- Well-suited to the data-stream model as the problem reduces to finding "heavy-hitters"
- Recall the Misra-Gries heavy hitters algorithm:
  - returns all elements whose frequency exceeds $\theta/2$
  - returns none with frequency below $\theta/4$
  - uses $O(k \log k/\theta)$ bits of space and k samples

Space-Efficient Property Testing

1. Find heavy hitters with Misra-Gries
2. Calculate their empirical frequencies using a fresh sample
3. Plug these values into the Canonical Testing algorithm

Space-sample trade-off

Theorem (Trade-off Theorem for (ϵ, δ^*) weakly continuous π)

Let S be the sample complexity of distinguishing $\pi > b - \epsilon$ from $\pi < a + \epsilon$. Then, for any $\delta < \delta^*$ there exists a stream algorithm that distinguishes $\pi > b + \epsilon$ from $\pi < a - \epsilon$ using $O(S 16^{\sqrt{\log n}} / \delta)$ samples and $O(S 16^{\sqrt{\log n}} \delta / \log n)$ space.

Space-sample trade-off

Theorem (Trade-off Theorem for (ϵ, δ^*) weakly continuous π)

Let S be the sample complexity of distinguishing $\pi > b - \epsilon$ from $\pi < a + \epsilon$. Then, for any $\delta < \delta^*$ there exists a stream algorithm that distinguishes $\pi > b + \epsilon$ from $\pi < a - \epsilon$ using $O(S16^{\sqrt{\log n}}/\delta)$ samples and $O(S16^{\sqrt{\log n}}\delta/\log n)$ space.

The result doesn't appear to be optimal:

Theorem

Let π be $(\epsilon/2, \delta)$-weakly-continuous and suppose there exists a $s(\epsilon)$-space algorithm that returns an additive $\epsilon/2$ approximation to π evaluated on a distribution defined empirically by the stream. Then there exists a stream algorithm using $O(\delta^{-2} n \log(n))$ samples and $s(\epsilon)$ space that is an ϵ additive approx for π.

Space-efficient computation of distribution properties and statistics

- Statistics: robustness allows us to use less space
- Property testing: robustness lets us trade off samples and space

Thank you!