COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Andrew McGregor

Lecture 6

Stream Processing: Have a massive dataset X with n items $x_1, x_2, \ldots, x_n$ that arrive in a continuous stream. Not nearly enough space to store all the items (in a single location).

- Still want to analyze and learn from this data.

Stream Processing: Have a massive dataset X with n items $x_1, x_2, \ldots, x_n$ that arrive in a continuous stream. Not nearly enough space to store all the items (in a single location).

- Still want to analyze and learn from this data.
- Typically must compress the data on the fly, storing a data structure from which you can still learn useful information.

Stream Processing: Have a massive dataset X with n items $x_1, x_2, \ldots, x_n$ that arrive in a continuous stream. Not nearly enough space to store all the items (in a single location).

- Still want to analyze and learn from this data.
- Typically must compress the data on the fly, storing a data structure from which you can still learn useful information.
- Often the compression is randomized. E.g., bloom filters.

Stream Processing: Have a massive dataset X with n items $x_1, x_2, \ldots, x_n$ that arrive in a continuous stream. Not nearly enough space to store all the items (in a single location).

- Still want to analyze and learn from this data.
- Typically must compress the data on the fly, storing a data structure from which you can still learn useful information.
- Often the compression is randomized. E.g., bloom filters.
- Compared to traditional algorithm design, which focuses on minimizing runtime, the big question here is how much space is needed to answer queries of interest.

SOME EXAMPLES

- **Sensor data:** images from telescopes (15 terabytes per night from the Large Synoptic Survey Telescope), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors (Smart Cities), electrical grid monitoring.

SOME EXAMPLES

- **Sensor data:** images from telescopes (15 terabytes per night from the Large Synoptic Survey Telescope), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors (Smart Cities), electrical grid monitoring.

SOME EXAMPLES

- **Sensor data:** images from telescopes (15 terabytes per night from the Large Synoptic Survey Telescope), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors (Smart Cities), electrical grid monitoring.
- **Internet Traffic**: 500 million Tweets per day, 5.6 billion Google searches, billions of ad-clicks and other logs from instrumented webpages, IPs routed by network switches, ...

SOME EXAMPLES

- **Sensor data:** images from telescopes (15 terabytes per night from the Large Synoptic Survey Telescope), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors (Smart Cities), electrical grid monitoring.
- **Internet Traffic**: 500 million Tweets per day, 5.6 billion Google searches, billions of ad-clicks and other logs from instrumented webpages, IPs routed by network switches, ...
- **Datasets in Machine Learning:** When training e.g. a neural network on a large dataset (ImageNet with 14 million images), the data is typically processed in a stream due to storage limitations.

SOME EXAMPLES

- **Sensor data:** images from telescopes (15 terabytes per night from the Large Synoptic Survey Telescope), readings from seismometer arrays monitoring and predicting earthquake activity, traffic cameras and travel time sensors (Smart Cities), electrical grid monitoring.
- **Internet Traffic**: 500 million Tweets per day, 5.6 billion Google searches, billions of ad-clicks and other logs from instrumented webpages, IPs routed by network switches, ...
- **Datasets in Machine Learning:** When training e.g. a neural network on a large dataset (ImageNet with 14 million images), the data is typically processed in a stream due to storage limitations.

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, output the number of distinct elements in the stream.

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, output the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \rightarrow 4 \text{ distinct elements}$$

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \rightarrow 4 \text{ distinct elements}$$

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \rightarrow 4 \text{ distinct elements}$$

Applications:

- Distinct IP addresses clicking on an ad or visiting a site.
- Number of distinct search engine queries.
- Counting distinct motifs in large DNA sequences.

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements in the stream. E.g.,

$$1, 5, 7, 5, 2, 1 \rightarrow 4 \text{ distinct elements}$$

Applications:

- Distinct IP addresses clicking on an ad or visiting a site.
- Number of distinct search engine queries.
- Counting distinct motifs in large DNA sequences.

Google Sawzall, Facebook Presto, Apache Drill, Twitter Algebird

DISTINCT ELEMENTS IDEAS

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

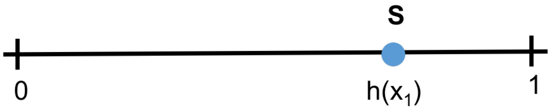
Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$

Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

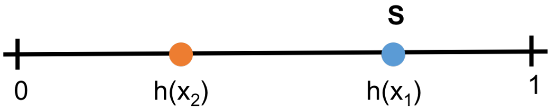
- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

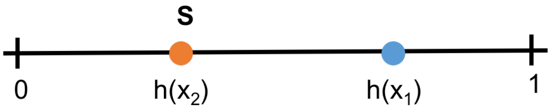
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

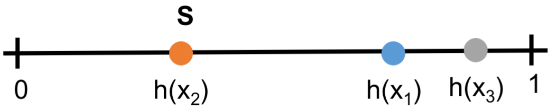
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

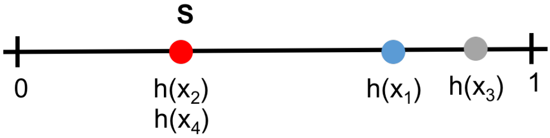
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

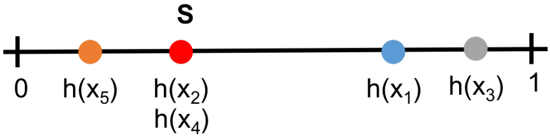
- Let $\mathbf{h} : U \rightarrow [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

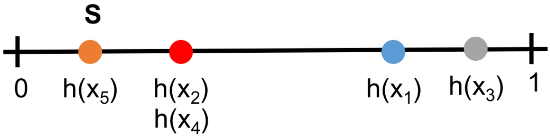
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



Distinct Elements (Count-Distinct) Problem: Given a stream $x_1, \ldots, x_n$, estimate the number of distinct elements.

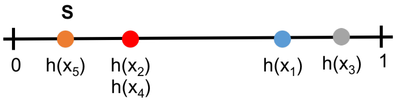
Min-Hashing for Distinct Elements (variant of Flajolet-Martin):

- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\tilde{d} = \frac{1}{s} - 1$



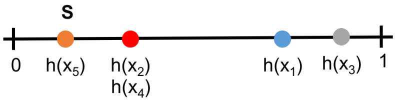
Min-Hashing for Distinct Elements:

- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\hat{d} = \frac{1}{s} - 1$



Min-Hashing for Distinct Elements:

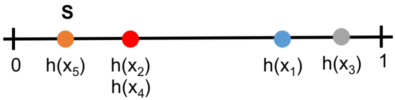
- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
    - $s := \min(s, \mathbf{h}(x_i))$
- Return $\hat{d} = \frac{1}{s} - 1$



- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.

Min-Hashing for Distinct Elements:

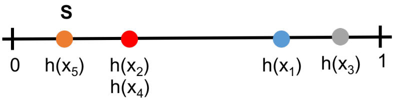
- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\hat{d} = \frac{1}{s} - 1$



- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where $d = \#$ distinct elements.
- Intuition: The larger d is, the smaller we expect s to be.

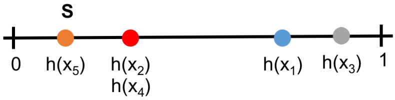
Min-Hashing for Distinct Elements:

- Let $\mathbf{h} : U \to [0,1]$ be a random hash function (with a real valued output)
- $s := 1$
- For $i = 1, \ldots, n$
  - $s := \min(s, \mathbf{h}(x_i))$
- Return $\hat{d} = \frac{1}{s} - 1$



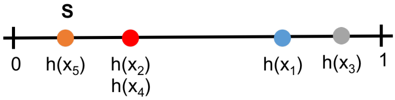
- After all items are processed, s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.
- Intuition: The larger d is, the smaller we expect s to be.
- Same idea as Flajolet-Martin algorithm and HyperLogLog, except they use discrete hash functions.

s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.



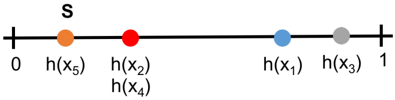
PERFORMANCE IN EXPECTATION

s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.



$$\mathbb{E}[\mathbf{s}] =$$

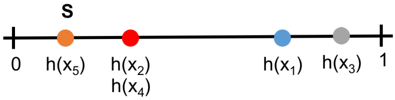
s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.



$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

PERFORMANCE IN EXPECTATION

s is the minimum of d points chosen uniformly at random on $[0, 1]$. Where $d = \#$ distinct elements.

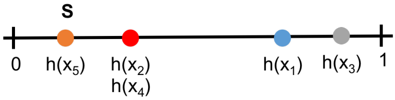


$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

- So estimate of $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ output by the algorithm is correct if **s** exactly equals its expectation.

PERFORMANCE IN EXPECTATION

s is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.

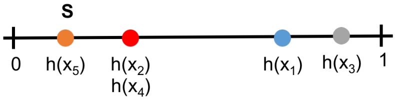


$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx\text{) + calculus)}$$

- So estimate of $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ output by the algorithm is correct if **s** exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$?

PERFORMANCE IN EXPECTATION

$\mathbf{s}$ is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.

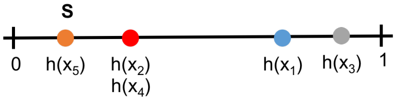


$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx\text{) + calculus)}$$

- So estimate of $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ output by the algorithm is correct if $\mathbf{s}$ exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$? No, but:

PERFORMANCE IN EXPECTATION

$\mathbf{s}$ is the minimum of d points chosen uniformly at random on $[0,1]$. Where $d = \#$ distinct elements.



$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (using } \mathbb{E}(\mathbf{s}) = \int_0^\infty \Pr(\mathbf{s} > x)dx) + \text{calculus)}$$

- So estimate of $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$ output by the algorithm is correct if $\mathbf{s}$ exactly equals its expectation. Does this mean $\mathbb{E}[\widehat{\mathbf{d}}] = d$? No, but:
- **Exercise:** Approximation is robust, i.e., if $|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \leq \epsilon \cdot \mathbb{E}[\mathbf{s}]$ for any $\epsilon \in (0, 1/2)$,

$$(1-4\epsilon)d \leq \widehat{\mathbf{d}} \leq (1+4\epsilon)d$$

.

So question is how well **s** concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \quad .$$

.

s: minimum of d distinct hashes chosen randomly over $[0, 1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

INITIAL CONCENTRATION BOUND

So question is how well **s** concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \text{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

.

$\mathbf{s}$: minimum of d distinct hashes chosen randomly over $[0, 1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

INITIAL CONCENTRATION BOUND

So question is how well **s** concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \text{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

Chebyshev's Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon \mathbb{E}[\mathbf{s}]\right] \leq \frac{\text{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} \qquad .$$

s: minimum of d distinct hashes chosen randomly over $[0, 1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

So question is how well **s** concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \operatorname{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

Chebyshev's Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon \mathbb{E}[\mathbf{s}]\right] \leq \frac{\operatorname{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} \leq \frac{1}{\epsilon^2}.$$

s: minimum of d distinct hashes chosen randomly over $[0, 1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

INITIAL CONCENTRATION BOUND

So question is how well **s** concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \operatorname{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

Chebyshev's Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon \mathbb{E}[\mathbf{s}]\right] \leq \frac{\operatorname{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} \leq \frac{1}{\epsilon^2}.$$

Bound is vacuous for any $\epsilon < 1$.

s: minimum of d distinct hashes chosen randomly over $[0, 1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

INITIAL CONCENTRATION BOUND

So question is how well **s** concentrates around its mean.

$$\mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ and } \operatorname{Var}[\mathbf{s}] \leq \frac{1}{(d+1)^2} \text{ (\textit{also via calculus})}.$$

Chebyshev's Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon \mathbb{E}[\mathbf{s}]\right] \leq \frac{\operatorname{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} \leq \frac{1}{\epsilon^2}.$$

Bound is vacuous for any $\epsilon < 1$. How can we improve accuracy?

> **s**: minimum of d distinct hashes chosen randomly over $[0, 1]$, computed by hashing algorithm. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

IMPROVING PERFORMANCE

Leverage the law of large numbers: improve accuracy via repeated independent trials.

IMPROVING PERFORMANCE

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h} : U \to [0, 1]$ be a random hash function
- $\mathbf{s} := 1$
- For $i = 1, \ldots, n$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_i))$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

IMPROVING PERFORMANCE

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s} := 1$
- For $i = 1, \ldots, n$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_i))$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

IMPROVING PERFORMANCE

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \rightarrow [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - $\mathbf{s} := \min(\mathbf{s}, \mathbf{h}(x_i))$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

IMPROVING PERFORMANCE

Leverage the law of large numbers: improve accuracy via repeated independent trials.

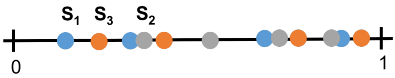
Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$

Leverage the law of large numbers: improve accuracy via repeated independent trials.

Hashing for Distinct Elements (Improved):

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,...,k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$



$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1}$$
$$\text{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of $\#$ distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}]$$
$$\mathrm{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\text{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

ANALYSIS

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathbf{s}]$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of $\#$ distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\text{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \text{Var}[\mathbf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathbf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon \mathbb{E}[\mathbf{s}]\right] \leq \frac{\mathrm{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\mathrm{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \mathrm{Var}[\mathbf{s}] \leq \frac{1}{k\cdot(d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[|\mathbf{s} - \mathbb{E}[\mathbf{s}]| \geq \epsilon\mathbb{E}[\mathbf{s}]\right] \leq \frac{\mathrm{Var}[\mathbf{s}]}{(\epsilon\mathbb{E}[\mathbf{s}])^2} = \frac{\mathbb{E}[\mathbf{s}]^2/k}{\epsilon^2\mathbb{E}[\mathbf{s}]^2} = \frac{1}{k\cdot\epsilon^2}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k}\mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\text{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \text{Var}[\mathbf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[\left|d - \widehat{\mathbf{d}}\right| \geq 4\epsilon \cdot d\right] \leq \frac{\text{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} = \frac{\mathbb{E}[\mathbf{s}]^2/k}{\epsilon^2 \mathbb{E}[\mathbf{s}]^2} = \frac{1}{k \cdot \epsilon^2}$$

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\text{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \text{Var}[\mathbf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[\left|d - \widehat{\mathbf{d}}\right| \geq 4\epsilon \cdot d\right] \leq \frac{\text{Var}[\mathbf{s}]}{(\epsilon\mathbb{E}[\mathbf{s}])^2} = \frac{\mathbb{E}[\mathbf{s}]^2/k}{\epsilon^2\mathbb{E}[\mathbf{s}]^2} = \frac{1}{k \cdot \epsilon^2}$$

How should we set k if we want $4\epsilon \cdot d$ error with probability $\geq 1 - \delta$?

> $\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0,1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\text{Var}[\mathbf{s}_j] \leq \frac{1}{(d+1)^2} \implies \text{Var}[\mathbf{s}] \leq \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

$$\Pr\left[\left|d - \widehat{\mathbf{d}}\right| \geq 4\epsilon \cdot d\right] \leq \frac{\text{Var}[\mathbf{s}]}{(\epsilon \mathbb{E}[\mathbf{s}])^2} = \frac{\mathbb{E}[\mathbf{s}]^2/k}{\epsilon^2 \mathbb{E}[\mathbf{s}]^2} = \frac{1}{k \cdot \epsilon^2}$$

How should we set k if we want $4\epsilon \cdot d$ error with probability $\geq 1 - \delta$? $k = \frac{1}{\epsilon^2 \cdot \delta}$.

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

$\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. Have already shown that for $j = 1, \ldots, k$:

$$\mathbb{E}[\mathbf{s}_j] = \frac{1}{d+1} \implies \mathbb{E}[\mathbf{s}] = \frac{1}{d+1} \text{ (linearity of expectation)}$$

$$\text{Var}[\mathbf{s}_j] \le \frac{1}{(d+1)^2} \implies \text{Var}[\mathbf{s}] \le \frac{1}{k \cdot (d+1)^2} \text{ (linearity of variance)}$$

Chebyshev Inequality:

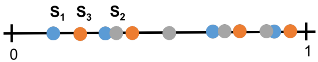
$$\Pr\left[\left|d - \widehat{\mathbf{d}}\right| \ge 4\epsilon \cdot d\right] \le \frac{\text{Var}[\mathbf{s}]}{(\epsilon\mathbb{E}[\mathbf{s}])^2} = \frac{\mathbb{E}[\mathbf{s}]^2/k}{\epsilon^2\mathbb{E}[\mathbf{s}]^2} = \frac{1}{k \cdot \epsilon^2} = \frac{\epsilon^2 \cdot \delta}{\epsilon^2} = \delta.$$

How should we set k if we want $4\epsilon \cdot d$ error with probability $\ge 1 - \delta$? $k = \frac{1}{\epsilon^2 \cdot \delta}$.

$\mathbf{s}_j$: minimum of d distinct hashes chosen randomly over $[0, 1]$. $\mathbf{s} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$. $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$: estimate of # distinct elements d.

Hashing for Distinct Elements:

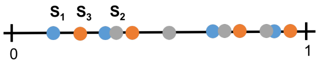
- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,. . . , k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$



- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.

Hashing for Distinct Elements:

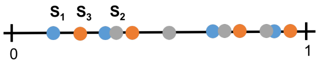
- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0,1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,. . . , k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k} \sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$



- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.
- Space complexity is $k = \frac{1}{\epsilon^2 \cdot \delta}$ real numbers $s_1, \ldots, s_k$.

Hashing for Distinct Elements:

- Let $\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_k : U \to [0, 1]$ be random hash functions
- $\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_k := 1$
- For $i = 1, \ldots, n$
  - For j=1,. . . , k, $\mathbf{s}_j := \min(\mathbf{s}_j, \mathbf{h}_j(x_i))$
- $\mathbf{s} := \frac{1}{k}\sum_{j=1}^{k} \mathbf{s}_j$
- Return $\widehat{\mathbf{d}} = \frac{1}{\mathbf{s}} - 1$



- Setting $k = \frac{1}{\epsilon^2 \cdot \delta}$, algorithm returns $\widehat{\mathbf{d}}$ with $|d - \widehat{\mathbf{d}}| \leq 4\epsilon \cdot d$ with probability at least $1 - \delta$.
- Space complexity is $k = \frac{1}{\epsilon^2 \cdot \delta}$ real numbers $s_1, \ldots, s_k$.
- $\delta = 5\%$ failure rate gives a factor 20 overhead in space complexity.

IMPROVED FAILURE RATE

How can we improve our dependence on the failure rate δ?

IMPROVED FAILURE RATE

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/4$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{4}{\epsilon^2}$ hash functions.

IMPROVED FAILURE RATE

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/4$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{4}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathbf{d}}_{\mathbf{1}}, \ldots, \widehat{\mathbf{d}}_{\mathbf{t}}$ be the outcomes of the t trials, return

$$\widehat{\mathbf{d}} = \textit{median}(\widehat{\mathbf{d}}_{\mathbf{1}}, \ldots, \widehat{\mathbf{d}}_{\mathbf{t}}) \ .$$

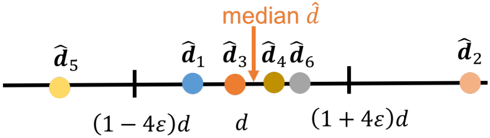
IMPROVED FAILURE RATE

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/4$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{4}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ be the outcomes of the t trials, return

$$\widehat{\mathbf{d}} = \textit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t) \ .$$



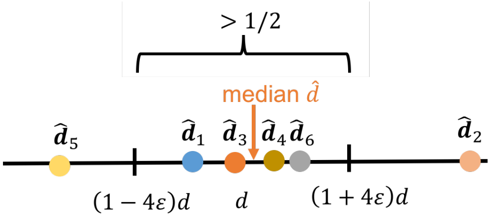
IMPROVED FAILURE RATE

How can we improve our dependence on the failure rate δ?

The median trick: Run $t = O(\log 1/\delta)$ trials each with failure probability $\delta' = 1/4$ – each using $k = \frac{1}{\delta' \epsilon^2} = \frac{4}{\epsilon^2}$ hash functions.

- Letting $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ be the outcomes of the t trials, return

$$\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t) \ .$$



- If $> 1/2$ of trials fall in $[(1-4\epsilon)d, (1+4\epsilon)d]$, then the median will.

THE MEDIAN TRICK

- $\widehat{\mathbf{d_1}}, \ldots, \widehat{\mathbf{d_t}}$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = \mathit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = \mathit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. .

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = \textit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. .

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right)$$

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least $3/4$. Let $\widehat{\mathbf{d}} = \textit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right)$$

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = \textit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq \frac{3}{4} \cdot t$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right)$$

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = \mathit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq \frac{3}{4} \cdot t$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right)$$

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_{\mathbf{1}}, \ldots, \widehat{\mathbf{d}}_{\mathbf{t}}$ are the outcomes of the t trials, each falling in

$$[(1 - 4\epsilon)d, (1 + 4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1 - 4\epsilon)d, (1 + 4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1 - 4\epsilon)d, (1 + 4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq \frac{3}{4} \cdot t$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1 - 4\epsilon)d, (1 + 4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right) \leq \Pr\left(\mathbf{X} \leq \frac{2}{3} \cdot \mathbb{E}[\mathbf{X}]\right)$$

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq \frac{3}{4} \cdot t$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right) \leq \Pr\left(\mathbf{X} \leq \frac{2}{3} \cdot \mathbb{E}[\mathbf{X}]\right)$$

Apply Chernoff bound:

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least 3/4. Let $\widehat{\mathbf{d}} = \mathit{median}(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq \frac{3}{4} \cdot t$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right) \leq \Pr\left(\mathbf{X} \leq \frac{2}{3} \cdot \mathbb{E}[\mathbf{X}]\right)$$

Apply Chernoff bound:

$$\Pr\left(|\mathbf{X} - \mathbb{E}[\mathbf{X}]| \geq \frac{1}{3}\mathbb{E}[\mathbf{X}]\right) \leq 2\exp\left(-\frac{\frac{1}{3}^2 \cdot \frac{3}{4}t}{2+1/3}\right) = e^{-\Theta(t)}.$$

THE MEDIAN TRICK

- $\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t$ are the outcomes of the t trials, each falling in

$$[(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least $3/4$. Let $\widehat{\mathbf{d}} = median(\widehat{\mathbf{d}}_1, \ldots, \widehat{\mathbf{d}}_t)$.

What is the probability that the median $\widehat{\mathbf{d}}$ falls in $[(1-4\epsilon)d, (1+4\epsilon)d]$?

- Let $\mathbf{X}$ be the # of trials falling in $[(1-4\epsilon)d, (1+4\epsilon)d]$. $\mathbb{E}[\mathbf{X}] \geq \frac{3}{4} \cdot t$.

$$\Pr\left(\widehat{\mathbf{d}} \notin [(1-4\epsilon)d, (1+4\epsilon)d]\right) \leq \Pr\left(\mathbf{X} \leq \frac{1}{2} \cdot t\right) \leq \Pr\left(\mathbf{X} \leq \frac{2}{3} \cdot \mathbb{E}[\mathbf{X}]\right)$$

Apply Chernoff bound:

$$\Pr\left(|\mathbf{X} - \mathbb{E}[\mathbf{X}]| \geq \frac{1}{3}\mathbb{E}[\mathbf{X}]\right) \leq 2\exp\left(-\frac{\frac{1}{3}^2 \cdot \frac{3}{4}t}{2+1/3}\right) = e^{-\Theta(t)}.$$

- Setting $t = O(\log(1/\delta))$ gives failure probability $e^{-\log(1/\delta)} = \delta$.

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns

$$\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least $1-\delta$.

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns

$$\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least $1-\delta$.

Total Space Complexity: t trials, each using $k = \frac{1}{\epsilon^2\delta'}$ hash functions, for $\delta' = 1/4$. Space is $\frac{4t}{\epsilon^2} = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ real numbers (the minimum value of each hash function).

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns

$$\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least $1-\delta$.

Total Space Complexity: t trials, each using $k = \frac{1}{\epsilon^2\delta'}$ hash functions, for $\delta' = 1/4$. Space is $\frac{4t}{\epsilon^2} = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ real numbers (the minimum value of each hash function).

No dependence on the number of distinct elements d or the number of items in the stream n! Both can be very large.

Upshot: The median of $t = O(\log(1/\delta))$ independent runs of the hashing algorithm for distinct elements returns

$$\widehat{\mathbf{d}} \in [(1-4\epsilon)d, (1+4\epsilon)d]$$

with probability at least $1-\delta$.

Total Space Complexity: t trials, each using $k = \frac{1}{\epsilon^2\delta'}$ hash functions, for $\delta' = 1/4$. Space is $\frac{4t}{\epsilon^2} = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$ real numbers (the minimum value of each hash function).

No dependence on the number of distinct elements d or the number of items in the stream n! Both can be very large.

A note on the median: The median is often used as a robust alternative to the mean, when there are outliers (e.g., heavy tailed distributions, corrupted data).