CS-GY 9223 D: Lecture 2
Streaming and Sketching Algorithms via Hashing

NYU Tandon School of Engineering, Prof. Christopher Musco

PROBLEM SET

I really appreciate everyone's high activity on Piazza and in office hours!

- Ly Cao set up a student Slack channel. You can join via link on the course website.
- Due this Friday 9/18 at midnight ET. Due to weirdness in NYU Classes, deadline appears to be 11:55pm. Don't worry if you submit after.
- Remember to list any collaborators. And remember to write up all solutions on your own.

NOTE ON MATHEMATICAL PROOFS

It can be hard to know how formal to be. I will try to provide feedback on first problem set for anyone who is either too rigorous or too loose. It's a learning process.

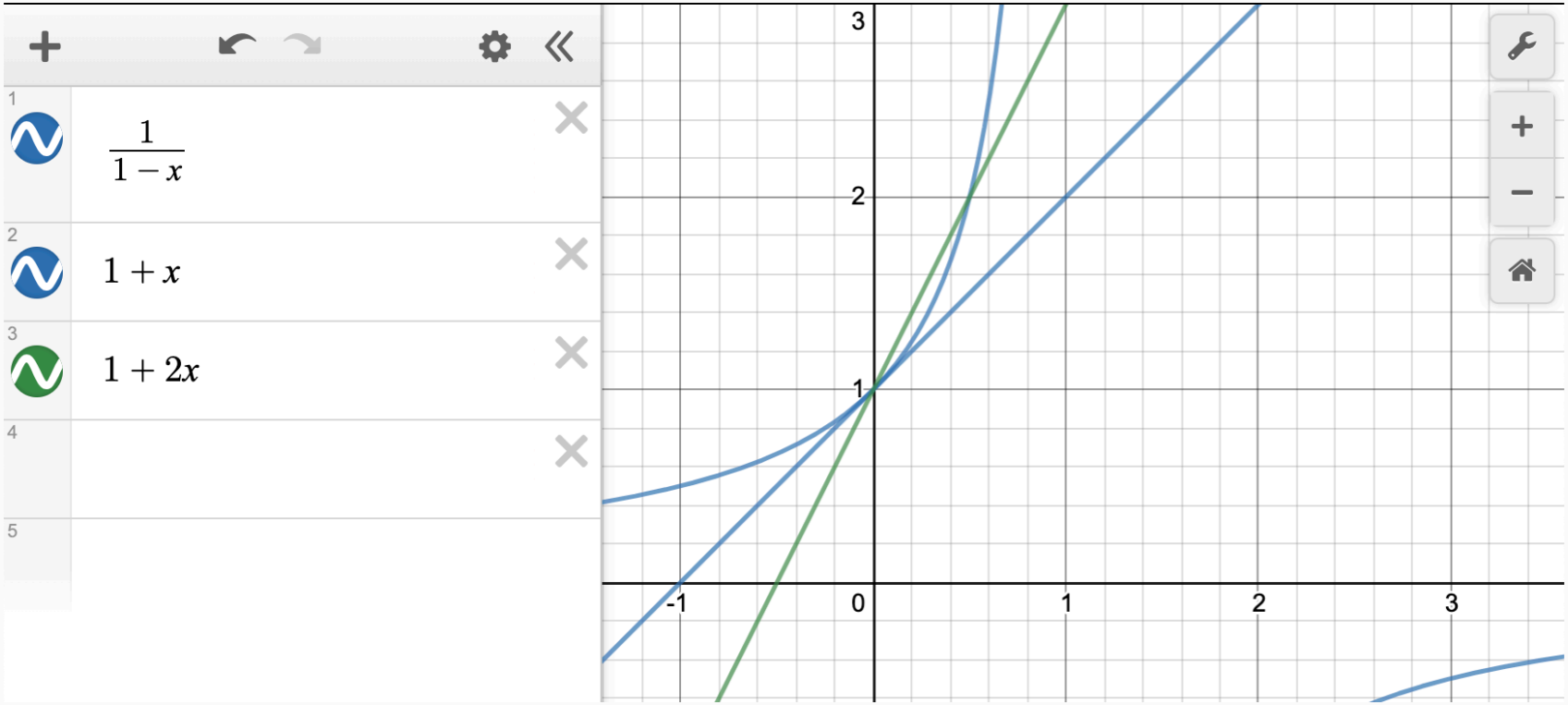
Things that are generally fine:

- Can assume input size n is $> C$ for some constant c. E.g. $n > 2, n > 10$.
- Similarly can assume $\epsilon < c$ for constant c. E.g. $\epsilon < .1$, $\epsilon < .01$.
- If I write $O(z)$, you are free to choose the constant. E.g., it's fine if your method only works for tables of size $1000 \cdot m^{1.5}$.
- Derivatives, integrals, etc. can be taken from e.g. WolframAlpha without working through steps.
- Basic inequalities can be used <u>without proof</u>, as long as you verify numerically. Don't need to include plot on problem set.

EXAMPLE INEQUALITY

$$1+\epsilon \leq \frac{1}{1-\epsilon} \leq 1+2\epsilon \text{ for } \epsilon \in [0, .5].$$

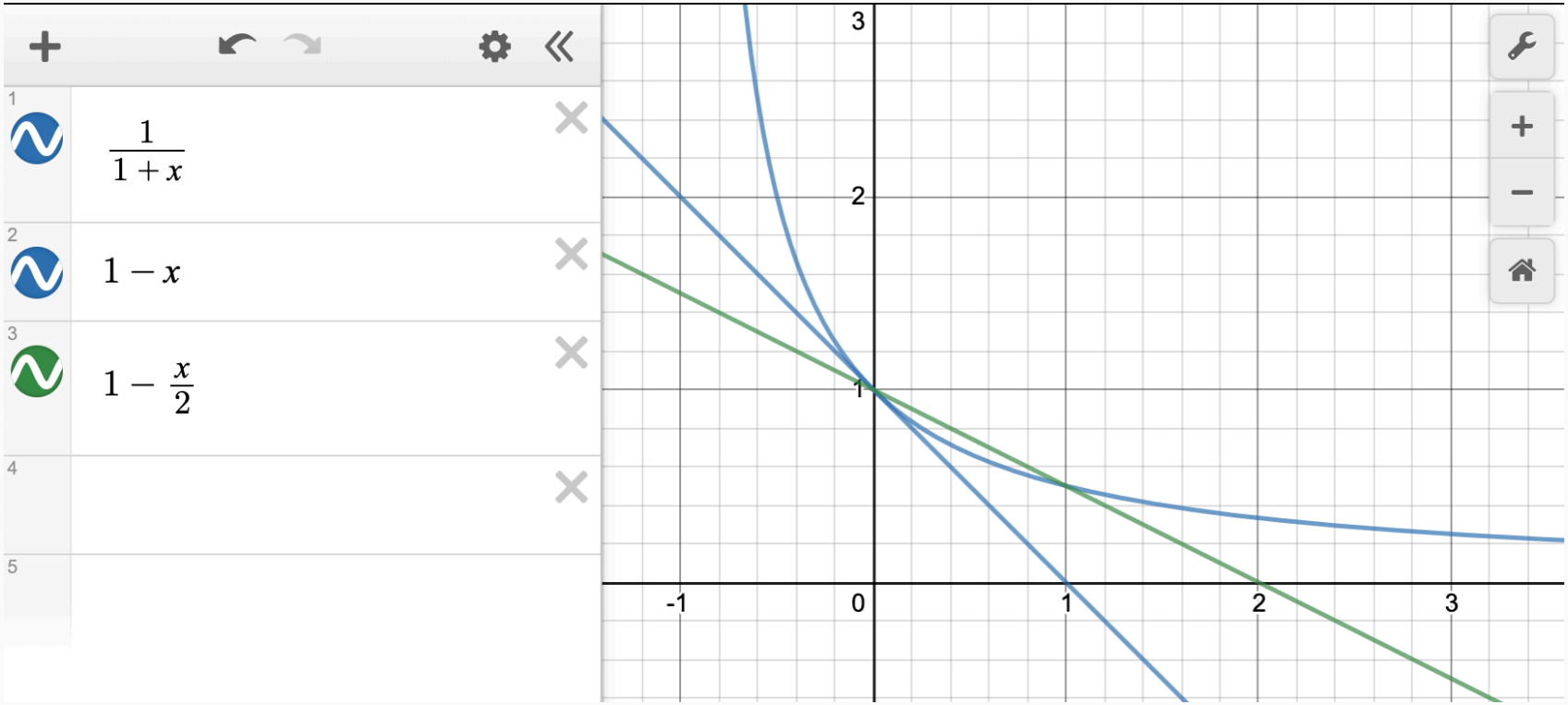
Proof by plotting:



EXAMPLE INEQUALITY

$$1 - \epsilon \leq \frac{1}{1+\epsilon} \leq 1 - .5\epsilon \text{ for } \epsilon \in [0, 1].$$

Proof by plotting:



Tip: When confronted with a complex expression, try to simplify by using big-Oh notation, or just rounding things off. Then clean-up your proof after you get to a solution.

Examples:

- $(m-1) \approx m$
- $\frac{1}{n} - \frac{1}{n^2} \approx \frac{1}{n}$
- $\left(\frac{m-1}{cm^{1.5}}\right)^2 \approx O\left(\frac{1}{m}\right)$.
- $\log(n/2) \approx \log(n)$

QUIZ REVIEW

Which of the following properties hold for *all* random variables X,Y? Check any that apply.

- [ ] E[a*X + b*Y] = a*E[X] + b*E[Y] for constants a,b.
- [ ] E[X]E[Y] = E[XY]
- [ ] Var[a*X + b*Y] = a^2*Var[X] + b^2*Var[Y] for constants a,b.
- [ ] Pr[X > a] <= E[X]/a for constant a.

Which of the following properties hold for *all independent* random variables X,Y? Check any that apply.

- [ ] E[a*X + b*Y] = a*E[X] + b*E[Y] for constants a,b.
- [ ] E[X]E[Y] = E[XY]
- [ ] Var[a*X + b*Y] = a^2*Var[X] + b^2*Var[Y] for constants a,b.
- [ ] Pr[X > a] <= E[X]/a for constant a.

QUIZ REVIEW

A uniformly random hash function h(x) is:

- 2-universal.
- Pairwise independent.
- All of the above.

A pairwise independent hash function h(x) is:

- 2-universal.
- Uniformly random.
- None of the above.

QUIZ REVIEW

List two possible advantages of Chebyshev's inequality over Markov's inequality.

Long answer text

True or False? The Union Bound only applies to independent random events.

- ○ True
- ○ False

QUIZ REVIEW

Let A be the random event that it rains on Dec. 1, 2020. Let B be the random event that it snows on Dec. 1 2020. Suppose A happens with 10% probability, and B happens with 5% probability. Using the union bound, give an upper bound on the probability that *either* it rains or snows on Dec. 1, 2020.

Short answer text

Do you suspect the bound you gave above is tight for this specific A and B? Why or why not?

Long answer text

Central question in randomized algorithms: How well does a random variable X concentrate around it's expectation $\mathbb{E}[X]$?

Three Concentration bounds.

Markov's Inequality. $\Pr[X > k\mathbb{E}[X]] \leq \frac{1}{k}$

- Requires that $X > 0$ always.

Chebyshev's Inequality. $\Pr[|X - \mathbb{E}[X]| > k\sigma] \leq \frac{1}{k^2}$

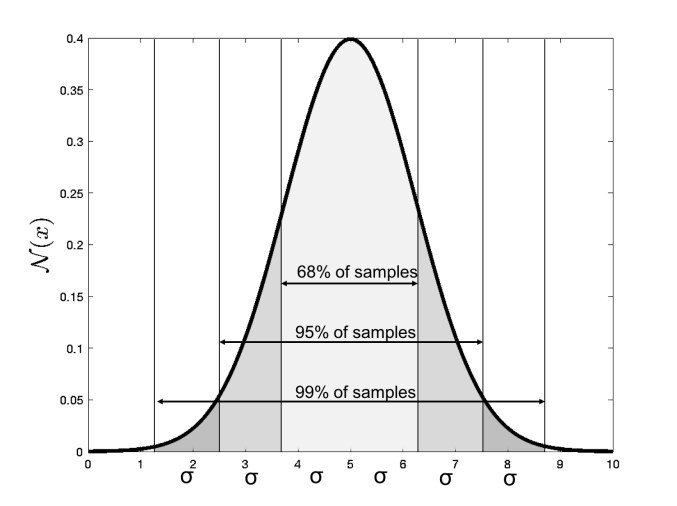
- Here $\sigma^2 = \mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2]$.

Exponential Tail Bounds (Chernoff/Bernstein).

- Will be covered in video lecture.

These bounds are not precise! They use only give a coarse idea of probability random variable deviates from it's expectation.

Motivating question: Is Chebyshev's Inequality tight?



68-95-99 rule for Gaussian bell-curve. $X \sim N(0, \sigma^2)$

Chebyshev's Inequality:

$$\Pr(|X - \mathbb{E}[X]| \geq 1\sigma) \leq 100\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 2\sigma) \leq 25\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 3\sigma) \leq 11\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 4\sigma) \leq 6\%.$$

Truth:

$$\Pr(|X - \mathbb{E}[X]| \geq 1\sigma) \approx 32\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 2\sigma) \approx 5\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 3\sigma) \approx 1\%$$
$$\Pr(|X - \mathbb{E}[X]| \geq 4\sigma) \approx .01\%$$

THIS LECTURE

- Introduce two paradigms for algorithm design: **sketching** and **streaming**.
- Learn about an application of hashing to the **district elements problem** and **estimating Jaccard similarity**.
- During first part of next weeks lecture (9/23) Dr. Aline Bessa will discuss a recent project that uses similar techniques for data set search and augmentation.

Abstract architecture of a streaming algorithm:

Have massive dataset $D = d_1, \ldots, d_n$ with n pieces of data that arrive in a sequential stream. There is far too much data to store or process it in a single location.

- Still want to analyze the data: i.e. fit a model or (approximately) compute some function $f(D)$.
- To do so, we must compress data "on-the-fly", storing some smaller data structure which still contains interesting information.
- Often can only take a <u>single-pass</u> over the data.

A typical goal is to minimize space complexity over computational complexity.

EASY EXAMPLE: MEAN

Design a single-pass streaming algorithm for computing the <u>mean</u> of a data set $d_1, \ldots, d_n$. How much space is required?

HARDER EXAMPLE: MEDIAN

Design a single-pass streaming algorithm for computing the median of a data set $d_1, \ldots, d_n$. How much space is required?

STREAMING ALGORITHMS IN PRACTICE

Sensor data: GPS or seismometer readings to detect geological anomalies, telescope images, satellite imagery, highway travel time sensors.

Web traffic and data: User data for website, including e.g. click data, web searches and API queries, posts and image uploads on social media.

Training machine learning models: Often done in a streaming setting when training dataset is huge, often with multiple passes.

Lots of software frameworks exist for easy development of streaming algorithms.

DISTINCT ELEMENTS PROBLEM

Input: $d_1, \ldots, d_n \in \mathcal{U}$ where $\mathcal{U}$ is a huge universe of items.

Output: Number of <u>distinct</u> inputs.

Example: $f(1, 10, 2, 4, 9, 2, 10, 4) \to 5$

Applications:

- Distinct users hitting a webpage.
- Distinct values in a database column (e.g. for estimating the size of group by queries)
- Number of distinct queries to a search engine.
- Distinct motifs in DNA sequence.

Implementations widely used at Google (Sawzall, Dremel, PowerDrill), Yahoo, Twitter, Facebook Presto, etc.

DISTINCT ELEMENTS PROBLEM

Input: $d_1, \ldots, d_n \in \mathcal{U}$ where $\mathcal{U}$ is a huge universe of items.

Output: Number of <u>distinct</u> inputs.

Example: $f(1, 10, 2, 4, 9, 2, 10, 4) \to 5$

Flajolet–Martin (simplified):

- Choose random hash function $h : \mathcal{U} \to [0, 1]$.
- $S = \infty$
- For $i = 1, \ldots, n$
  - $S \leftarrow \min(S, h(d_i))$
- Return: $\frac{1}{S} - 1$

HOLD UP...

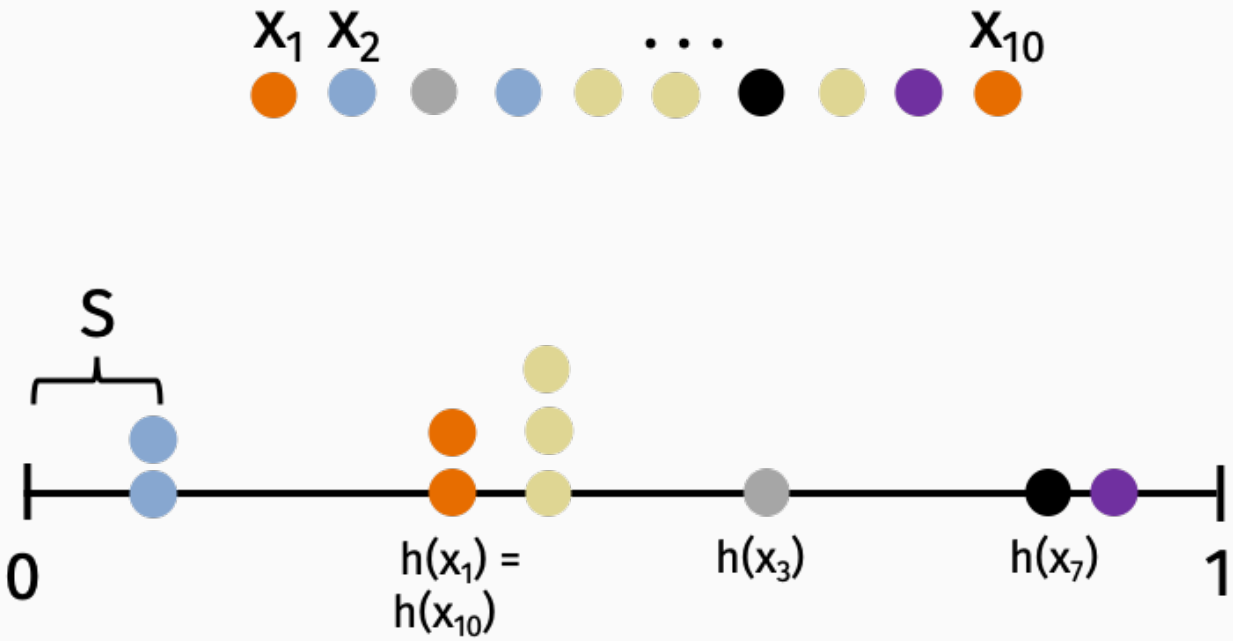
The hash function h maps from $\mathcal{U}$ to a random point in $[0, 1]$?

Hashing to real numbers:

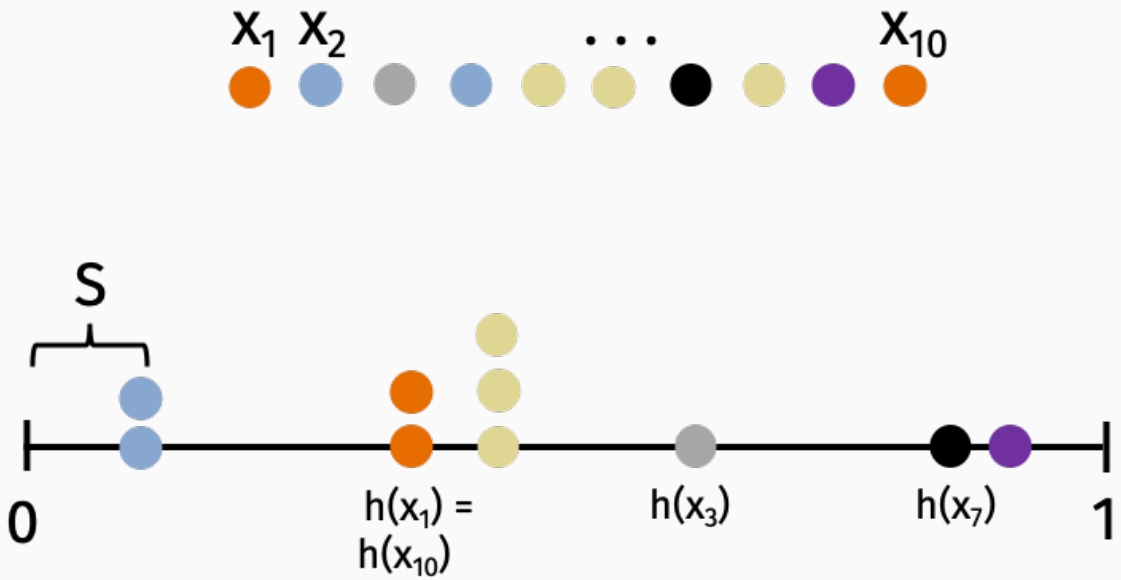
- Impossible to implement $h(x)$ in reality, but you can replace it with $\frac{g(x)}{k}$, where g is a hash function that maps to $\{0, 1, \ldots, k\}$ for sufficiently large k.
- All results hold if this "discrete" hash is used instead, but the analysis is simpler if we assume access to h.
- Just like when we assumed fully random hash functions, this is a useful abstraction which makes understanding and analyzing the underlying algorithms easier.

Flajolet–Martin (simplified):

- Choose random hash function $h : \mathcal{U} \to [0, 1]$.
- $S = \infty$
- For $i = 1, \ldots, n$
  - $S \leftarrow \min(S, h(d_i))$
- Return: $\tilde{D} = \frac{1}{S} - 1$



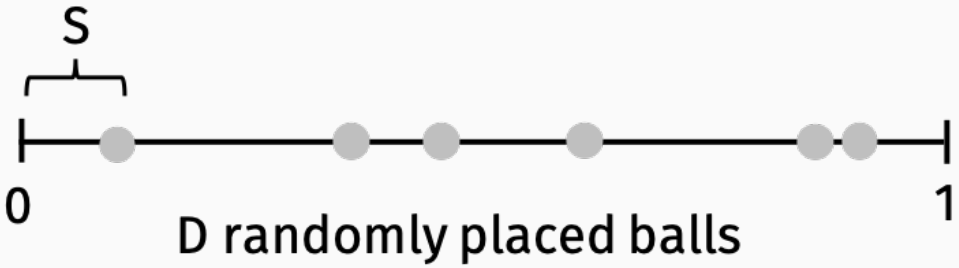
Let D equal the number of distinct elements in our stream.



D unique locations after hashing

Intuition: When D is larger, S will be smaller. Makes sense to return the estimate $\tilde{D} = \frac{1}{S} - 1$.

What is $\mathbb{E}S$?



Let D equal the number of distinct elements in our stream.

Lemma

$\mathbb{E}S = \frac{1}{D+1}$.

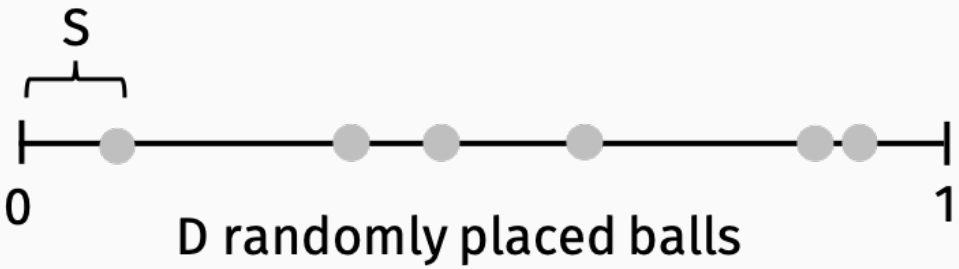
Proof:

$$\begin{aligned}\mathbb{E}[S] &= \int_0^1 \Pr[S \geq \lambda] d\lambda \\ &= \int_0^1 (1-\lambda)^D d\lambda \\ &= \frac{-(1-\lambda)^{D+1}}{D+1}\Bigg|_{\lambda=0}^{1} \\ &= \frac{1}{D+1}\end{aligned}$$

Exercise: Why?

PROOF "FROM THE BOOK"

$\mathbb{E}[S] = \Pr[(D+1)^{\text{st}}$ item has the smallest hash value$]$.



By symmetry, this equals $\frac{1}{D+1}$ (since every ball is equally likely to be first).

$\mathbb{E}S = \frac{1}{D+1}$. **Estimate:** $\tilde{D} = \frac{1}{S} - 1$.

This does not imply that $\mathbb{E}[\tilde{D}] = D$, but we have for $\epsilon < \frac{1}{2}$:

If $|S - \mathbb{E}S| \leq \epsilon \cdot \mathbb{E}S$, then:

$$(1 - 4\epsilon)D \leq \tilde{D} \leq (1 + 4\epsilon)D.$$

Exercise: Prove this to yourself.

So, it suffices to show that S concentrates around its mean. We will use Chebyshev's inequality as our concentration bound.

Lemma

$\mathrm{Var}[S] = \mathbb{E}[S^2] - \mathbb{E}[S]^2 = \frac{2}{(D+1)(D+2)} - \frac{1}{(D+1)^2} \leq \frac{1}{(D+1)^2}$.

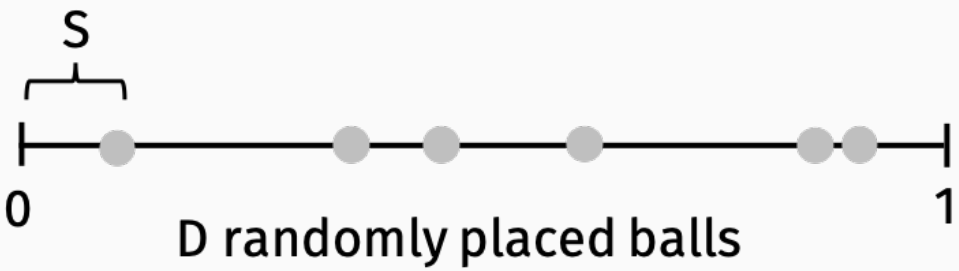
Proof:

$$\begin{aligned}
\mathbb{E}[S^2] &= \int_0^1 \Pr[S^2 \geq \lambda] d\lambda \\
&= \int_0^1 \Pr[S \geq \sqrt{\lambda}] d\lambda \\
&= \int_0^1 (1 - \sqrt{\lambda})^D d\lambda \\
&= \frac{2}{(D+1)(D+2)}
\end{aligned}$$

www.wolframalpha.com/input/?i=integral+from+0+to+1+of+%281-sqrt%28x%29%29%5ED

PROOF "FROM THE BOOK"

$\mathbb{E}[S] = ??.$



- $\mathbb{E}[S] = \frac{1}{D+1} = \mu$.
- $\mathrm{Var}[S] \leq \mu^2$. Standard deviation: $\sigma \leq \mu$.
- Want to bound $\Pr[|S - \mu| \leq \epsilon\mu] \leq \delta$.

Chebyshev's: $\Pr[|S - \mu| \leq \epsilon\mu] = \Pr[|S - \mu| \leq \epsilon\sigma] \leq \frac{1}{\epsilon^2}$.

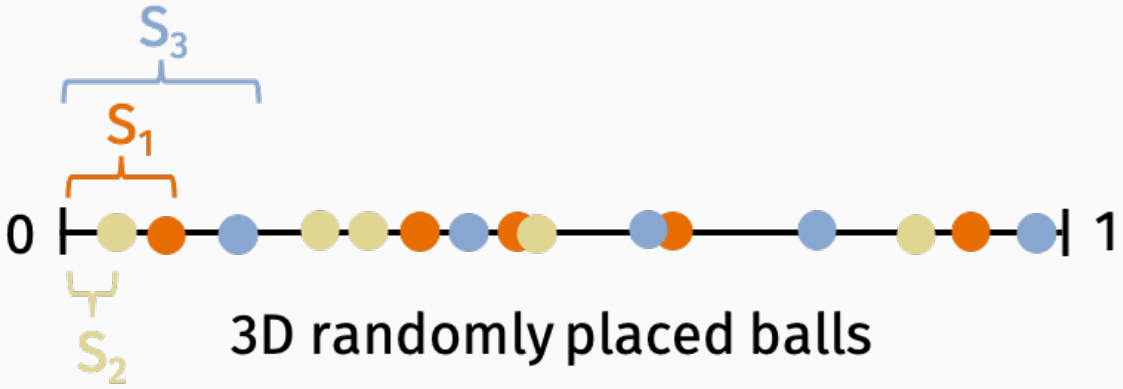
Vacuous bound. Our variance is way too high!

Trick of the trade: Repeat many independent trials and take the mean to get a better estimator.

Given i.i.d. (independent, identically distributed) random variables $X_1, \ldots, X_k$ with mean μ and variance σ^2, what is:

- $\mathbb{E}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] =$
- $\mathrm{Var}\left[\frac{1}{k}\sum_{i=1}^{k} X_i\right] =$

Using independent hash functions, maintain k independent sketches $S_1, \ldots, S_k$.



Flajolet–Martin:

- Choose k random hash function $h_1, \ldots, h_k : \mathcal{U} \to [0, 1]$.
- $S_1 = \infty, \ldots, S_k = \infty$
- For $i = 1, \ldots, n$
  - $S_j \leftarrow \min(S, h_j(d_i))$ for all $j \in 1, \ldots, k$.
- $S = (S_1 + \ldots + S_k)/k$
- Return: $\frac{1}{S} - 1$

1 estimator:

- $\mathbb{E}[S] = \frac{1}{D+1} = \mu$.
- $\mathrm{Var}[S] = \mu^2$

k estimators:

- $\mathbb{E}[S] = \frac{1}{D+1} = \mu$.
- $\mathrm{Var}[S] = \mu^2/k$
- By Chebyshev, $\Pr[|S - \mathbb{E}S| \geq c\mu/\sqrt{k}] \leq \frac{1}{c^2}$.

Setting $c = 1/\sqrt{\delta}$ and $k = O\left(\frac{1}{\epsilon^2\delta}\right)$ gives:

$$\Pr[|S - \mu| \geq \epsilon\mu] \leq \delta.$$

Total space complexity: $O\left(\frac{1}{\epsilon^2\delta}\right)$ to estimate distinct elements up to error ϵ with success probability $1 - \delta$.

NOTE ON FAILURE PROBABILITY

$O\left(\frac{1}{\epsilon^2\delta}\right)$ space is an impressive bound:

- Achieves any accuracy desired. $1/\epsilon^2$ dependence cannot be improved.
- No dependence on number of distinct elements D. Naive algorithm takes $O(D)$ space.
- But... $1/\delta$ dependence is not ideal. For 95% success rate, pay a $\frac{1}{5\%} = 20$ factor overhead in space.

We will discuss how to get a better bound depending on $O(\log(1/\delta))$ in the online lecture.

In practice, we cannot hash to real numbers on $[0, 1]$. Instead, map to bit vectors.

Real Flajolet-Martin / HyperLogLog:

$\mathbf{h}(x_1)$	**1010010**
$\mathbf{h}(x_2)$	**1001100**
$\mathbf{h}(x_3)$	**1001110**
⋮	
$\mathbf{h}(x_n)$	**1011000**

- Estimate # distinct elements based on maximum number of trailing zeros **m**.
- The more distinct hashes we see, the higher we expect this maximum to be.

With D distinct elements what do we expect **m** to be?

Real Flajolet-Martin / HyperLogLog:

$\mathbf{h}(x_1)$	**1010010**
$\mathbf{h}(x_2)$	**1001100**
$\mathbf{h}(x_3)$	**1001110**
⋮	
$\mathbf{h}(x_n)$	**1011000**

- Estimate # distinct elements based on maximum number of trailing zeros **m**.
- The more distinct hashes we see, the higher we expect this maximum to be.

$$\Pr(\mathbf{h}(x_i) \text{ has } \log D \text{ trailing zeros}) =$$

So with D distinct hashes, expect to see 1 with $\log D$ trailing zeros. Expect $\mathbf{m} \approx \log D$. **m** takes $O(\log \log D)$ bits to store.

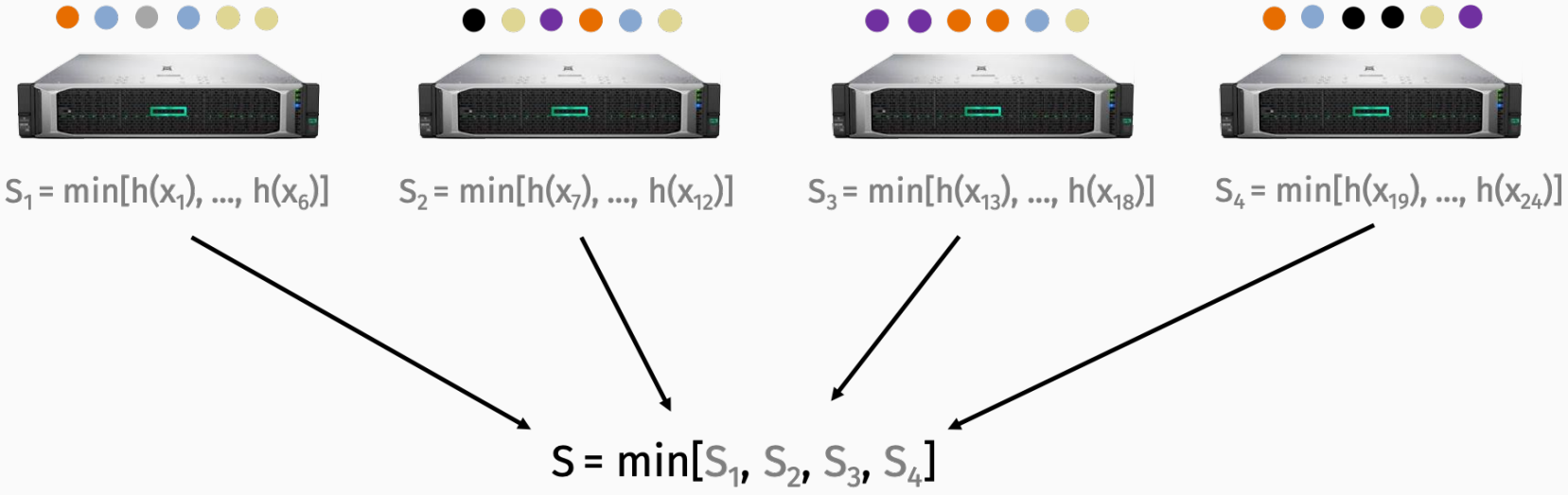
Total Space: $O\left(\frac{\log\log D}{\epsilon^2} + \log D\right)$ for an ϵ approximate count.

"Using an auxiliary memory smaller than the size of this abstract, the LogLog algorithm makes it possible to estimate in a single pass and within a few percents the number of different words in the whole of Shakespeare's works." – Flajolet, Durand.

Using HyperLogLog to count 1 billion distinct items with 2% accuracy:

$$\begin{aligned}\text{space used} &= O\left(\frac{\log\log D}{\epsilon^2} + \log D\right)\\ &= \frac{1.04 \cdot \lceil \log_2 \log_2 D \rceil}{\epsilon^2} + \lceil \log_2 D \rceil \text{ bits}\\ &= \frac{1.04 \cdot 5}{.02^2} + 30 = 13030 \text{ bits} \approx 1.6\ kB!\end{aligned}$$

DISTRIBUTED DISTINCT ELEMENTS



MinHash summaries are "mergeable". No need to share lists of distinct elements if those elements are stored on different machines. Just share minimum hash value.

HYPERLOGLOG IN PRACTICE

Implementations: Google PowerDrill, Facebook Presto, Twitter Algebird, Amazon Redshift.

Use Case: Exploratory SQL-like queries on tables with 100's of billions of rows.

- Count number of distinct users in Germany that made at least one search containing the word 'auto' in the last month.
- Count number of distinct subject lines in emails sent by users that have registered in the last week, in comparison to number of emails sent overall (to estimate rates of spam accounts).

Answering a query requires a (distributed) linear scan over the database: 2 seconds in Google's distributed implementation.

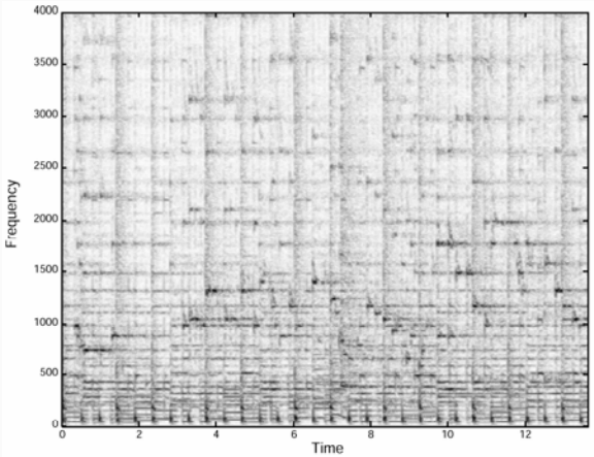
"The system has been in production since end of 2008 and was made available for internal users across all of Google mid 2009. Each month it is used by more than 800 users sending out about 4 million SQL queries. After a hard day's work, one of our top users has spent over 6 hours in the UI, triggering up to 12 thousand queries. When using our column-store as a backend, this may amount to scanning as much as 525 trillion cells in (hypothetical) full scans."

Abstract architecture of a sketching algorithm:

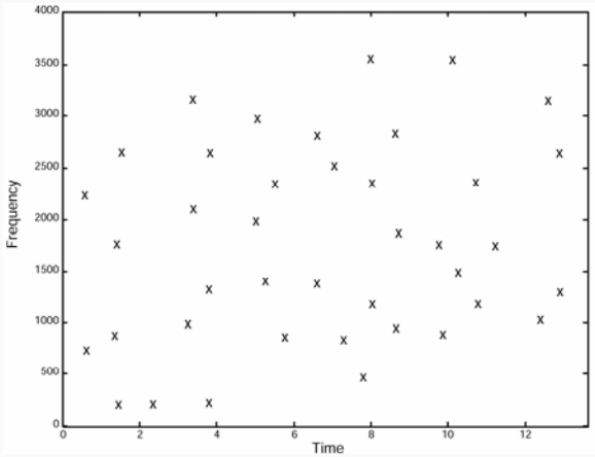
- Given a (high dimensional) dataset $D = d_1, \ldots, d_n$ with n pieces of data each in $\mathbb{R}^d$.
- **Sketch phase:** For each $i \in 1, \ldots, n$, compute $s_i = C(d_i)$, where C is some compression function and $s_i \in \mathbb{R}^k$ for $k \ll d$.
- **Process phase:** Use (more compact) dataset $s_1, \ldots, s_n$ to approximately compute something about D.

Sketching phase is easily distributed, parallelized, etc. Better space complexity, communication complexity, runtime, all at once.

How does **Shazam** match a song clip against a library of 8 million songs (32 TB of data) in a fraction of a second?



Spectrogram extracted from audio clip.



Processed spectrogram: used to construct audio "fingerprint" $\mathbf{q} \in \{0,1\}^d$.

Each clip is represented by a high dimensional binary vector $\mathbf{q}$.

1	0	1	1	0	0	0	1	0	0	0	0	1	1	0	1

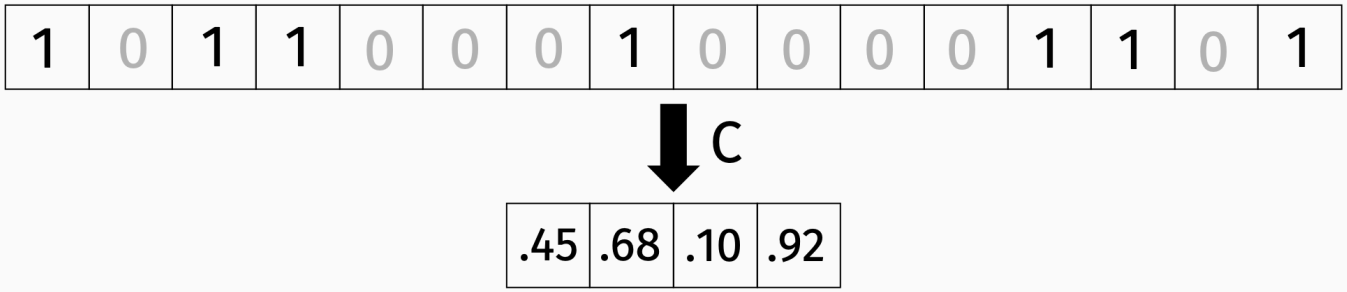
Given $\mathbf{q}$, find any nearby "fingerprint" $\mathbf{y}$ in a database – i.e. any $\mathbf{y}$ with $\text{dist}(\mathbf{y}, \mathbf{q})$ small.

Challenges:

- Database is possibly huge: $O(nd)$ bits.
- Expensive to compute $\text{dist}(\mathbf{y}, \mathbf{q})$: $O(d)$ time.

Goal: Design a more compact sketch for comparing $\mathbf{q}, \mathbf{y} \in \{0,1\}^d$. Ideally $\ll d$ space/time complexity.

$$C(\mathbf{q}) \in \mathbb{R}^k$$
$$C(\mathbf{y}) \in \mathbb{R}^k$$



Homomorphic Compression:

$C(\mathbf{q})$ should be similar to $C(\mathbf{y})$ if $\mathbf{q}$ is similar to $\mathbf{y}$.

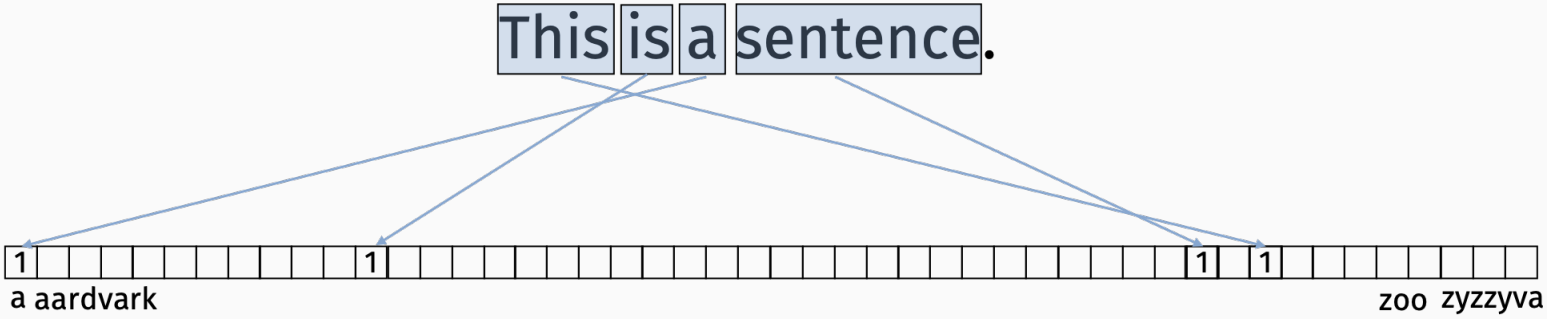
Definition (Jaccard Similarity)

$$J(\mathbf{q}, \mathbf{y}) = \frac{|\mathbf{q} \cap \mathbf{y}|}{|\mathbf{q} \cup \mathbf{y}|} = \frac{\text{\# of non-zero entries in common}}{\text{total \# of non-zero entries}}$$

Natural similarity measure for binary vectors. $0 \leq J(\mathbf{q}, \mathbf{y}) \leq 1$.

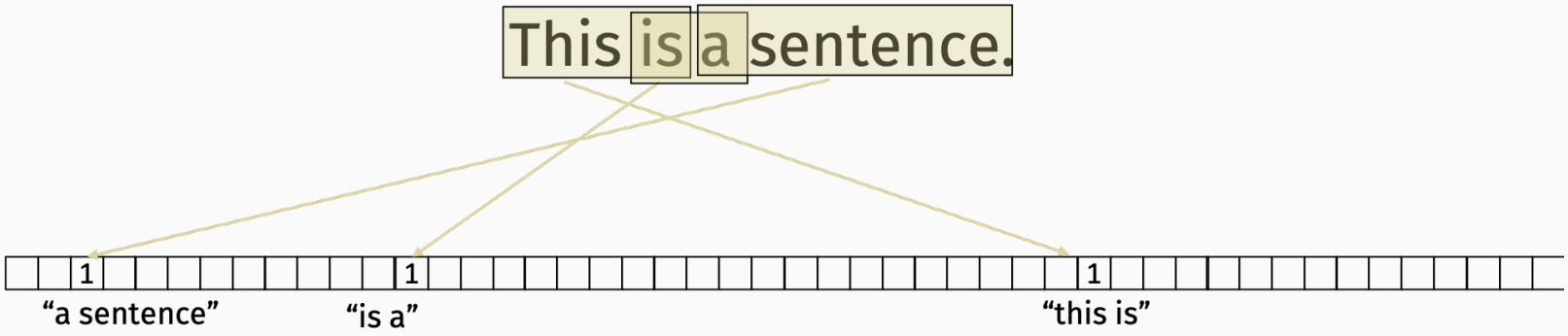
Can be applied to any data which has a natural binary representation (more than you might think).

"Bag-of-words" model:



How many words do a pair of documents have in common?

"Bag-of-words" model:



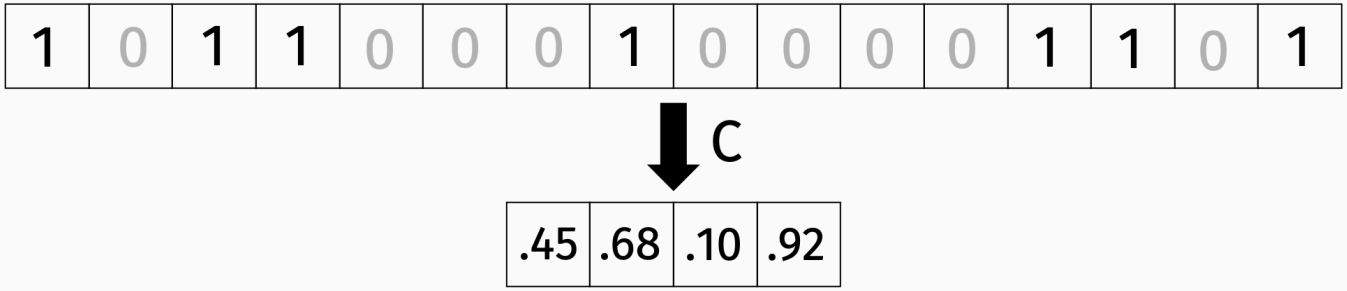
How many bigrams do a pair of documents have in common?

APPLICATIONS: DOCUMENT SIMILARITY

- Finding duplicate or new duplicate documents or webpages.
- Change detection for high-speed web caches.
- Finding near-duplicate emails or customer reviews which could indicate spam.

Other types of data with a natural binary representation?

Goal: Design a compact sketch $C : \{0, 1\} \to \mathbb{R}^k$:



Homomorphic Compression: Want to use $C(\mathbf{q}), C(\mathbf{y})$ to approximately compute the Jaccard similarity $J(\mathbf{q}, \mathbf{y})$.