Learning Goals

- Streaming Algorithms
- Idea of AMS
- k-wise Independence

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics
    - A network switch has a limited memory, and network traffic “streams” through it
    - At the end of the day, we may be interested in statistics such as
        - How many different requests have there been?
        - What is the most frequent request?
        - Variance of the package sizes?

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics
  - A network switch has a limited memory, and network traffic “streams” through it
  - At the end of the day, we may be interested in statistics such as
    - How many different requests have there been?
    - What is the most frequent request?
    - Variance of the package sizes?
- Input: a sequence of indices $i_1, \ldots, i_n \in \{1, \cdots, d\}$

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics
  - A network switch has a limited memory, and network traffic “streams” through it
  - At the end of the day, we may be interested in statistics such as
    - How many different requests have there been?
    - What is the most frequent request?
    - Variance of the package sizes?
- Input: a sequence of indices $i_1, \dots, i_n \in \{1, \cdots, d\}$
- *Frequency vector*: $x \in \mathbb{Z}^d$, with

$$x_j := |\{k : i_k = j\}|.$$

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics
    - A network switch has a limited memory, and network traffic “streams” through it
    - At the end of the day, we may be interested in statistics such as
        - How many different requests have there been?
        - What is the most frequent request?
        - Variance of the package sizes?
- Input: a sequence of indices $i_1, \ldots, i_n \in \{1, \cdots, d\}$
- *Frequency vector*: $x \in \mathbb{Z}^d$, with

$$x_j := |\{k : i_k = j\}|.$$

- Output: certain statistic of x, such as $||x||_p$, $||x||_0$, etc.

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics
  - A network switch has a limited memory, and network traffic "streams" through it
  - At the end of the day, we may be interested in statistics such as
    - How many different requests have there been?
    - What is the most frequent request?
    - Variance of the package sizes?
- Input: a sequence of indices $i_1, \ldots, i_n \in \{1, \cdots, d\}$
- *Frequency vector*: $x \in \mathbb{Z}^d$, with

$$x_j := |\{k : i_k = j\}|.$$

- Output: certain statistic of x, such as $||x||_p$, $||x||_0$, etc.
- The algorithm must use only $O(\log d)$ space.

Streaming Model

- Sometimes a device with limited storage processes a huge amount of data and must return statistics
    - A network switch has a limited memory, and network traffic "streams" through it
    - At the end of the day, we may be interested in statistics such as
        - How many different requests have there been?
        - What is the most frequent request?
        - Variance of the package sizes?
- Input: a sequence of indices $i_1, \ldots, i_n \in \{1, \cdots, d\}$
- *Frequency vector*: $x \in \mathbb{Z}^d$, with

$$x_j := |\{k : i_k = j\}|.$$

- Output: certain statistic of x, such as $||x||_p$, $||x||_0$, etc.
- The algorithm must use only $O(\log d)$ space.
- We usually allow some error in the output

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
  - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
    - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.
    - Initiate $y = 0 \in \mathbb{R}^t$.

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
    - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.
    - Initiate $y = 0 \in \mathbb{R}^t$.
    - When we see $i_k = j$, add the j-th column of L to y.

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
    - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.
    - Initiate $y = 0 \in \mathbb{R}^t$.
    - When we see $i_k = j$, add the j-th column of L to y.
    - Return $||y||$.

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
  - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.
  - Initiate $y = 0 \in \mathbb{R}^t$.
  - When we see $i_k = j$, add the j-th column of L to y.
  - Return $||y||$.
- Guarantee: for any $\delta > 0$, if we set $t = O(\log(\frac{1}{\delta})/\epsilon^2)$, with probability at least $1 - \delta$, we have $(1 - \epsilon)||x|| \leq ||y|| \leq (1 + \epsilon)||x||$.

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
    - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.
    - Initiate $y = 0 \in \mathbb{R}^t$.
    - When we see $i_k = j$, add the j-th column of L to y.
    - Return $||y||$.
- Guarantee: for any $\delta > 0$, if we set $t = O(\log(\frac{1}{\delta})/\epsilon^2)$, with probability at least $1 - \delta$, we have $(1 - \epsilon)||x|| \leq ||y|| \leq (1 + \epsilon)||x||$.
- Issue: we must store $t \times d$ real numbers drawn from a Gaussian distribution!

AMS

- Alon, Matias, Szegedy studied in 1996 the streaming problem for $||x||_2 = (\sum_i x_i^2)^{1/2}$, for which they won the Gödel prize
- Naïve solution using JL-transform:
  - Maintain $L \in \mathbb{R}^{t \times d}$ whose entries are i.i.d. from $\mathcal{N}(0, \frac{1}{t})$.
  - Initiate $y = 0 \in \mathbb{R}^t$.
  - When we see $i_k = j$, add the j-th column of L to y.
  - Return $||y||$.
- Guarantee: for any $\delta > 0$, if we set $t = O(\log(\frac{1}{\delta})/\epsilon^2)$, with probability at least $1 - \delta$, we have $(1 - \epsilon)||x|| \leq ||y|| \leq (1 + \epsilon)||x||$.
- Issue: we must store $t \times d$ real numbers drawn from a Gaussian distribution!
  - Sampling them anew each time does not work — we must use the same linear transform for all the indices.

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much “randomness”

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much "randomness"
    - We were in a similar situation when we thought about hashing.

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much "randomness"
    - We were in a similar situation when we thought about hashing.
    - The solution there was that we weakened the requirement on randomness (universal hashing), and have in return hash functions that take less space to store

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much "randomness"
  - We were in a similar situation when we thought about hashing.
  - The solution there was that we weakened the requirement on randomness (universal hashing), and have in return hash functions that take less space to store
  - We had a small seed of randomness, and used that to grow a whole hashing function

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much “randomness”
    - We were in a similar situation when we thought about hashing.
    - The solution there was that we weakened the requirement on randomness (universal hashing), and have in return hash functions that take less space to store
    - We had a small seed of randomness, and used that to grow a whole hashing function
- Let’s try something similar.

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much "randomness"
  - We were in a similar situation when we thought about hashing.
  - The solution there was that we weakened the requirement on randomness (universal hashing), and have in return hash functions that take less space to store
  - We had a small seed of randomness, and used that to grow a whole hashing function
- Let's try something similar.
- Recall the idea behind JL: if $G_1, \cdots, G_d$ are i.i.d. from $\mathcal{N}(0, 1)$, then $\sum_i G_i x_i \sim \mathcal{N}(0, ||x||^2)$.

Reducing the Memory Needed

- We have an algorithm that unfortunately needs to store too much "randomness"
    - We were in a similar situation when we thought about hashing.
    - The solution there was that we weakened the requirement on randomness (universal hashing), and have in return hash functions that take less space to store
    - We had a small seed of randomness, and used that to grow a whole hashing function
- Let's try something similar.
- Recall the idea behind JL: if $G_1, \cdots, G_d$ are i.i.d. from $\mathcal{N}(0, 1)$, then $\sum_i G_i x_i \sim \mathcal{N}(0, ||x||^2)$.
- In general, if $G_1, \cdots, G_d$ are independent random variables, then $\mathrm{Var}[\sum_i G_i x_i] = \sum_i x_i^2 \mathrm{Var}[G_i]$.

Proof of Claim

Claim

If $G_1, \cdots, G_d$ are independent random variables, then $\mathrm{Var}[\sum_i G_i x_i] = \sum_i x_i^2 \mathrm{Var}[G_i]$.

Proof of Claim

Claim

If $G_1, \cdots, G_d$ are independent random variables, then $\mathrm{Var}[\sum_i G_i x_i] = \sum_i x_i^2 \mathrm{Var}[G_i]$.

Proof.

$$\begin{aligned}
\mathrm{Var}\left[\sum_i G_i x_i\right] &= \mathbf{E}\left[\left(\sum_i G_i x_i - \mathbf{E}\left[\sum_i G_i x_i\right]\right)^2\right] \\
&= \sum_i \mathbf{E}\left[(G_i x_i - \mathbf{E}\left[G_i x_i\right])^2\right] + \sum_{i \neq j} \mathbf{E}\left[(G_i x_i - \mathbf{E}\left[G_i x_i\right]) \cdot (G_j x_j - \mathbf{E}\left[G_j x_j\right])\right] \\
&= \sum_i x_i^2 \mathrm{Var}\left[G_i\right] + \sum_{i \neq j} \mathbf{E}\left[G_i x_i - \mathbf{E}\left[G_i x_i\right]\right] \cdot \mathbf{E}\left[G_j x_j - \mathbf{E}\left[G_j x_j\right]\right] \\
&= \sum_i x_i^2 \mathrm{Var}\left[G_i\right].
\end{aligned}$$

Pairwise Independence

The only place where we used independence was for $i \neq j$, $\mathbf{E}[G_i G_j] = \mathbf{E}[G_i]\,\mathbf{E}[G_j]$. But this is much weaker than requiring *mutual independence* for all $G_1, \cdots, G_n$.

Pairwise Independence

The only place where we used independence was for $i \neq j$, $\mathbf{E}[G_i G_j] = \mathbf{E}[G_i]\,\mathbf{E}[G_j]$. But this is much weaker than requiring *mutual independence* for all $G_1, \cdots, G_n$.

Definition

Random variables $X_1, \cdots, X_n$ are said to be *pairwise independent* if for any $i \neq j$, X_i and X_j are independent, i.e., for any a, b, $\mathbf{Pr}[X_i = a \wedge X_j = b] = \mathbf{Pr}[X_i = a] \cdot \mathbf{Pr}[x_j = b]$.

Pairwise Independence

The only place where we used independence was for $i \neq j$, $\mathbf{E}[G_i G_j] = \mathbf{E}[G_i]\,\mathbf{E}[G_j]$. But this is much weaker than requiring *mutual independence* for all $G_1, \cdots, G_n$.

Definition

Random variables $X_1, \cdots, X_n$ are said to be *pairwise independent* if for any $i \neq j$, X_i and X_j are independent, i.e., for any a, b, $\mathbf{Pr}[X_i = a \wedge X_j = b] = \mathbf{Pr}[X_i = a] \cdot \mathbf{Pr}[x_j = b]$.

In fact, we showed

Claim

If $G_1, \cdots, G_d$ are *pairwise* independent random variables, then $\mathrm{Var}[\sum_i G_i x_i] = \sum_i x_i^2 \,\mathrm{Var}[G_i]$.

Example of Pairwise Independent Random Variables

Let our sample space be $\{1, 2, 3, 4\}$, each outcome having probability $\frac{1}{4}$.

Example of Pairwise Independent Random Variables

Let our sample space be $\{1, 2, 3, 4\}$, each outcome having probability $\frac{1}{4}$.
Let Y_1 take values $0, 0, 1, 1$ for the four outcomes, respectively.

Example of Pairwise Independent Random Variables

Let our sample space be $\{1, 2, 3, 4\}$, each outcome having probability $\frac{1}{4}$.
Let Y_1 take values $0, 0, 1, 1$ for the four outcomes, respectively.
Let Y_2 take values $0, 1, 1, 0$ for the four outcomes, respectively.

Example of Pairwise Independent Random Variables

Let our sample space be $\{1, 2, 3, 4\}$, each outcome having probability $\frac{1}{4}$.
Let Y_1 take values $0, 0, 1, 1$ for the four outcomes, respectively.
Let Y_2 take values $0, 1, 1, 0$ for the four outcomes, respectively. Let Y_3 take values $0, 1, 0, 1$ for the four outcomes, respectively.

Example of Pairwise Independent Random Variables

Let our sample space be $\{1, 2, 3, 4\}$, each outcome having probability $\frac{1}{4}$.
Let Y_1 take values $0, 0, 1, 1$ for the four outcomes, respectively.
Let Y_2 take values $0, 1, 1, 0$ for the four outcomes, respectively. Let Y_3 take values $0, 1, 0, 1$ for the four outcomes, respectively.
Then Y_1, Y_2, Y_3 are pairwise independent but not mutually independent.

Construction of Pairwise Independent Hashing

- Recall our construction of universal hashing:
  - for a prime number q, let $\mathbb{F}_q$ denote the equivalent classes of $0, \ldots, q-1 \mod q$. All operations below are understood to be $\mod q$.
  - Let U be $\mathbb{F}_q^m$, for any $\vec{s} = (s_1, \ldots, s_m) \in \mathbb{F}_q^m$, define hash function

$$h_{\vec{s}}(u = (u_1, \ldots, u_m)) \coloneqq \sum_i s_i u_i.$$

Construction of Pairwise Independent Hashing

- Recall our construction of universal hashing:
  - for a prime number q, let $\mathbb{F}_q$ denote the equivalent classes of $0, \ldots, q-1 \mod q$. All operations below are understood to be $\mod q$.
  - Let U be $\mathbb{F}_q^m$, for any $\vec{s} = (s_1, \ldots, s_m) \in \mathbb{F}_q^m$, define hash function

$$h_{\vec{s}}(u = (u_1, \ldots, u_m)) \coloneqq \sum_i s_i u_i.$$

- We can see this as a random number generator: $\vec{s}$ is the seed, drawn uniformly at random from $\mathbb{F}_q^k$; u is arbitrary and fixed in $\mathbb{F}_q^k$.

Construction of Pairwise Independent Hashing

- Recall our construction of universal hashing:
  - for a prime number q, let $\mathbb{F}_q$ denote the equivalent classes of $0, \ldots, q-1 \mod q$. All operations below are understood to be $\mod q$.
  - Let U be $\mathbb{F}_q^m$, for any $\vec{s} = (s_1, \ldots, s_m) \in \mathbb{F}_q^m$, define hash function

$$h_{\vec{s}}(u = (u_1, \ldots, u_m)) := \sum_i s_i u_i.$$

- We can see this as a random number generator: $\vec{s}$ is the seed, drawn uniformly at random from $\mathbb{F}_q^k$; u is arbitrary and fixed in $\mathbb{F}_q^k$.
- Consider the case $m = 1$. For any $b \in \mathbb{F}_q$, $\mathbf{Pr}_s[h_s(u) = b] = \frac{1}{q}$.

Construction of Pairwise Independent Hashing

- Recall our construction of universal hashing:
    - for a prime number q, let $\mathbb{F}_q$ denote the equivalent classes of $0, \ldots, q-1 \mod q$. All operations below are understood to be $\mod q$.
    - Let U be $\mathbb{F}_q^m$, for any $\vec{s} = (s_1, \ldots, s_m) \in \mathbb{F}_q^m$, define hash function

$$h_{\vec{s}}(u = (u_1, \ldots, u_m)) := \sum_i s_i u_i.$$

- We can see this as a random number generator: $\vec{s}$ is the seed, drawn uniformly at random from $\mathbb{F}_q^k$; u is arbitrary and fixed in $\mathbb{F}_q^k$.
- Consider the case $m = 1$. For any $b \in \mathbb{F}_q$, $\mathbf{Pr}_s[h_s(u) = b] = \frac{1}{q}$.
- Now if we sample independent s_1, s_2 uniformly from $\mathbb{F}_q$, then for any $u \in \mathbb{F}_q$, $h_{s_1,s_2}(u) := s_1 u + s_2$ is a random number on $\mathbb{F}_q$.

Claim

Random variables $h_{s_1,s_2}(1), \cdots, h_{s_1,s_2}(q-1)$ are pairwise independent random variables, each distributed uniformly on $\mathbb{F}_q$.

Claim

Random variables $h_{s_1,s_2}(1), \cdots, h_{s_1,s_2}(q-1)$ are pairwise independent random variables, each distributed uniformly on $\mathbb{F}_q$.

Proof.

For any $b_1, b_2 \in \mathbb{F}_q$, and for any $u \neq v \in \mathbb{F}_q$, the equation

$$\begin{cases} s_1 u + s_2 = b_1 \\ s_1 v + s_2 = b_2 \end{cases} \Rightarrow \begin{pmatrix} 1 & u \\ 1 & v \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

has a unique solution (since the coefficient matrix is full rank for $u \neq v$.)

Claim

Random variables $h_{s_1,s_2}(1), \cdots, h_{s_1,s_2}(q-1)$ are pairwise independent random variables, each distributed uniformly on $\mathbb{F}_q$.

Proof.

For any $b_1, b_2 \in \mathbb{F}_q$, and for any $u \neq v \in \mathbb{F}_q$, the equation

$$\left\{ \begin{array}{l} s_1 u + s_2 = b_1 \\ s_1 v + s_2 = b_2 \end{array} \right. \Rightarrow \begin{pmatrix} 1 & u \\ 1 & v \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

has a unique solution (since the coefficient matrix is full rank for $u \neq v$.) Therefore $\mathbf{Pr}[h_{s_1,s_2}(u) = b_1 \wedge h_{s_1,s_2}(v) = b_2] = \frac{1}{q^2}$.

Claim

Random variables $h_{s_1,s_2}(1), \cdots, h_{s_1,s_2}(q-1)$ are pairwise independent random variables, each distributed uniformly on $\mathbb{F}_q$.

Proof.

For any $b_1, b_2 \in \mathbb{F}_q$, and for any $u \neq v \in \mathbb{F}_q$, the equation

$$\begin{cases} s_1 u + s_2 = b_1 \\ s_1 v + s_2 = b_2 \end{cases} \Rightarrow \begin{pmatrix} 1 & u \\ 1 & v \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

has a unique solution (since the coefficient matrix is full rank for $u \neq v$.)

Therefore $\mathbf{Pr}[h_{s_1,s_2}(u) = b_1 \wedge h_{s_1,s_2}(v) = b_2] = \frac{1}{q^2}$.

This implies that $h_{s_1,s_2}(u)$ is uniformly distributed on $\mathbb{F}_q$. ☐

k-wise Independence

Definition

Random variables $X_1, \cdots, X_n$ are said to be *k-wise independent* if any k of them are mutually independent.

k-wise Independence

Definition

Random variables $X_1, \cdots, X_n$ are said to be *k-wise independent* if any k of them are mutually independent.

Definition

A family $\mathcal{H}$ of hash functions from U to $\{0, \ldots, m\}$ is *k-universal* if for any k distinct key values $u_1, \ldots, u_k \in U$, and any k (not necessarily distinct) hash addresses $b_1, \ldots, b_k \in \{0, \ldots, m-1\}$,

$$\mathbf{Pr}_{h \sim \mathcal{H}} \left[h(u_1) = b_1 \wedge \cdots \wedge h(u_k) = b_k \right] = \left(\frac{1}{m} \right)^k.$$

Construction of *k*-wise independent random variables

For prime q, let U be $\mathbb{F}_q$. Let random seeds $s_1, \ldots, s_k$ be independent uniform samples from $\mathbb{F}_q$. Define

$$h_{(s_1,\ldots,s_k)}(u) := s_1 u^{k-1} + s_2 u^{k_2} + \ldots + s_{k-1} u + s_k .$$

Construction of k-wise independent random variables

For prime q, let U be $\mathbb{F}_q$. Let random seeds $s_1, \ldots, s_k$ be independent uniform samples from $\mathbb{F}_q$. Define

$$h_{(s_1,\ldots,s_k)}(u) := s_1 u^{k-1} + s_2 u^{k_2} + \ldots + s_{k-1} u + s_k .$$

Theorem

The set of $h_{\vec{s}}$ thus defined is a k-universal hash family.

Construction of k-wise independent random variables

For prime q, let U be $\mathbb{F}_q$. Let random seeds $s_1, \ldots, s_k$ be independent uniform samples from $\mathbb{F}_q$. Define

$$h_{(s_1,\ldots,s_k)}(u) := s_1 u^{k-1} + s_2 u^{k_2} + \ldots + s_{k-1} u + s_k.$$

Theorem

The set of $h_{\vec{s}}$ thus defined is a k-universal hash family.

Proof.

For any distinct $u_1, \ldots, u_k \in \mathbb{F}_q$, and $b_1, \ldots, b_k \in \mathbb{F}_q$ that are not necessarily distinct, we show that there is a unique $\vec{s} = (s_1, \ldots, s_k)$ such that $h_{\vec{s}}(u_i) = b_i$ for $i = 1, \cdots, k$.

Proof of k-Universality (Cont.)

(Continued).

$$\begin{cases} s_1 u_1^{k-1} + \ldots + s_{k-1} u_1 + s_K = b_1 \\ s_1 u_2^{k-1} + \ldots + s_{k-1} u_2 + s_K = b_2 \\ \ldots \\ s_1 u_k^{k-1} + \ldots + s_{k-1} u_k + s_k = b_k \end{cases}$$

$$\Leftrightarrow \begin{pmatrix} u_1^{k-1} & u_1^{k-1} & \ldots & u_1 & 1 \\ u_2^{k-1} & u_2^{k-1} & \ldots & u_2 & 1 \\ & \ldots & & \ldots & \\ u_k^{k-1} & u_k^{k-1} & \ldots & u_k & 1 \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ \ldots \\ s_k \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \ldots \\ b_k \end{pmatrix}.$$

The coefficient matrix is a *van der Monder* matrix. For distinct $u_1, \ldots, u_k$ it has full rank. So the system has a unique solution. □

Brief Introduction to Finite Fields

- In the construction of universal hashing, our hash function mapped $U = \mathbb{F}_q^m$ to $\mathbb{F}_q$. Our construction of k-universal hashing so far only allows mapping from $\mathbb{F}_q$ to $\mathbb{F}_q$.

Brief Introduction to Finite Fields

- In the construction of universal hashing, our hash function mapped $U = \mathbb{F}_q^m$ to $\mathbb{F}_q$. Our construction of k-universal hashing so far only allows mapping from $\mathbb{F}_q$ to $\mathbb{F}_q$.
- What if we'd like h to map from $\mathbb{F}_q^m$ to $\mathbb{F}_q^\ell$, for $\ell < m$?

Brief Introduction to Finite Fields

- In the construction of universal hashing, our hash function mapped $U = \mathbb{F}_q^m$ to $\mathbb{F}_q$. Our construction of k-universal hashing so far only allows mapping from $\mathbb{F}_q$ to $\mathbb{F}_q$.
- What if we'd like h to map from $\mathbb{F}_q^m$ to $\mathbb{F}_q^\ell$, for $\ell < m$?
- If we have k-universal hashing from $\mathbb{F}_q^m$ to $\mathbb{F}_q^m$, then we may take, say, the first k coordinates of the hash code.

Brief Introduction to Finite Fields

- In the construction of universal hashing, our hash function mapped $U = \mathbb{F}_q^m$ to $\mathbb{F}_q$. Our construction of k-universal hashing so far only allows mapping from $\mathbb{F}_q$ to $\mathbb{F}_q$.
- What if we'd like h to map from $\mathbb{F}_q^m$ to $\mathbb{F}_q^\ell$, for $\ell < m$?
- If we have k-universal hashing from $\mathbb{F}_q^m$ to $\mathbb{F}_q^m$, then we may take, say, the first k coordinates of the hash code.
- The whole construction would go through if $\mathbb{F}_q^m$ supports the same operations as $\mathbb{F}_q$.

Brief Introduction to Finite Fields

- In the construction of universal hashing, our hash function mapped $U = \mathbb{F}_q^m$ to $\mathbb{F}_q$. Our construction of k-universal hashing so far only allows mapping from $\mathbb{F}_q$ to $\mathbb{F}_q$.
- What if we'd like h to map from $\mathbb{F}_q^m$ to $\mathbb{F}_q^\ell$, for $\ell < m$?
- If we have k-universal hashing from $\mathbb{F}_q^m$ to $\mathbb{F}_q^m$, then we may take, say, the first k coordinates of the hash code.
- The whole construction would go through if $\mathbb{F}_q^m$ supports the same operations as $\mathbb{F}_q$.
  - Obviously, $\mathbb{F}_q^m$ as a vector space supports addition and subtraction.

Brief Introduction to Finite Fields

- In the construction of universal hashing, our hash function mapped $U = \mathbb{F}_q^m$ to $\mathbb{F}_q$. Our construction of k-universal hashing so far only allows mapping from $\mathbb{F}_q$ to $\mathbb{F}_q$.
- What if we'd like h to map from $\mathbb{F}_q^m$ to $\mathbb{F}_q^\ell$, for $\ell < m$?
- If we have k-universal hashing from $\mathbb{F}_q^m$ to $\mathbb{F}_q^m$, then we may take, say, the first k coordinates of the hash code.
- The whole construction would go through if $\mathbb{F}_q^m$ supports the same operations as $\mathbb{F}_q$.
  - Obviously, $\mathbb{F}_q^m$ as a vector space supports addition and subtraction.
  - How do we define multiplication between vectors while satisfying commutativity, associativity and distributive law?

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree *n irreducible polynomial.*

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree *n irreducible polynomial*.
- Example: On $\mathbb{F}_2$, the polynomial x^2+x+1 is irreducible.

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree n *irreducible polynomial*.
- Example: On $\mathbb{F}_2$, the polynomial x^2+x+1 is irreducible.
- $(1,1)\cdot(1,0)=(0,1)$ because $(x+1)x=x^2+x\equiv 1 \mod (x^2+x+1)$.

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree n *irreducible polynomial*.
- Example: On $\mathbb{F}_2$, the polynomial x^2+x+1 is irreducible.
- $(1,1)\cdot(1,0)=(0,1)$ because $(x+1)x = x^2+x \equiv 1 \mod (x^2+x+1)$.
- Alternatively, you may think of *extending* the field $\mathbb{F}_2$ with an additional element α satisfying $\alpha^2=\alpha+1$.

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree *n irreducible polynomial.*
- Example: On $\mathbb{F}_2$, the polynomial x^2+x+1 is irreducible.
- $(1,1)\cdot(1,0)=(0,1)$ because $(x+1)x = x^2+x \equiv 1 \mod (x^2+x+1)$.
- Alternatively, you may think of *extending* the field $\mathbb{F}_2$ with an additional element α satisfying $\alpha^2=\alpha+1$.
    - In much of the same way, the complex field is the extension of the real field with the addition of i that solves $i^2=-1$.

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree *n irreducible polynomial.*
- Example: On $\mathbb{F}_2$, the polynomial x^2+x+1 is irreducible.
- $(1,1)\cdot(1,0)=(0,1)$ because $(x+1)x=x^2+x\equiv 1 \mod (x^2+x+1)$.
- Alternatively, you may think of *extending* the field $\mathbb{F}_2$ with an additional element α satisfying $\alpha^2=\alpha+1$.
  - In much of the same way, the complex field is the extension of the real field with the addition of i that solves $i^2=-1$.
  - So $(\alpha+1)\alpha=\alpha^2+\alpha=1$.

Field Extension

- Answer: we see a vector in $\mathbb{F}_q^m$ as coefficients of a polynomial of degree $m-1$, and do multiplication of vectors as polynomial multiplications modulo a degree n *irreducible polynomial*.
- Example: On $\mathbb{F}_2$, the polynomial x^2+x+1 is irreducible.
- $(1,1)\cdot(1,0)=(0,1)$ because $(x+1)x = x^2+x \equiv 1 \mod (x^2+x+1)$.
- Alternatively, you may think of *extending* the field $\mathbb{F}_2$ with an additional element α satisfying $\alpha^2=\alpha+1$.
  - In much of the same way, the complex field is the extension of the real field with the addition of i that solves $i^2=-1$.
  - So $(\alpha+1)\alpha=\alpha^2+\alpha=1$.
- One can show that degree n irreducible polynomials always exist for $\mathbb{F}_q$. So we can construct fields $\mathbb{F}_{p^m}$ for any positive integer m.

JL with *k*-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
    - We'll decide k later.

JL with k-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
    - We'll decide k later.
    - For a k-universal hash function $h \sim \mathcal{H}$ with seed s, let $L_j = h_s(j)$.

JL with *k*-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
  - We'll decide k later.
  - For a k-universal hash function $h \sim \mathcal{H}$ with seed s, let $L_j = h_s(j)$.
- Consider $y := \sum_i L_i x_i$.

JL with *k*-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
    - We'll decide k later.
    - For a k-universal hash function $h \sim \mathcal{H}$ with seed s, let $L_j = h_s(j)$.
- Consider $y := \sum_i L_i x_i$.
    - $\mathbf{E}[y] = 0$ because $\mathbf{E}[L_i] = 0$ for each i.

JL with *k*-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
  - We'll decide k later.
  - For a k-universal hash function $h \sim \mathcal{H}$ with seed s, let $L_j = h_s(j)$.
- Consider $y := \sum_i L_i x_i$.
  - $\mathbf{E}[y] = 0$ because $\mathbf{E}[L_i] = 0$ for each i.
  - The variance of $L_i x_i$ is $\mathbf{E}[L_i^2 x_i^2] = x_i^2$. As long as $L_1, \cdots, L_d$ are pairwise independent, we have $\mathrm{Var}[y] = \sum_i x_i^2 = ||x||^2$. On the other hand, we have $\mathrm{Var}[y] = \mathbf{E}[y^2] - (\mathbf{E}[y])^2 = \mathbf{E}[y^2]$.

JL with *k*-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
  - We'll decide k later.
  - For a k-universal hash function $h \sim \mathcal{H}$ with seed s, let $L_j = h_s(j)$.
- Consider $y \coloneqq \sum_i L_i x_i$.
  - $\mathbf{E}[y] = 0$ because $\mathbf{E}[L_i] = 0$ for each i.
  - The variance of $L_i x_i$ is $\mathbf{E}[L_i^2 x_i^2] = x_i^2$. As long as $L_1, \cdots, L_d$ are pairwise independent, we have $\mathrm{Var}[y] = \sum_i x_i^2 = ||x||^2$. On the other hand, we have $\mathrm{Var}[y] = \mathbf{E}[y^2] - (\mathbf{E}[y])^2 = \mathbf{E}[y^2]$.
  - We would like to estimate $||x||^2$, so we would like y^2 to concentrate around its expectation.

JL with *k*-wise Independent Hash

- Let's use k-wise independent variables $L_1, \cdots, L_d$, each distributed evenly on $\{-1, +1\}$, to emulate JL.
  - We'll decide k later.
  - For a k-universal hash function $h \sim \mathcal{H}$ with seed s, let $L_j = h_s(j)$.
- Consider $y := \sum_i L_i x_i$.
  - $\mathbf{E}[y] = 0$ because $\mathbf{E}[L_i] = 0$ for each i.
  - The variance of $L_i x_i$ is $\mathbf{E}[L_i^2 x_i^2] = x_i^2$. As long as $L_1, \cdots, L_d$ are pairwise independent, we have $\mathrm{Var}[y] = \sum_i x_i^2 = ||x||^2$. On the other hand, we have $\mathrm{Var}[y] = \mathbf{E}[y^2] - (\mathbf{E}[y])^2 = \mathbf{E}[y^2]$.
  - We would like to estimate $||x||^2$, so we would like y^2 to concentrate around its expectation.
  - We cannot afford the Chernoff bound. But we may use Chebyshev inequality if we can bound $\mathrm{Var}[y^2]$!

$$\mathbf{Pr}\left[|y^2 - \mathbf{E}\left[y^2\right]| > \alpha\right] \leq \frac{\mathrm{Var}[y^2]}{\alpha^2}.$$

Variance of $\sum_i y^2$

$$\mathrm{Var}\left[y^2\right] \leq \mathbf{E}\left[y^4\right] = \mathbf{E}\left[\left(\sum_i L_i x_i\right)^4\right]$$

$$= \sum_{j_1, j_2, j_3, j_4 \in [n]} \mathbf{E}\left[L_{j_1} L_{j_2} L_{j_3} L_{j_4}\right] x_{j_1} x_{j_2} x_{j_3} x_{j_4}.$$

Variance of $\sum_i y^2$

$$\mathrm{Var}\left[y^2\right] \leq \mathbf{E}\left[y^4\right] = \mathbf{E}\left[\left(\sum_i L_i x_i\right)^4\right]$$

$$= \sum_{j_1,j_2,j_3,j_4 \in [n]} \mathbf{E}\left[L_{j_1} L_{j_2} L_{j_3} L_{j_4}\right] x_{j_1} x_{j_2} x_{j_3} x_{j_4}.$$

Now to simplify the analysis, we will require that $L_1, \cdots, L_d$ be 4-wise independent.

Variance of $\sum_i y^2$

$$\mathrm{Var}\left[y^2\right] \leq \mathbf{E}\left[y^4\right] = \mathbf{E}\left[\left(\sum_i L_i x_i\right)^4\right]$$
$$= \sum_{j_1, j_2, j_3, j_4 \in [n]} \mathbf{E}\left[L_{j_1} L_{j_2} L_{j_3} L_{j_4}\right] x_{j_1} x_{j_2} x_{j_3} x_{j_4}.$$

Now to simplify the analysis, we will require that $L_1, \cdots, L_d$ be 4-wise independent.

Whenever some $j \in [n]$ appears only once among j_1, j_2, j_3, j_4, the term $\mathbf{E}[L_{j_1} L_{j_2} L_{j_3} L_{j_4}] = 0$.

Variance of $\sum_i y^2$

$$\operatorname{Var}\left[y^2\right] \le \mathbf{E}\left[y^4\right] = \mathbf{E}\left[\left(\sum_i L_i x_i\right)^4\right]$$

$$= \sum_{j_1,j_2,j_3,j_4 \in [n]} \mathbf{E}\left[L_{j_1} L_{j_2} L_{j_3} L_{j_4}\right] x_{j_1} x_{j_2} x_{j_3} x_{j_4}.$$

Now to simplify the analysis, we will require that $L_1, \cdots, L_d$ be 4-wise independent.

Whenever some $j \in [n]$ appears only once among j_1, j_2, j_3, j_4, the term $\mathbf{E}[L_{j_1} L_{j_2} L_{j_3} L_{j_4}] = 0$.

Only two kinds of factors remain non-zero:

- $j_1 = j_2 = j_3 = j_4 = j$, each such term appears once, contributing x_j^4 to the sum.

Variance of $\sum_i y^2$

$$\mathrm{Var}\left[y^2\right] \le \mathbf{E}\left[y^4\right] = \mathbf{E}\left[\left(\sum_i L_i x_i\right)^4\right]$$
$$= \sum_{j_1, j_2, j_3, j_4 \in [n]} \mathbf{E}\left[L_{j_1} L_{j_2} L_{j_3} L_{j_4}\right] x_{j_1} x_{j_2} x_{j_3} x_{j_4}.$$

Now to simplify the analysis, we will require that $L_1, \cdots, L_d$ be 4-wise independent.

Whenever some $j \in [n]$ appears only once among j_1, j_2, j_3, j_4, the term $\mathbf{E}[L_{j_1} L_{j_2} L_{j_3} L_{j_4}] = 0$.

Only two kinds of factors remain non-zero:

- $j_1 = j_2 = j_3 = j_4 = j$, each such term appears once, contributing x_j^4 to the sum.
- j_1, j_2, j_3, j_4 are split into two equal pairs. For each $i_1, i_2 \in [n]$, $i_1 < i_2$, these terms contribute altogether $6x_{i_1}^2 x_{i_2}^2$.

Multiple Samples

So we have $\mathrm{Var}[y^2] \leq \sum_{j\in[n]} x_j^4 + 6\sum_{i_1<i_2} x_{i_1}^2 x_{i_2}^2 \leq 2||x||^4$.

Multiple Samples

So we have $\mathrm{Var}[y^2] \leq \sum_{j \in [n]} x_j^4 + 6 \sum_{i_1 < i_2} x_{i_1}^2 x_{i_2}^2 \leq 2||x||^4$. Therefore $\mathbf{Pr}[|y^2 - ||x||^2| > \alpha] \leq 2||x||^4/\alpha^2$. We are interested in $\alpha = \epsilon||x||^2$.

Multiple Samples

So we have $\mathrm{Var}[y^2] \leq \sum_{j\in[n]} x_j^4 + 6\sum_{i_1<i_2} x_{i_1}^2 x_{i_2}^2 \leq 2||x||^4$. Therefore $\mathbf{Pr}[|y^2 - ||x||^2| > \alpha] \leq 2||x||^4/\alpha^2$. We are interested in $\alpha = \epsilon||x||^2$.

- To make the error rate smaller, let's have t independent estimates $y_1, \ldots, y_t$.

Multiple Samples

So we have $\mathrm{Var}[y^2] \leq \sum_{j \in [n]} x_j^4 + 6 \sum_{i_1 < i_2} x_{i_1}^2 x_{i_2}^2 \leq 2||x||^4$. Therefore $\mathbf{Pr}[|y^2 - ||x||^2| > \alpha] \leq 2||x||^4/\alpha^2$. We are interested in $\alpha = \epsilon||x||^2$.

- To make the error rate smaller, let's have t independent estimates $y_1, \ldots, y_t$.
    - This uses a matrix $L \in \{+1, -1\}^{t \times d}$, whose rows are indepedent, but within each row, $L_{i,1}, \cdots, L_{i,d}$ are only 4-wise independent.

Multiple Samples

So we have $\mathrm{Var}[y^2] \leq \sum_{j\in[n]} x_j^4 + 6\sum_{i_1<i_2} x_{i_1}^2 x_{i_2}^2 \leq 2||x||^4$. Therefore $\mathbf{Pr}[|y^2 - ||x||^2| > \alpha] \leq 2||x||^4/\alpha^2$. We are interested in $\alpha = \epsilon||x||^2$.

- To make the error rate smaller, let's have t independent estimates $y_1, \ldots, y_t$.
  - This uses a matrix $L \in \{+1, -1\}^{t\times d}$, whose rows are indepedent, but within each row, $L_{i,1}, \cdots, L_{i,d}$ are only 4-wise independent.
- The variance of $\frac{1}{t}\sum_i y_i$ is bounded by $\frac{2||x||^4}{t}$.

Multiple Samples

So we have $\mathrm{Var}[y^2] \leq \sum_{j\in[n]} x_j^4 + 6\sum_{i_1<i_2} x_{i_1}^2 x_{i_2}^2 \leq 2||x||^4$. Therefore $\mathbf{Pr}[|y^2 - ||x||^2| > \alpha] \leq 2||x||^4/\alpha^2$. We are interested in $\alpha = \epsilon||x||^2$.

- To make the error rate smaller, let's have t independent estimates $y_1, \ldots, y_t$.
  - This uses a matrix $L \in \{+1,-1\}^{t\times d}$, whose rows are indepedent, but within each row, $L_{i,1}, \cdots, L_{i,d}$ are only 4-wise independent.
- The variance of $\frac{1}{t}\sum_i y_i$ is bounded by $\frac{2||x||^4}{t}$.
- So as long as $\frac{2}{\epsilon^2 t} \leq \delta$, i.e., $t \geq \frac{2}{\epsilon^2\delta}$, we would have that $\mathbf{Pr}[|\frac{1}{t}\sum_i y_i - ||x||^2| > \epsilon||x||^2] < \delta$.

Space requirement

- We need to store $y_1, \ldots, y_t$ throughout the algorithm, each using $O(\log d)$ space.

Space requirement

- We need to store $y_1, \ldots, y_t$ throughout the algorithm, each using $O(\log d)$ space.
- We need to store the hash functions we use to generate each row of L.

Space requirement

- We need to store $y_1, \ldots, y_t$ throughout the algorithm, each using $O(\log d)$ space.
- We need to store the hash functions we use to generate each row of L.
    - For k-universal hashing from $[d]$, the seed takes space $O(k \log d)$.

Space requirement

- We need to store $y_1, \ldots, y_t$ throughout the algorithm, each using $O(\log d)$ space.
- We need to store the hash functions we use to generate each row of L.
    - For k-universal hashing from $[d]$, the seed takes space $O(k \log d)$.
    - We used 4-universal hashing, so each hash function takes $O(\log d)$ space, and there are t of them.
- Altogether the space used is $O(\frac{\log d}{\epsilon^2 \delta})$.