CS-GY 9223 I: Lecture 3
Sketching, the Johnson-Lindenstrauss lemma + applications

NYU Tandon School of Engineering, Prof. Christopher Musco

STREAMING ALGORITHMS

Abstract architecture of a streaming algorithm:

- Given a dataset $D = d_1, \ldots, d_n$ with n pieces of data, we want to output $f(D)$ for some function f.
- Maintain state S_t with $\ll |D|$ space at each time step t.
- **Update phase:** Receive $d_1, \ldots, d_n$ in sequence, update $S_t \leftarrow U(S_{t-1}, d_t)$.
- **Process phase:** Using S_n, compute approximation to $f(D)$.

Typical setup for training models in machine learning, required for large scale data monitoring (e.g. processing sensor data, time series, seismic data, satellite imagery, etc.)

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

Output: Number of distinct inputs, D.

Example: $f(1, 10, 10, 4, 9, 1, 1, 4) \rightarrow 4$

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

Output: Number of distinct inputs, D.

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

Naive solution takes $O(D)$ space and is exact. We want something that uses much less space, but is approximate.

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

Output: Number of distinct inputs, D.

Example: $f(1, 10, 10, 4, 9, 1, 1, 4) \rightarrow 4$

Naive solution takes $O(D)$ space and is exact. We want something that uses much less space, but is approximate.

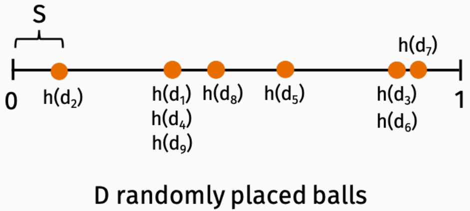
In practice: Approximate COUNT(DISTINCT) in huge databases (of weblogs, biological data, etc.).

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

Output: Number of distinct inputs.

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

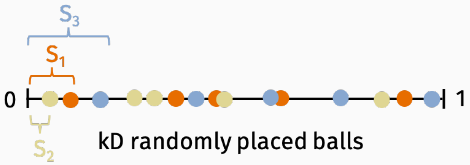
Basic estimator:



$\mathbb{E}[S] = \frac{1}{D+1}$. Estimate $D \approx \frac{1}{S} - 1$.

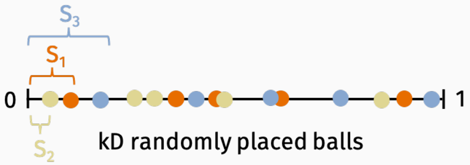
REPEAT TO BOOST ACCURACY

- Choose k random hash function $h_1, \ldots, h_k : \mathcal{U} \to [0,1]$.
- Maintain k estimators $S_1, \ldots, S_k$.

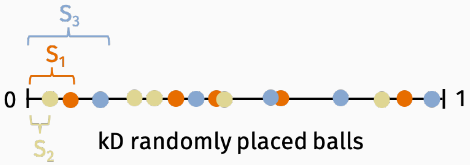


REPEAT TO BOOST ACCURACY

- Choose k random hash function $h_1, \dots, h_k : \mathcal{U} \to [0, 1]$.
- Maintain k estimators $S_1, \dots, S_k$.
- **Set:** $S' = \frac{1}{k} \sum_{i=1}^{k} S_k$.
- **Estimate:** $D \approx \frac{1}{S'} - 1$.



- Choose k random hash function $h_1, \ldots, h_k : \mathcal{U} \to [0, 1]$.
- Maintain k estimators $S_1, \ldots, S_k$.
- **Set:** $S' = \frac{1}{k}\sum_{i=1}^{k} S_k$.
- **Estimate:** $D \approx \frac{1}{S'} - 1$.



- Choose k random hash function $h_1, \ldots, h_k : \mathcal{U} \to [0,1]$.
- Maintain k estimators $S_1, \ldots, S_k$.
- **Set:** $S' = \frac{1}{k}\sum_{i=1}^{k} S_{\cancel{p}\, i}$.
- **Estimate:** $D \approx \frac{1}{S'} - 1$.

$$\mathrm{Var}\left[\frac{1}{k}\sum_{i=1}^{k} S_i\right] = \frac{1}{k^2}\mathrm{Var}\left[\sum_{i=1}^{k} S_i\right] = \frac{1}{k^2}\underbrace{\sum_{i=1}^{k}\mathrm{Var}[S_i]}_{k\cdot\sigma^2} = \frac{1}{k}\sigma^2$$

σ^2/k

If $\mathrm{Var}[S] = \sigma^2$, $\mathrm{Var}[S'] = ?$

$1/k \sum_{i=1}^{k} S_i \to$ each S_i has variance σ^2.

Applying Chebyshev's inequality: Need $O(1/\epsilon^2)$ estimators to return $\tilde{D}$ satisfying:

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

with probability 9/10.

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

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

Durand

Real Flajolet-~~Martin~~ / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	1011000

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

Real Flajolet-Martin / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	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.

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	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?

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	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?

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

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	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?

$$\Pr(\mathbf{h}(x_i) \text{ has } x \text{ trailing zeros}) = \frac{1}{2^x}$$

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	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?

$$\Pr(\mathbf{h}(x_i) \text{ has } x \text{ trailing zeros}) = \frac{1}{2^x}$$

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	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?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log D \text{ trailing zeros}) = \frac{1}{2^{\log D}}$$

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
⋮	
$h(x_n)$	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?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log D \text{ trailing zeros}) = \frac{1}{2^{\log D}} = \frac{1}{D}.$$

Flajolet-Durand / HyperLogLog:

$h(x_1)$	1010010
$h(x_2)$	1001100
$h(x_3)$	1001110
$\vdots$	
$h(x_n)$	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?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log D \text{ trailing zeros}) = \frac{1}{2^{\log D}} = \frac{1}{D}.$$

So with D distinct hashes, expect to see 1 with $\log D$ trailing zeros. Expect $\mathbf{m} \approx \log D$.

Flajolet-Durand / HyperLogLog:

h(x_1)	**1010010**
h(x_2)	**1001100**
h(x_3)	**1001110**
⋮	
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?

$$\Pr(\mathbf{h}(x_i) \text{ has } \log D \text{ trailing zeros}) = \frac{1}{2^{\log D}} = \frac{1}{D}.$$

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.

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.

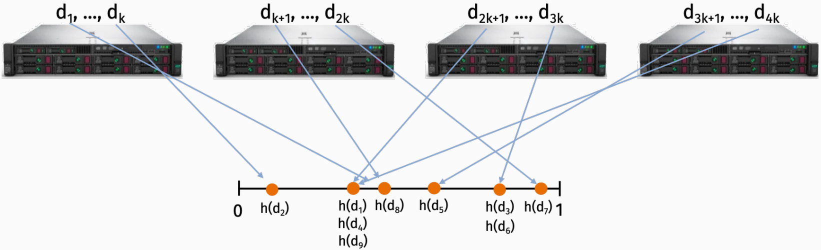
Total Space: $O\left(\frac{\log\log D}{\epsilon^2} + \log D\right)$ for an ϵ approximate count.

"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



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).

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 dataset $D = d_1, \ldots, d_n$ with n pieces of data, we want to output $f(D)$ for some function f.

Abstract architecture of a sketching algorithm:

- Given a dataset $D = d_1, \ldots, d_n$ with n pieces of data, we want to output $f(D)$ for some function f.
- **Sketch phase:** For each $i \in 1, \ldots, n$, compute $s_i = C(d_i)$, where C is some compression function and $|s_i| \ll d_i$.

Abstract architecture of a sketching algorithm:

- Given a dataset $D = d_1, \ldots, d_n$ with n pieces of data, we want to output $f(D)$ for some function f.
- **Sketch phase:** For each $i \in 1, \ldots, n$, compute $s_i = C(d_i)$, where C is some compression function and $|s_i| \ll d_i$.
- **Process phase:** Using (lower dimensional) dataset $s_1, \ldots, s_n$, compute an approximation to $f(D)$.

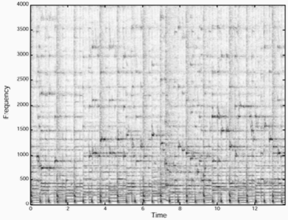
Abstract architecture of a sketching algorithm:

- Given a dataset $D = d_1, \ldots, d_n$ with n pieces of data, we want to output $f(D)$ for some function f.
- **Sketch phase:** For each $i \in 1, \ldots, n$, compute $s_i = C(d_i)$, where C is some compression function and $|s_i| \ll d_i$.
- **Process phase:** Using (lower dimensional) dataset $s_1, \ldots, s_n$, compute an approximation to $f(D)$.

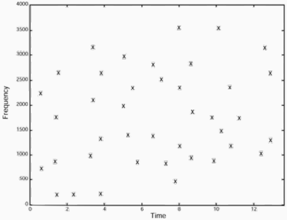
Sketching phase is easily distributed, parallelized, etc. Better space complexity, communication complexity, runtime, all at once.

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

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.

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.

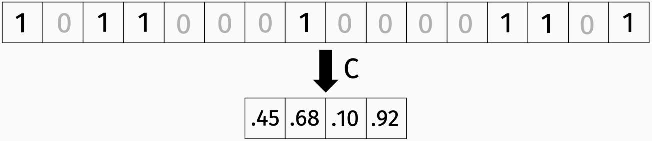
Challenges:

- 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.

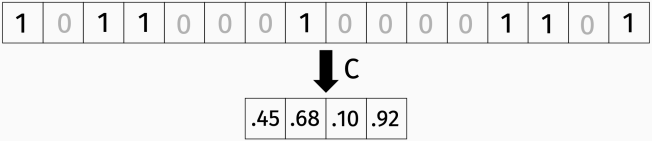
Goal: Design a more compact sketch for comparing $\mathbf{q}, \mathbf{y} \in \{0,1\}^d$. Ideally $\ll d$ space/time complexity.

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



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)

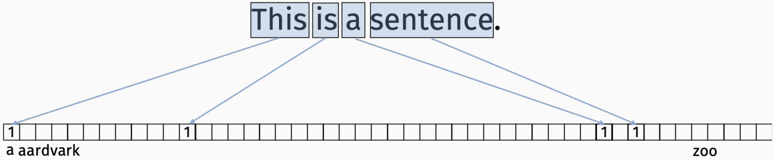
$$J(\mathbf{q}, \mathbf{y}) = \frac{|\mathbf{q} \cap \mathbf{y}|}{|\mathbf{q} \cup \mathbf{y}|} = \frac{\text{\# of non-zero entries in common}}{\text{total \# of non-zero entries}}$$

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

Other applications:

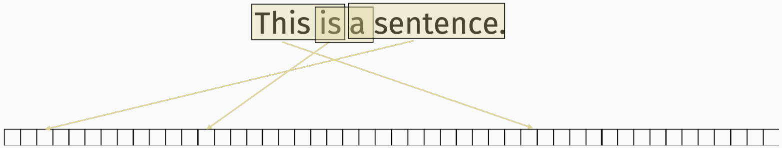
- Change detection in documents (high speed web caches).
- Analyzing seismic data (matching signatures of earthquakes).
- User recommendations on social networking sites.

"Bag-of-words" model:



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

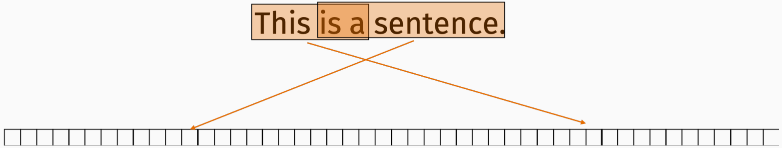
"Bag-of-words" model:



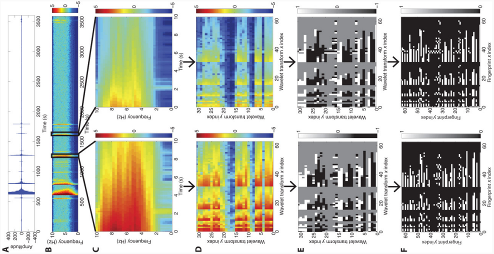
How many **bigrams** do a pair of documents have in common?

JACCARD SIMILARITY FOR DOCUMENT COMPARISON

"Bag-of-words" model:

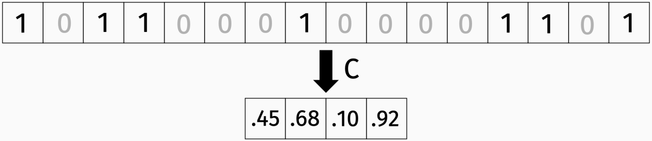


How many **trigrams** do a pair of documents have in common?



Feature extract pipeline for earthquake data.

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



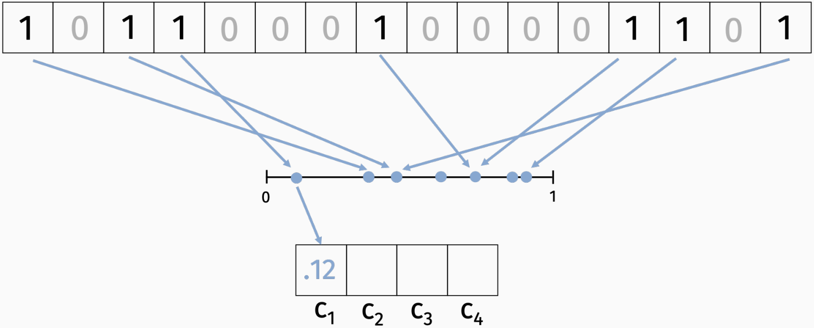
Homomorphic Compression: Want to use $C(\mathbf{q}), C(\mathbf{y})$ to approximately compute the Jaccard similarity $J(\mathbf{q}, \mathbf{y})$.

MinHash (Broder, '97):

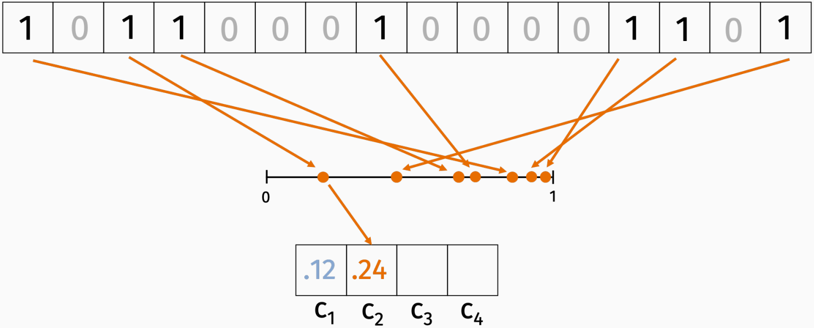
- Choose k random hash functions $h_1, \ldots, h_k : \{1, \ldots, n\} \to [0, 1]$.
- For $i \in 1, \ldots, k$, let $c_i = \min_{j, \mathbf{q}_j = 1} h_i(j)$.
- $C(\mathbf{q}) = [c_1, \ldots, c_k]$.

MinHash (Broder, '97):

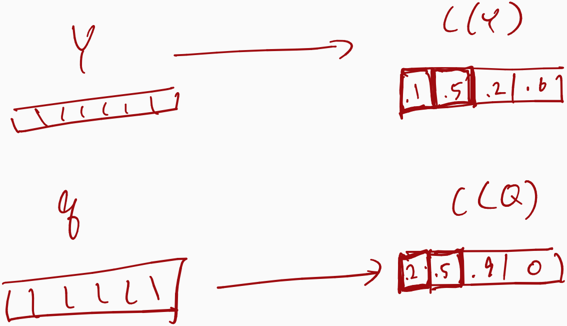
- Choose k random hash functions $h_1, \ldots, h_k : \{1, \ldots, n\} \to [0, 1]$.
- For $i \in 1, \ldots, k$, let $c_i = \min_{j, \mathbf{q}_j = 1} h_i(j)$.
- $C(\mathbf{q}) = [c_1, \ldots, c_k]$.



- Choose k random hash functions $h_1, \ldots, h_k : \{1, \ldots, n\} \rightarrow [0, 1]$.
- For $i \in 1, \ldots, k$, let $c_i = \min_{j, \mathbf{q}_j = 1} h_i(j)$.
- $C(\mathbf{q}) = [c_1, \ldots, c_k]$.



Claim: $\Pr[c_i(q) = c_i(y)] = J(q, y)$.



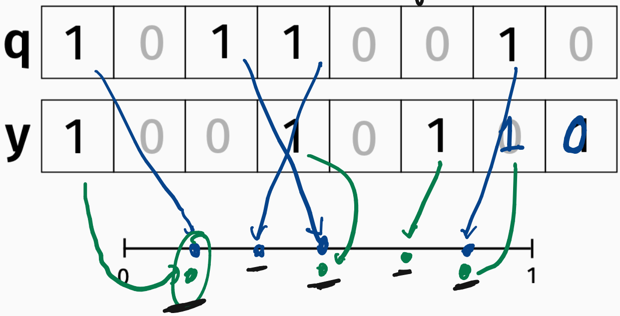
$J(q,y) = 1$

$\Pr[c_i(q) = c_i(y)]$

$= 1$

min min

Claim: $\Pr[c_i(q) = c_i(y)] = J(q, y)$. $= \dfrac{q \cap y}{q \cup y}$



Total # of distinct real values: $q \cup y$

total # of "repeat" positions: $q \cap y$

$i=1$

Return: $\boxed{\tilde{J} = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}[c_i(\mathbf{q}) = c_i(\mathbf{y})].}$ $i=3$

Unbiased estimate for Jaccard similarity: $\boxed{\mathbb{E}\tilde{J} = J(\mathbf{q}, \mathbf{y}).}$

C(q)	.12	.24	.76	.35	C(y)	.12	.98	.76	.11

1 2 . . . . k

Chernoff bound: Analysis is the same as summing random coin flips. As long as $k = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, then with prob. $\boxed{1 - \delta}$,

$$J(\mathbf{q}, \mathbf{y}) - \epsilon \leq \tilde{J}(C(\mathbf{q}), C(\mathbf{y})) \leq J(\mathbf{q}, \mathbf{y}) + \epsilon.$$

Pr this does not happen

$\leq \delta$

Chernoff bound: Analysis is the same as summing random coin flips. As long as $\underbrace{k = O\left(\frac{\log(1/\cancel{\delta}\,\Delta)}{\epsilon^2}\right)}$, then with prob. $1 - \cancel{\delta}$, Δ

$$J(\mathbf{q}, \mathbf{y}) - \epsilon \leq \tilde{J}(C(\mathbf{q}), C(\mathbf{y})) \leq J(\mathbf{q}, \mathbf{y}) + \epsilon.$$

And $\tilde{J}$ only takes $O(k)$ time to compute! Independent of original fingerprint dimension d.

$P[A \text{ or } B] \leq \underline{\underline{\Pr\{A\}}} + \underline{\underline{\Pr[B]}}$

$\tilde{J} = \frac{1}{k}\sum_{i=1}^{k} \mathbb{1}[c_i(\mathbf{q}) = c_i(\mathbf{y})]$

Suffices to prove:

1) • $\underline{\tilde{J} \leq J + \epsilon}$ with probability $(1 - \cancel{\delta}\,\Delta/2)$.

2) • $\underline{\tilde{J} \geq J - \epsilon}$ with probability $(1 - \cancel{\delta}\,\Delta/2)$.

A

$\Pr[\tilde{J} > J + \epsilon] \leq \Delta/2$

$\Pr[\tilde{J} < J - \epsilon] \leq \Delta/2$

B

Pr[neither A nor B] happens with prob > $1 - \Delta$

$\boxed{\Pr[A \text{ or } B] \leq \Delta}$

Theorem (Chernoff Bound, 1)

Let $X_1, X_2, \ldots, X_k$ be independent $\{0,1\}$-valued random variables and let $p_i = \mathbb{E}[X_i]$, where $0 < p_i < 1$. Then the sum $X = \sum_{i=1}^k X_i$, which has mean $\mu = \sum_{i=1}^k p_i$, satisfies

Want to prove:

$\Pr[\tilde{J} \geq J + \epsilon] \leq \frac{\Delta}{2}$ if $k = O\left(\frac{\log(1/\Delta)}{\epsilon^2}\right)$

$$\Pr[X \geq (1+\delta)\mu] \leq e^{\frac{-\delta^2\mu}{3+3\delta}}.$$

$\tilde{J} = \frac{1}{k}\sum_{i=1}^{k} \underbrace{\mathbb{1}[c_i(q) = c_i(y)]}_{\{0,1\}\text{ random variable that's one w/ prob. } J} \longrightarrow$ call this X_i, so, $\tilde{J} = \frac{1}{k}\underbrace{\sum_{i=1}^{k} X_i}_{X}$ $\quad \tilde{J} = \frac{1}{k}X$.

1) $\mu = \sum_{i=1}^{k} p_i = kJ$

2) $\Pr[X \geq (1+\delta)kJ] \leq e^{-\delta^2 kJ/3+3\delta}$

3) $\Pr[\tilde{J} \geq (1+\delta)J] \leq e^{-\delta^2 kJ/3+3\delta}$

4) Set $\delta = \epsilon/J$. $\rightarrow$ plug in

5) $\Pr[\tilde{J} \geq J+\epsilon] \leq e^{-(\epsilon^2 k/J)/3+3\epsilon/J}$.

6) $\Pr[\tilde{J} \geq J+\epsilon] \leq e^{-\epsilon^2 k/3J+3\epsilon} \leq e^{-\epsilon^2 k/6}$

since $J, \epsilon < 1$

7) Set $k = \frac{12\log(1/\Delta)}{\epsilon^2} = O\left(\frac{\log 1/\Delta}{\epsilon^2}\right)$

✓ 8) $\Pr[\tilde{J} \geq J+\epsilon] \leq e^{-2\log(1/\Delta)} = \Delta^2 \leq \frac{\Delta}{2}$ for $\Delta \leq \frac{1}{2}$

Theorem (Chernoff Bound, 2)

Let $X_1, X_2, \ldots, X_k$ be independent $\{0,1\}$-valued random variables and let $p_i = \mathbb{E}[X_i]$, where $0 < p_i < 1$. Then the sum $\cancel{S}$ X $= \sum_{i=1}^{k} X_i$, which has mean $\mu = \sum_{i=1}^{k} p_i$, satisfies

Want to prove:

$\Pr[\tilde{J} \leq J - \varepsilon] \leq \frac{\Delta}{2}$ if $k = O\left(\frac{\log 1/\Delta}{\varepsilon^2}\right)$.

$$\Pr[X \leq (1-\delta)\mu] \leq e^{\frac{-\delta^2\mu}{3}}.$$

→ NOTE This is trivially true when $J \leq \varepsilon$! $\Pr[\tilde{J} \leq \text{negative } \#] = 0$ always positive.

for all $0 < \delta < 1$.

So only left to prove for $J > \varepsilon$.

Same set up as previous page.

1) $\mu = kJ$.

2) $\Pr[X \leq (1-\delta)kJ] \leq e^{-\delta^2 kJ/3}$

when $J > \varepsilon$, $\varepsilon/J < 1$, so we're okay!

3) $\Pr[\tilde{J} \leq J - \delta J] \leq e^{-\delta^2 kJ/3}$

4) Set $\delta = \varepsilon/J$. Theorem only applies when $\delta < 1$!

5) $\Pr[\tilde{J} \leq J - \varepsilon J] \leq e^{-\varepsilon^2 k/3J} \leq e^{-\varepsilon^2 k/3}$

6) Set $k = \frac{12 \log(1/\Delta)}{\varepsilon^2}$ as before.

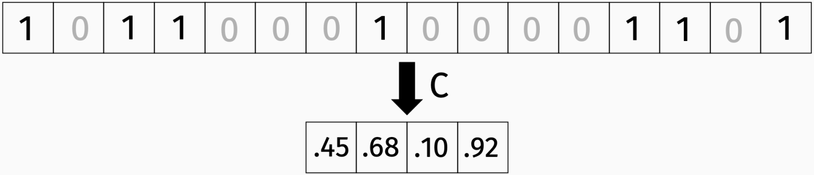
7) $\Pr[\tilde{J} \leq J - \varepsilon J] \leq e^{-4 \log(1/\Delta)} = \Delta^4$

8) $\Pr[\tilde{J} < J - \varepsilon J] \leq \frac{\Delta}{2}$ for $\Delta < 1/2$.

✓ $\leq \Delta/2$

One incredibly powerful theorem:

The Johnson-Lindenstrauss Lemma.


1 0 1 1 0 0 0 1 0 0 0 0 1 1 0 1
C
.45 .68 .10 .92

.62	.93	.00	.11	.31	.45	.21	.17	.12	.89	.88	.50	.42	.86	.34	.71

$$\Big\downarrow C$$

.45	.68	.10	.92

1															d
.62	.93	.00	.11	.31	.45	.21	.17	.12	.89	.88	.50	.42	.86	.34	.71

$\Downarrow$ C

B

.45	.68	.10	.92

Euclidean norm / distance:

- Given $\mathbf{q} \in \mathbb{R}^d$, $\|\mathbf{q}\|_2 = \sqrt{\sum_{i=1}^{d} q(i)^2}$.
- Given $\mathbf{q}, \mathbf{y} \in \mathbb{R}^d$, distance defined as $\|\mathbf{q} - \mathbf{y}\|_2$.

.62	.93	.00	.11	.31	.45	.21	.17	.12	.89	.88	.50	.42	.86	.34	.71

$$\Big\downarrow \mathbf{C}$$

.45	.68	.10	.92

Euclidean norm / distance:

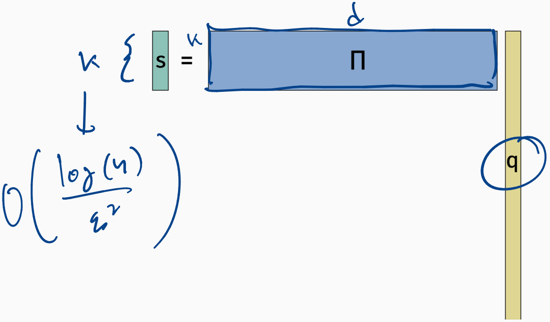
- Given $\mathbf{q} \in \mathbb{R}^d$, $\|\mathbf{q}\|_2 = \sqrt{\sum_{i=1}^d q(i)^2}$.
- Given $\mathbf{q}, \mathbf{y} \in \mathbb{R}^d$, distance defined as $\|\mathbf{q} - \mathbf{y}\|_2$.

Can we find compact sketches that preserve Euclidean distance, just as we did for Jaccard similarity?

Lemma (Johnson-Lindenstrauss, 1984) JL

For any set of n data points $\mathbf{q}_1, \ldots, \mathbf{q}_n \in \mathbb{R}^d$ *there exists a linear map* $\mathbf{\Pi} : \mathbb{R}^d \to \mathbb{R}^k$ *where* $k = O\left(\frac{\log n}{\epsilon^2}\right)$ *such that for all* i, j,

$$(1-\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2 \leq \|\mathbf{\Pi}\mathbf{q}_i - \mathbf{\Pi}\mathbf{q}_j\|_2 \leq (1+\epsilon)\|\mathbf{q}_i - \mathbf{q}_j\|_2.$$



Remarkably, Π can be chosen <u>completely at random</u>!

One possible construction: Random Gaussian.

$$\mathbf{\Pi}_{\underline{i,j}} = \frac{1}{\sqrt{k}}\underline{\mathcal{N}(0,1)} \qquad \frac{1}{\sqrt{n}}$$

Remarkably, Π can be chosen completely at random!

One possible construction: Random Gaussian.

$$\mathbf{\Pi}_{i,j} = \frac{1}{\sqrt{k}}\mathcal{N}(0,1)$$

The map $\mathbf{\Pi}$ is **oblivious to the data set**. This stands in constrast to e.g. PCA, amoungst other differences.

Remarkably, Π can be chosen completely at random!

One possible construction: Random Gaussian.

$$\mathbf{\Pi}_{i,j} = \frac{1}{\sqrt{k}}\mathcal{N}(0,1)$$

The map $\mathbf{\Pi}$ is **oblivious to the data set**. This stands in constrast to e.g. PCA, amoungst other differences.

[Indyk, Motwani 1998] [Arriage, Vempala 1999] [Achlioptas 2001] [Dasgupta, Gupta 2003].

Many other possible choices suffice – you can use random $\{+1,-1\}$ variables, sparse random matrices, pseudorandom $\mathbf{\Pi}$. Each with different advantages. We should have time to discuss a few examples next lecture.

Intermediate result: (which we already know how to prove)

Lemma (Distributional JL Lemma)

Let $\mathbf{\Pi} \in \mathbb{R}^{k\times d}$ be chosen so that each entry equals $\frac{1}{\sqrt{k}}\mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ denotes a standard Gaussian random variable.

If we choose $k = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, then for <u>any vector $\mathbf{q}$</u>, with probability $(1-\delta)$:

$$(1-\epsilon)\|\mathbf{q}\|_2 \leq \|\mathbf{\Pi}\mathbf{q}\|_2 \leq (1+\epsilon)\|\mathbf{q}\|_2$$

Intermediate result: (which we already know how to prove)

Lemma (Distributional JL Lemma)

Let $\mathbf{\Pi} \in \mathbb{R}^{k \times d}$ *be chosen so that each entry equals* $\frac{1}{\sqrt{k}}\mathcal{N}(0,1)$, *where* $\mathcal{N}(0,1)$ *denotes a standard Gaussian random variable.*

If we choose $k = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, *then for any vector* $\mathbf{q}$, *with probability* $(1-\delta)$:

$q_i - q_j$ for any q

$$(1-\epsilon)\|\mathbf{q}\|_2^2 \leq \|\mathbf{\Pi q}\|_2^2 \leq (1+\epsilon)\|\mathbf{q}\|_2^2$$

$\|\Pi(q_i - q_j)\|_2 = \|\Pi q_i - \Pi q_j\|_2$

In class exercise: Given this lemma, prove the Johnson-Lindenstrauss lemma.

Implies: $\rightarrow$ If $K = O(\log n / \varepsilon^2)$ then with prob. $9/10$

$(1-\varepsilon)\|q_i - q_j\|_2 \leq \|\Pi q_i - \Pi q_j\|_2 \leq (1+\varepsilon)\|q_i - q_j\|_2$ for all $i,j \in \{1,\dots,n\}$

For any i,j if $k = O\left(\frac{\log(1/\delta)}{\varepsilon^2}\right)$

then w/ probability $(1-\delta)$ $\boxed{S_{ij}}$

$$(1-\varepsilon)\|q_i - q_j\|_2 \le \|\Pi q_i - \Pi q_j\|_2 \le (1+\varepsilon)\|q_i - q_j\|_2$$

$q_1 \dots q_n$ $\binom{n}{2}$ i,j pairs $\to O(n^2)$

$\delta = \log\left(\frac{1}{10n^2}\right)$. $\Pr(S_{ij}) \ge 1 - \frac{1}{10n^2}$ for all i,j

$\Pr(S_{12}$ and S_{13} and S_{14} $\dots)$

$$\ge 1 - \frac{\binom{n}{2}}{10n^2}$$

$$\ge 1 - \frac{1}{10} = 9/10$$

$k = \frac{\log(10n^2)}{\varepsilon^2} \to O(\log n)$ $\binom{n}{2}$ ✓

IN CLASS EXERCISE

Want to argue that with high probability, $\|\mathbf{\Pi q}\|_2 = (1 \pm \epsilon)\|\mathbf{q}\|_2$.
It suffices to prove that, with probability $(1 - \delta)$,

$$(1 - \epsilon)\|\mathbf{q}\|_2^2 \leq |\mathbf{\Pi q}\|_2^2 \leq (1 + \epsilon)\|\mathbf{q}\|_2^2$$

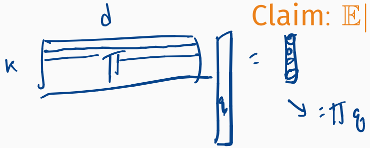
Want to argue that with high probability, $\|\mathbf{\Pi q}\|_2 = (1 \pm \epsilon)\|\mathbf{q}\|_2$.
It suffices to prove that, with probability $(1 - \delta)$,

$$(1-\epsilon)\|\mathbf{q}\|_2^2 \leq |\mathbf{\Pi q}\|_2^2 \leq (1+\epsilon)\|\mathbf{q}\|_2^2$$

Claim: $\mathbb{E}\|\mathbf{\Pi q}\|_2^2 = \|\mathbf{q}\|_2^2$.

Want to argue that with high probability, $\|\mathbf{\Pi q}\|_2 = (1 \pm \epsilon)\|\mathbf{q}\|_2$.
It suffices to prove that, with probability $(1 - \delta)$,

$$(1-\epsilon)\|\mathbf{q}\|_2^2 \leq \|\mathbf{\Pi q}\|_2^2 \leq (1+\epsilon)\|\mathbf{q}\|_2^2$$



Claim: $\mathbb{E}\|\mathbf{\Pi q}\|_2^2 = \|\mathbf{q}\|_2^2$.

$\mathbb{E}\left[\|\Pi q\|_2^2\right] = \|q\|_2^2$

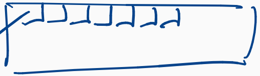
$$\|\mathbf{\Pi q}\|_2^2 = \sum_{i=1}^{k} [\mathbf{\Pi q}]_i^2$$

$\mathbb{E}[y^2] = k\,\mathbb{E}[[\Pi q]_i^2]$
$= \|q\|_2^2$

$$\mathbb{E}\|\mathbf{\Pi q}\|_2^2 = \underbrace{\sum_{i=1}^{k} \mathbb{E}\left([\mathbf{\Pi q}]_i^2\right)}_{k\,\mathbb{E}[[\Pi q]_i^2]} = \|q\|_2^2$$

PROOF OF DISTRIBUTIONAL JL

Goal: Let $Y = \sqrt{k} \cdot [\Pi q]_i$. $\boxed{\text{Evaluate } \mathbb{E}[Y^2]}$



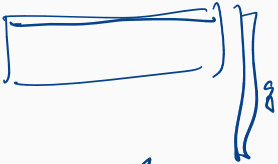
$$\left[\tfrac{1}{\sqrt{k}} X_1, \dots \tfrac{1}{\sqrt{k}} X_d\right]$$

$$Y = \sqrt{k} \sum_{j=1}^{d} \frac{X_j}{\sqrt{k}} \cdot q_j$$

$$1/\sqrt{k}\, N(0,1)$$

$$X_1, \dots X_d \sim N(0,1)$$

$$Y = \sum_{j=1}^{d} X_j \cdot q_j$$



$$\mathbb{E}[Y] = \sum_{j=1}^{d} q_j \underbrace{\mathbb{E}[X_j]}_{\to 0} = Y$$

$$\mathrm{Var}[Y] = \mathbb{E}[Y^2] - \cancel{\mathbb{E}[Y]^2}\; 0$$

$$\mathrm{Var}[Y] = \sum_{j=1}^{d} \mathrm{Var}[X_j \cdot q_j] = \sum_{j=1}^{d} q_j^2 \underbrace{\mathrm{Var}[X_j]}_{\to 1} = \sum_{j=1}^{d} q_j^2 = \|q\|_2^2$$

What type of random variable is $\boxed{[\mathbf{\Pi q}]_i^2?}$ $= Y^2$

What type of random variable is $[\mathbf{\Pi q}]_i^2$?

Fact (Stability of Gaussian random variables)

$$\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

$$O(1) \cdot \underbrace{\sum_{i=1}^{d} q_i X_i}_{N(0, \|q\|_2^2)} \quad X_i \to N(0,1)$$

What type of random variable is $[\mathbf{\Pi q}]_i^2$?

Fact (Stability of Gaussian random variables)

$$\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

$[\mathbf{\Pi q}]_i^2$ is the square of a Gaussian random variable and $\|\mathbf{\Pi x}\|_2^2$ is a sum of k squared Gaussian random variables.

What type of random variable is $[\mathbf{\Pi q}]_i^2$?

Fact (Stability of Gaussian random variables)

$$\mathcal{N}(\mu_1, \sigma_1^2) + \mathcal{N}(\mu_2, \sigma_2^2) = \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$$

$[\mathbf{\Pi q}]_i^2$ is the square of a Gaussian random variable and $\|\mathbf{\Pi x}\|_2^2$ is a sum of k squared Gaussian random variables.

"Chi-squared random variable with k degrees of freedom."

Lemma

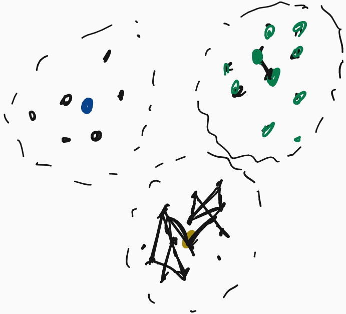
Let X be a chi-squared random variable with k degrees of freedom.

$$\Pr[|\mathbb{E}X - X| \geq \epsilon\mathbb{E}X] \leq 2e^{-k\epsilon^2/8}$$

k-means clustering: Give data points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, find centers $\mu_1, \ldots, \mu_k$ to minimize:

k

$$Cost(\mu_1, \ldots, \mu_k) = \sum_{i=1}^{n} \min_{j=1,\ldots,k} \|\mu_j - \mathbf{x}_i\|_2^2$$

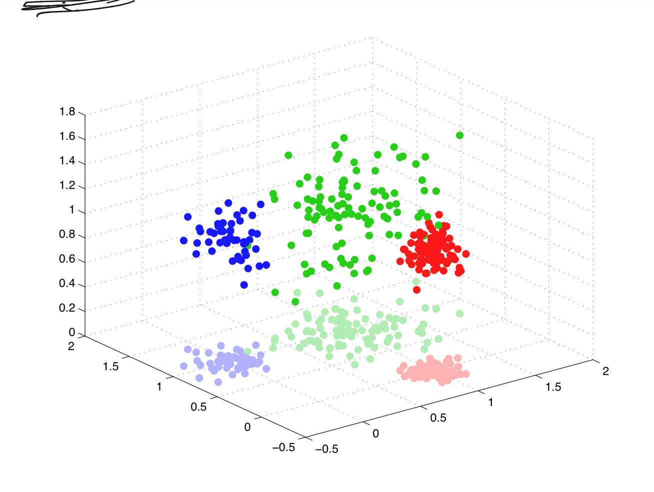


Equivalent formulation: Find clusters $C_1, \ldots, C_k \subseteq \{1, \ldots, n\}$ to minimize:

$X_1, \ldots, X_n$

$$Cost(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|\mathbf{x}_u - \mathbf{x}_v\|_2^2.$$

Approximation algorithm: Find optimal clusters $\tilde{C}_1, \ldots, \tilde{C}_k$ for the $k = O(\frac{\log n}{\epsilon^2})$ dimension data set $\mathbf{\Pi X}_1, \ldots, \mathbf{\Pi X}_n$.



$$Cost(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \|\mathbf{x}_u - \mathbf{x}_v\|_2^2.$$

$$\widetilde{Cost}(C_1, \ldots, C_k) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{u,v \in C_j} \underline{\|\Pi x_u - \Pi x_v\|_2^2}$$

$$(1-\epsilon)\|x_u - x_v\|_2^2 \leq \quad \Downarrow \quad \leq (1+\epsilon)\|x_u - x_v\|_2^2$$

$$(1-\epsilon)\, Cost(C_1, \ldots C_k) \leq \widetilde{Cost}(C_1, \ldots C_k) \leq (1+\epsilon)\cdot Cost(C_1, \ldots, C_k)$$