Dimensionality reduction

Outline

- Dimensionality Reductions or data projections
- Random projections
- Singular Value Decomposition and Principal Component Analysis (PCA)

The curse of dimensionality

- The efficiency of many algorithms depends on the number of dimensions **d**
  - Distance/similarity computations are at least linear to the number of dimensions
  - Index structures fail as the dimensionality of the data increases

Goals

- Reduce dimensionality of the data
- Maintain the meaningfulness of the data

Dimensionality reduction

- Dataset **X** consisting of **n** points in a **d**–dimensional space
- Data point $\mathbf{x_i \epsilon R^d}$ (**d**–dimensional real vector):

  $\mathbf{x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]}$
- Dimensionality reduction methods:
  - **Feature selection:** choose a subset of the features
  - **Feature extraction:** create new features by combining new ones

Dimensionality reduction

- Dimensionality reduction methods:
  - **Feature selection:** choose a subset of the features
  - **Feature extraction:** create new features by combining new ones
- Both methods map vector $x_i \epsilon R^d$, to vector $y_i \epsilon R^k$, $(k<<d)$

- $F : R^d \rightarrow R^k$

Linear dimensionality reduction

- Function **F** is a **linear** projection
- $y_i = x_i A$
- $Y = X A$
- **Goal:** Y is as **close** to X as possible

Closeness: Pairwise distances

- **Johnson-Lindenstrauss lemma:** Given $\varepsilon>0$, and an integer n, let k be a positive integer such that $k \geq k_0 = O(\varepsilon^{-2} \log n)$. For every set X of n points in R^d there exists $F: R^d \rightarrow R^k$ such that for all $x_i, x_j \in X$

$$(1-\varepsilon)||x_i - x_j||^2 \leq ||F(x_i) - F(x_j)||^2 \leq (1+\varepsilon)||x_i - x_j||^2$$

What is the intuitive interpretation of this statement?

JL Lemma: Intuition

- Vectors $x_i \epsilon R^d$, are projected onto a k–dimensional space ($k << d$): $y_i = x_i A$
- If $||x_i||=1$ for all i, then,

  $||x_i - x_j||^2$ is approximated by $(d/k)||y_i - y_j||^2$
- **Intuition:**
  - The expected squared norm of a projection of a unit vector onto a random subspace through the origin is k/d
  - The probability that it deviates from expectation is very small

Finding random projections

- Vectors $x_i \epsilon R^d$, are projected onto a k-dimensional space ($k << d$)
- Random projections can be represented by linear transformation matrix A
- $y_i = x_i A$

- What is the matrix A?

Finding random projections

- Vectors $x_i \epsilon R^d$, are projected onto a k–dimensional space ($k \ll d$)
- Random projections can be represented by linear transformation matrix A
- $y_i = x_i A$

- What is the matrix A?

Finding matrix A

- Elements A(i,j) can be Gaussian distributed
- Achlioptas* has shown that the Gaussian distribution can be replaced by

$$A(i,j) = \begin{cases} +1 \text{ with prob } \frac{1}{6} \\ 0 \text{ with prob } \frac{2}{3} \\ -1 \text{ with prob } \frac{1}{6} \end{cases}$$

- All zero mean, unit variance distributions for A(i,j) would give a mapping that satisfies the **JL** lemma

- **Why is Achlioptas result useful?**

Datasets in the form of matrices

Given **n** objects and **d** features describing the objects. (Each object has **d** numeric values describing it.)

Dataset

An **n-by-d** matrix **A**, A_{ij} shows the "**importance**" of feature **j** for object **i**.

Every row of **A** represents an object.

Goal

1. **Understand** the structure of the data, e.g., the underlying process generating the data.
2. **Reduce the number of features** representing the data

Market basket matrices

$$\begin{array}{cc} & \textbf{d}\text{ products} \\ & \text{(e.g., milk, bread, wine, etc.)} \\ \begin{array}{l}\textbf{n}\\\text{customers}\end{array} & \left(\begin{array}{c} \\ A \\ \\ \end{array}\right) \end{array}$$

A_{ij} = quantity of j-th product purchased by the i-th customer

Find a subset of the products that characterize customer behavior

Social-network matrices

d groups
(e.g., BU group, opera, etc.)

n users

$$\begin{pmatrix} \\ \\ A \\ \\ \end{pmatrix}$$

A_{ij} = partiticipation of the **i**-th user in the **j**-th group

Find a subset of the groups that accurately clusters social-network users

Document matrices

d terms
(e.g., theorem, proof, etc.)

n
documents

$$\begin{pmatrix} \\ & A & \\ \\ \end{pmatrix}$$

A_{ij} = frequency of the **j**-th term in the **i**-th document

Find a subset of the terms that accurately clusters the documents

Recommendation systems

$$\begin{array}{cc} & d \text{ products} \\ n \text{ customers} & \begin{pmatrix} \\ \\ A \\ \\ \end{pmatrix} \end{array}$$

A_{ij} = frequency of the j-th product is bought by the i-th customer

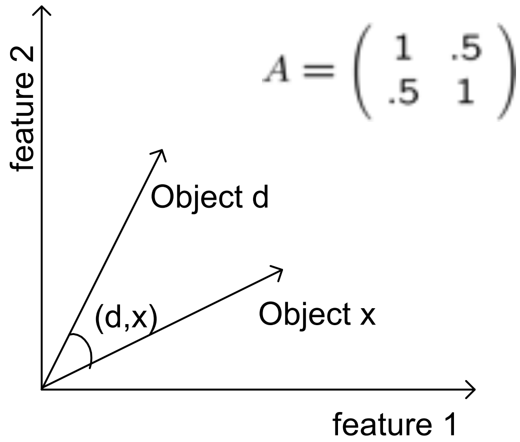
Find a subset of the products that accurately describe the behavior or the customers

The Singular Value Decomposition (SVD)

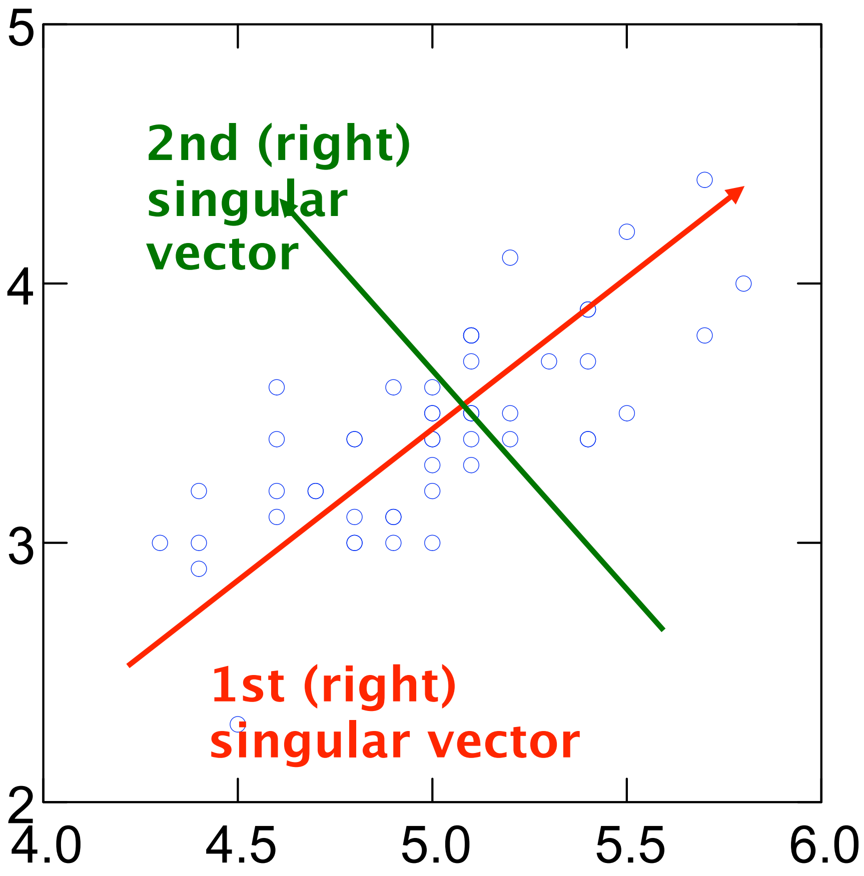
Data matrices have **n** rows (one for each object) and **d** columns (one for each feature).

<ins>Rows:</ins> vectors in a Euclidean space,

Two objects are "**close**" if the angle between their corresponding vectors is small.



SVD: Example



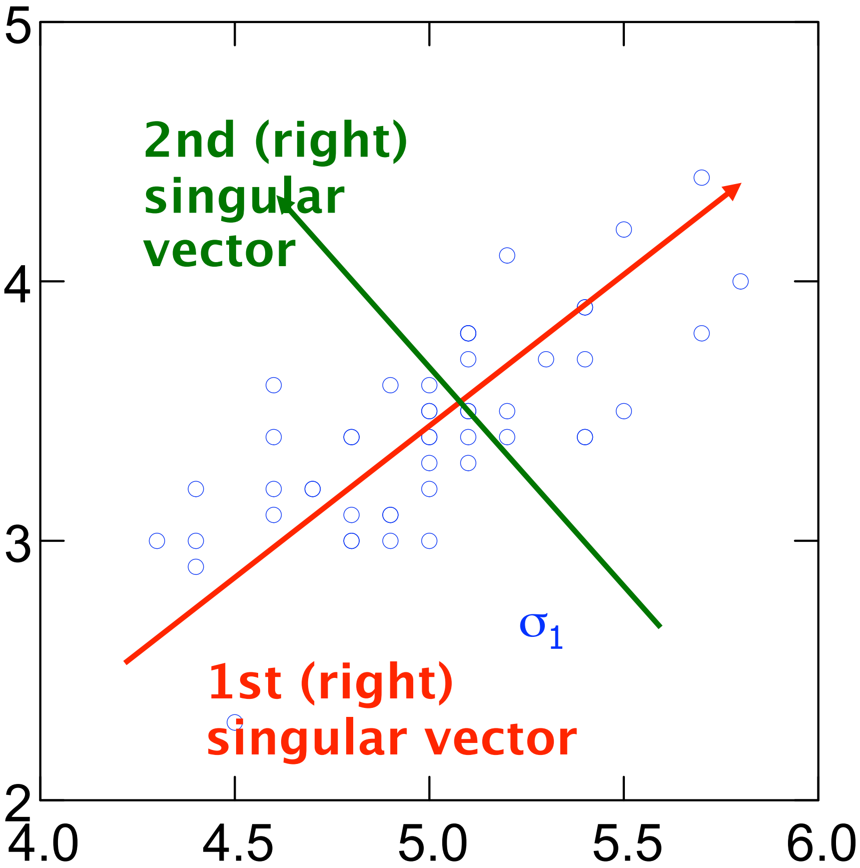
Input: 2-d dimensional points

Output:

<u>1st (right) singular vector:</u> direction of maximal variance,

<u>2nd (right) singular vector:</u> direction of maximal variance, after removing the projection of the data along the first singular vector.

Singular values



σ_1: measures how much of the data variance is explained by the first singular vector.

σ_2: measures how much of the data variance is explained by the second singular vector.

SVD decomposition

$$\underbrace{\begin{pmatrix} \quad A \quad \end{pmatrix}}_{n \times d} = \underbrace{\begin{pmatrix} \quad U \quad \end{pmatrix}}_{n \times \ell} \cdot \underbrace{\begin{pmatrix} \Sigma & \\ 0 & \end{pmatrix}}_{\ell \times \ell} \cdot \underbrace{\begin{pmatrix} \quad V \quad \end{pmatrix}^T}_{\ell \times d}$$

U (V): orthogonal matrix containing the left (right) singular vectors of **A**.

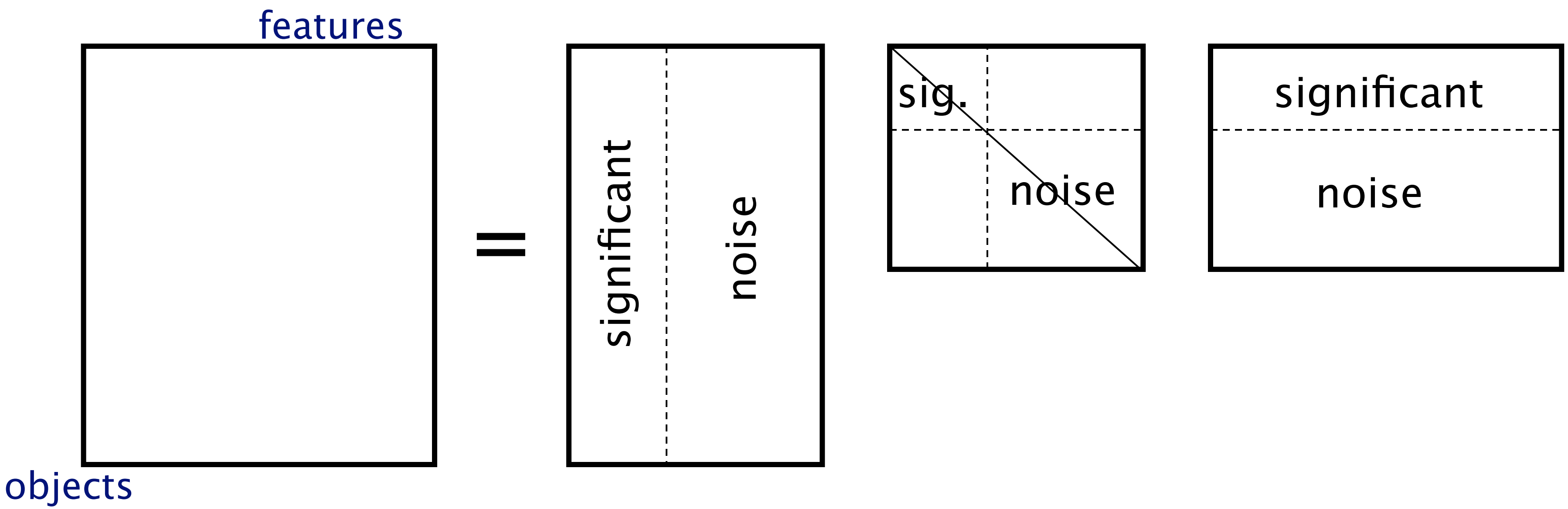
Σ: diagonal matrix containing the **singular values** of **A**: ($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_\ell$)

Exact computation of the SVD takes **$O(\min\{mn^2, m^2n\})$** time.

The top k left/right singular vectors/values can be **computed faster** using Lanczos/Arnoldi methods.

SVD and Rank-k approximations

$$\mathbf{A} \quad = \quad \mathbf{U} \quad \Sigma \quad \mathbf{V}^T$$



Rank-k approximations (A_k)

$$\begin{pmatrix} \\ \\ A_k \\ \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \\ U_k \\ \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \\ \Sigma_k \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \\ V_k^T \\ \\ \end{pmatrix}$$

n x d **n x k** **k x k** **k x d**

U_k (V_k): orthogonal matrix containing the top k left (right) singular vectors of A.
Σ_k: diagonal matrix containing the top k singular values of A

A_k is an approximation of A

Rank-k approximations (A_k)

$$A_k = U_k \cdot \Sigma_k \cdot V_k^T$$

n x d **n x k** **k x d**

A_k is the **best** approximation of A

U_k (V_k): orth
(right) singu
Σ_k: diagonal
values of A

A_k is an approximation of A

SVD as an optimization problem

Find C to minimize:

$$\min_{C} \left\| \underset{n\times d}{A} - \underset{n\times k}{C} \; \underset{k\times d}{X} \right\|_F^2$$

Frobenius norm:

$$\|A\|_F^2 = \sum_{i,j} A_{ij}^2$$

Given C it is easy to find X from standard least squares.
However, the fact that we can find the optimal C is fascinating!

SVD is “**the Rolls-Royce and the Swiss Army Knife of Numerical Linear Algebra.**”*

*Dianne O’Leary, MMDS ’06

Reference

Simple and Deterministic Matrix Sketching
Author: Edo Liberty, Yahoo! Labs
KDD 2013, Best paper award

Thanks Edo Liberty for the slides

Sketches of streaming matrices

- A nxd matrix
- Rows of A arrive in a stream
- Task: compute

$$AA^T = \sum_{i=1}^{n} A_i A_i^t$$

Sketches of streaming matrices

- A dxn matrix
- Rows of A arrive in a stream
- Task: compute

$$AA^T = \sum_{i=1}^{n} A_i A_i^t$$

- Naive solution: Compute AA^T in time $O(nd^2)$ and space $O(d^2)$

- Think of d=10^6, n = 10^6

Goal

- **Efficiently** compute a **concisely representable** matrix B such that

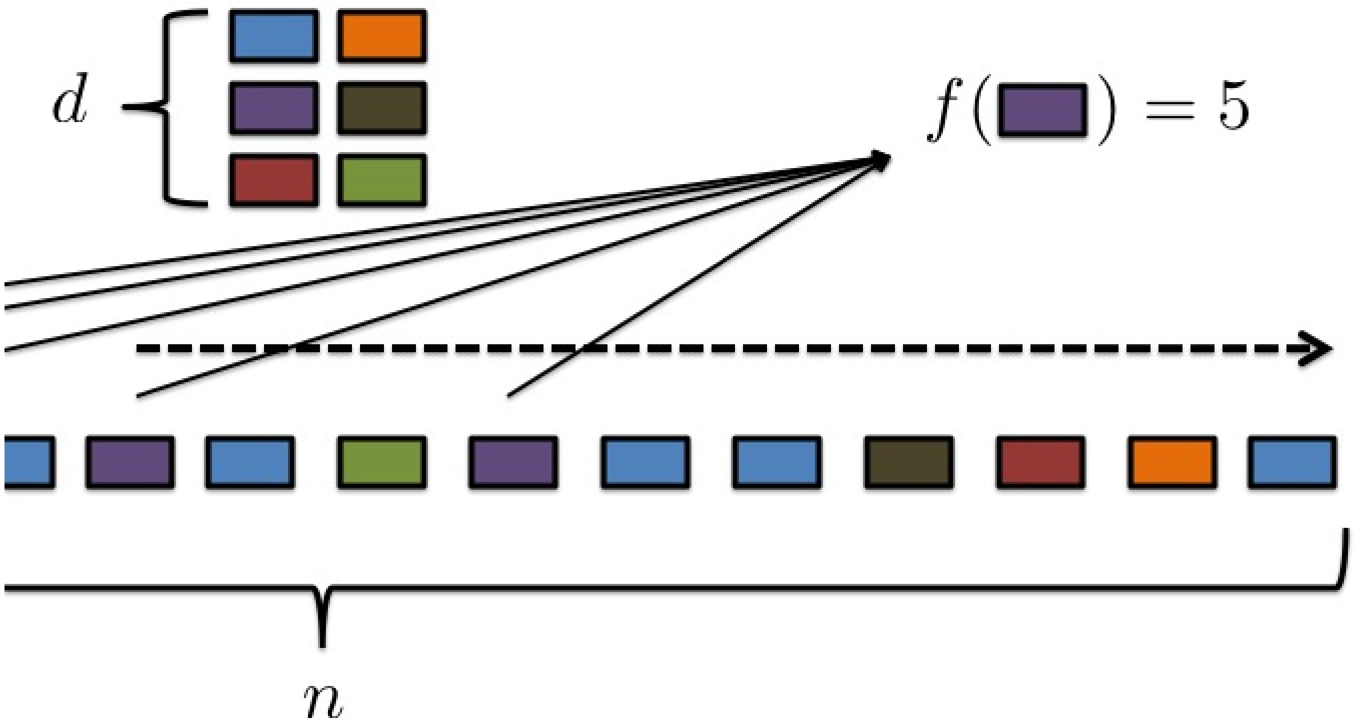
$$B \approx A \text{ or } BB^T \approx AA^T$$

woking with B is good enough for many tasks

- **Efficiently** maintain matrix B with only $\ell = 2/\epsilon$ such that

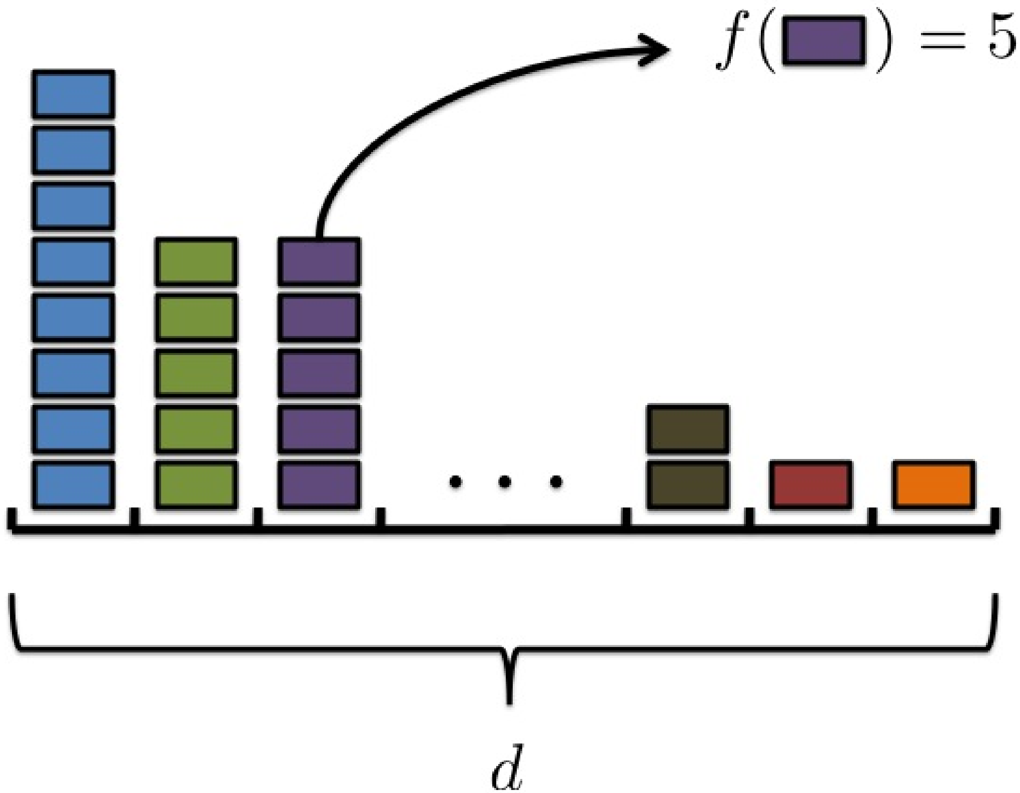
$$||AA^T - BB^T||_2 \leq \epsilon ||A||_f^2$$

Frequent items



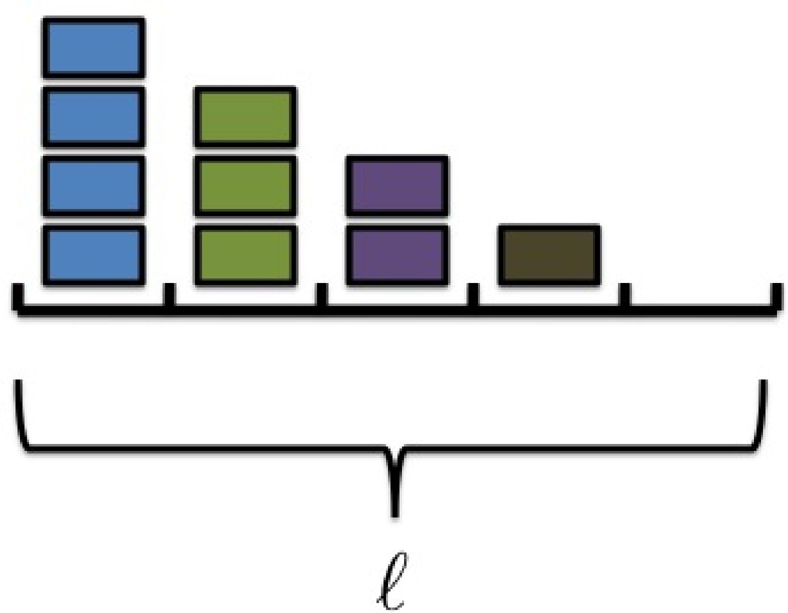
- obtain the frequency f(i) of each item in a stream of items

Frequent items



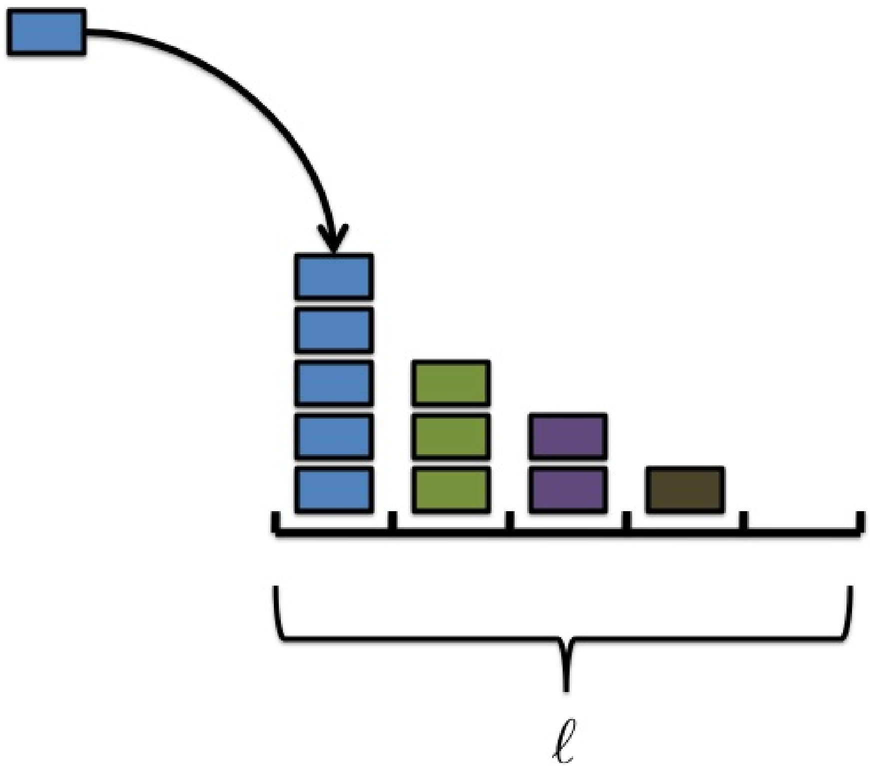
- With d counters it's easy but not good enough

Frequent Items



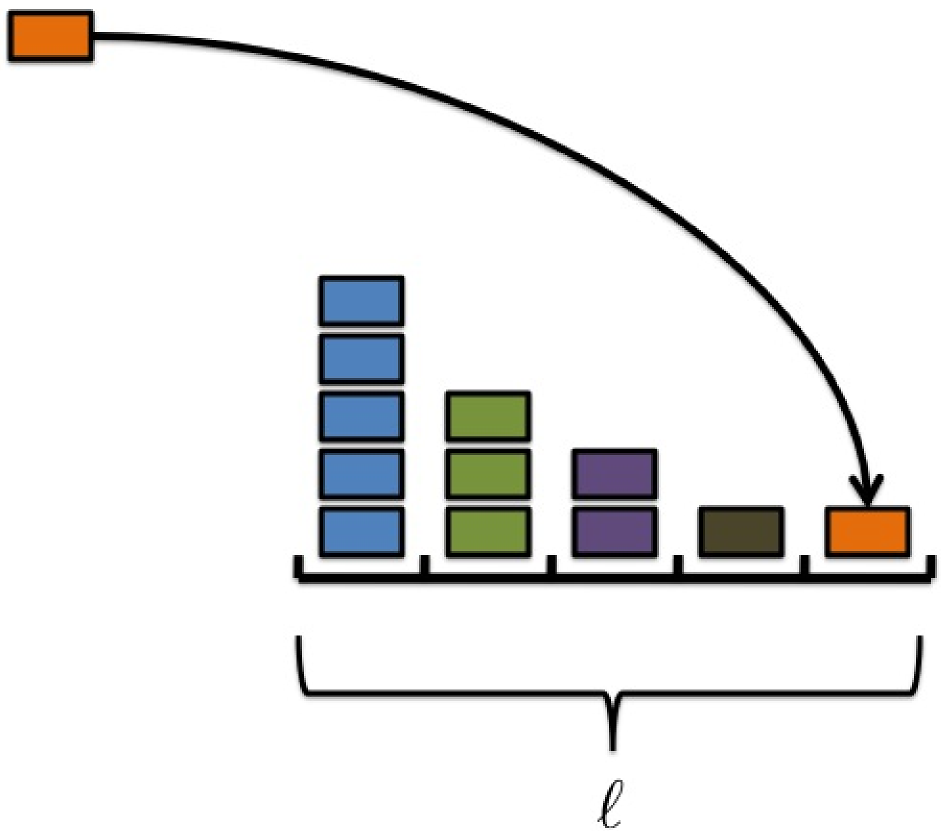
- Lets keep **less than** a fixed number of counters

Frequent items



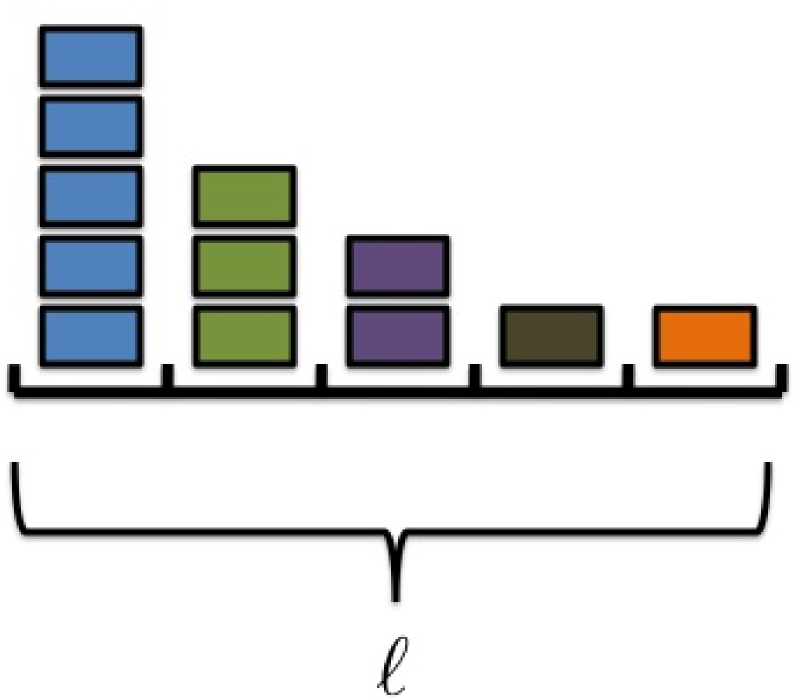
- If an item has a counter we add 1 to that counter

Frequent items



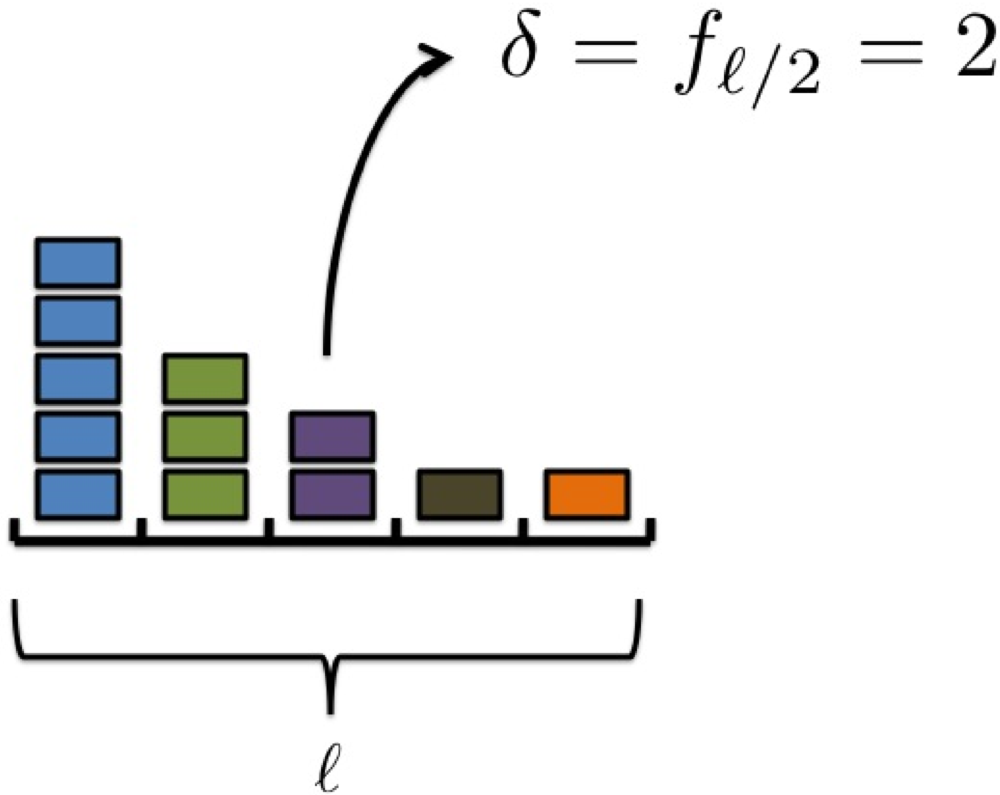
- Otherwise, we create a new counter for it and set it to 1

Frequent items



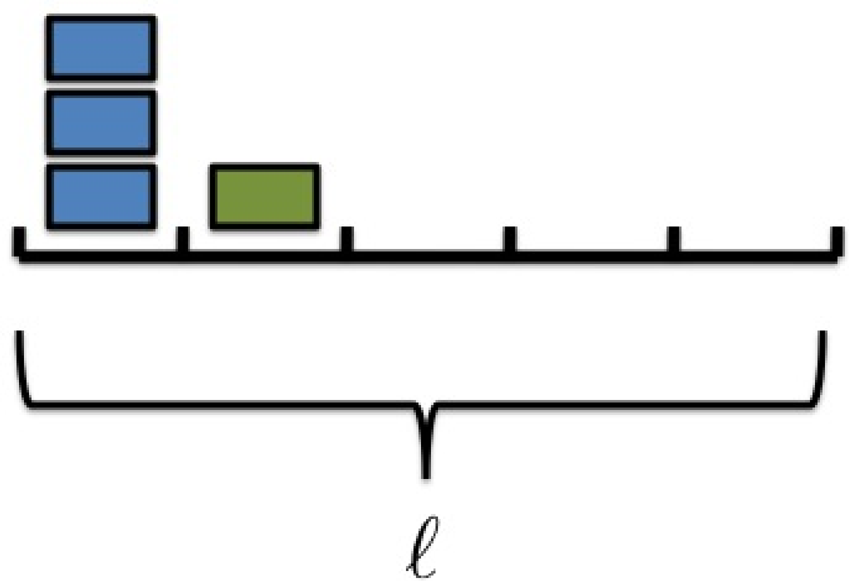
- But now we do not have less than ℓ counters

Frequent items



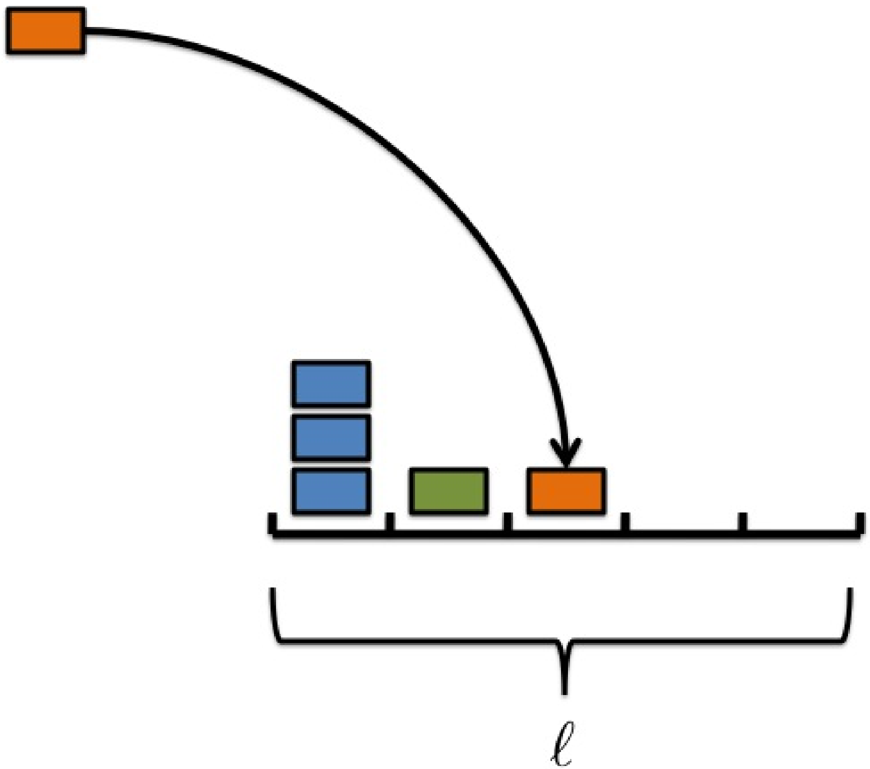
- Let δ be the median counter value at time t

Frequent items



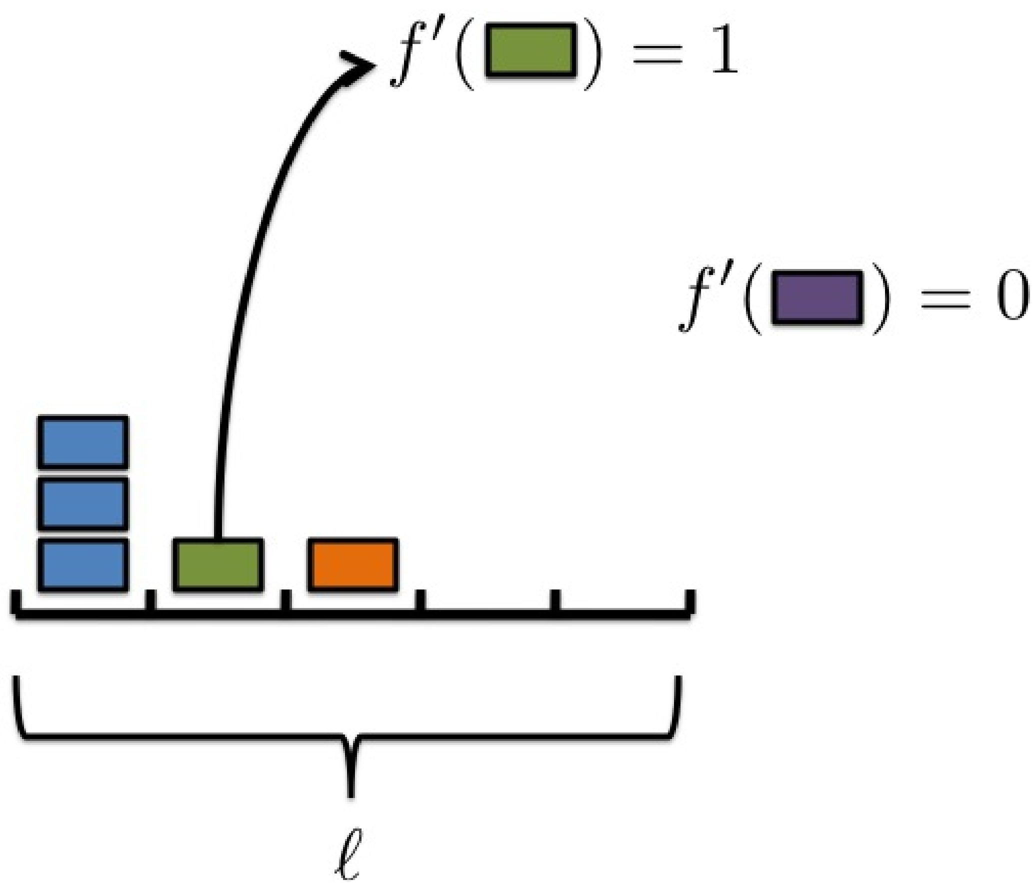
- Decrease all counters by δ (or set to zero if less than δ)

Frequent items



- And continue....

Frequent items



- The approximated counts are f’

Frequent items

- We increase the count by only 1 for each item appearance

$$f'(i) \leq f(i)$$

- Because we decrease each counter by at most δ_t at time t

$$f'(i) \geq f(i) - \sum_t \delta_t$$

- Calculating the total approximated frequencies:

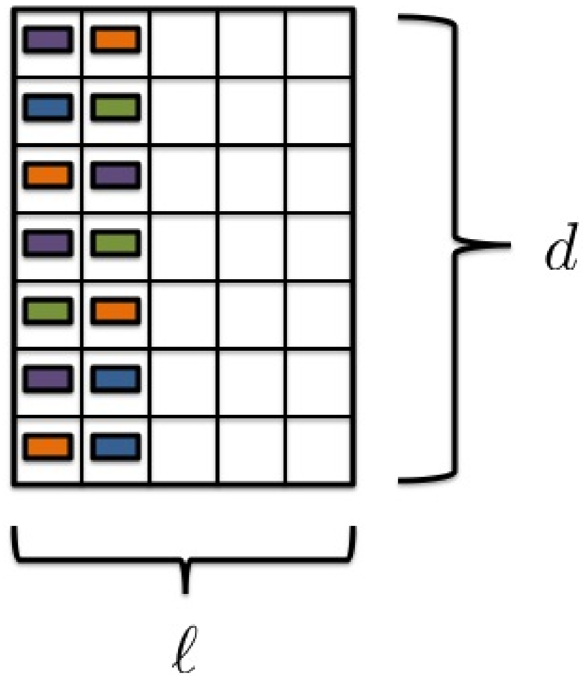
$$0 \leq \sum_i f'(i) \leq \sum_t (1 - (\ell/2)\delta_t) = n - (\ell/2) \sum_t \delta_t$$

$$\sum_t \delta_t \leq 2n/\ell$$

- Setting $\ell = 2/\epsilon$

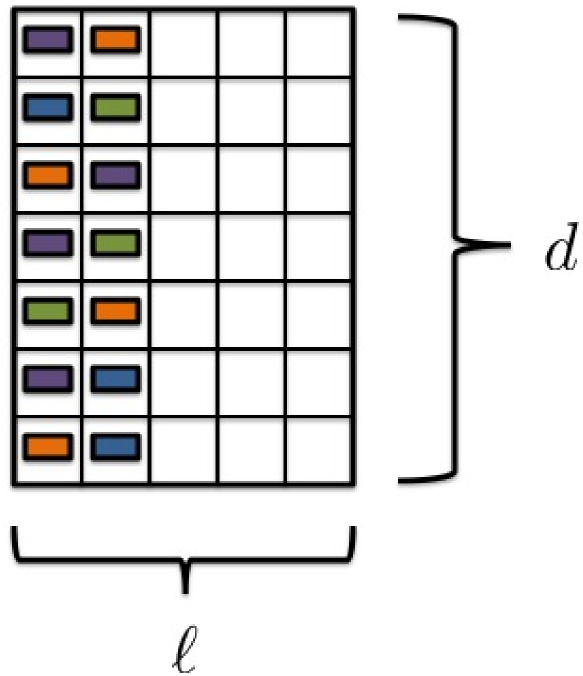
$$|f(i) - f'(i)| \leq \epsilon n$$

Frequent directions



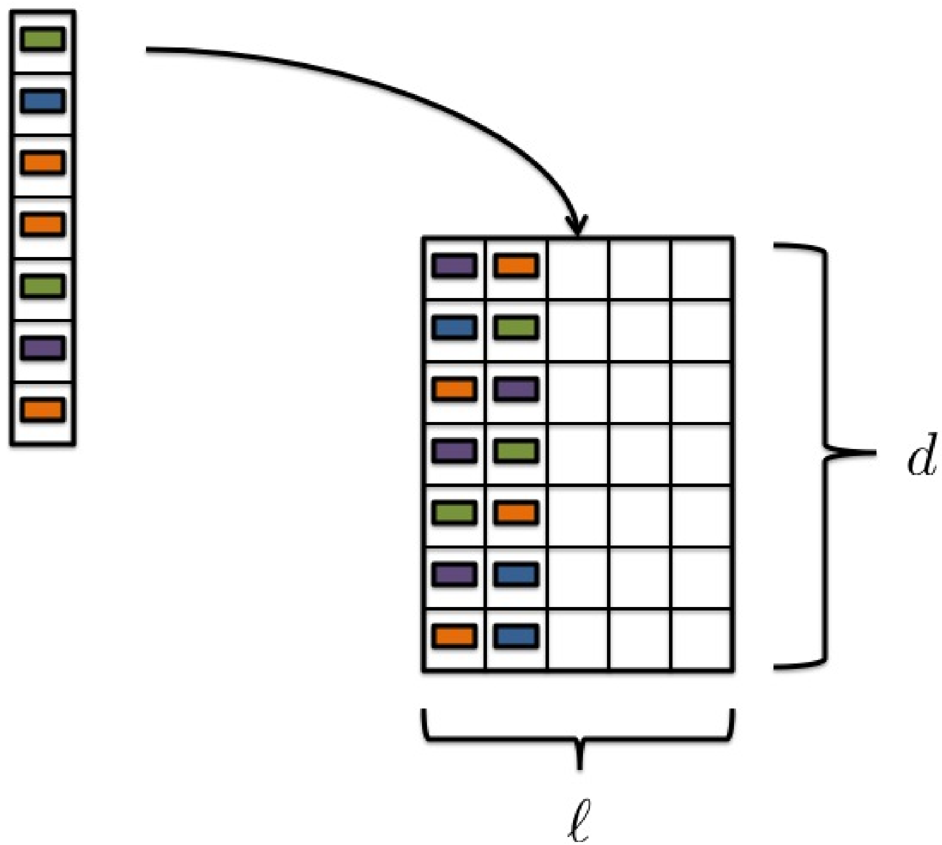
- We keep a sketch of at most ℓ columns

Frequent directions



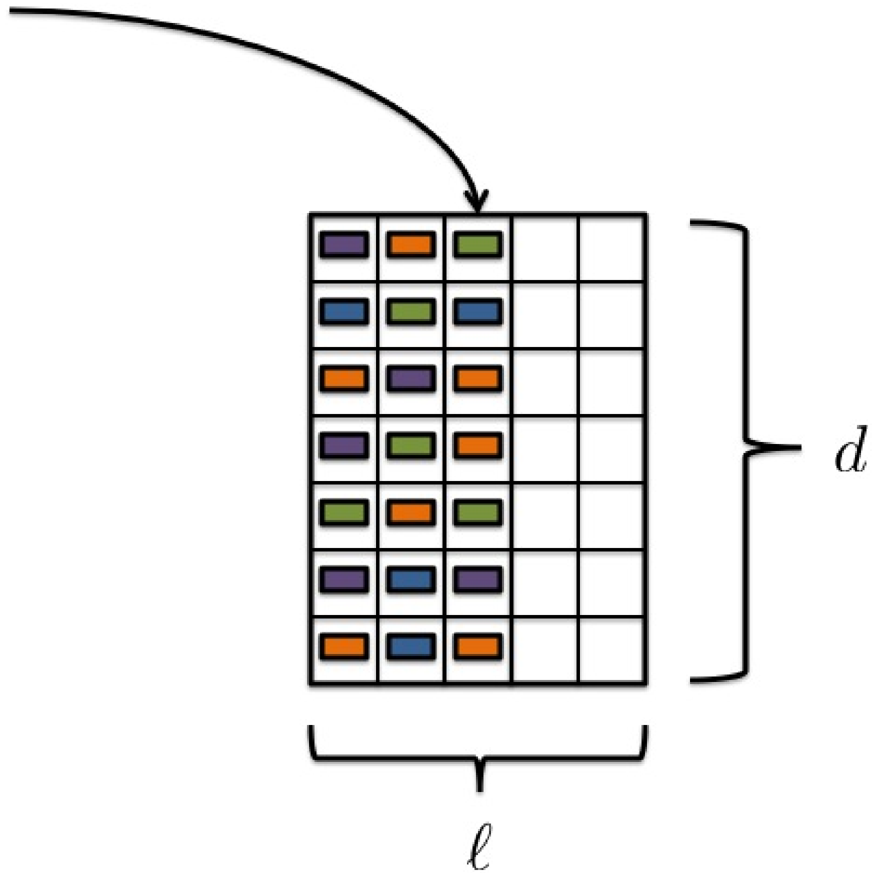
- Maintain the invariant that some of the columns are empty (zero-valued)

Frequent directions



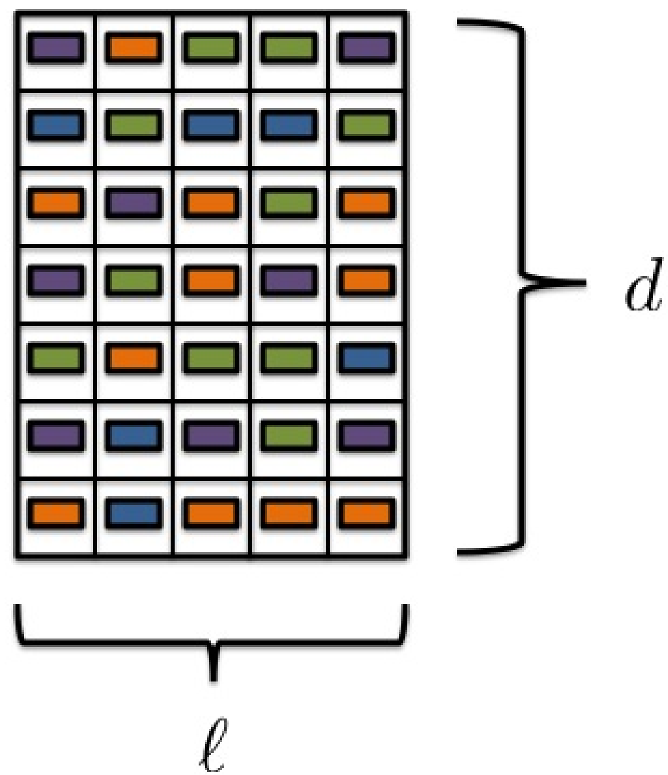
- Input vectors are simply stored in empty columns

Frequent directions



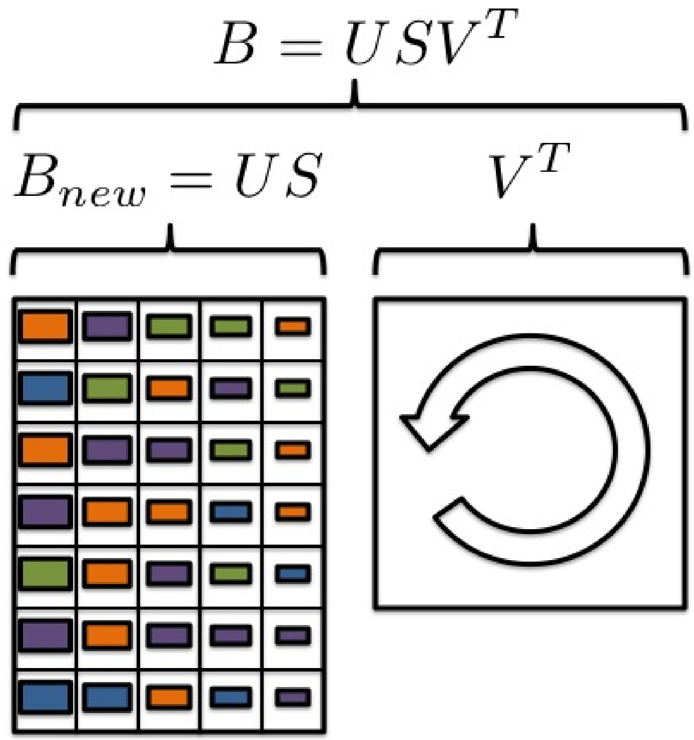
- Input vectors are simply stored in empty columns

Frequent directions



- When the sketch is ``full” we need to zero out some columns

Frequent directions

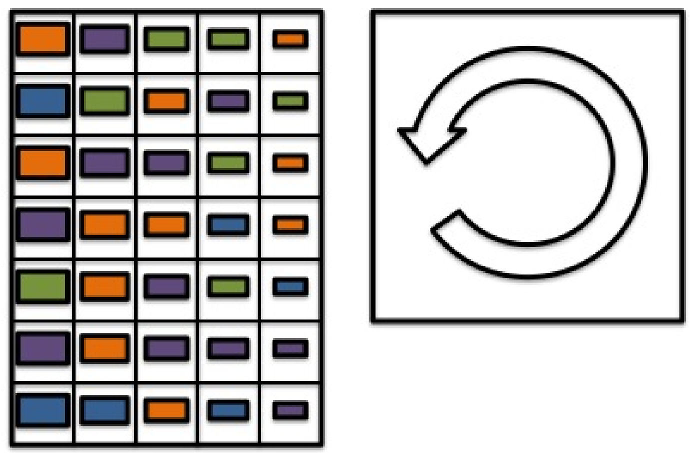


- Using SVD we compute $B = USV^T$ and set $B_{new} = US$

Frequent directions

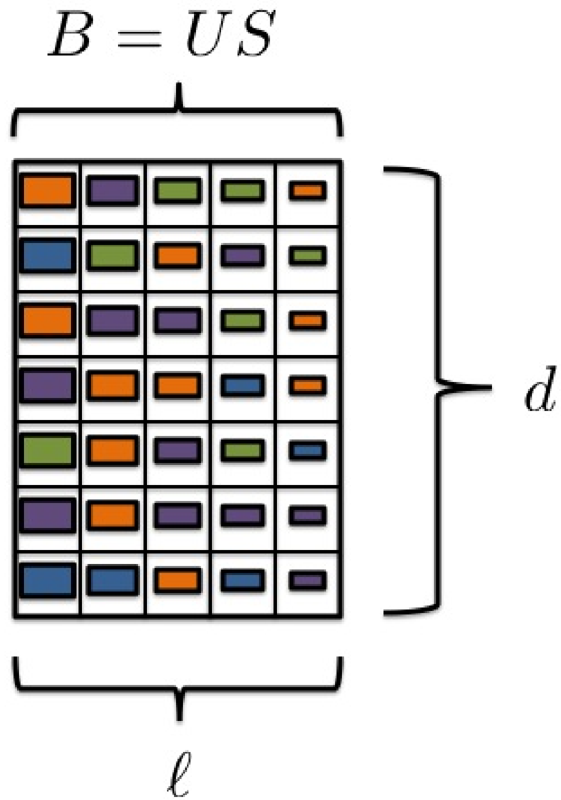
$$B = USV^T$$

$$B_{new} = US \qquad V^T$$



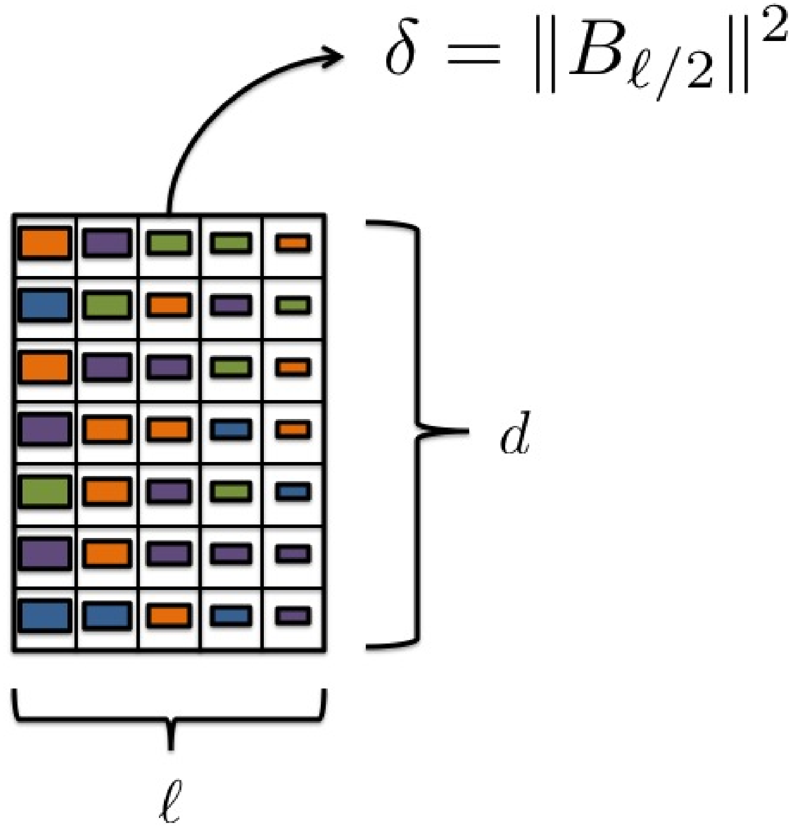
- Note that $BB^T = B_{new}B_{new}^T$ so we don't ``lose" anything

Frequent directions



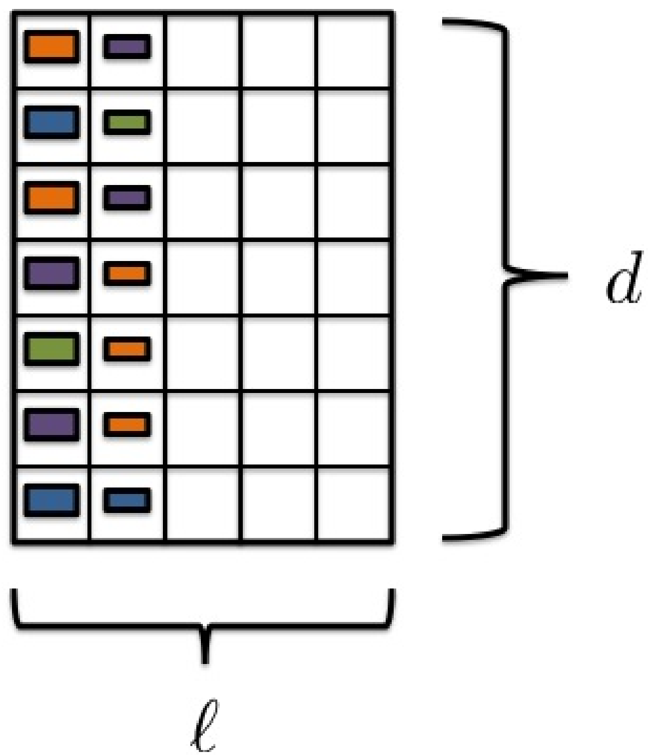
- The columns of B are now orthogonal and in decreasing magnitude order

Frequent directions



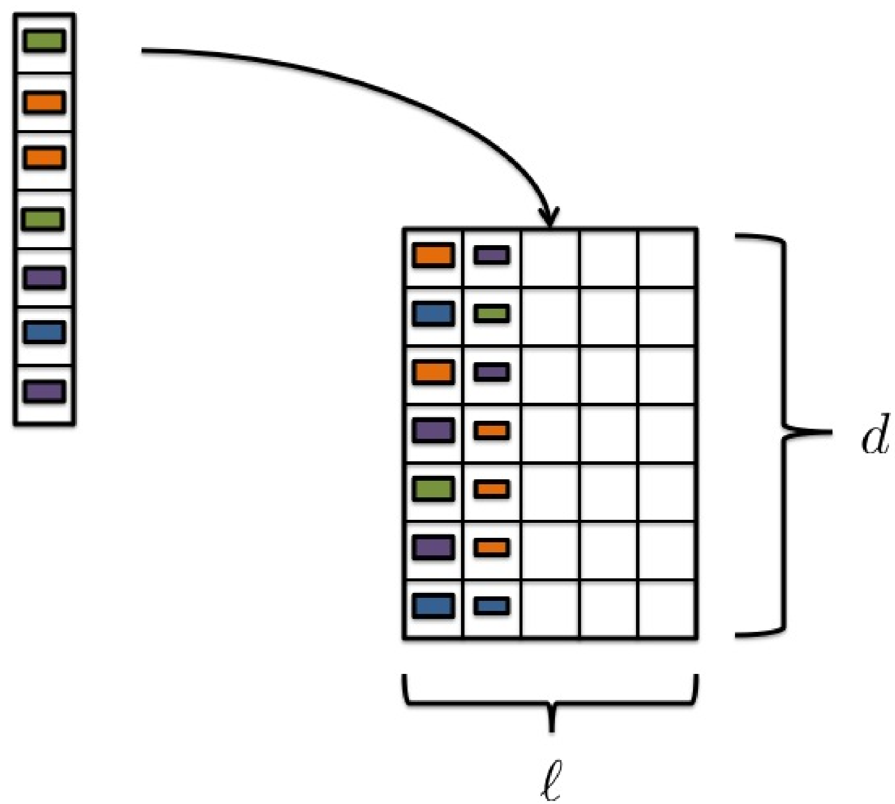
- Let $\delta = ||B_{\ell/2}||^2$

Frequent directions



- Reduce column $\ell_2^2 - \mathrm{norms}$ by δ (or nullify if less)

Frequent directions



- Start aggregating columns again

Frequent directions

Input: $\ell, \;\; A \in \mathbb{R}^{d \times n}$
$B \leftarrow$ all zeros matrix $\in \mathbb{R}^{d \times \ell}$
for $i \in [n]$ **do**
  Insert A_i into a zero valued column of B
  if B has no zero valued colums **then**
    $[U, \Sigma, V] \leftarrow SVD(B)$
    $\delta \leftarrow \sigma^2_{\ell/2}$
    $\check{\Sigma} \leftarrow \sqrt{\max(\Sigma^2 - I_\ell \delta, 0)}$
    $B \leftarrow U\check{\Sigma}$ # At least half the columns of B are zero.
Return: B

Frequent directions: proof

- Step 1:

$$||AA^T - BB^T|| \leq \sum_{t=1}^{n} \delta_t$$

- Step 2:

$$\sum_{t=1}^{n} \delta_t \leq 2||A||_f^2/\ell$$

- Setting $\ell = 2/\epsilon$ yields

$$||AA^T - BB^T|| \leq \epsilon||A||_f^2$$

Error as a function of ℓ