Chapter 1: Dimension Reduction

Ilya Razenshteyn **(MIT CSAIL)**

Motivation

- Compress high-dimensional data while not loosing much
- A concrete example:
  - Bag of words
  - Hashing trick: will see later
- **Theory vs Practice:**
  - Johnson–Lindenstrauss (JL)
  - Fast JL
  - Hashing trick
  - PCA

Word	Count
once	10
upon	3
time	4

Problem statement

- **Dataset:** n points in R^d, denote by X
- **Goal:** embed X into R^m with $m \ll d$ while preserving pairwise Euclidean distances up to multiplicative $(1 \pm \varepsilon)$

$$(1-\varepsilon) \cdot \|x_1 - x_2\|_2 \leq \|f(x_1) - f(x_2)\|_2 \leq (1+\varepsilon) \cdot \|x_1 - x_2\|_2,$$

where $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}$.

- **Parameters:** fixed n, d, ε, minimize m.
- **Worst-case** vs. **data-dependent** bounds

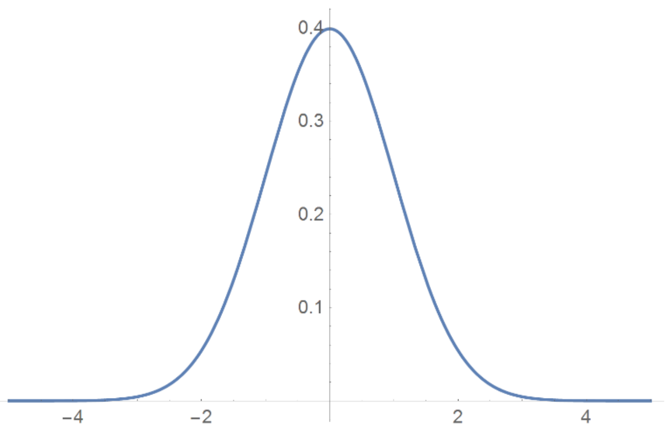
Naïve bound

- **Recall:** given X from R^d with $|X| = n$, embed X into R^m while $(1 \pm \varepsilon)$-preserving **pairwise Euclidean distances**
- **Exercise:** get $\varepsilon = 0$ with $m = n - 1$ (meaningful if $n \ll d$)
  - Tight
  - No dependence on d
  - Crucially uses the structure of Euclidean distance

Johnson–Lindenstrauss (JL) lemma

- **Recall:** given X from R^d with $|X| = n$, embed X into R^m while $(1 \pm \varepsilon)$-preserving **pairwise Euclidean distances**
- **[Johnson, Lindenstrauss 1984]:** one can get
$$m = O\left(\frac{\log n}{\varepsilon^2}\right).$$
- **[Alon 2003]:** tight up to $\log(1/\varepsilon)$.
- **[Larsen, Nelson 2016]**: tight!
- Proof technique: **probabilistic method**
  - Random object is good with positive probability → it exists!
- More specifically: **random projections**

Detour: normal distribution



- The density of $N(0,1)$ is:

$$f(t) = \frac{1}{\sqrt{2\pi}} \cdot e^{-t^2/2}$$

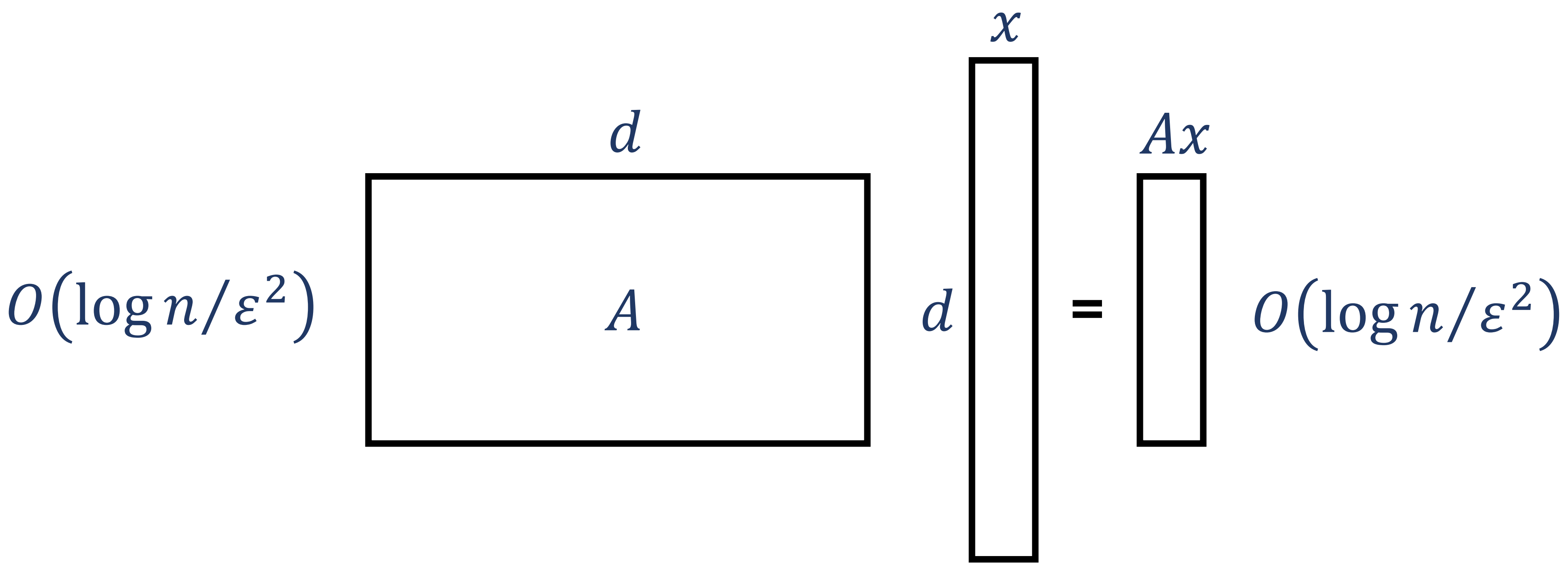
- Properties: if X_1, X_2, …, X_d are i.i.d. $N(0,1)$'s, then:
  - $(X_1, X_2, \ldots, X_d)$ is spherically symmetrical
  - $\alpha_1 X_1 + \alpha_2 X_2 + \cdots \alpha_d X_d$ is distributed as $\|\alpha\|_2 \cdot N(0,1)$ (2-stability)
- Normalized Gaussian vector is a **uniform unit vector**

Detour 2: concentration inequalities

- **CLT:** average of i.i.d. nice random variables converges to a Gaussian with matching first two moments
- Often, want a *finitary* statement:
  - **Example:** let X be a sum of n i.i.d. ± 1's.
  - **Claim:** $\Pr[X \geq t\sqrt{n}] \leq e^{-\Omega(t^2)}$
- Lots of statements of this sort, proved very similarly:
  - Chernoff
  - Hoeffding
  - Azuma
  - Hinchin…

Oblivious dimension reduction

- A universal distribution on embeddings that works with high probability for **any given dataset!**



Proof of Johnson–Lindenstrauss I

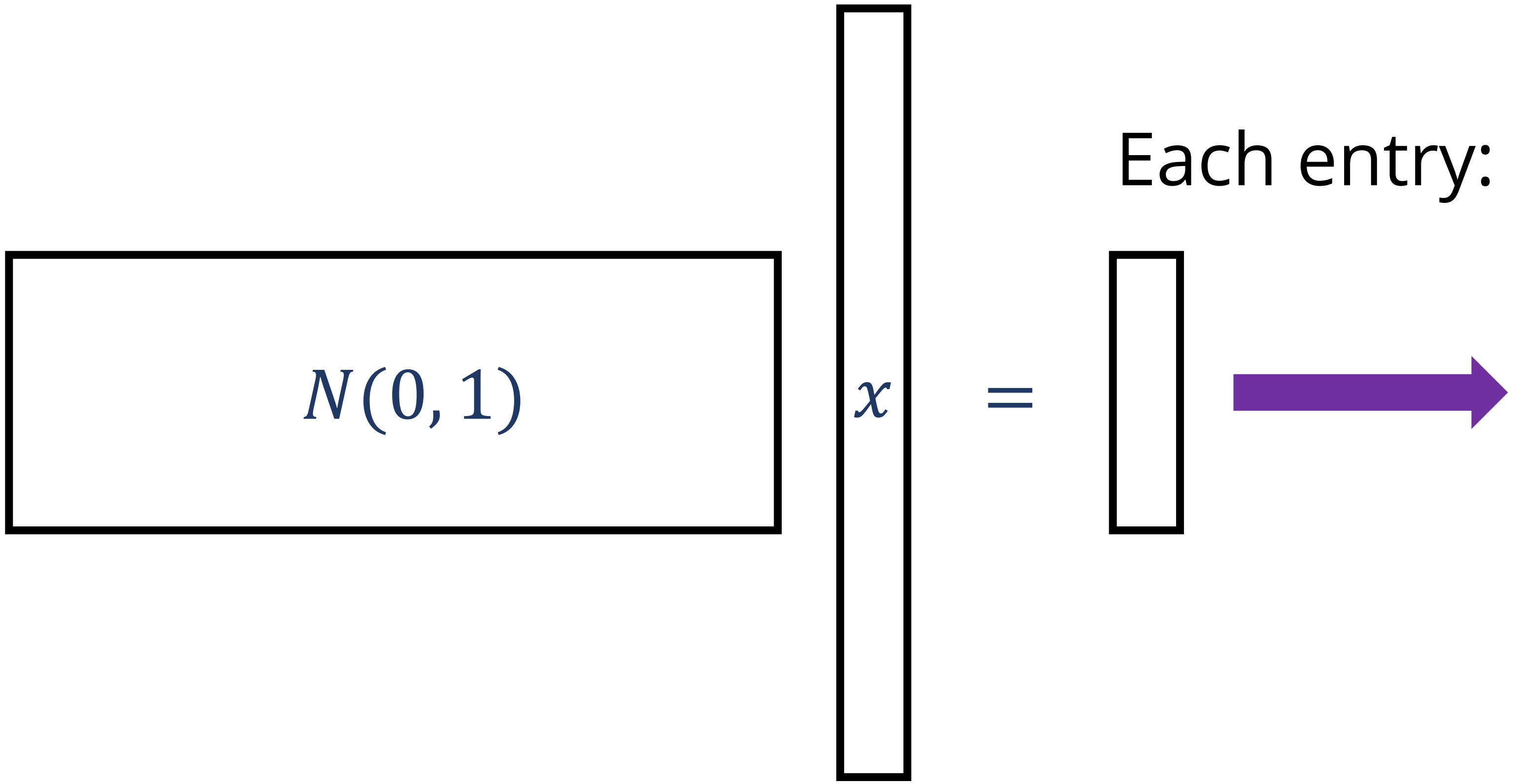
- **[Dasgupta, Gupta 2003]** Let A be an $m \times d$ matrix with i.i.d. $N(0, 1)$ entries
- **The main claim:** for every $\varepsilon, \delta > 0$, there exists
$$m = O\big(\log(1/\delta)/\varepsilon^2\big)$$

s.t. for every x one has with probability $1 - \delta$:
$$(1 - \varepsilon)m \cdot \|x\|_2^2 \leq \|Ax\|_2^2 \leq (1 + \varepsilon)m \cdot \|x\|_2^2$$

- **Implies JL:** set $\delta = 1/10n^2$ and use the **union bound**
  - Crucially use linearity of the map
  - A is not explicit, but can be constructed quickly w.h.p.

Proof by picture

Each entry: $\|x\|_2 \cdot N(0,1)$



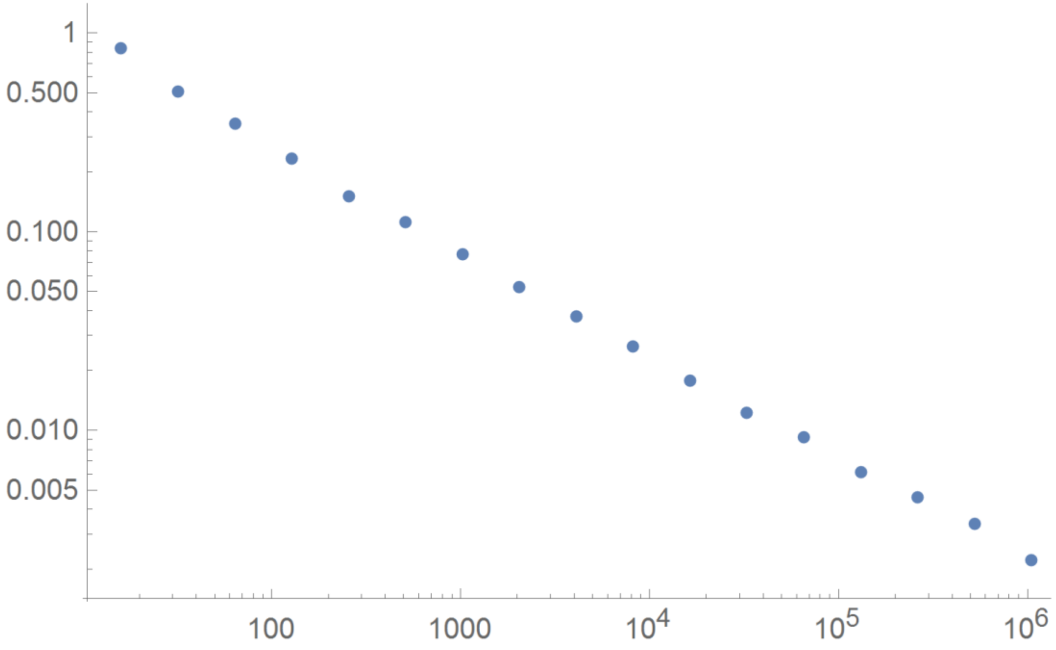
$\|x\|_2^2 \cdot \chi^2(m)$

Proof of Johnson–Lindenstrauss II

- **The main claim (reformulated):** for every ε, m
$$\Pr\left[\|Ax\|_2^2 \in (1 \pm \varepsilon)m \ \cdot \|x\|_2^2\right] \geq 1 - e^{-\Omega(\varepsilon^2 m)}$$
- **Step 1:** elements of Ax are i.i.d. $\|x\|_2 \cdot N(0,1)$
- **Step 2:** $\|Ax\|_2^2$ is distributed as $\|x\|_2^2 \cdot \chi^2(m)$
- **Step 3:** $\Pr[\chi^2(m) \in (1 \pm \varepsilon)m] \geq 1 - e^{-\Omega(\varepsilon^2 m)}$ (see, e.g., **[Laurent, Massart 2000]**)

Concrete numbers

- Fix $\delta = 0.1$, trade-off between m and ε
- $1 / \varepsilon^2$ makes the construction quite impractical
- But:
  - Seldom need to preserve **all** the pairwise distances
  - Random projections are *very* useful (will see later)



Fast dimension reduction

- Applying a random projection requires $O(dm)$ time
  - Too slow for reducing dimension from, say, **1M** to **1K**
  - Takes **225 ms** on *one core* of Intel Core i5-2500 (C++, using Eigen)
- **Never implement your own matrix-vector or matrix-matrix multiplication**
  - Specialized libraries (OpenBLAS, Eigen) exploit vectorization, memory caches, multithreading etc.
  - The above takes **920 ms** if done (relatively) naively
- Can we do dimension reduction faster?
  - Will improve to **< 5 ms** by **better algorithms**

The plan

- **[Ailon, Chazelle 2006]:** fast random projection
- For “dense” vectors can uniformly subsample coordinates
- Reduction from the general case to the dense case

The dense case

- Trying to preserve the norm of a d-dimensional vector x
  - Can assume w.l.o.g. that $\|x\|_2 = 1$
- Assume that all the entries of x are at most τ
  - Best case: $\tau = 1/\sqrt{d}$; worst case: $\tau = 1$
- **Intuition:** if the energy is spread, then subsampling works!
- By **Hoeffding inequality**, need to sample

$$m = O\left(d\tau^2 \log(1/\delta)/\varepsilon^2\right)$$

coordinates to $(1 \pm \varepsilon)$-preserve the norm w/prob. $1 - \delta$

- Between $m = O\left(\log(1/\delta)/\varepsilon^2\right)$ and $m = O\left(d \cdot \log(1/\delta)/\varepsilon^2\right)$

Reduction to the dense case

- **First idea:** apply a random rotation
  - Preserves the norm of any vector
  - Makes any fixed vector dense w.h.p. (energy of a random unit vector is spread)
- **But,** applying a random rotation takes time $O(d^2)$
- **Crucial idea:** complete randomness is unnecessary
- Will see a distribution on rotations that has the above two properties, but takes only $O(d \log d)$ time to apply

Pseudo-random rotations

- Introduced in **[Ailon, Chazelle 2006]**
- **Fast Hadamard Transform**
  - Preserves distances
  - Can be computed in time $O(d \log d)$
  - "Mixes well"

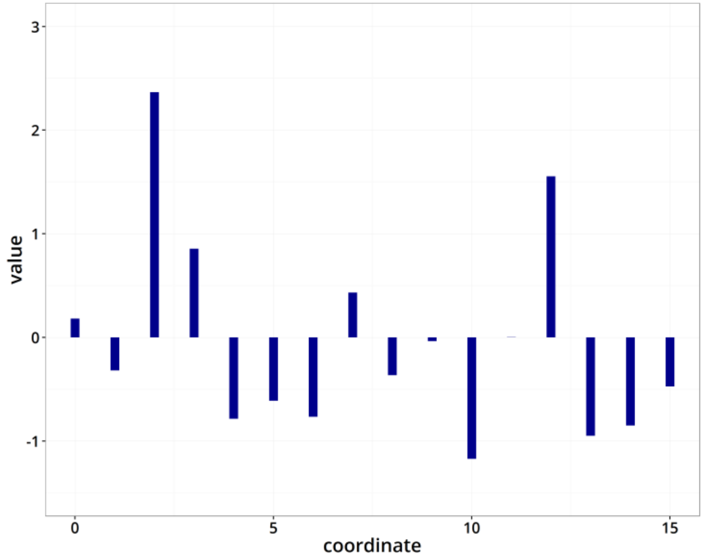
$$x = (x_1, x_2, \dots, x_d)$$

Flip signs

$$x' = (\pm x_1, \pm x_2, \dots, \pm x_d)$$

Hadamard

$$Hx'$$



Hadamard transform

- Defined recursively
- $H_0 = (1)$
- $H_{k+1} = \frac{1}{\sqrt{2}} \cdot \begin{pmatrix} H_k & H_k \\ H_k & -H_k \end{pmatrix}$
- **Exercise**: can multiply by $H_{\log d}$ in time $O(d \log d)$
- By **Khintchin's inequality**: with probability $1 - \delta$ the entries of Hx' are bounded by $O\left(\sqrt{\frac{\log(d/\delta)}{d}}\right)$
    - What we use: all entries are $\pm 1/\sqrt{d}$

Each entry of Hx' is: $\frac{\pm x_1 \pm x_2 \pm \cdots \pm x_d}{\sqrt{d}}$

Overall

- We reduce dimension to
$$O\left(\frac{\log\frac{d}{\delta}\cdot\log\frac{1}{\delta}}{\varepsilon^2}\right)$$
in time $O(d\log d)$.
- Implementation details:
  - Don't ever try to implement your own FFT, use FFTW…
  - … unless it pays off! FFTW turns out to be sub-optimal for FHT
  - Use **https://github.com/falconn-lib/ffht** **[R, Schmidt 2015]**
- Again, not so useful by itself, but the FHT idea is used a lot!

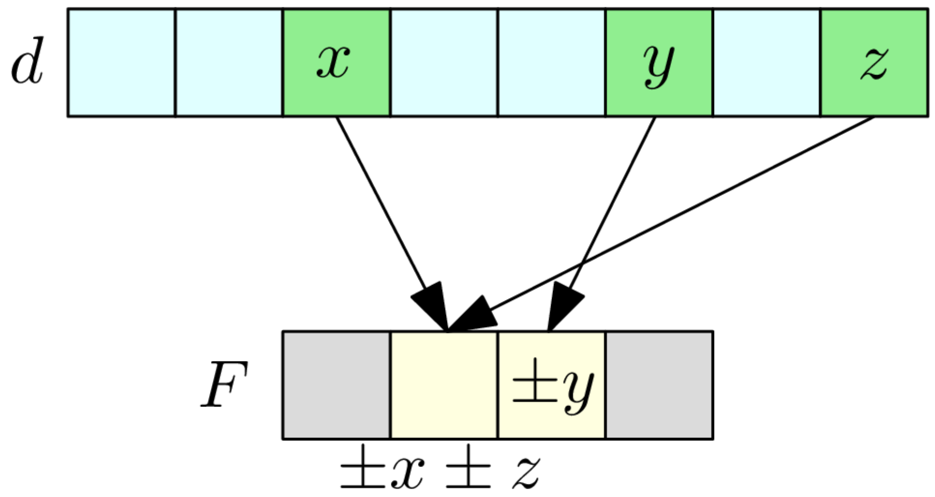
Lower bounds

- **[Alon 2003]:** need $\Omega\left(\frac{\log n}{\varepsilon^2 \log(1/\varepsilon)}\right)$ dimensions (tight for that example)
- **[Larsen, Nelson 2016]:** need $\Omega\left(\frac{\log n}{\varepsilon^2}\right)$ dimensions

Dimension reduction for ℓ_1

- Dimension reduction for ℓ_1? ($\|x\|_1 = |x_1| + |x_2| + \cdots + |x_d|$)
  - **[Brinkman, Charikar 2003]:** not as great! Need dimension $n^{\Omega(1/D^2)}$ for distortion D
  - **[Lee, Naor 2004]:** a simple example (the diamond graph)
- **[Batson, Spielman, Srivastava 2009]:** can achieve dimension $O(n/\varepsilon^2)$ (via spectral sparsifiers)
- Weaker notions of dimension reduction **[Kushilevitz, Ostrovsky, Rabani 2000], [Indyk 2000]**

Hashing trick (a.k.a. CountSketch)



Good expectation and variance, but bad concentration, still useful in practice (streaming, similarity search, randomized linear algebra)

Size in bits?

- Johnson–Lindenstrauss gives $O(n \log n/\varepsilon^2)$ real numbers
- Can obtain essentially the same number of **bits**! **[Indyk, Wagner 2017] [Indyk, Wagner, R 2017]**

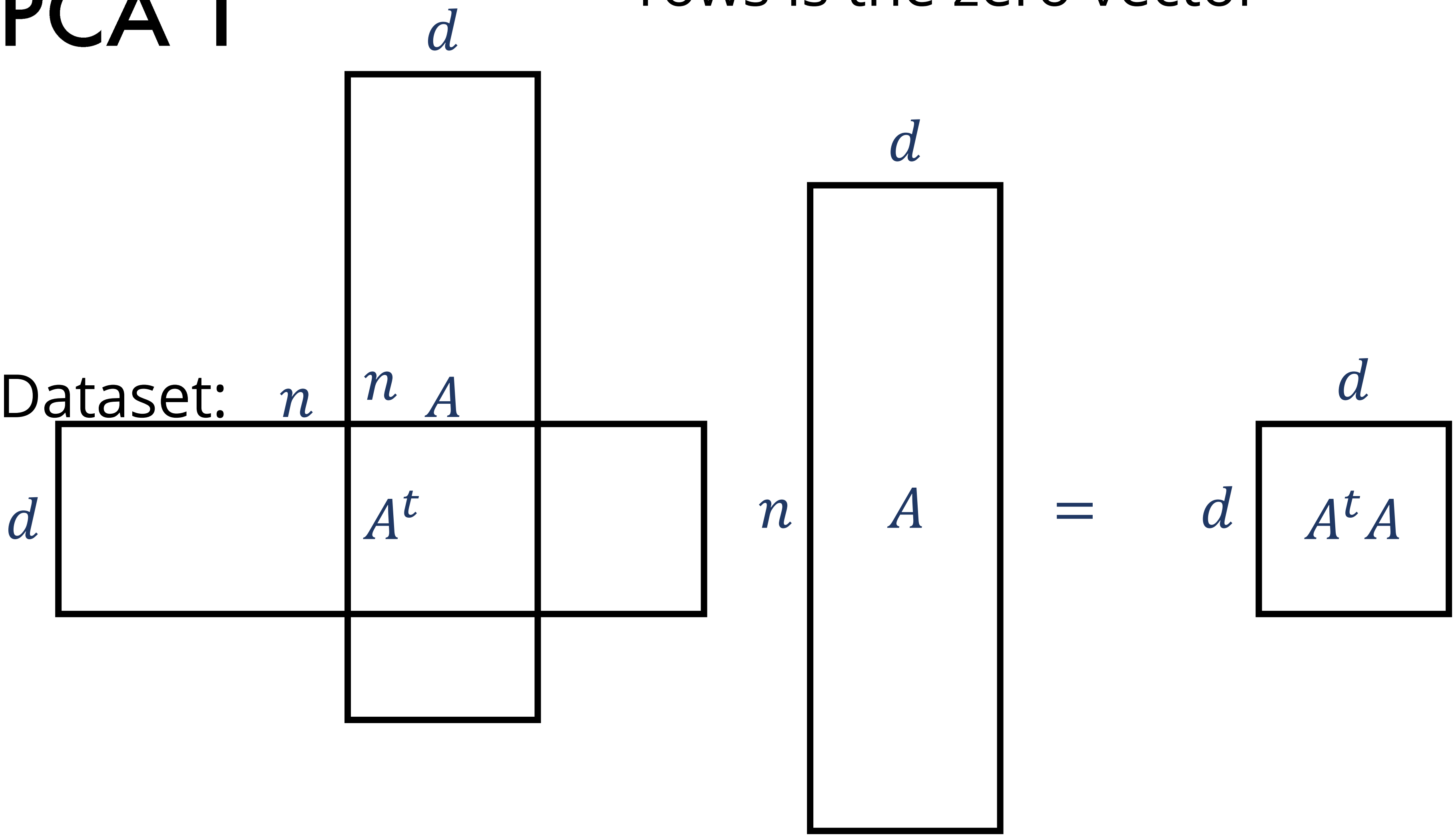
Principal component analysis (PCA)

- Great practical heuristic for dimension reduction
- Fit a Gaussian to data
- Not only dimension reduction, but de-noising as well!

PCA 1

Assume: the mean of rows is the zero vector

Dataset:



PCA 2

- Matrix $A^t A$ is symmetric positive semi-definite, hence its eigenvalues $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_d(A) \geq 0$ are real and non-negative
- Let $v_1, v_2, \ldots, v_d \in R^d$ be an orthonormal eigenbasis (v_i corresponds to $\lambda_i(A)$)
- For $1 \leq k \leq d$, project the dataset onto the span of $v_1, v_2, \ldots, v_k$

Properties of PCA

- The direction v_1 maximizes the variance of the projection
- v_2 maximizes the variance conditional on being orthogonal to v_1
- …
- If dataset lies in a low-dimensional space (modulo small noise), PCA should discover it
- Quite often, most of the variance can be "explained" by a few directions; in this case, PCA works very well

An example: PCA for MNIST

- 60000 28x28 images (784 dimensions)
- Let's try to do PCA on it