

# Machine Learning for Data Science (CS4786)

## Lecture 5

Random Projections

Course Webpage :

<http://www.cs.cornell.edu/Courses/cs4786/2016sp/>

Recap

# DIMENSIONALITY REDUCTION

Given feature vectors  $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^d$ , compress the data points into low dimensional representation  $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^K$  where  $K \ll d$

## **Principal Component Analysis:**

- **Find directions that maximize variance (spread)**
- **Find directions that minimize reconstruction error**

# PRINCIPAL COMPONENT ANALYSIS

1.  $\Sigma = \text{cov}\left(X\right)$

2.  $W = \text{eigs}\left(\Sigma, K\right)$

3.  $Y = X - \mu \times W$

# RECONSTRUCTION

4.

$$\hat{X} = Y \times W^T + \mu$$

# TWO VIEW DIMENSIONALITY REDUCTION

- Data can be split into pairs  $(\mathbf{x}_1, \mathbf{x}'_1), \dots, (\mathbf{x}_n, \mathbf{x}'_n)$  where  $\mathbf{x}_t$ 's are  $d_1$  dimensional and  $\mathbf{x}'_t$ 's are  $d_2$  dimensional
- Goal: Compress  $\mathbf{x}_1, \dots, \mathbf{x}_n$  into  $K$  dimensional vectors  $\mathbf{y}_1, \dots, \mathbf{y}_n$  (or  $\mathbf{x}'_1, \dots, \mathbf{x}'_n$  into  $\mathbf{y}'_1, \dots, \mathbf{y}'_n$  or both)
  - Retain information redundant between the two views

## Canonical Correlation Analysis:

- Find directions that maximize correlations between the projections in the two views

# CCA ALGORITHM

$$1. \quad X = \begin{pmatrix} n & \begin{matrix} X_1 \\ X_2 \end{matrix} \\ d_1 & d_2 \end{pmatrix}$$

$$2. \quad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix} = \text{cov} \left( \begin{matrix} X \end{matrix} \right)$$

$$3. \quad W_1 = \text{eigs} \left( \Sigma_{11}^{-1} \Sigma_{12} \Sigma_{22}^{-1} \Sigma_{21}, K \right)$$

# CCA ALGORITHM

$$4. \quad Y_1 = (X_1 - \mu_1) \times W_1$$

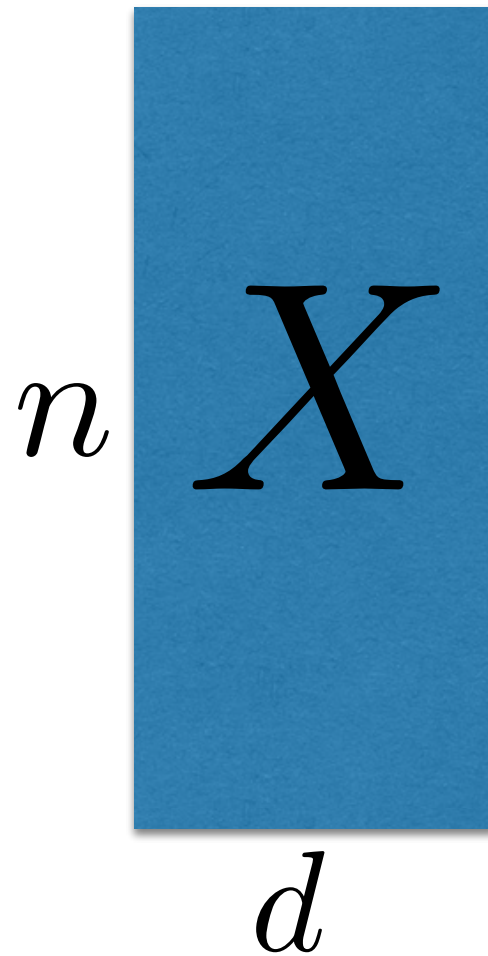


# CCA DEMO REDO?

# BACK TO SINGLE VIEW: RECAP

The diagram illustrates the matrix multiplication  $X \times W = Y$ . Matrix  $X$  is an  $n \times d$  matrix, represented by a blue rectangle. Its height is labeled  $n$  and its width is labeled  $d$ . The rows of  $X$  are labeled  $\mathbf{x}_1^\top$  at the top and  $\mathbf{x}_n^\top$  at the bottom, with a vertical dotted line indicating intermediate rows. Matrix  $W$  is a  $d \times K$  matrix, represented by a narrower blue rectangle. Its width is labeled  $K$ . Matrix  $Y$  is an  $n \times K$  matrix, represented by a blue rectangle of the same height as  $X$  but with width  $K$ . Its rows are labeled  $\mathbf{y}_1^\top$  at the top and  $\mathbf{y}_n^\top$  at the bottom, with a vertical dotted line indicating intermediate rows. The multiplication is shown as  $X \times W = Y$ .

# The Tall, THE FAT AND THE UGLY



# The Tall, THE FAT AND THE UGLY

$$\begin{array}{c} d \\ \boxed{X^T} \\ n \end{array} \times \begin{array}{c} n \\ \boxed{X} \\ d \end{array} \Bigg/ n = \begin{array}{c} d \\ \boxed{\Sigma} \\ d \end{array}$$

$$\begin{array}{c} d \\ \boxed{W} \\ K \end{array} = \text{Eigs} \left( \begin{array}{c} \boxed{\Sigma} \\ d \end{array}, K \right)$$

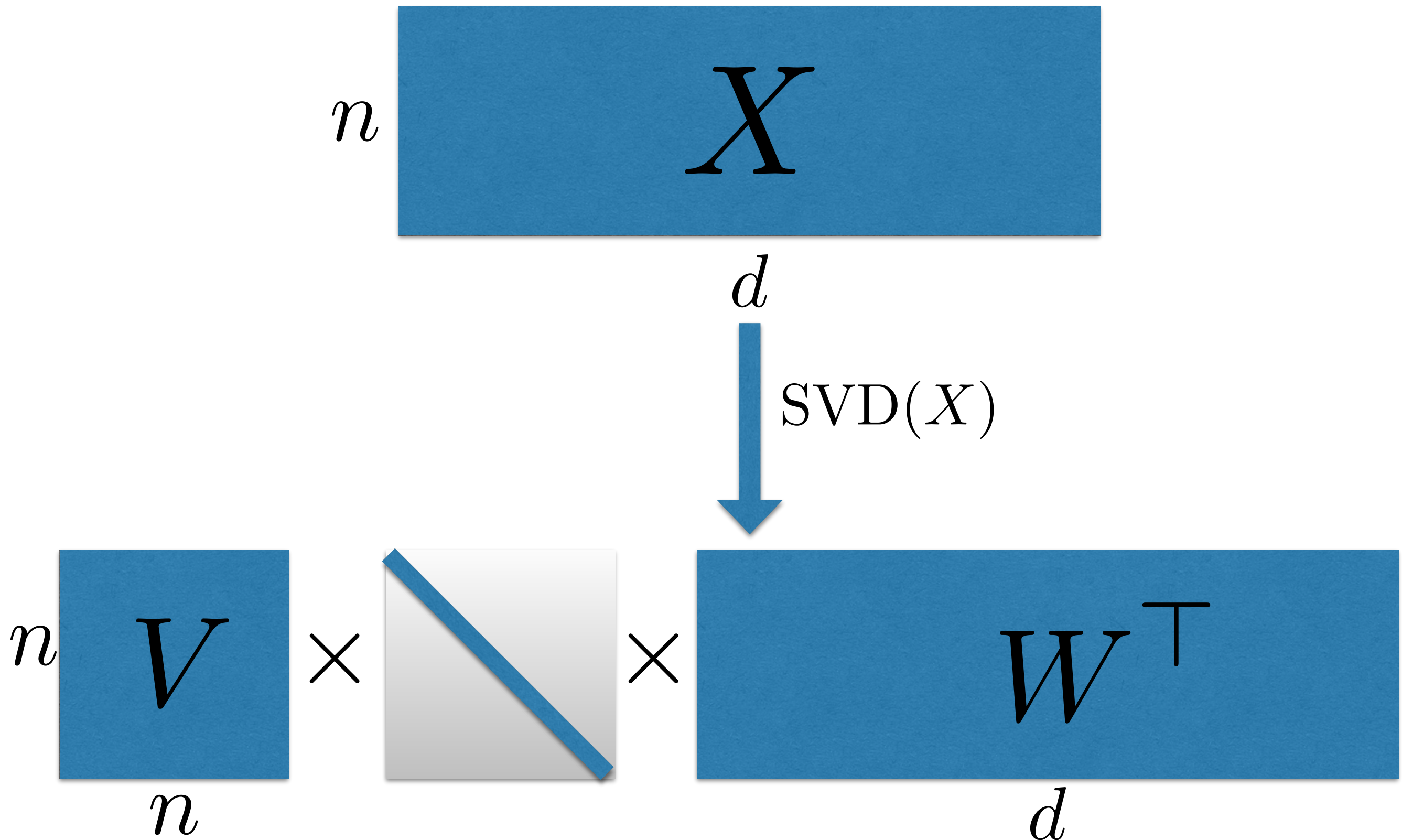
# THE TALL, the Fat AND THE UGLY

$n$

$X$

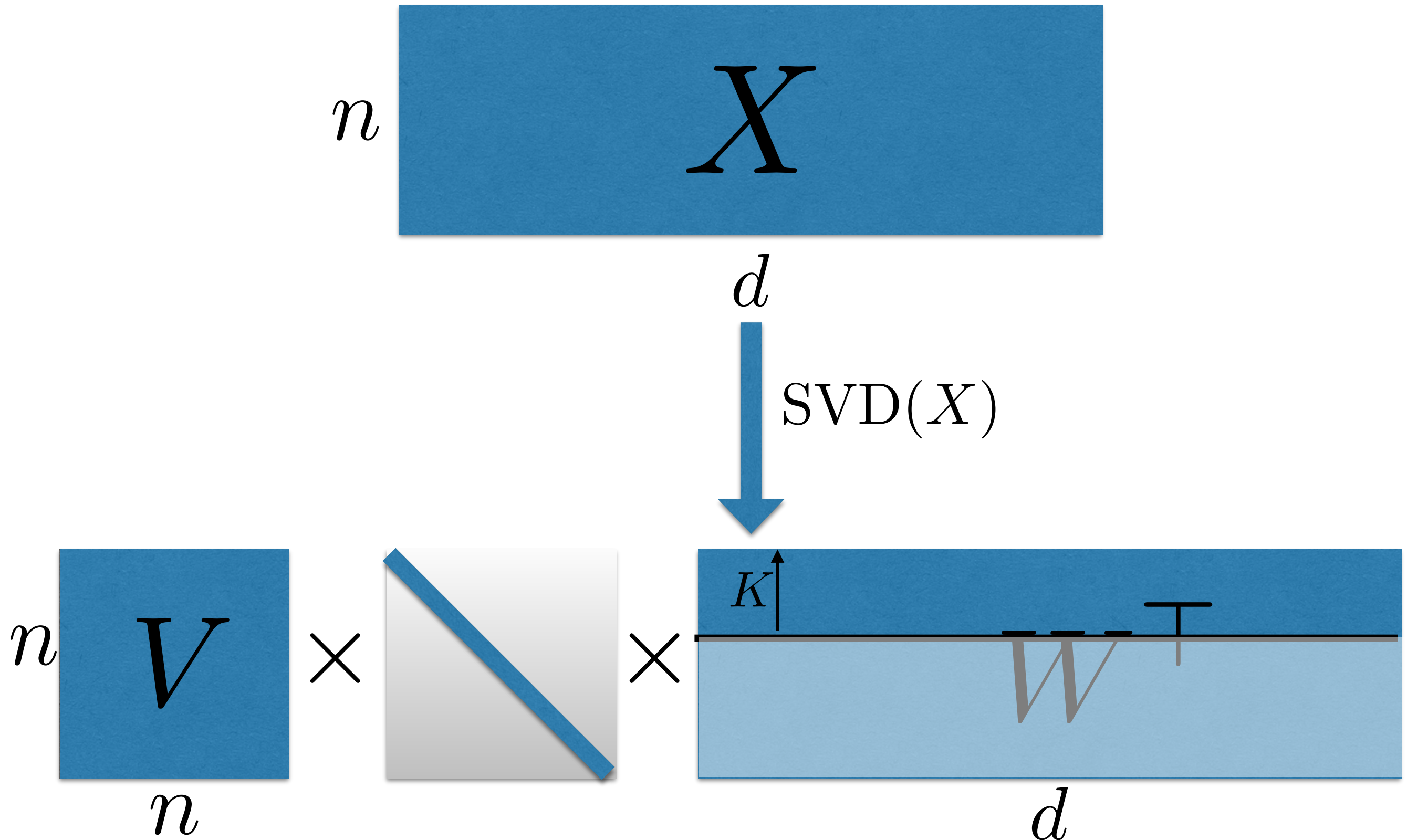
$d$

# THE TALL, the Fat AND THE UGLY





# THE TALL, the Fat AND THE UGLY



# THE TALL, THE FAT AND the Ugly

$X$



- $d$  and  $n$  so large we can't even store in memory
- Only have time to be linear in  $\text{size}(X) = n \times d$

I there any hope?



# PICK A RANDOM W

$$Y = X \times \left[ \begin{array}{ccc} +1 & \dots & -1 \\ -1 & \dots & +1 \\ +1 & \dots & -1 \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ +1 & \dots & -1 \end{array} \right] \Bigg/ \sqrt{K}$$

# RANDOM PROJECTION

- What does “it works” even mean?

Distances between all pairs of data-points in low dim. projection is roughly the same as their distances in the high dim. space.

That is, when  $K$  is “large enough”, with “high probability”, for all pairs of data points  $i, j \in \{1, \dots, n\}$ ,

$$(1 - \epsilon) \|\mathbf{y}_i - \mathbf{y}_j\|_2 \leq \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq (1 + \epsilon) \|\mathbf{y}_i - \mathbf{y}_j\|_2$$

# WHY SHOULD RANDOM PROJECTIONS EVEN WORK?!

Consider any vector  $\tilde{\mathbf{x}} \in \mathbb{R}^d$  and let  $\tilde{\mathbf{y}} = W^\top \tilde{\mathbf{x}}$ . Note that

$$\begin{aligned}\tilde{\mathbf{y}}[j]^2 &= \left( \sum_{i=1}^d W[i, j] \cdot \tilde{\mathbf{x}}[i] \right)^2 = \sum_{i, i'} (W[i, j] \cdot \tilde{\mathbf{x}}[i]) \cdot (W[i', j] \cdot \tilde{\mathbf{x}}[i']) \\ &= \sum_{i, i'} (W[i, j] \cdot W[i', j]) \cdot (\tilde{\mathbf{x}}[i] \cdot \tilde{\mathbf{x}}[i'])\end{aligned}$$

# WHY SHOULD RANDOM PROJECTIONS EVEN WORK?!

Hence,

$$\mathbb{E}[\tilde{\mathbf{y}}[j]^2] = \sum_{i,i'=1}^d \mathbb{E}[(W[i,j] \cdot W[i',j])] \cdot (\tilde{\mathbf{x}}[i] \cdot \tilde{\mathbf{x}}[i'])$$

if  $i \neq i'$ ,  $W[i,j]$  and  $W[i',j]$  are independent and so

$$\begin{aligned} &= \sum_{i=1}^d \mathbb{E}[(W[i,j]^2)] \tilde{\mathbf{x}}[i]^2 + \sum_{i \neq i'} (\mathbb{E}[W[i,j]] \cdot \mathbb{E}[W[i',j]]) \cdot (\tilde{\mathbf{x}}[i] \cdot \tilde{\mathbf{x}}[i']) \\ &= \sum_{i=1}^d \tilde{\mathbf{x}}[i]^2 / \sqrt{K}^2 = \|\tilde{\mathbf{x}}\|_2^2 / K \end{aligned}$$

# WHY SHOULD RANDOM PROJECTIONS EVEN WORK?!

Hence,

$$\mathbb{E}[\|\tilde{\mathbf{y}}\|_2^2] = \sum_{j=1}^K \mathbb{E}[\tilde{\mathbf{y}}[j]^2] = \sum_{j=1}^K \|\tilde{\mathbf{x}}\|_2^2 / K = \|\tilde{\mathbf{x}}\|_2^2$$

If we let  $\tilde{\mathbf{x}} = \mathbf{x}_s - \mathbf{x}_t$  then

$$\tilde{\mathbf{y}} = W^\top \tilde{\mathbf{x}} = W^\top \mathbf{x}_s - W^\top \mathbf{x}_t = \mathbf{y}_s - \mathbf{y}_t$$

Hence for any  $s, t \in \{1, \dots, n\}$ ,

$$\mathbb{E}[\|\mathbf{y}_s - \mathbf{y}_t\|_2^2] = \|\mathbf{x}_s - \mathbf{x}_t\|_2^2$$

Lets try this in Matlab ...

# WHY SHOULD RANDOM PROJECTIONS EVEN WORK?!

For large  $K$ , not only true in expectation but also with high probability

For any  $\epsilon > 0$ , if  $K \approx \log(n/\delta) / \epsilon^2$ , with probability  $1 - \delta$  over draw of  $W$ , for all pairs of data points  $i, j \in \{1, \dots, n\}$ ,

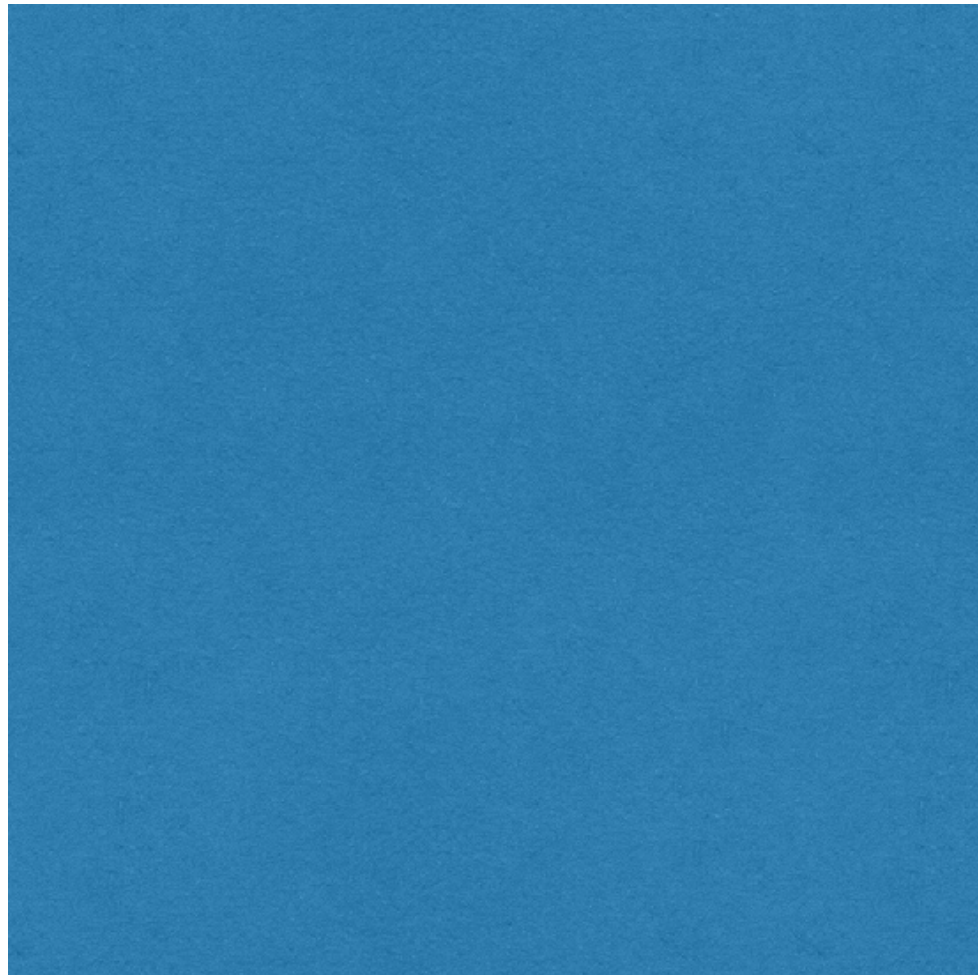
$$(1 - \epsilon) \|\mathbf{y}_i - \mathbf{y}_j\|_2 \leq \|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq (1 + \epsilon) \|\mathbf{y}_i - \mathbf{y}_j\|_2$$

Lets try on Matlab ...

This is called the Johnson-Lindenstrauss lemma or JL lemma for short.

# WHY IS THIS SO RIDICULOUSLY MAGICAL?

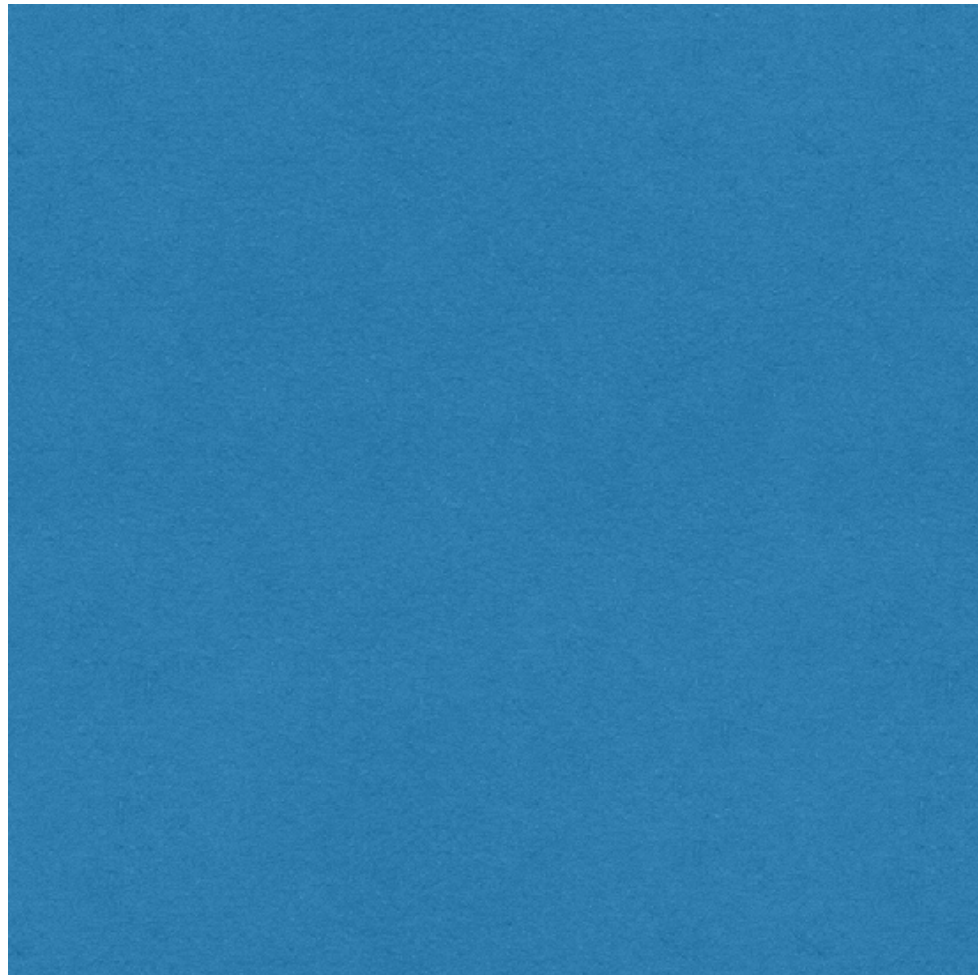
$n =$   
1000



$d = 1000$

# WHY IS THIS SO RIDICULOUSLY MAGICAL?

$n =$   
1000



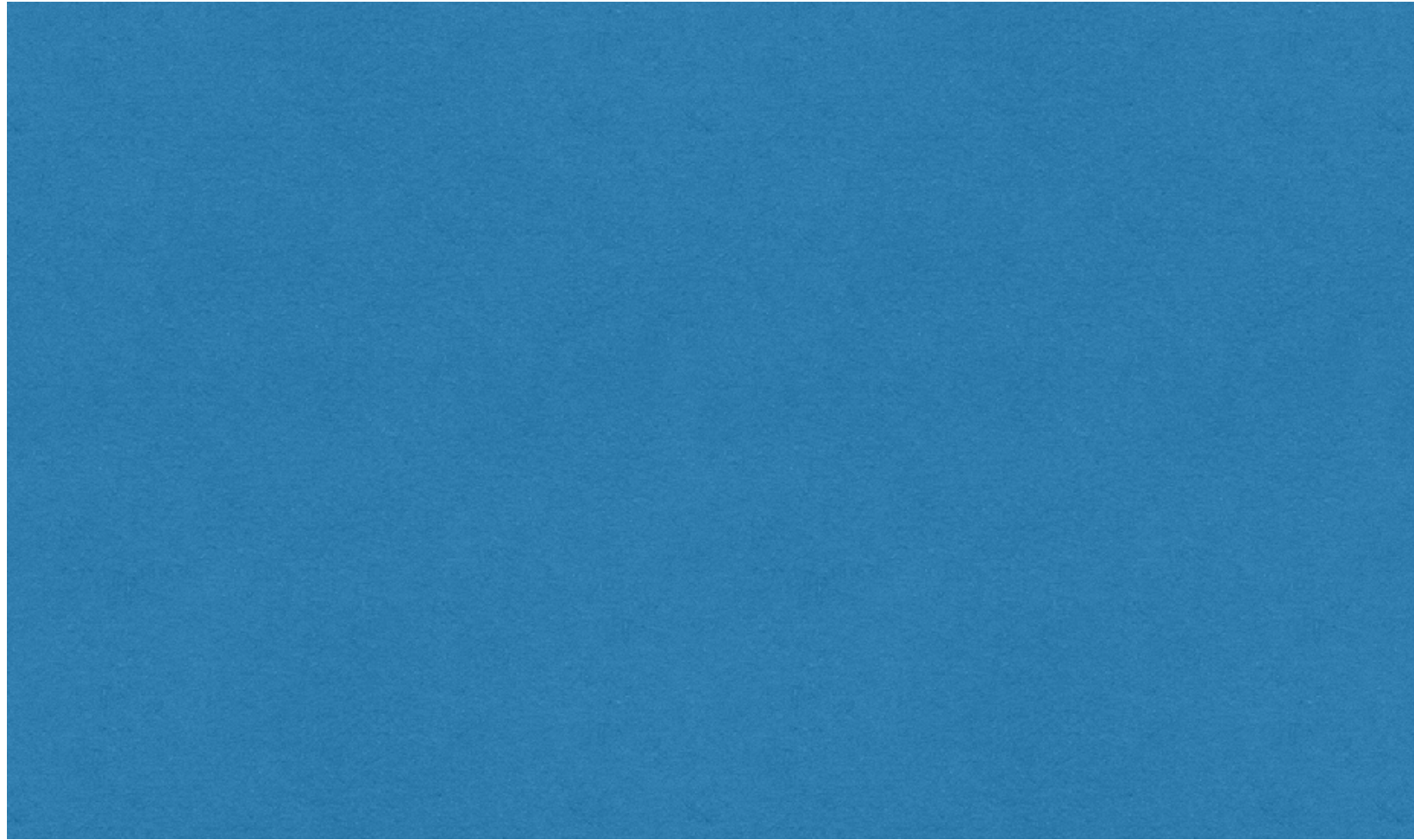
$d = 1000$

If we take  $K = 69.1/\epsilon^2$ , with probability 0.99 distances are preserved to accuracy  $\epsilon$



# WHY IS THIS SO RIDICULOUSLY MAGICAL?

$n =$   
1000



$d = 10000$

If we take  $K = 69.1/\epsilon^2$ , with probability 0.99 distances are preserved to accuracy  $\epsilon$

# WHY IS THIS SO RIDICULOUSLY MAGICAL?

$n =$   
1000

$d = 1000000$

If we take  $K = 69.1/\epsilon^2$ , with probability  
0.99 distances are preserved to accuracy  $\epsilon$