# Machine Learning for Data Science (CS4786)
# Lecture 10

PCA and Random Projections

Course Webpage :
http://www.cs.cornell.edu/Courses/cs4786/2017fa/

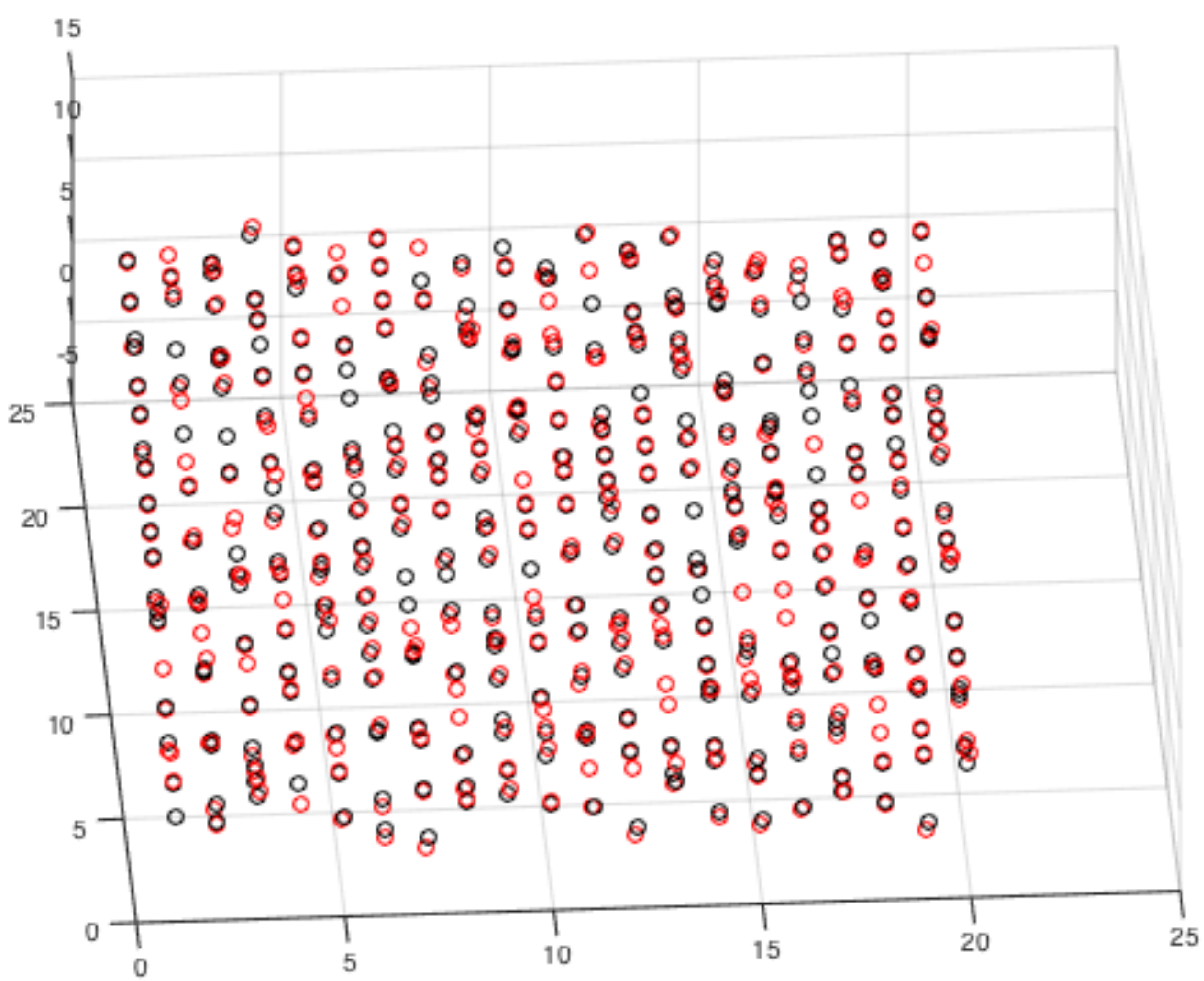
# Principal Component Analysis

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

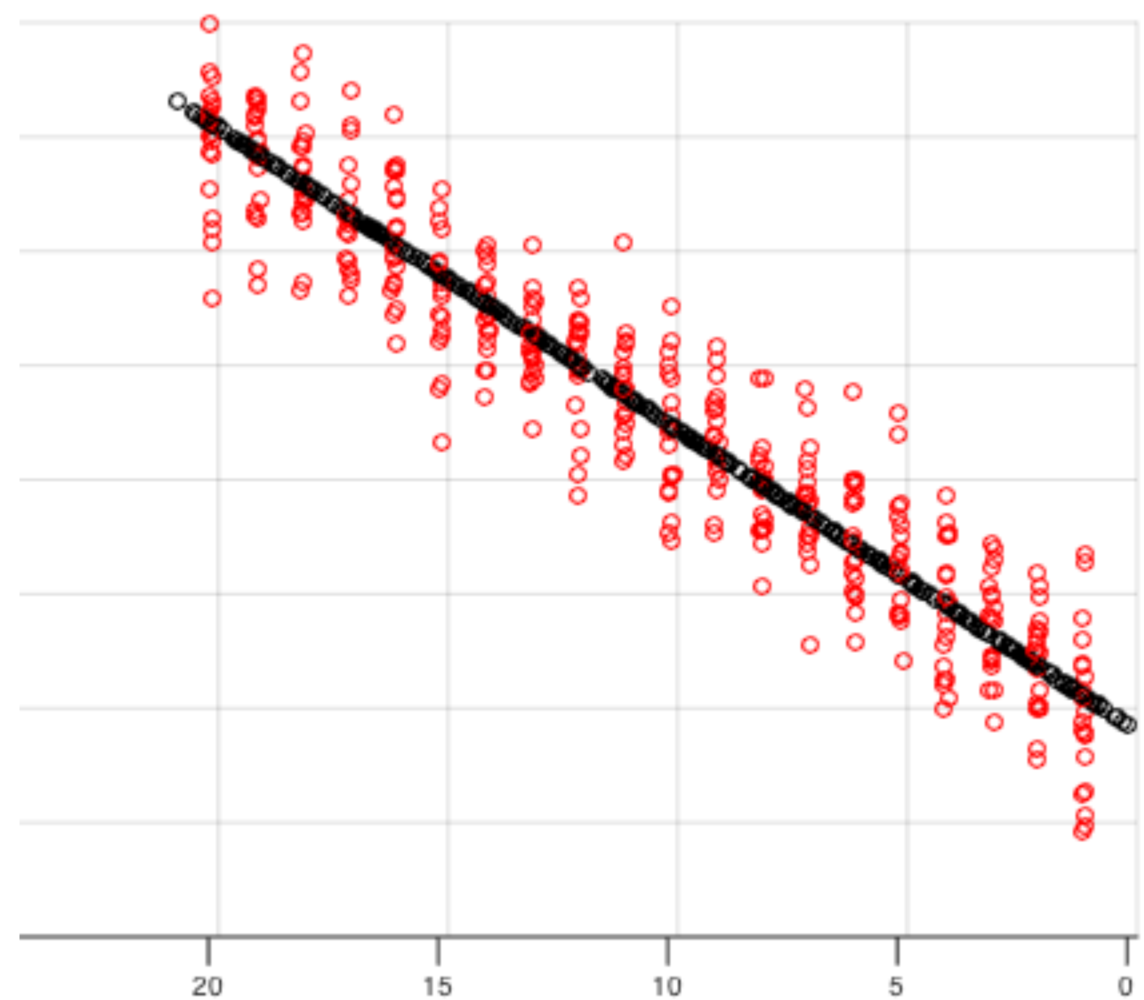
2. $$W = \mathrm{eigs}\left( \Sigma, K \right)$$

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

## Maximize Spread




## Minimize Reconstruction Error



# Demo

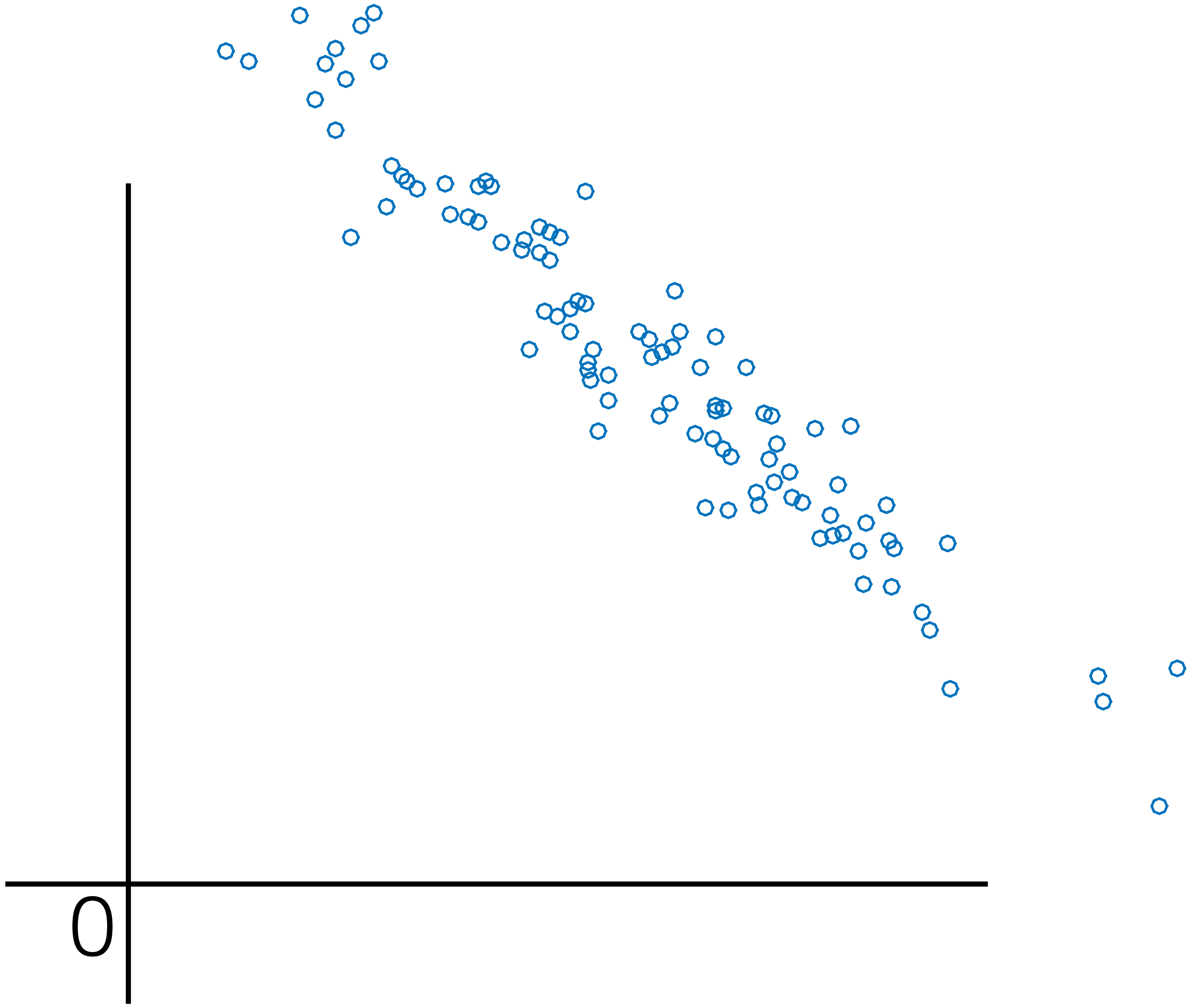
# Orthonormal Projections

- Think of $\mathbf{w}_1, \ldots, \mathbf{w}_K$ as coordinate system for PCA (in a $K$ dimensional subspace)
- $\mathbf{y}$ values provide coefficients in this system
- Without loss of generality, $\mathbf{w}_1, \ldots, \mathbf{w}_K$ can be orthonormal, i.e. $\mathbf{w}_i \perp \mathbf{w}_j$ & $\|\mathbf{w}_i\| = 1$.

$$\|\mathbf{w}_i\|_2^2 = \sum_{k=1}^{d} \mathbf{w}_i[k]^2$$

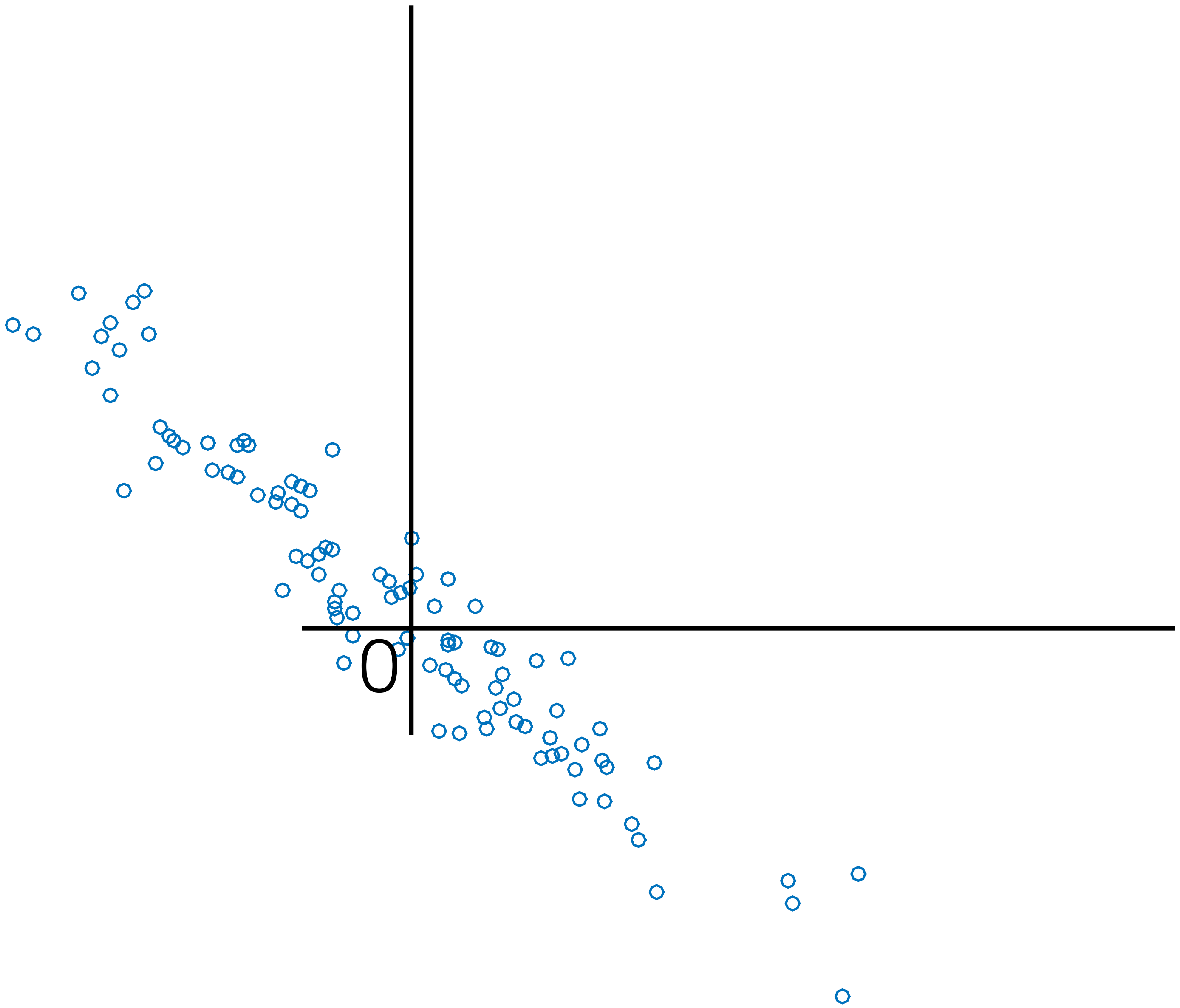
$$\mathbf{w}_i \perp \mathbf{w}_j \Rightarrow \sum_{k=1}^{d} \mathbf{w}_i[k]\mathbf{w}_j[k] = 0$$

# Centering Data




Compressing these data points…

# CENTERING DATA




… is same as compressing these.

# ORTHONORMAL PROJECTIONS

- (Centered) Data-points as linear combination of some orthonormal basis, i.e.

$$\mathbf{x}_t = \mu + \sum_{j=1}^{d} \mathbf{y}_t[j]\mathbf{w}_j$$

where $\mathbf{w}_1, \ldots, \mathbf{w}_d \in \mathbb{R}^d$ are the orthonormal basis and $\mu = \frac{1}{n}\sum_{t=1}^{n} x_t$.

- Represent data as linear combination of just $K$ orthonormal basis,

$$\hat{\mathbf{x}}_t = \mu + \sum_{j=1}^{K} \mathbf{y}_t[j]\mathbf{w}_j$$

# PCA: Minimizing Reconstruction Error

- Goal: find the basis that minimizes reconstruction error,

$$
\begin{aligned}
\sum_{t=1}^{n} \|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2^2 &= \sum_{t=1}^{n} \left\| \sum_{j=1}^{K} \mathbf{y}_t[j]\mathbf{w}_j + \mu - \mathbf{x}_t \right\|_2^2 \\
&= \sum_{t=1}^{n} \left\| \sum_{j=1}^{K} \mathbf{y}_t[j]\mathbf{w}_j + \mu - \sum_{j=1}^{d} \mathbf{y}_t[j]\mathbf{w}_j - \mu \right\|_2^2 \\
&= \sum_{t=1}^{n} \left\| \sum_{j=K+1}^{d} \mathbf{y}_t[j]\mathbf{w}_j \right\|_2^2 \qquad (\text{but } \|a+b\|_2^2 = \|a\|_2^2 + \|b\|_2^2 + 2a^\top b) \\
&= \sum_{t=1}^{n} \left( \sum_{j=K+1}^{d} \mathbf{y}_t[j]^2 \|\mathbf{w}_j\|_2^2 + 2 \sum_{j=K+1}^{d} \sum_{i=j+1}^{d} \mathbf{y}_t[j]\mathbf{y}_t[i]\mathbf{w}_j^\top \mathbf{w}_i \right) \\
&= \sum_{t=1}^{n} \sum_{j=k+1}^{d} \mathbf{y}_t[j]^2 \|\mathbf{w}_j\|_2^2 \qquad (\text{last step because } \mathbf{w}_j \perp \mathbf{w}_i)
\end{aligned}
$$

# PCA: Minimizing Reconstruction Error

$$
\begin{aligned}
\frac{1}{n}\sum_{t=1}^{n}\|\hat{\mathbf{x}}_t - \mathbf{x}_t\|_2^2 &= \frac{1}{n}\sum_{t=1}^{n}\sum_{j=k+1}^{d}\mathbf{y}_t[j]^2\|\mathbf{w}_j\|_2^2 \qquad (\text{but } \|\mathbf{w}_j\| = 1)\\
&= \frac{1}{n}\sum_{t=1}^{n}\sum_{j=k+1}^{d}\mathbf{y}_t[j]^2 \qquad (\text{now } \mathbf{y}_j = \mathbf{w}_j^\top(\mathbf{x}_t - \mu))\\
&= \frac{1}{n}\sum_{t=1}^{n}\sum_{j=k+1}^{d}\left(\mathbf{w}_j^\top(\mathbf{x}_t - \mu)\right)^2\\
&= \frac{1}{n}\sum_{t=1}^{n}\sum_{j=k+1}^{d}\mathbf{w}_j^\top(\mathbf{x}_t - \mu)(\mathbf{x}_t - \mu)^\top\mathbf{w}_j\\
&= \sum_{j=k+1}^{d}\mathbf{w}_j^\top\Sigma\mathbf{w}_j
\end{aligned}
$$

# PCA: Minimizing Reconstruction Error

Minimize w.r.t. $\mathbf{w}_1, \ldots, \mathbf{w}_K$'s that are orthonormal,

$$\underset{\forall j,\ \|\mathbf{w}_j\|_2 = 1}{\operatorname{argmin}} \sum_{j=k+1}^{d} \mathbf{w}_j^\top \Sigma \mathbf{w}_j$$

- Solution, (discard) $\mathbf{w}_{K+1}, \ldots, \mathbf{w}_d$ are bottom $d - K$ eigenvectors
- Hence $\mathbf{w}_1, \ldots, \mathbf{w}_K$ are the top $K$ eigenvectors

# Principal Component Analysis

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

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

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

# Reconstruction

4. $$\widehat{X} = Y \times W^\top + \mu$$

# When $d >> n$

- If $d >> n$ then $\Sigma$ is large
- But we only need top $K$ eigen vectors.
- Idea: use SVD

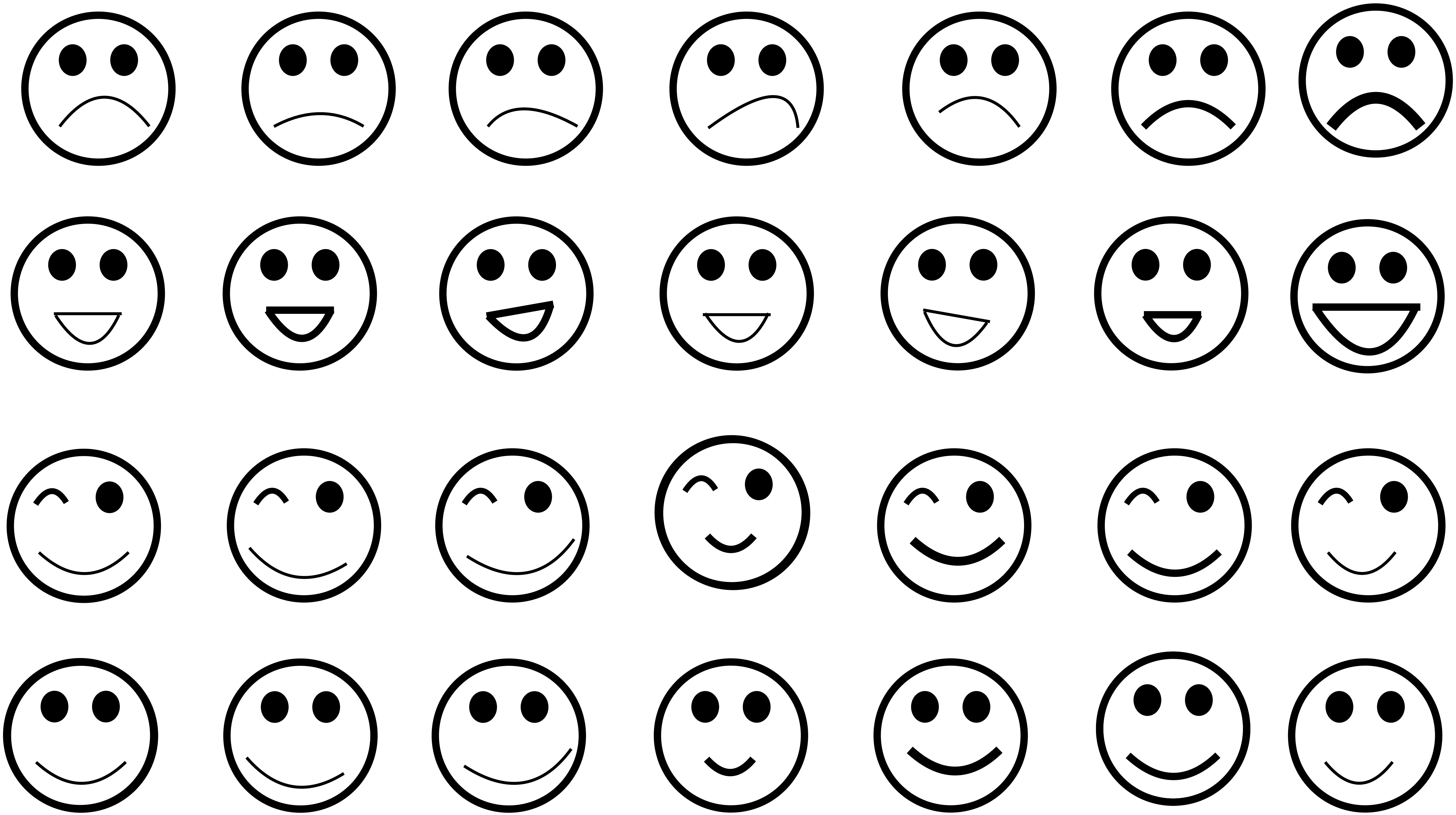
$$X - \mu = UDV^{\top} \qquad \begin{aligned} V^{\top}V &= I \\ U^{\top}U &= I \end{aligned}$$

Then note that, $\Sigma = (X-\mu)^{\top}(X-\mu) = VD^2V$

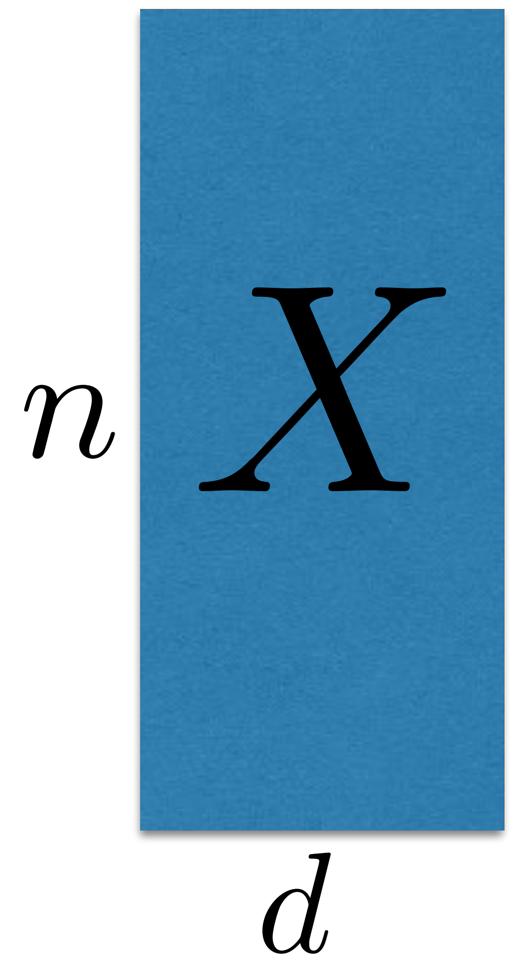
- Hence, matrix $V$ is the same as matrix $W$ got from eigen decomposition of $\Sigma$, eigenvalues are diagonal elements of $D^2$
- Alternative algorithm:

$$[U, V] = \mathrm{SVD}(X-\mu, K) \quad W = V$$

# Principal Component Analysis: Demo



# The Tall, the Fat and the Ugly

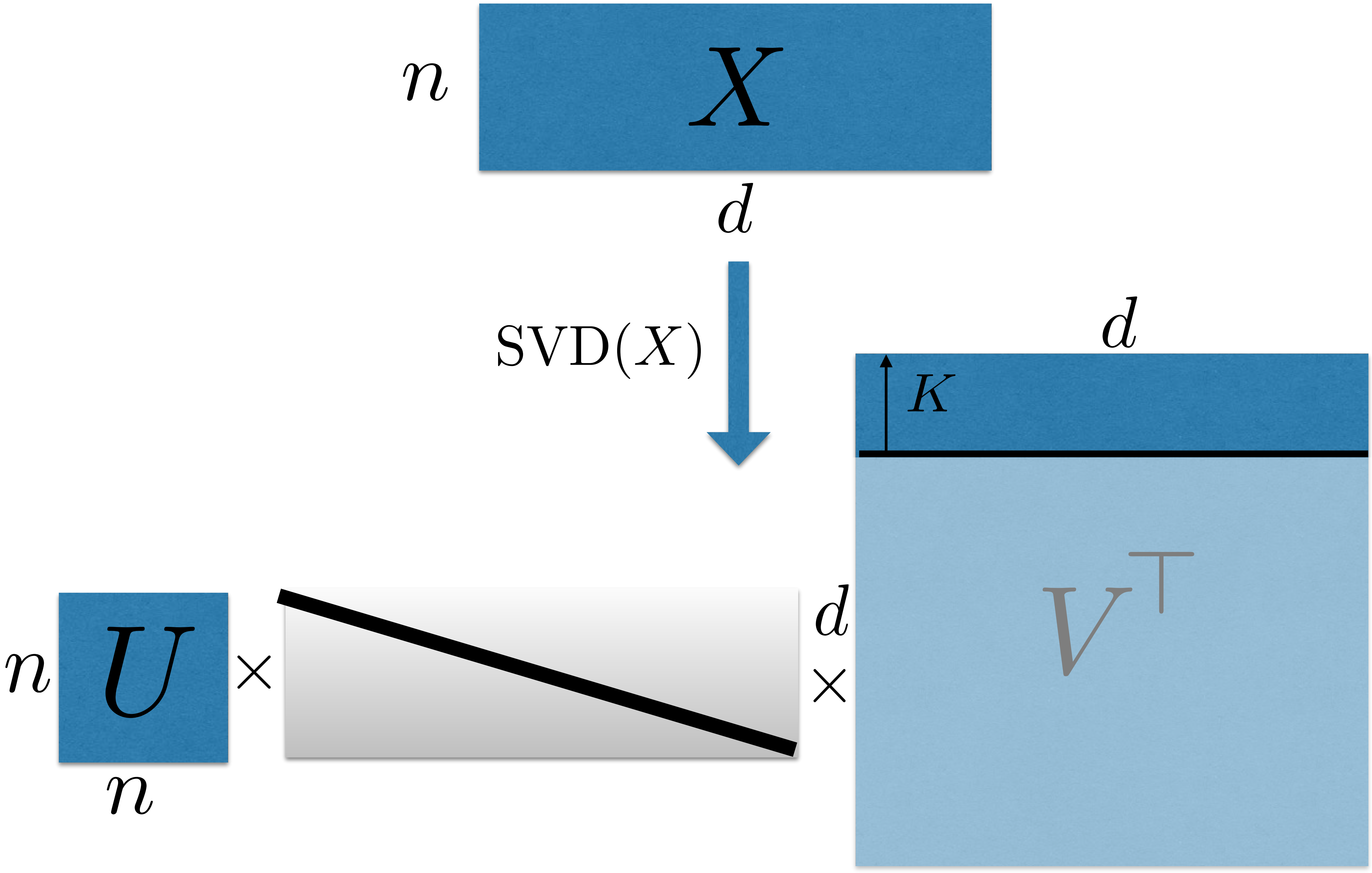


# The Tall, the Fat and the Ugly

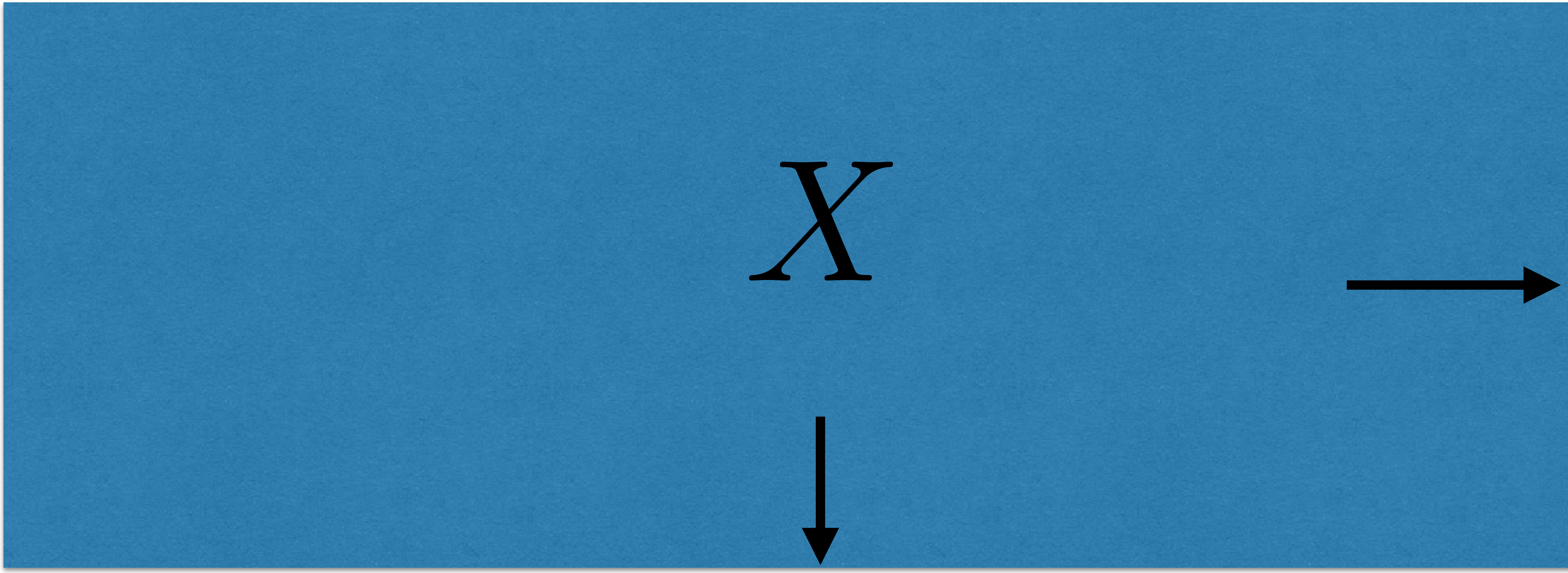
$$ {}_{d}\underset{n}{X^{\top}} \times {}_{n}\underset{d}{X} \Big/ n = {}_{d}\overset{d}{\Sigma} $$

$$ {}_{d}\underset{K}{W} = \mathrm{Eigs}\left(\Sigma, K\right) $$

# THE TALL, **the Fat** AND THE UGLY



# The Tall, the Fat and **the Ugly**




- $d$ 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

$$Y = X \times \left. \begin{array}{c} \begin{bmatrix} +1 & \dots & -1 \\ -1 & \dots & +1 \\ +1 & \dots & -1 \\ & \vdots & \\ +1 & \dots & -1 \end{bmatrix} d \\ K \end{array} \right/ \sqrt{K}$$

# Why should Random Projections even work?!

# Random Projection

- What does “it works” even mean?

# 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 $i, j \in \{1, \ldots, n\}$,

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

# Why should Random Projections even work?!

Say $K = 1$. Consider any vector $\tilde{\mathbf{x}} \in \mathbb{R}^d$ and let $\tilde{\mathbf{y}} = \tilde{\mathbf{x}}\, W$. Note that

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

However $W^2[i,1] = 1/K = 1$ when $K = 1$

$$= \sum_{i=1}^{d} \tilde{\mathbf{x}}^2[i] + \sum_{i'>i} \left(W[i,1] \cdot W[i',1]\right) \cdot \left(\tilde{\mathbf{x}}[i] \cdot \tilde{\mathbf{x}}[i']\right)$$

# WHY SHOULD RANDOM PROJECTIONS EVEN WORK?!

Hence,

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

However $W[i,1]$ and $W[i',1]$ are independent and so

$$\mathbb{E}\left[W[i,1] \cdot W[i',1]\right] = \mathbb{E}[W[i,1]] \cdot \mathbb{E}\left[W[i',1]\right] = 0$$

Using this we conclude that

$$\mathbb{E}\left[\tilde{\mathbf{y}}^2\right] = \sum_{i=1}^{d} \tilde{\mathbf{x}}^2[i] = \|\tilde{\mathbf{x}}\|^2$$

# Why should Random Projections even work?!

Hence,

$$\mathbb{E}\left[|\tilde{\mathbf{y}}|^2\right] = \|\tilde{\mathbf{x}}\|_2^2$$

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

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

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

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

Lets try this in Matlab . . .

# Why should Random Projections even work?!

- Setting $K$ large is like getting $K$ samples.
- Specifically since we take W to be random signs normalized by $\sqrt{K}$, for each $j \in [K]$, for any $\tilde{\mathbf{x}}$ if $\tilde{\mathbf{y}} = \tilde{\mathbf{x}}\, W$, then

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

Hence we can conclude that

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

This is like taking an average of $K$ independent measurements whose expectations are $\|\tilde{\mathbf{x}}\|_2^2$

# 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 $i, j \in \{1, \ldots, n\}$,

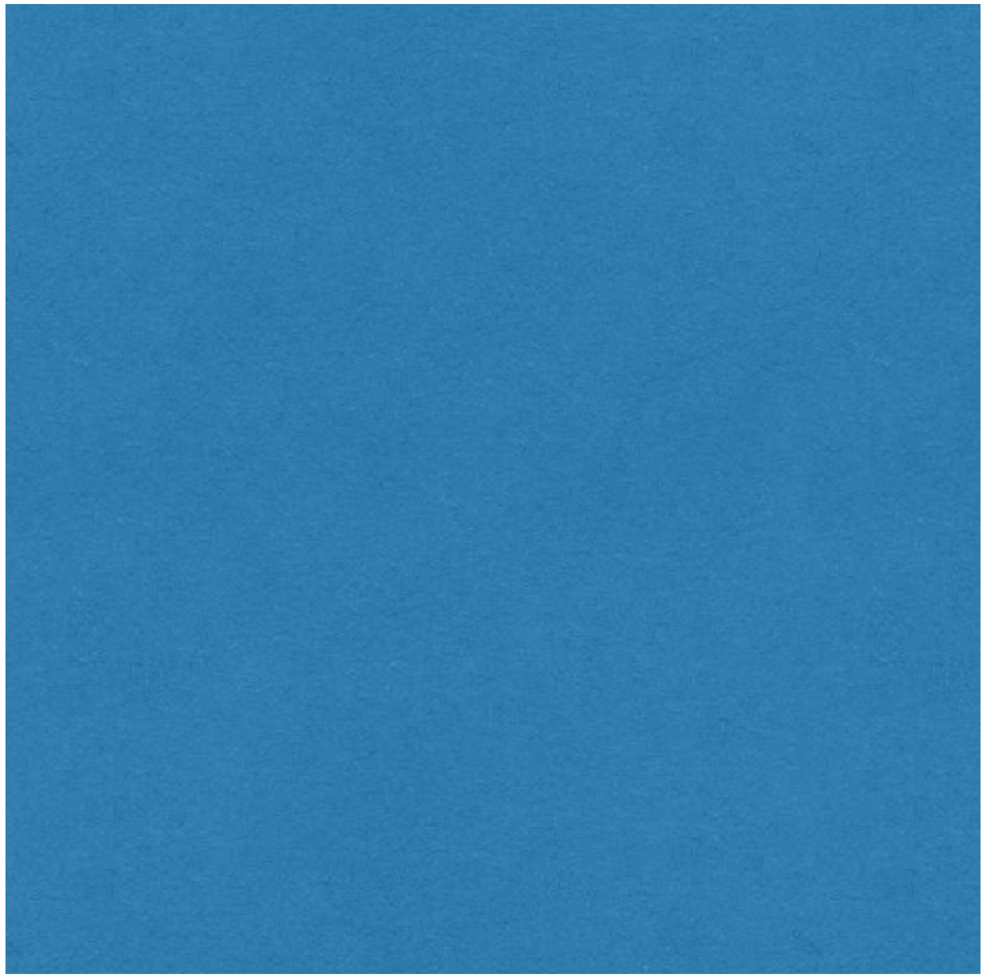
$$(1-\epsilon)\left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 \leq \left\|\mathbf{x}_i - \mathbf{x}_j\right\|_2 \leq (1+\epsilon)\left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2$$

Lets try on Matlab ...

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

# Why is this so Ridiculously Magical?

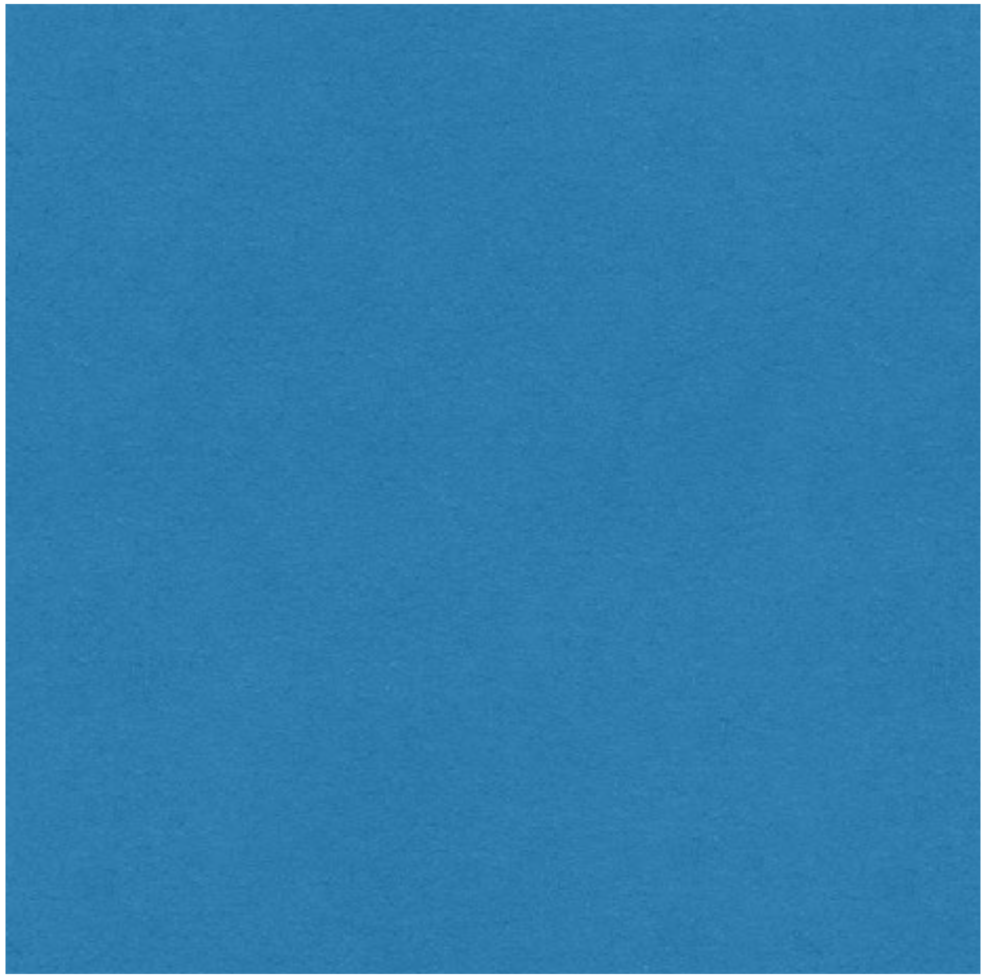
n=
1000



d = 1000

# WHY IS THIS SO RIDICULOUSLY MAGICAL?

n=
1000

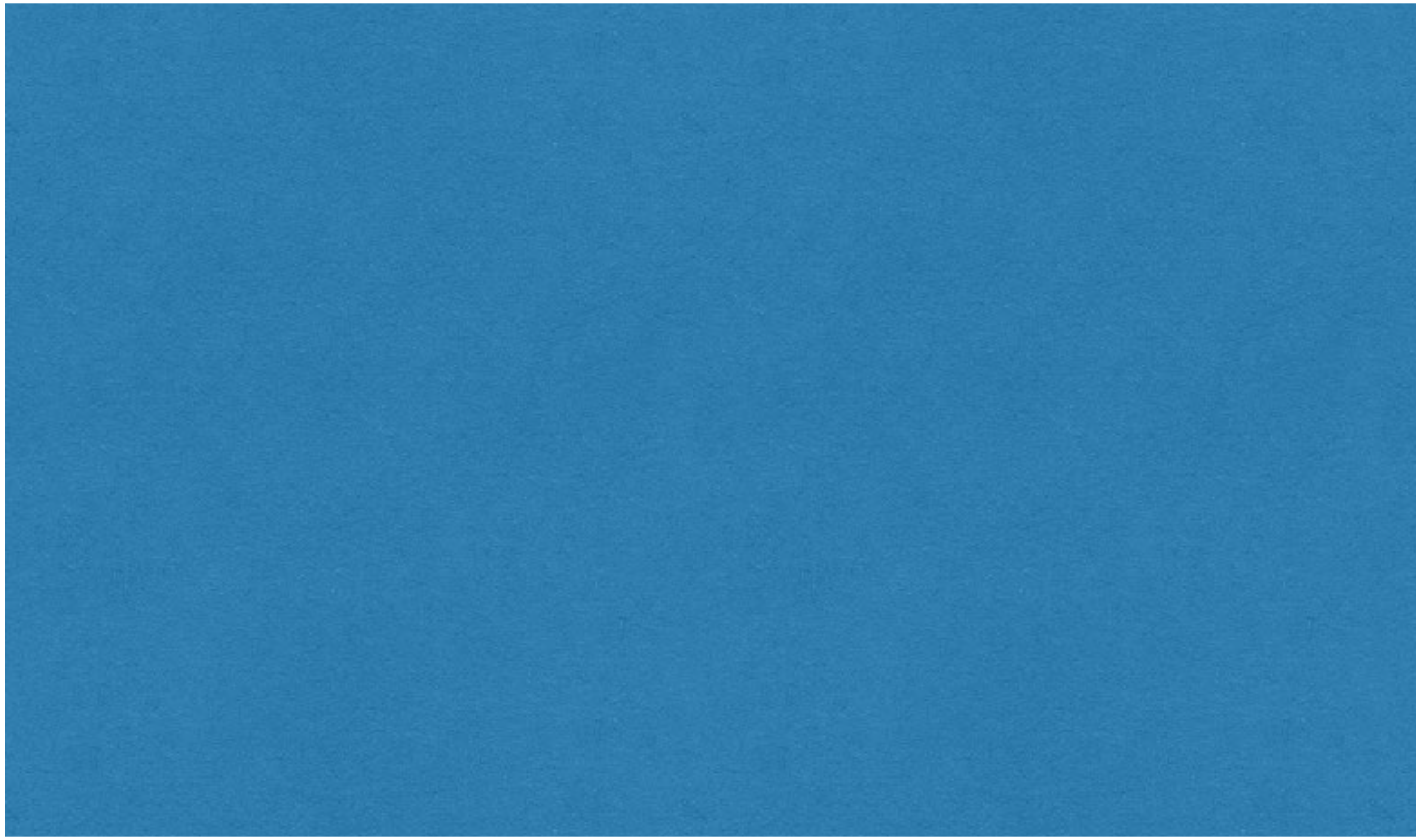


d = 1000

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

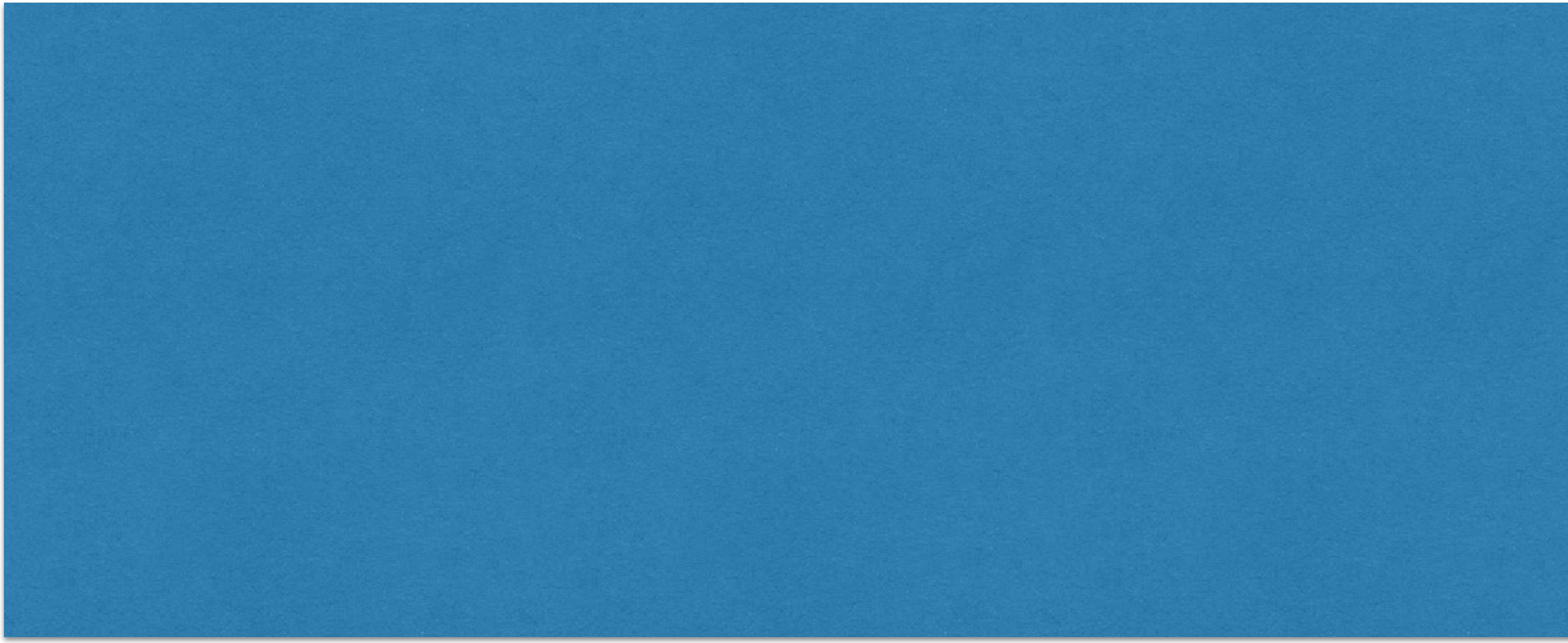


d = 10000

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 $\epsilon$