

Data Modelling Methods-V

CS771: Introduction to Machine Learning
Purushottam Kar



Outline of today's discussion

- Recap of PCA, PPCA
- Details of PCA and applications

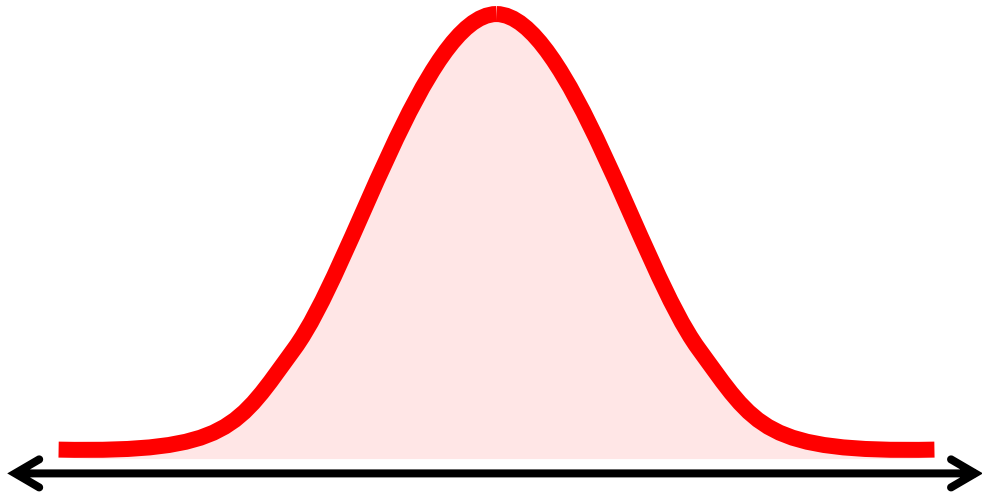
Recap

Modelling Low-dimensional structure in data

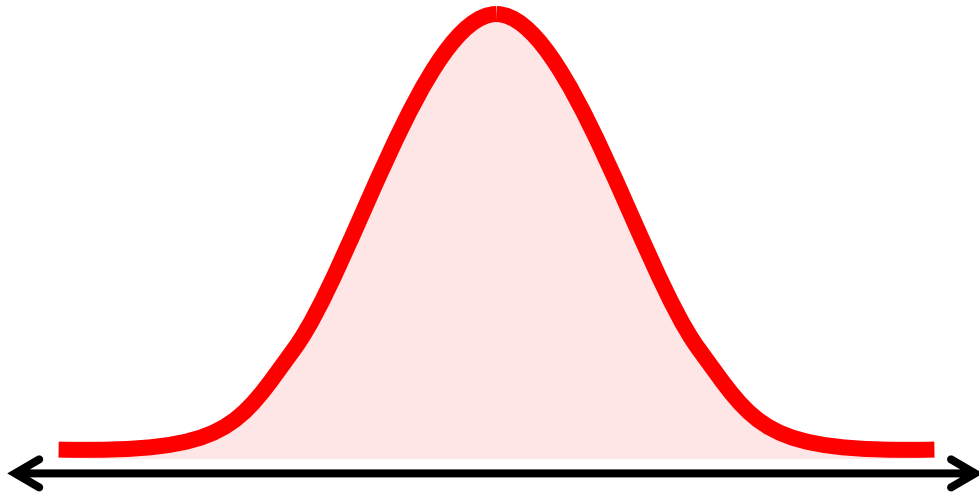
Low-dimensional Structure in Data



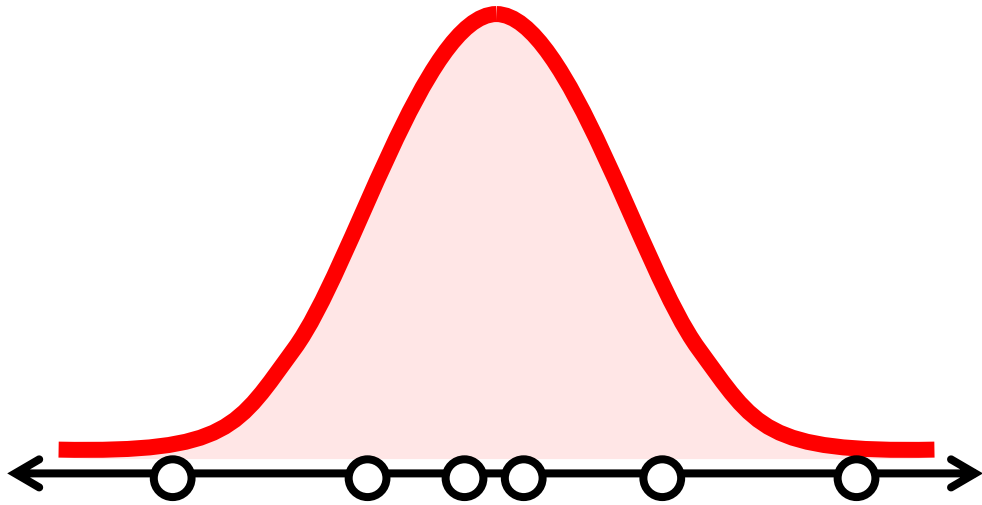
Low-dimensional Structure in Data



Low-dimensional Structure in Data



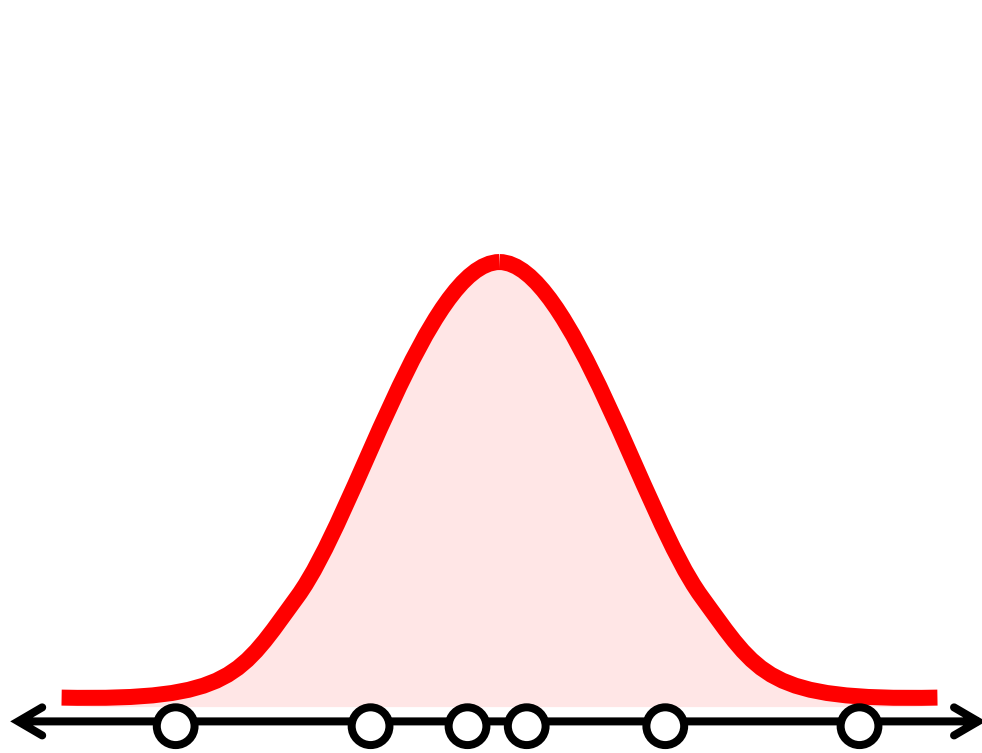
Low-dimensional Structure in Data



$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$



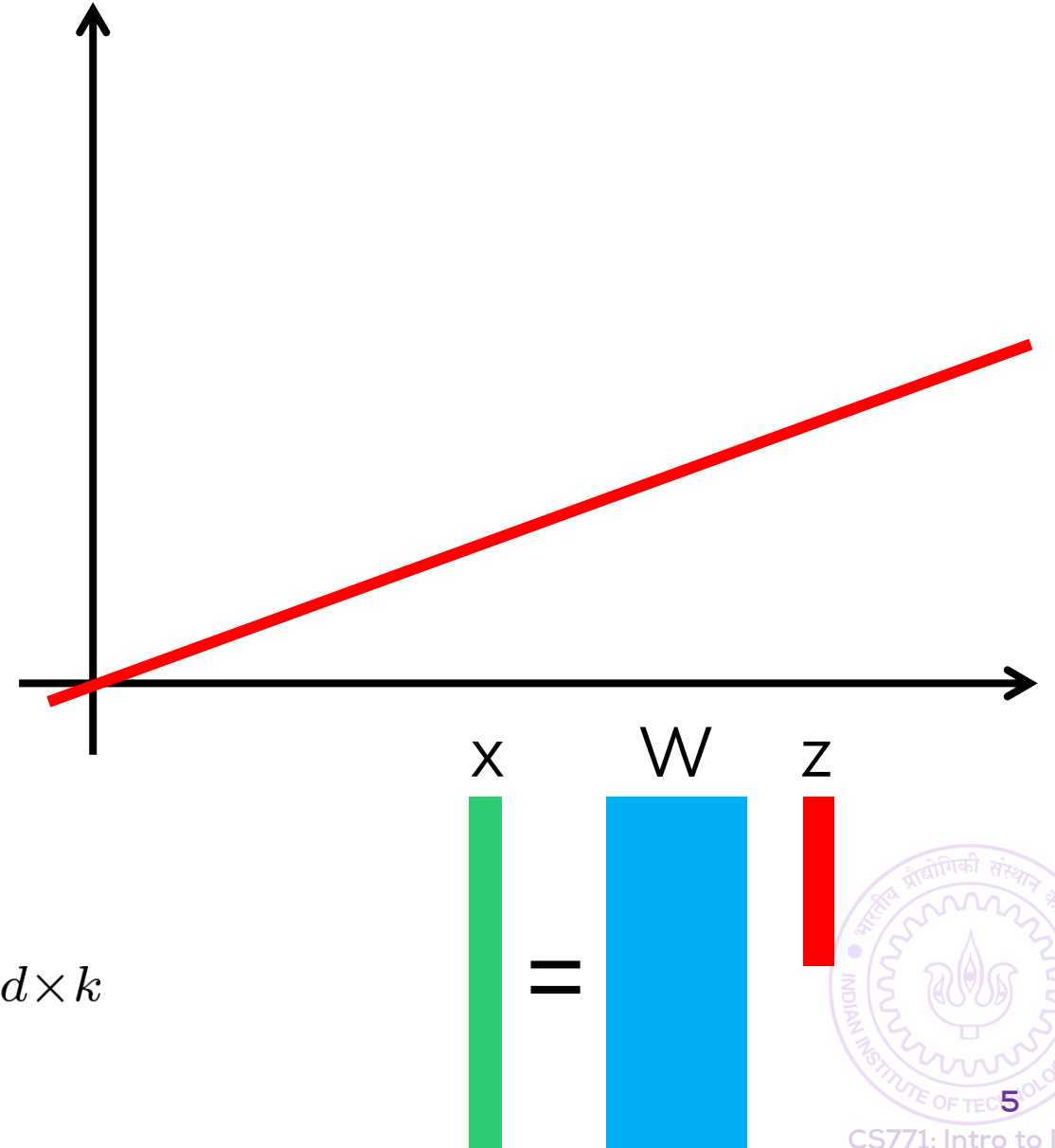
Low-dimensional Structure in Data



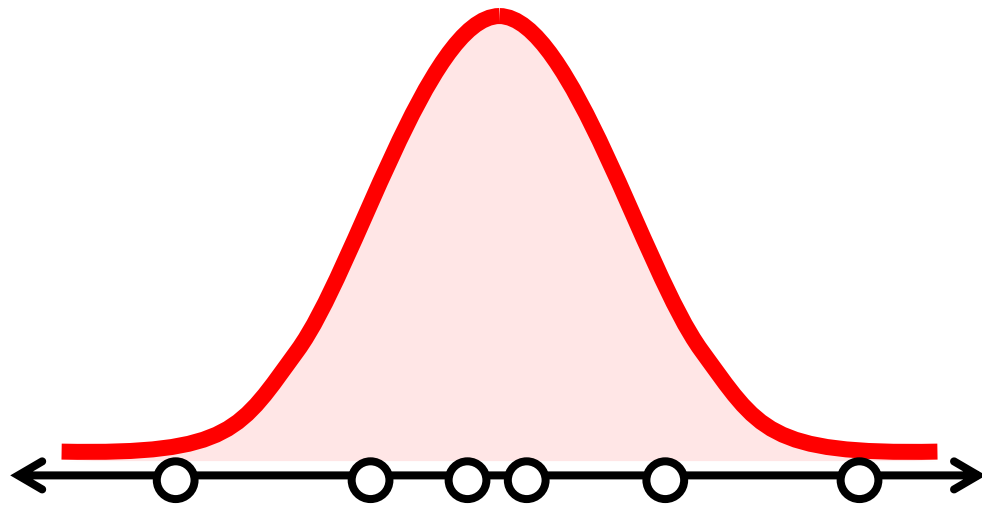
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W \mathbf{z}^i$$

$$W \in \mathbb{R}^{d \times k}$$



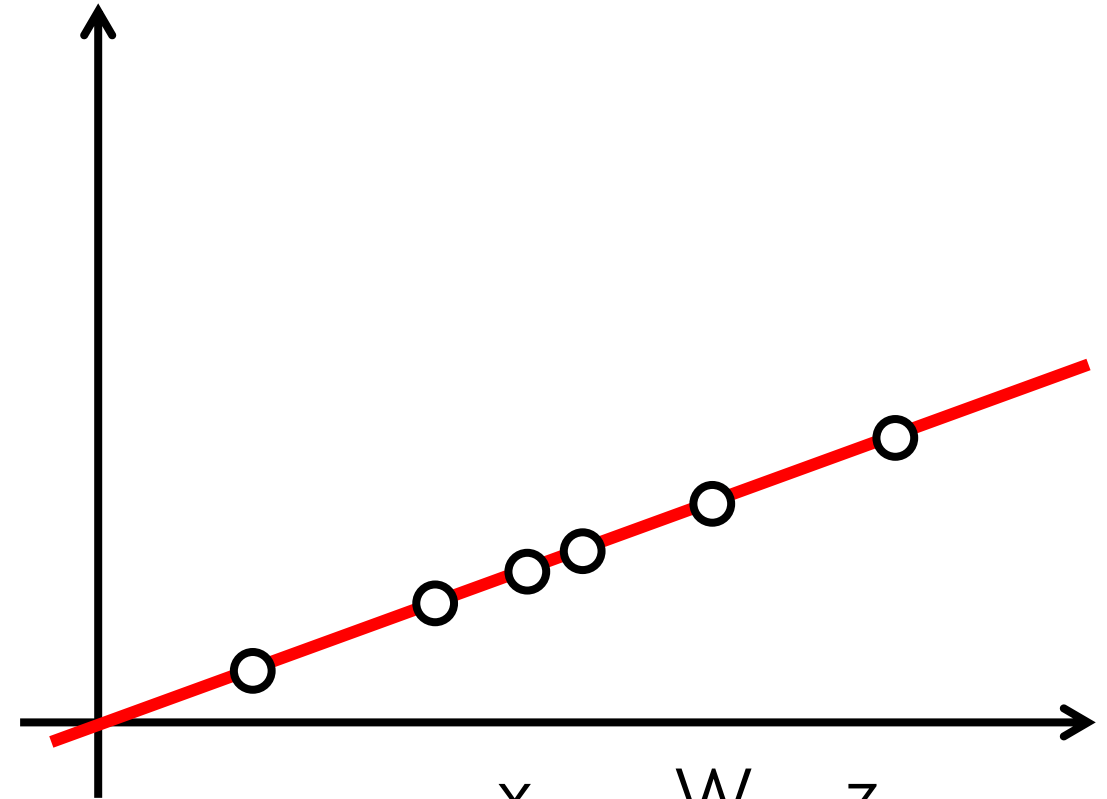
Low-dimensional Structure in Data



$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W \mathbf{z}^i$$

$$W \in \mathbb{R}^{d \times k}$$



\mathbf{x}



$=$

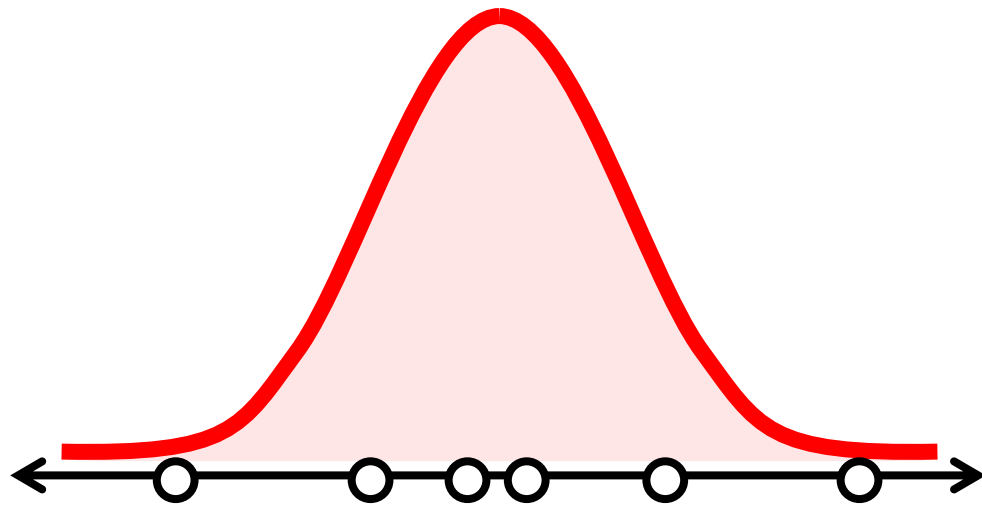
W



\mathbf{z}



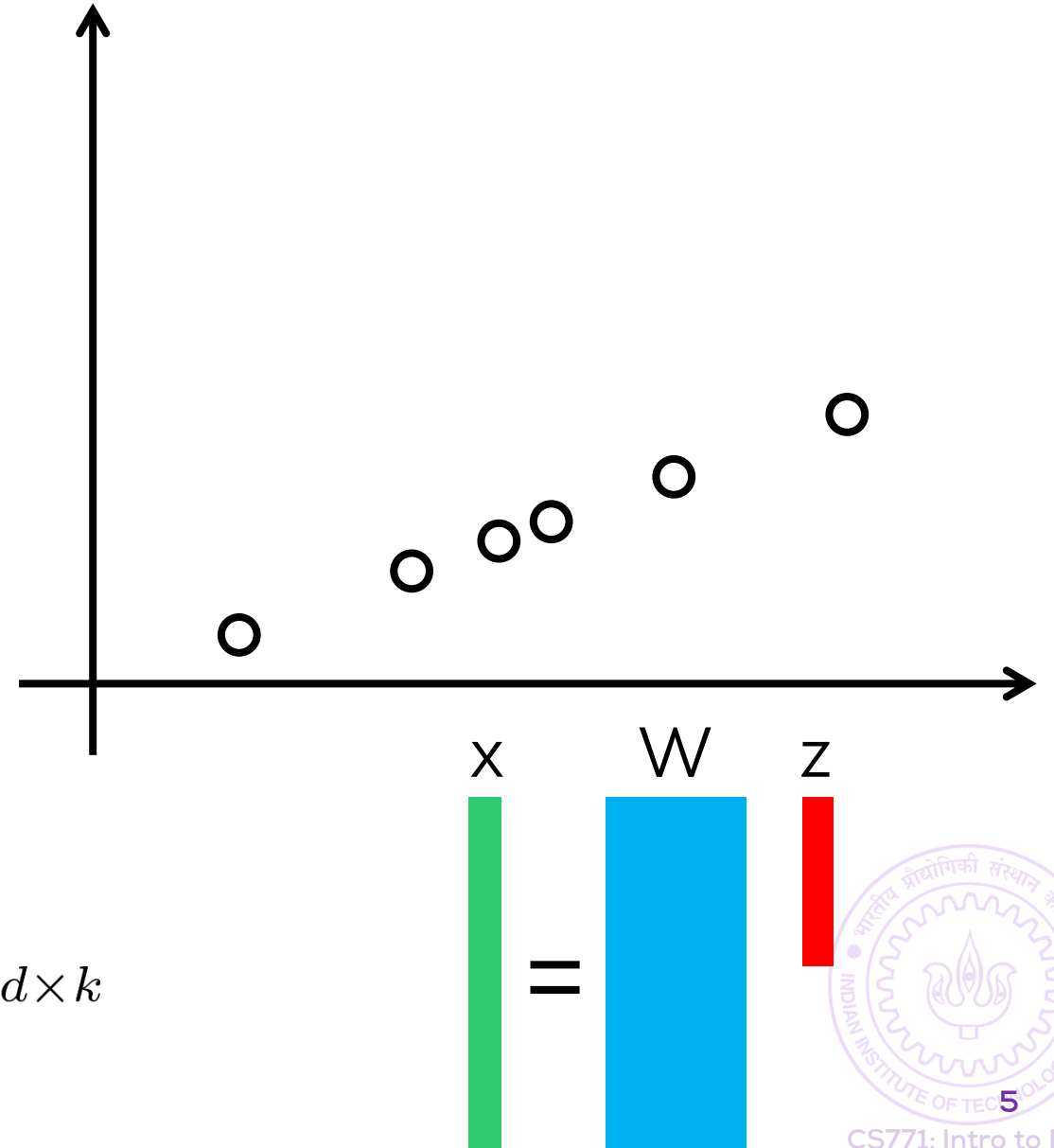
Low-dimensional Structure in Data



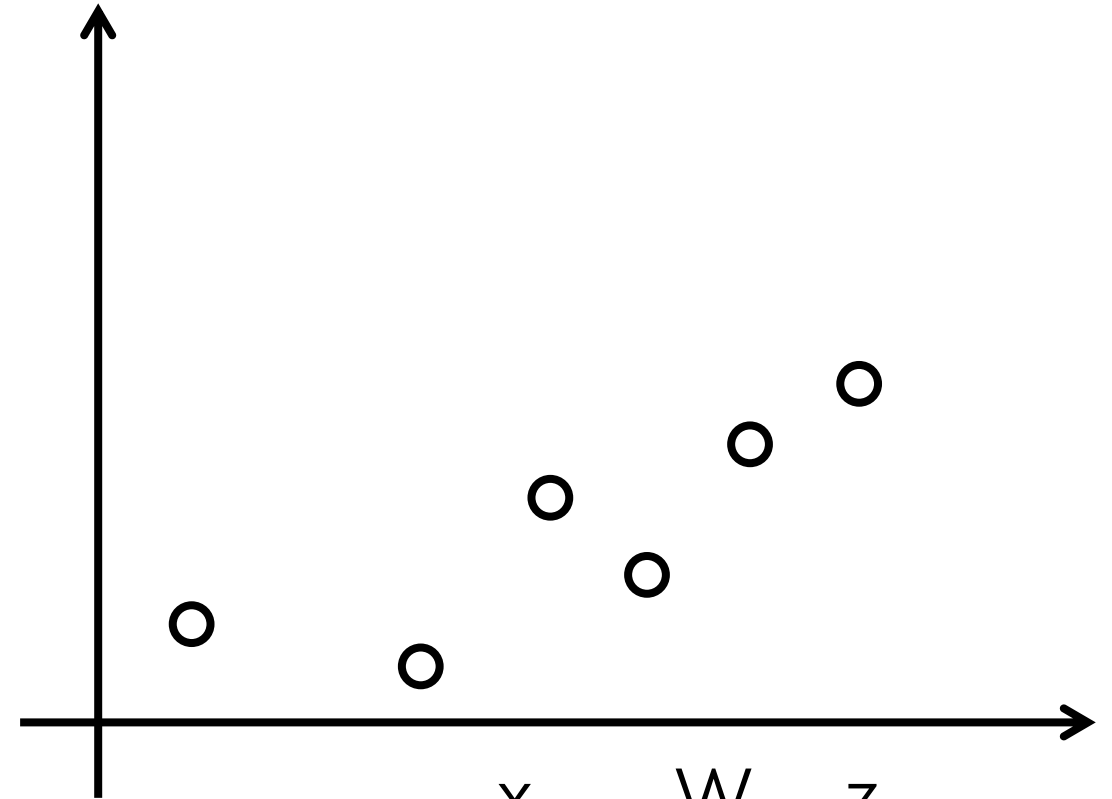
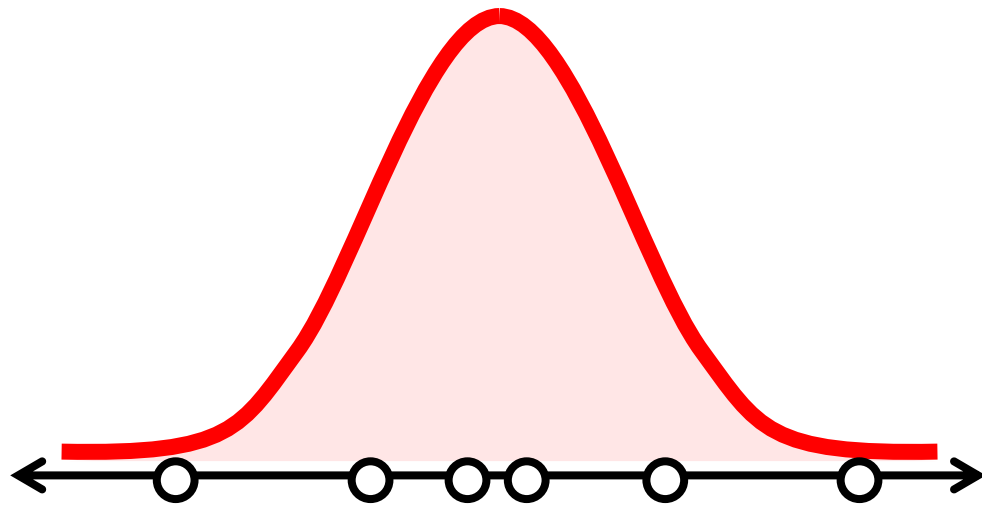
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W \mathbf{z}^i$$

$$W \in \mathbb{R}^{d \times k}$$



Low-dimensional Structure in Data

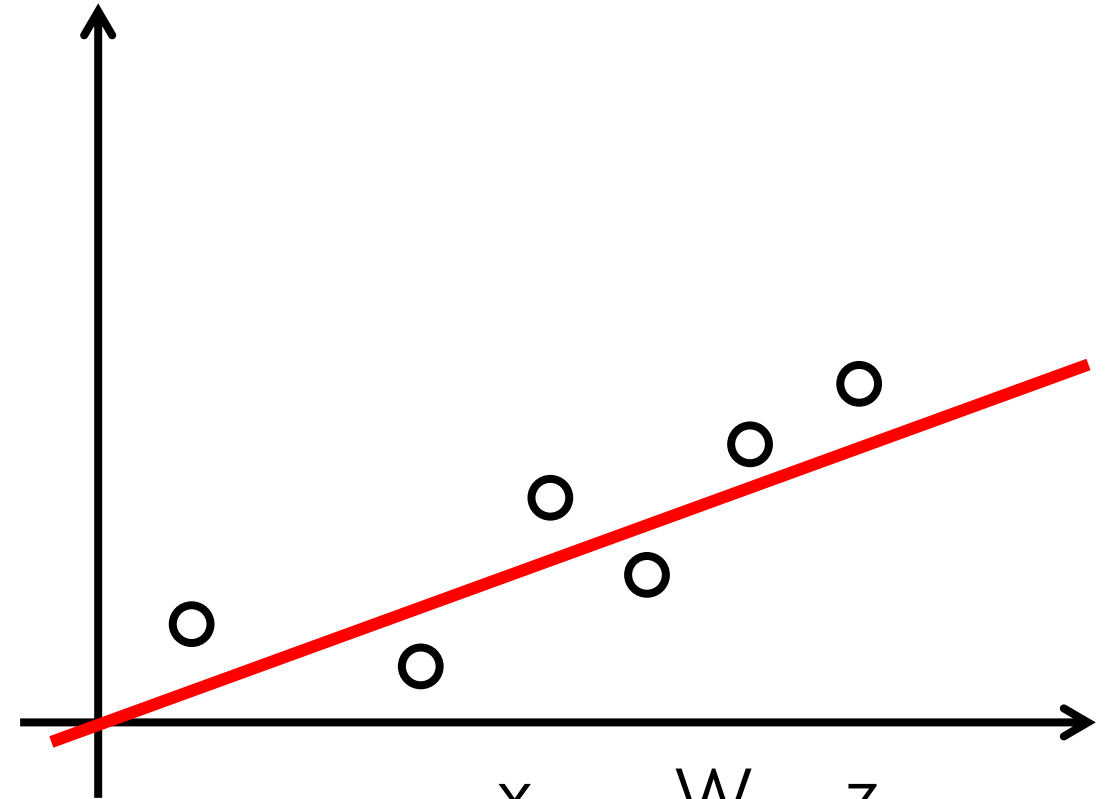
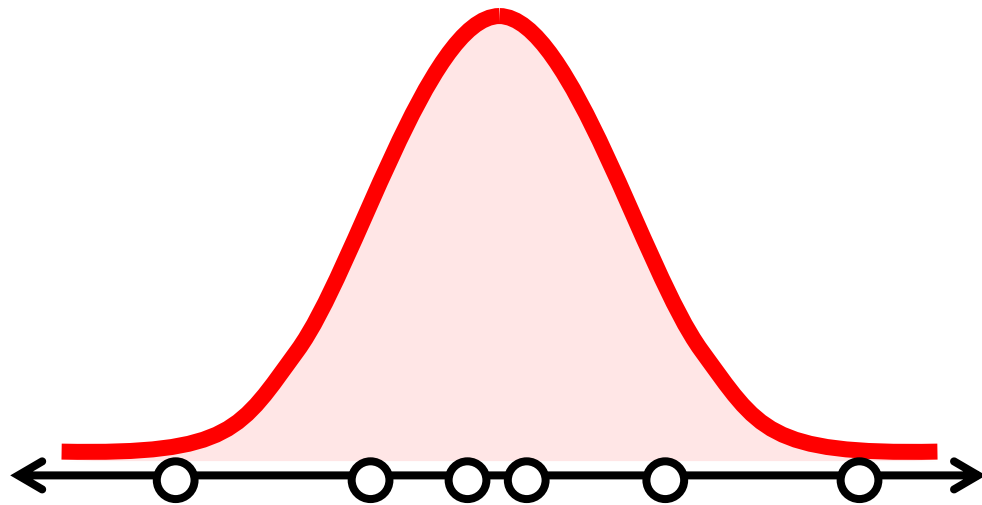


$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i, \quad \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

$$W \in \mathbb{R}^{d \times k}$$

Low-dimensional Structure in Data



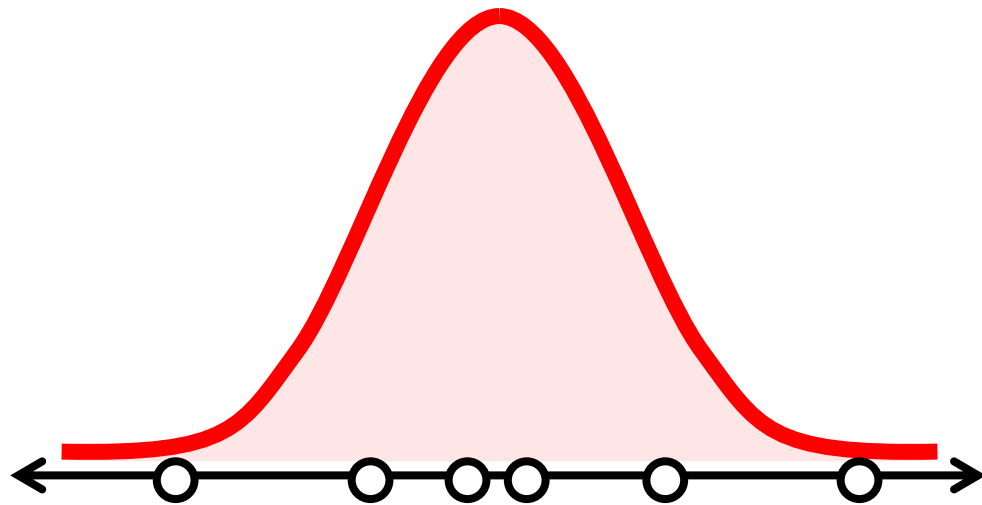
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i, \quad \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

$$W \in \mathbb{R}^{d \times k}$$

$$\begin{array}{c} \mathbf{x} \\ \text{green bar} \end{array} = \begin{array}{c} W \\ \text{blue bar} \end{array} \begin{array}{c} \mathbf{z} \\ \text{red bar} \end{array}$$

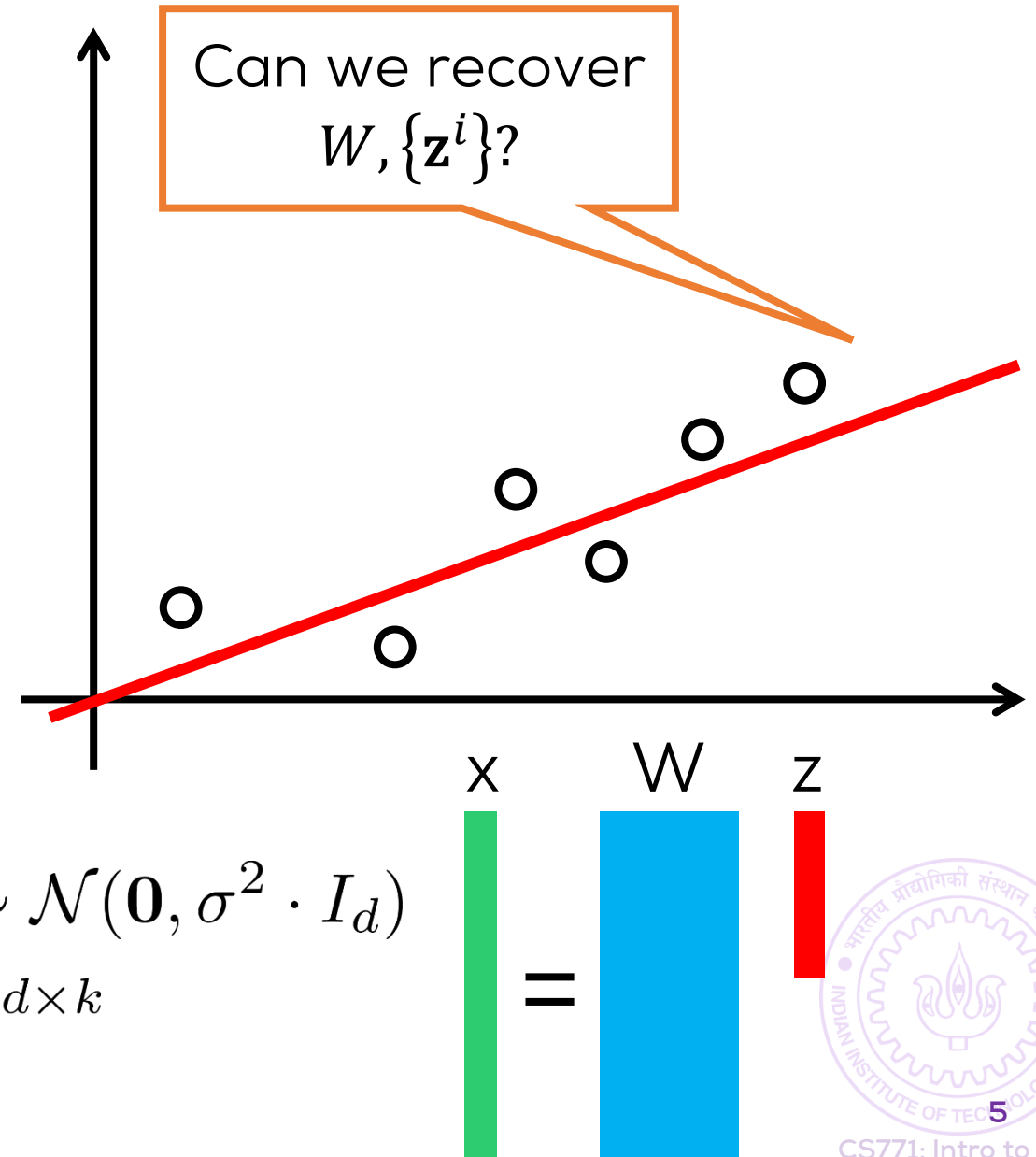
Low-dimensional Structure in Data



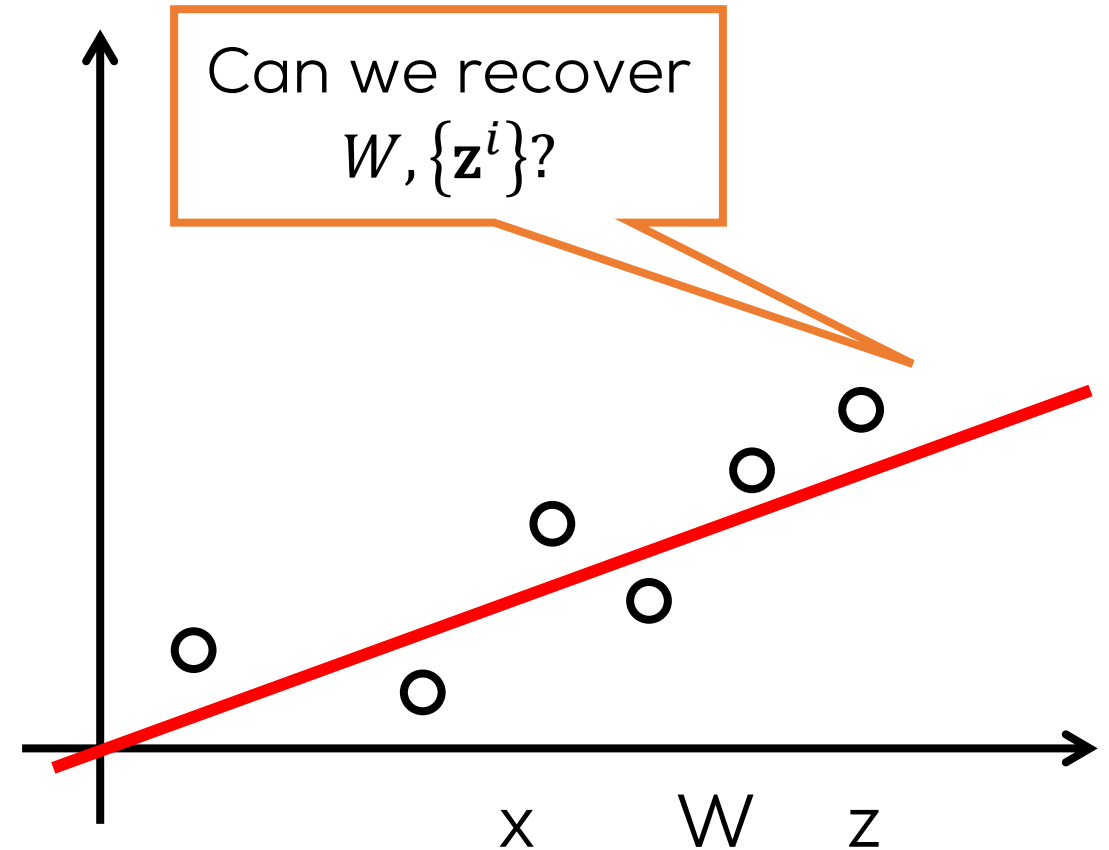
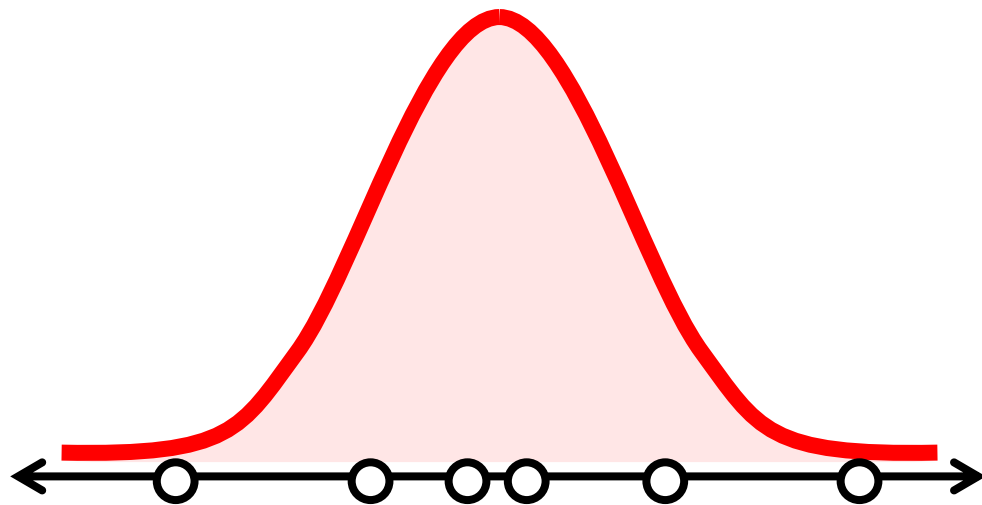
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i, \quad \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

$$W \in \mathbb{R}^{d \times k}$$



Low-dimensional Structure in Data



$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \quad \mathbf{x}^i = W \mathbf{z}^i + \boldsymbol{\epsilon}^i, \quad \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

Dictionary/Factor
Loading matrix

$$W \in \mathbb{R}^{d \times k}$$

Discovering low-dim structure is super-useful

- Space savings: store k -dim \mathbf{z}^i instead of d -dim \mathbf{x}^i , $k \ll d$
- Discover meaningful structure in data captured by W

Discovering low-dim structure is super-useful

- Space savings: store k -dim \mathbf{z}^i instead of d -dim \mathbf{x}^i , $k \ll d$
- Discover meaningful structure in data captured by W

Original Collection of Images



Credits: Piyush Rai, CS771, 2016-17-I



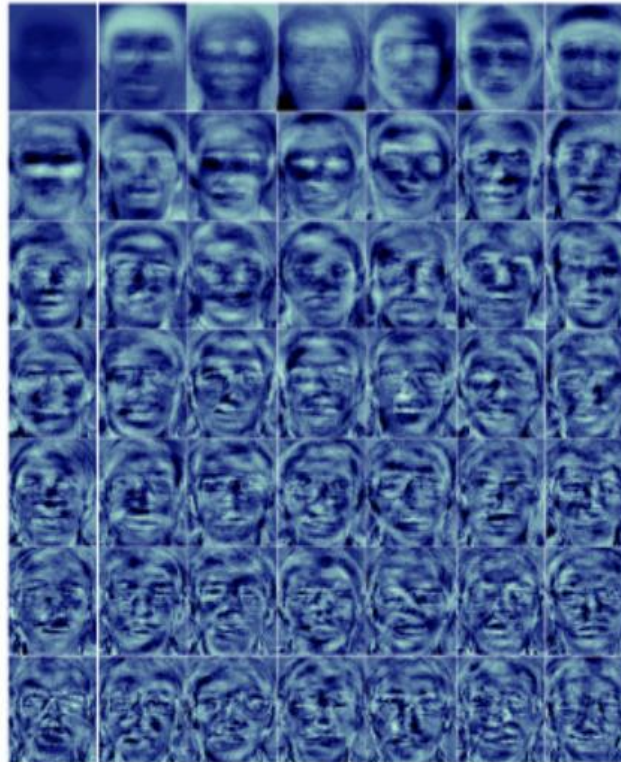
Discovering low-dim structure is super-useful

- Space savings: store k -dim \mathbf{z}^i instead of d -dim \mathbf{x}^i , $k \ll d$
- Discover meaningful structure in data captured by W

Original Collection of Images



K=49 Eigenvectors
("eigenfaces") learned
by PCA on this data



[7-1]



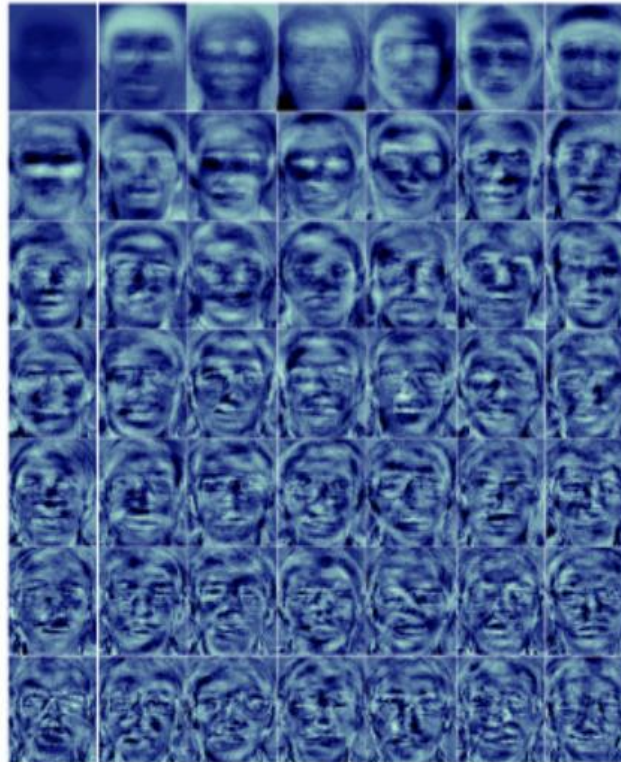
Discovering low-dim structure is super-useful

- Space savings: store k -dim \mathbf{z}^i instead of d -dim \mathbf{x}^i , $k \ll d$
- Discover meaningful structure in data captured by W

Original Collection of Images



K=49 Eigenvectors
("eigenfaces") learned
by PCA on this data

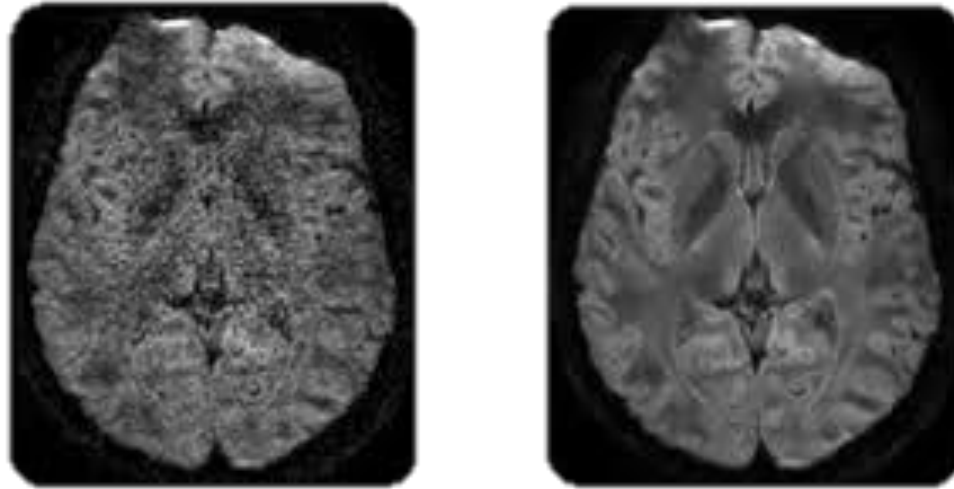


Each image's reconstructed version



Discovering low-dim structure is super-useful

- Noise removal: low-dim \mathbf{z}^i contains all useful info, rest is noise



- Or ... the “noise” could be the useful stuff (fore/background sep)



What we have and what we want

- In secret, someone generates a low-dim data $\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \in \mathbb{R}^k$
- The point is projected into a high-dim space $W\mathbf{z}^i \in \mathbb{R}^d$ ($W \in \mathbb{R}^{d \times k}$)
- Noise is added to the point $\mathbf{x}^i | \mathbf{z}^i \sim \mathcal{N}(W\mathbf{z}^i, \sigma^2 \cdot I_d)$
- We get to see $\mathbf{x}^i, i = 1 \dots n$ but only for say $n \approx dk$ points
- Want to recover W, σ and \mathbf{z}^i
- It turns out that likelihood $\mathbb{P}[\mathbf{x}^i | \sigma, W] = \mathcal{N}(0, \Sigma)$ where $\Sigma = \sigma^2 \cdot I_d + WW^\top \in \mathbb{R}^{d \times d}$
- However, estimating Σ directly is bad
 - Too many samples $n \approx d^2$ required
 - Not clear how to extract W, σ from Σ

An MLE Estimate for W, σ

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is

$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log |C| + \text{tr}(C^{-1}S))$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$

- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}$, $\lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [\mathbf{u}^1, \dots, \mathbf{u}^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j$

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is

$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log |C| + \text{tr}(C^{-1}S))$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$

- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}$, $\lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [\mathbf{u}^1, \dots, \mathbf{u}^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j$

Need to find top k eigenvectors and all the eigenvalues

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is

$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log |C| + \text{tr}(C^{-1}S))$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$

- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}$, $\lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [\mathbf{u}^1, \dots, \mathbf{u}^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j$

Need to find top k
eigenvectors and all the
eigenvalues

Lets set $\sigma = 0$ for now

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k}$$

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \dots, k$
 1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$
 2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \dots, k$
 1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$
 2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

The peeling technique

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \dots, k$
 1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$
 2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

Overall
 $O(d^2 k)$ time

The peeling
technique

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \dots, k$
 1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$
 2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

Overall
 $O(d^2 k)$ time

The peeling
technique

How to recover \mathbf{z}^i ?
Wait a bit.

The Power Method

THE POWER METHOD

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize \mathbf{x}^0 randomly $\sim \mathcal{N}(\mathbf{0}, I)$
3. For $t = 1, 2, \dots, T$

$$\mathbf{y}^t = S\mathbf{x}^{t-1}$$

$$\mathbf{x}^t = \frac{\mathbf{y}^t}{\|\mathbf{y}^t\|_2}$$

4. Repeat until convergence
5. Return eigenvector estimate as \mathbf{x}^T
6. Return eigenvalue estimate as $\|S\mathbf{x}^T\|_2$

The Power Method

THE POWER METHOD

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize \mathbf{x}^0 randomly $\sim \mathcal{N}(\mathbf{0}, I)$
3. For $t = 1, 2, \dots, T$

$$\mathbf{y}^t = S\mathbf{x}^{t-1}$$
$$\mathbf{x}^t = \frac{\mathbf{y}^t}{\|\mathbf{y}^t\|_2}$$

Takes only
 $O(d^2)$ time

4. Repeat until convergence
5. Return eigenvector estimate as \mathbf{x}^T
6. Return eigenvalue estimate as $\|S\mathbf{x}^T\|_2$

The Power Method

THE POWER METHOD

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize \mathbf{x}^0 randomly $\sim \mathcal{N}(\mathbf{0}, I)$
3. For $t = 1, 2, \dots, T$

$$\mathbf{y}^t = S\mathbf{x}^{t-1}$$
$$\mathbf{x}^t = \frac{\mathbf{y}^t}{\|\mathbf{y}^t\|_2}$$

4. Repeat until convergence
5. Return eigenvector estimate as \mathbf{x}^T
6. Return eigenvalue estimate as $\|S\mathbf{x}^T\|_2$

Takes only
 $O(d^2)$ time

Overall
 $O(d^2)$ time

Probabilistic PCA (not assuming $\sigma = 0$)

PROBABILISTIC PCA

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \dots, d$
 1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$
 2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Let $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \hat{\lambda}_j$
5. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j - \hat{\sigma}_{\text{MLE}}} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

Probabilistic PCA (not assuming $\sigma = 0$)

PROBABILISTIC PCA

1. Matrix $S \in \mathbb{R}^{d \times d}$

2. Initialize $S^0 \leftarrow S$

3. For $j = 1, \dots, d$

1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$

2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$

4. Let $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \hat{\lambda}_j$

5. Return $\hat{W}_{\text{MLE}} = \left[\sqrt{\hat{\lambda}_j - \hat{\sigma}_{\text{MLE}}} \cdot \hat{\mathbf{u}}_j \right]_{j=1, \dots, k}$

Recall that

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$$

Probabilistic PCA (not assuming $\sigma = 0$)

PROBABILISTIC PCA

1. Matrix $S \in \mathbb{R}^{d \times d}$

2. Initialize $S^0 \leftarrow S$

3. For $j = 1, \dots, d$

1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$

2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$

4. Let $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \hat{\lambda}_j$

5. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j - \hat{\sigma}_{\text{MLE}}} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

Recall that

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$$

How to recover \mathbf{z}^i ?
Wait a bit.

Probabilistic PCA (not assuming $\sigma = 0$)

PROBABILISTIC PCA

1. Matrix $S \in \mathbb{R}^{d \times d}$

2. Initialize $S^0 \leftarrow S$

3. For $j = 1, \dots, d$

1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$

2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$

4. Let $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \hat{\lambda}_j$

5. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j - \hat{\sigma}_{\text{MLE}}} \cdot \hat{\mathbf{u}}_j]_{j=1, \dots, k}$

Recall that

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$$

Takes $O(d^3)$
time ☹

How to recover \mathbf{z}^i ?
Wait a bit.

Probabilistic PCA (not assuming $\sigma = 0$)

PROBABILISTIC PCA

1. Matrix $S \in \mathbb{R}^{d \times d}$

2. Initialize $S^0 \leftarrow S$

3. For $j = 1, \dots, d$

1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow \text{POWER-METHOD}(S^{j-1})$

2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$

4. Let $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \hat{\lambda}_j$

5. Return $\hat{W}_{\text{MLE}} = [\sqrt{\hat{\lambda}_j - \hat{\sigma}_{\text{MLE}}} \cdot \hat{\mathbf{u}}_j]_{j=1}^k$

Recall that

$$\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$$

Can we do better?

Takes $O(d^3)$ time ☹

How to recover \mathbf{z}^i ?
Wait a bit.

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is

$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log |C| + \text{tr}(C^{-1}S))$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$

- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}$, $\lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [u^1, \dots, u^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j$

Need to find top k eigenvectors and all the eigenvalues

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is
$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log |C| + \text{tr}(C^{-1}S))$$
where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$
- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}, \Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}, \lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [u^1, \dots, u^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j = \frac{1}{d-k} (\text{tr}(S) - \sum_{j=1}^k \lambda_j)$

Need to find top k eigenvectors and all the eigenvalues

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is
$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log|C| + \text{tr}(C^{-1}S))$$
where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$
- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}, \Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}, \lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [u^1, \dots, u^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j = \frac{1}{d-k} (\text{tr}(S) - \sum_{j=1}^k \lambda_j)$

Need to find top k
eigenvectors and all the
eigenvalues

Exercise!

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$, log-likelihood is
$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log 2\pi + \log|C| + \text{tr}(C^{-1}S))$$
where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$
- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}, \Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}, \lambda_1 \geq \lambda_2 \geq \dots$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [u^1, \dots, u^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j = \frac{1}{d-k} (\text{tr}(S) - \sum_{j=1}^k \lambda_j)$

Need to find top k
eigenvectors and top k
eigenvalues

Exercise!

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(\mathbf{W}X, \sigma^2 I_d)$

Power method takes log-likelihood is only $O(d^2 k)$ time and not $O(d^3)$ time to solve this problem ☺

$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log |C| - \frac{1}{|C|} \sum_{i=1}^n \mathbf{x}^i \mathbf{x}^i \mathbf{C}^{-1} \mathbf{x}^i)$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i \mathbf{x}^i \mathbf{C}^{-1}$
- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}, \Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}, \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I}$
- where $U_k = [u^1, \dots, u^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j = \frac{1}{d-k} (\text{tr}(S) - \sum_{j=1}^k \lambda_j)$

Need to find top k eigenvectors and top k eigenvalues

Exercise!

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}(\mathbf{W}X, \sigma^2 I_d)$

$$\log \mathbb{P}[X | W, \sigma] = \frac{n}{2} (d \log |C| - \frac{1}{|C|} \sum_{i=1}^n \mathbf{x}^i \mathbf{x}^i \mathbf{C}^{-1} \mathbf{x}^i)$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i \mathbf{x}^i \mathbf{C}^{-1} \mathbf{x}^i$

Power method takes only $O(d^2 k)$ time and $O(dk)$ space (not $O(d^3)$ time to solve this problem ☺)

Improvement over last lecture's claim
- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
 - $U = [\mathbf{u}^1, \dots, \mathbf{u}^d] \in \mathbb{R}^{d \times d}, \Lambda = \text{diag}(\lambda_1, \dots, \lambda_d) \in \mathbb{R}^{d \times d}, \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$
- $\hat{W}_{\text{MLE}} = U_k \sqrt{\Lambda_k - \hat{\sigma}_{\text{MLE}}^2 \cdot I_k}$
- where $U_k = [u^1, \dots, u^k]$ and $\Lambda_k = [\lambda_1, \dots, \lambda_k]$
- Top k eigenvalues and eigenvectors
- $\hat{\sigma}_{\text{MLE}}^2 = \frac{1}{d-k} \sum_{j=k+1}^d \lambda_j = \frac{1}{d-k} (\text{tr}(S) - \sum_{j=1}^k \lambda_j)$

Need to find top k eigenvectors and top k eigenvalues

Exercise!

FA Interpretations for PCA

Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data

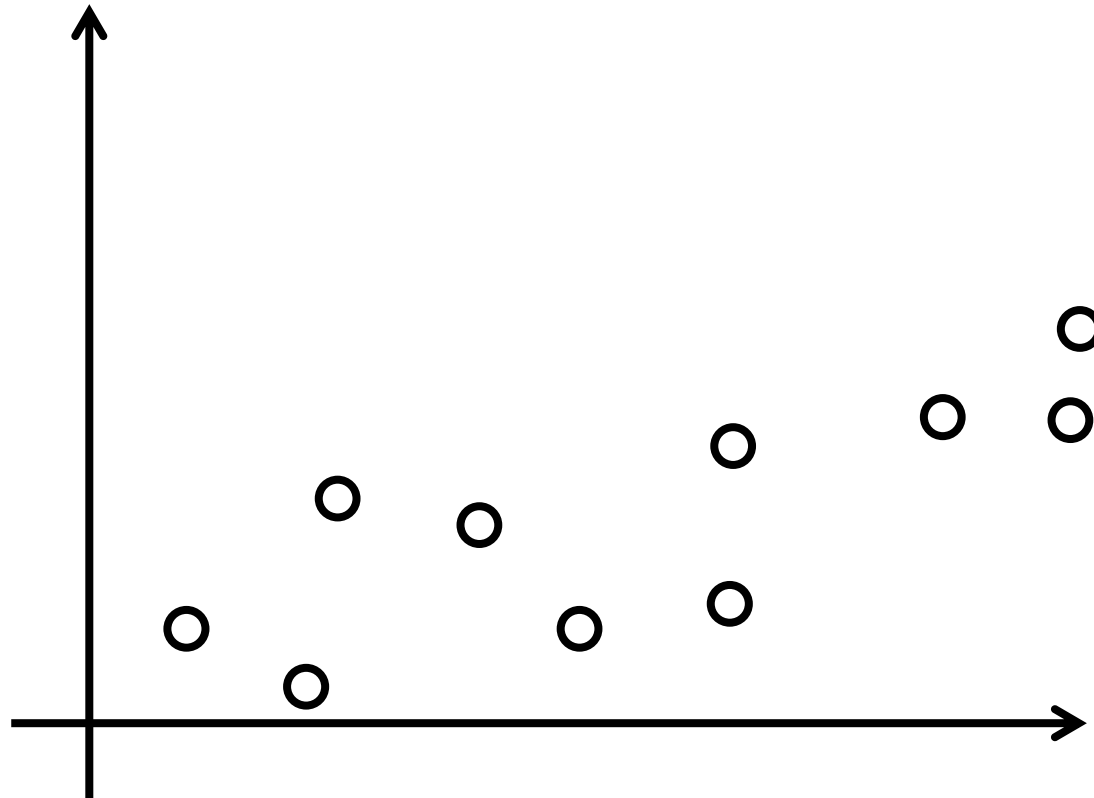
Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



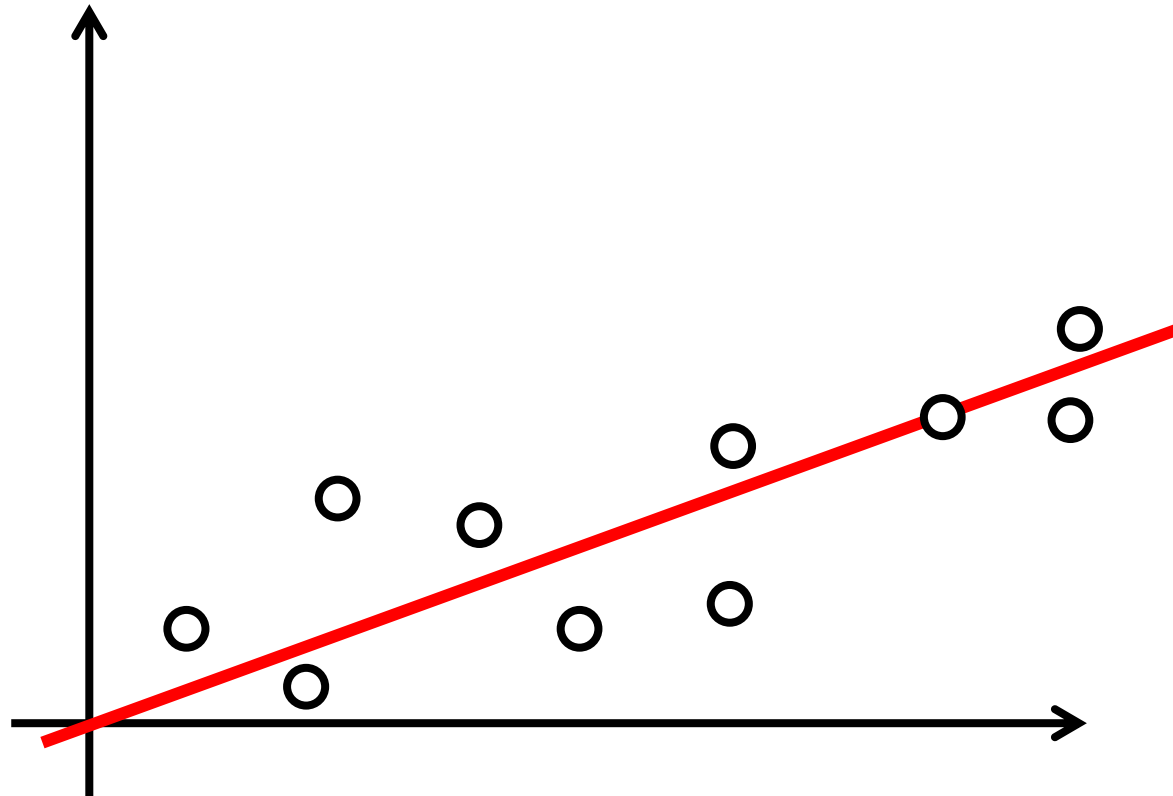
Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



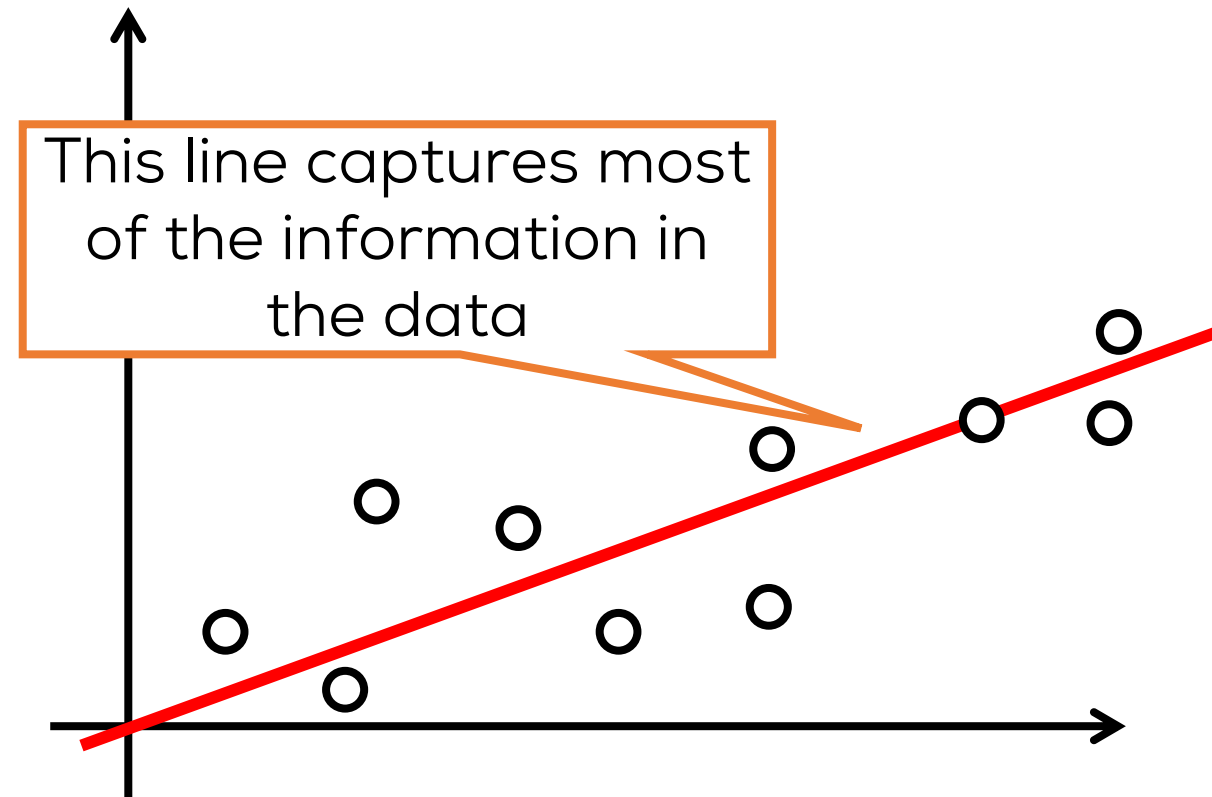
Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



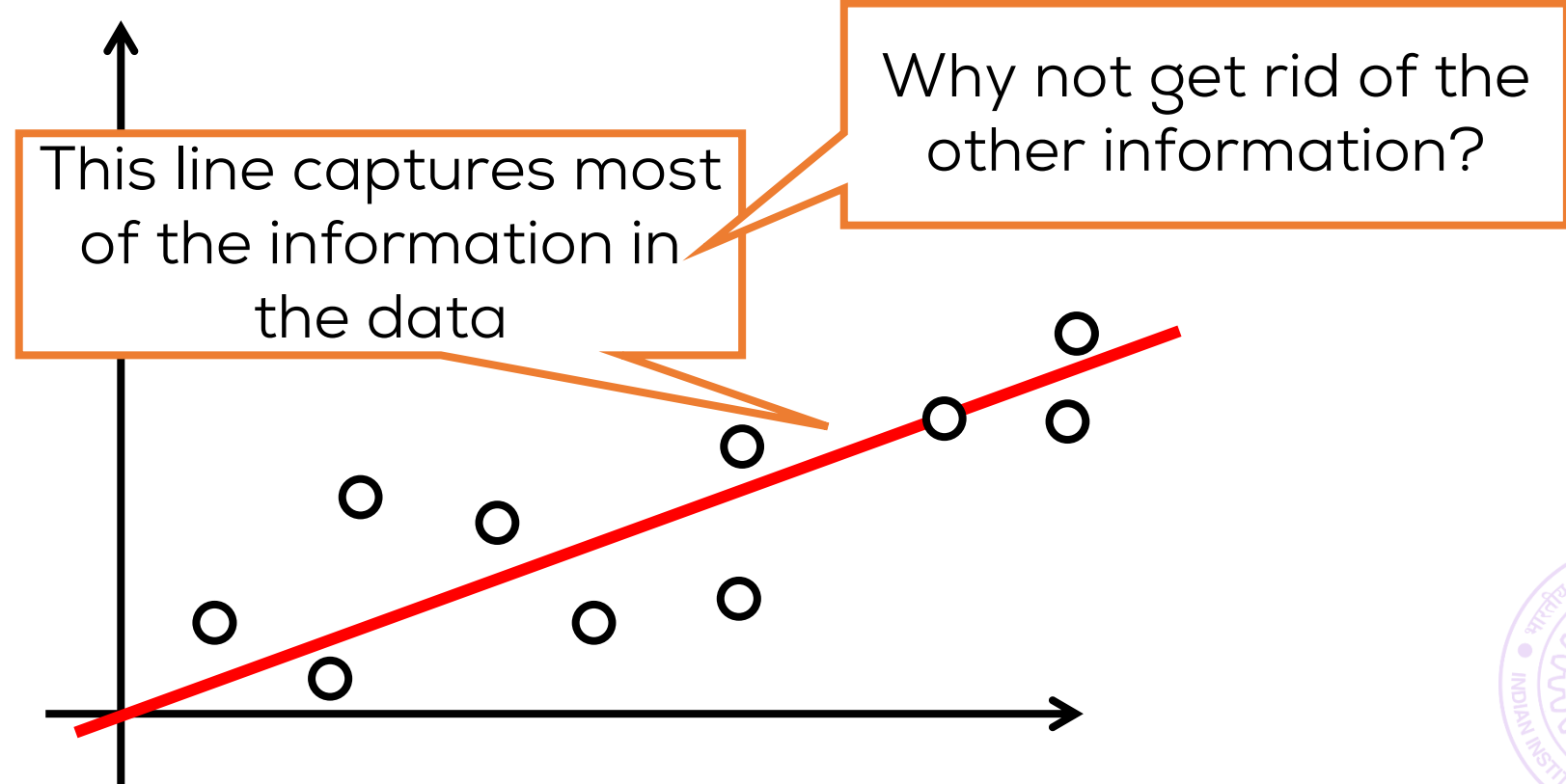
Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



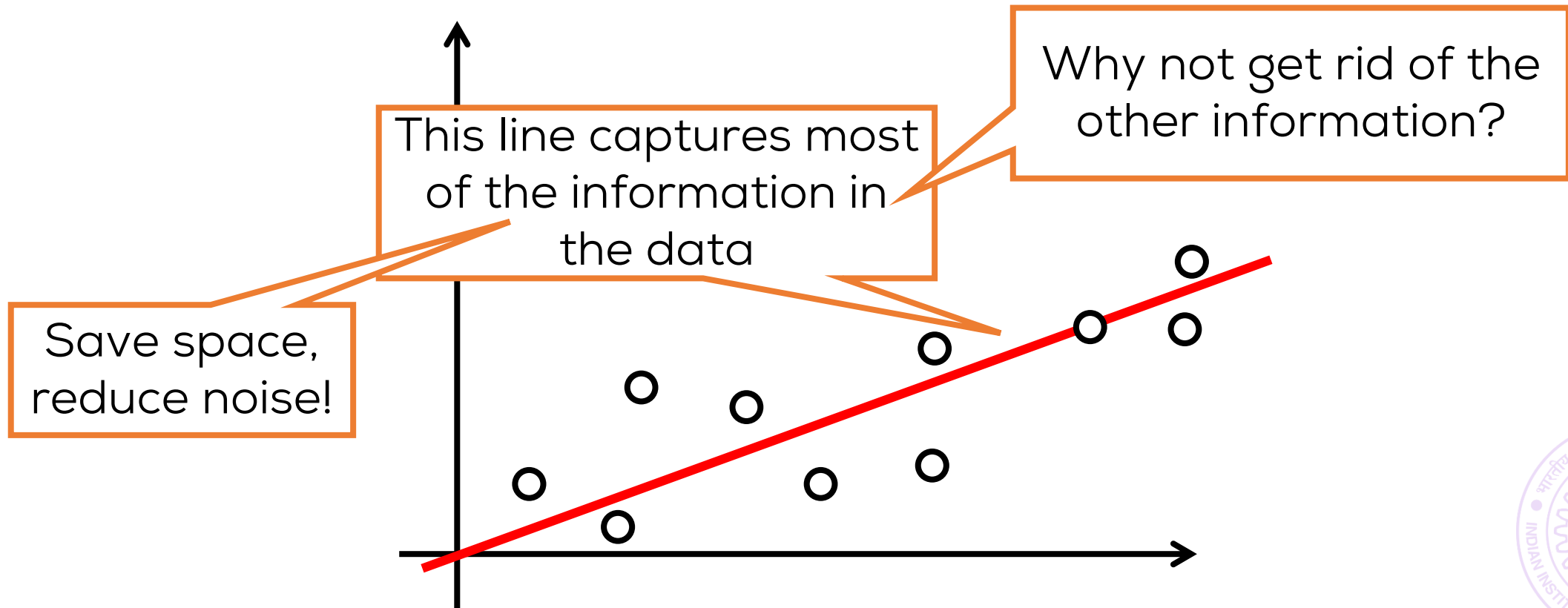
Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



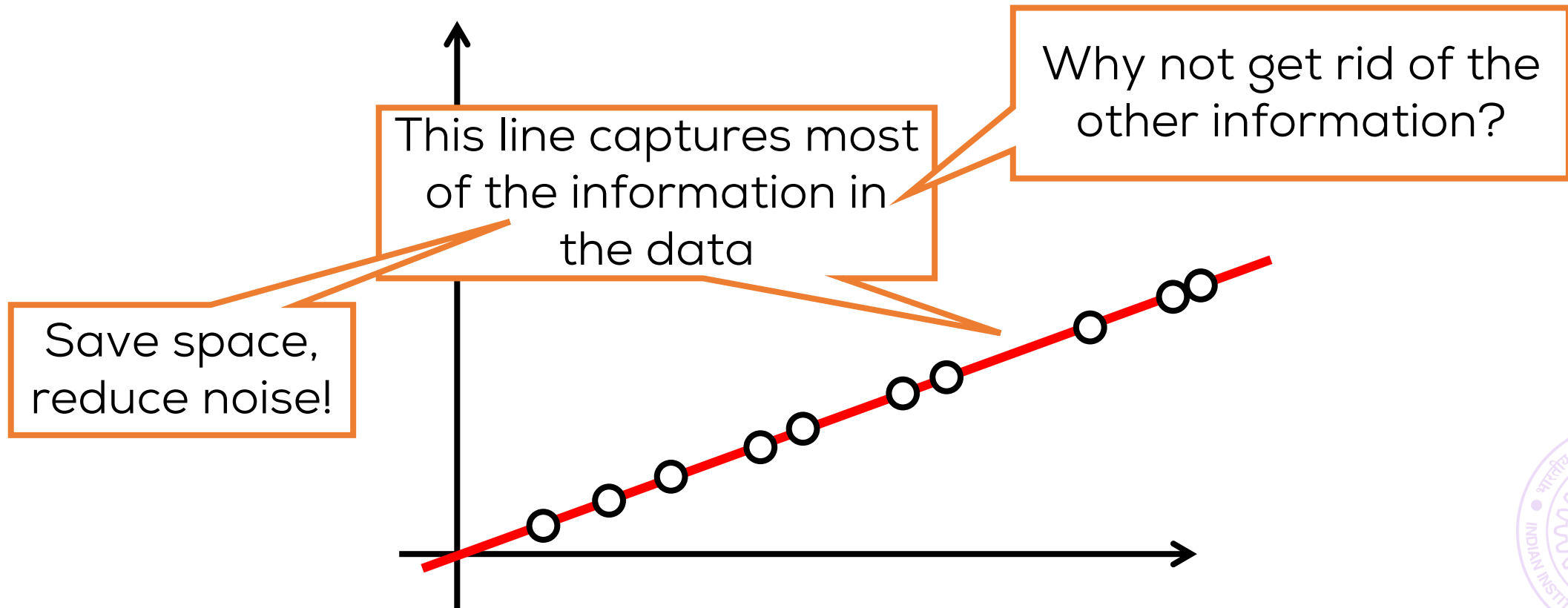
Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



Principal Component Analysis

- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the “closest” low-dim representation of data
- Also seeks to capture the maximum “variance” in the data



PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find a line in \mathbb{R}^d such that all points lie “close” to the line (actually finding the best 1-d representation for the points)
- Every line in d -dimensions is indexed by a unit vector $\mathbf{w} \in \mathbb{R}^d$ (assume the line passes through the origin for simplicity)
- Every point on this line is of the form $z \cdot \mathbf{w}$ for some $z \in \mathbb{R}$
- Closest point on the line (Euclidean dist.) to a point \mathbf{x} is $\langle \mathbf{w}, \mathbf{x} \rangle \cdot \mathbf{w}$
- Given data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$ and line corresponding to $\mathbf{w} \in \mathbb{R}^d$, can find out how well the line “fits” the points

$$\sum_{i=1}^n \|\mathbf{x}^i - \langle \mathbf{w}, \mathbf{x}^i \rangle \cdot \mathbf{w}\|_2^2$$

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find a line in \mathbb{R}^d such that all points lie “close” to the line (actually finding the best 1-d representation for the points)
- So our problem reduces to the following optimization problem

$$\arg \min_{\|\mathbf{w}\|_2=1} \sum_{i=1}^n \|\mathbf{x}^i - \langle \mathbf{w}, \mathbf{x}^i \rangle \cdot \mathbf{w}\|_2^2 = \sum_{i=1}^n \|\mathbf{x}^i - \mathbf{w}\mathbf{w}^\top \mathbf{x}^i\|_2^2$$

- Simple calculations show that the above is equivalent to

$$\arg \max_{\|\mathbf{w}\|_2=1} \mathbf{w}^\top \mathbf{S} \mathbf{w}$$

$$\text{where } \mathbf{S} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top \in \mathbb{R}^{d \times d}$$

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find a line in \mathbb{R}^d such that all points lie “close” to the line (actually finding the best 1-d representation for the points)

- So our problem further reduces to

$$\arg \max_{\|\mathbf{w}\|_2=1} \mathbf{w}^\top S \mathbf{w}$$

Power method
can find this in
 $O(d^2)$ time ☺

- The solution to (2) is the leading eigenvector of S (the one corresponding to the largest eigenvalue λ_1) i.e. $\mathbf{w} = \mathbf{u}^1$
- **Proof:** write $S = U\Lambda U^\top$, $U = [\mathbf{u}^1, \dots, \mathbf{u}^d]$, $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_d)$, $\lambda_1 \geq \lambda_2 \geq \dots$
- Then $\mathbf{w}^\top S \mathbf{w} = \sum_{j=1}^d \lambda_j (\langle \mathbf{u}^j, \mathbf{w} \rangle)^2$. Makes sense to align \mathbf{w} with \mathbf{u}^1 fully.
- **Exercise:** complete the above proof (Hint: U is a basis for \mathbb{R}^d)
- **Alternate proof:** find and solve the dual problem!



PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find the k -dimensional hyperplane in \mathbb{R}^d such that all points lie “closest” to that hyperplane
- Every k -dimensional hyperplane in d -dimensions is indexed by k orthonormal unit vectors $\mathbf{w}^1, \mathbf{w}^2, \dots, \mathbf{w}^k \in \mathbb{R}^d$
(assume the plane passes through the origin for simplicity)
- Let us represent this using a matrix $W \in \mathbb{R}^{d \times k}$ with $W^\top W = I_k \in \mathbb{R}^{k \times k}$
- Be careful ... we may not have $WW^\top = I_d \in \mathbb{R}^{d \times d}$
- Every point on the plane is of the form $W\mathbf{z}$ for some $\mathbf{z} \in \mathbb{R}^k$
- Is this \mathbf{z} that same latent variable \mathbf{z} we had earlier?
- Yeah ... we will now see how to recover \mathbf{z}^i as well 😊

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find the k -dimensional hyperplane in \mathbb{R}^d such that all points lie “closest” to that hyperplane
- The closest point (in Euclidean dist.) to \mathbf{x} on the plane given by W is $WW^\top \mathbf{x}$. (Hint: use the fact that U has orthonormal columns)
- We can thus, setup an opt. problem to find the best W

$$\arg \min_{W^\top W = I_k} \sum_{i=1}^n \|\mathbf{x}^i - WW^\top \mathbf{x}^i\|_2^2 \equiv \arg \max_{W^\top W = I_k} \text{tr}(W^\top S W)$$

- One can show that the optimal solution are the top k eigenvectors of S i.e. $W = U_k = [\mathbf{u}^1, \dots, \mathbf{u}^k] \in \mathbb{R}^{d \times k}$

PCA as minimizing reconstruction error

- **Proof:** by induction. Base case $k = 1$ already done

- Do a bit of rewriting

$$\begin{aligned}\arg \max_{W^T W = I_k} \text{tr}(W^T S W) &= \arg \max_{\substack{\tilde{W}^T \tilde{W} = I_{k-1} \\ \tilde{W}^T \mathbf{v} = \mathbf{0}}} \arg \max_{\|\mathbf{v}\|_2 = 1} \text{tr} \left([\mathbf{v} \ \tilde{W}]^T S [\mathbf{v} \ \tilde{W}] \right) \\ &= \arg \max_{\substack{\tilde{W}^T \tilde{W} = I_{k-1} \\ \tilde{W}^T \mathbf{v} = \mathbf{0}}} \left\{ \arg \max_{\|\mathbf{v}\|_2 = 1} \mathbf{v}^T S \mathbf{v} + \text{tr}(\tilde{W}^T S \tilde{W}) \right\} \\ &= (\mathbf{u}^1)^T S \mathbf{u}^1 + \arg \max_{\substack{\tilde{W}^T \tilde{W} = I_{k-1} \\ \tilde{W}^T \mathbf{u}^1 = \mathbf{0}}} \text{tr}(\tilde{W}^T S \tilde{W}) = (\mathbf{u}^1)^T S \mathbf{u}^1 + \arg \max_{\tilde{W}^T \tilde{W} = I_{k-1}} \text{tr}(\tilde{W}^T \tilde{S} \tilde{W})\end{aligned}$$

where $\tilde{S} = S - \lambda_1 \cdot \mathbf{u}^1 (\mathbf{u}^1)^T$

The peeling technique

- Now apply induction to get $\tilde{W} = [\mathbf{u}^2, \mathbf{u}^3, \dots, \mathbf{u}^k]$

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find the k -dimensional hyperplane in \mathbb{R}^d such that all points lie “closest” to that hyperplane
- The closest point (in Euclidean dist.) to \mathbf{x} on the plane given by W is $WW^\top \mathbf{x}$. (Hint: use the fact that U has orthonormal columns)
- We can thus, setup an opt. problem to find the best W

$$\arg \min_{W^\top W = I_k} \sum_{i=1}^n \|\mathbf{x}^i - WW^\top \mathbf{x}^i\|_2^2 \equiv \arg \max_{W^\top W = I_k} \text{tr}(W^\top SW)$$

- One can show that the optimal solution are the top k eigenvectors of S i.e. $W = U_k = [\mathbf{u}^1, \dots, \mathbf{u}^k] \in \mathbb{R}^{d \times k}$

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find the k -dimensional hyperplane in \mathbb{R}^d such that all points lie “closest” to that hyperplane
- The closest point (in Euclidean dist.) to \mathbf{x} on the plane given by W is $WW^\top \mathbf{x}$. (Hint: use the fact that U has orthonormal columns)
- We can thus, setup an opt. problem to find the best W

$$\arg \min_{W^\top W = I_k} \sum_{i=1}^n \|\mathbf{x}^i - WW^\top \mathbf{x}^i\|_2^2 \equiv \arg \max_{W^\top W = I_k} \text{tr}(W^\top S W)$$

- One can show that the optimal solution are the top k eigenvectors of S i.e. $W = U_k = [\mathbf{u}^1, \dots, \mathbf{u}^k] \in \mathbb{R}^{d \times k}$

Power method
can find these
in $O(d^2 k)$ time

Exercise!

Exercise!

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$
- Goal: find the k -dimensional hyperplane in \mathbb{R}^d such that all points lie “closest” to that hyperplane
- Having recovered the optimal W we can now recover \mathbf{z}^i
- Recall that the closest point on the plane to \mathbf{x}^i is $WW^\top \mathbf{x}^i$
- However, recall that every point in the plane is expressed as $W\mathbf{z}$
- This means that $WW^\top \mathbf{x}^i = W\mathbf{z}^i$
- Multiply both sides by W^\top and use the fact that $W^\top W = I_k$
$$W^\top WW^\top \mathbf{x}^i = W^\top W\mathbf{z}^i$$
$$\mathbf{z}^i = W^\top \mathbf{x}^i$$

Wait a second!

- Didn't the PML discussion tell us that $W_{\text{MLE}} = U_k \sqrt{\Lambda_k}$
- Now the FA discussion is telling us $W = U_k$. What gives??
- Note that the FA forced W to have orthonormal columns $W^T W = I_k$
- This is why we got $W = U_k$ since we also have $U_k^T U_k = I_k$
- Both FA/PML are valid. They just shift normalization constants
- FA view: $W_{\text{FA}} = U_k$ and $\mathbf{z}_{\text{FA}}^i = W_{\text{FA}}^T \mathbf{x}^i$
- PML view: $W_{\text{PML}} = U_k \sqrt{\Lambda_k}$ and $\mathbf{z}_{\text{PML}}^i = \Lambda_k^{-1} W_{\text{PML}}^T \mathbf{x}^i$
- The PML \mathbf{z}^i are "normalized" since PML models $\mathbf{z}^i \sim N(\mathbf{0}, I_k)$
- The FA W is "normalized" as they are constrained to be so
- Note that $W_{\text{FA}} \mathbf{z}_{\text{FA}}^i = W_{\text{PML}} \mathbf{z}_{\text{PML}}^i = U_k U_k^T \mathbf{x}^i$



Some practical issues

- We kept assuming that “lines” and “planes” (that approximate the data points) pass through the origin
- This may not be true in general. To make this true, we need to *mean-centre* the data.
- Let $\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i$. Convert data to $\tilde{\mathbf{x}}^i = \mathbf{x}^i - \boldsymbol{\mu}$ and work with $\tilde{\mathbf{x}}^i$
- Use $S = \frac{1}{n} \sum_{i=1}^n \tilde{\mathbf{x}}^i (\tilde{\mathbf{x}}^i)^\top$ to perform PCA, PPCA
- How to decide k ?
 - Choose k that is small but gives reasonable reconstruction
 - Some tuning required on training data itself
 - Other criterion also used (model selection – later lectures)

PCA as maximizing capture of variance

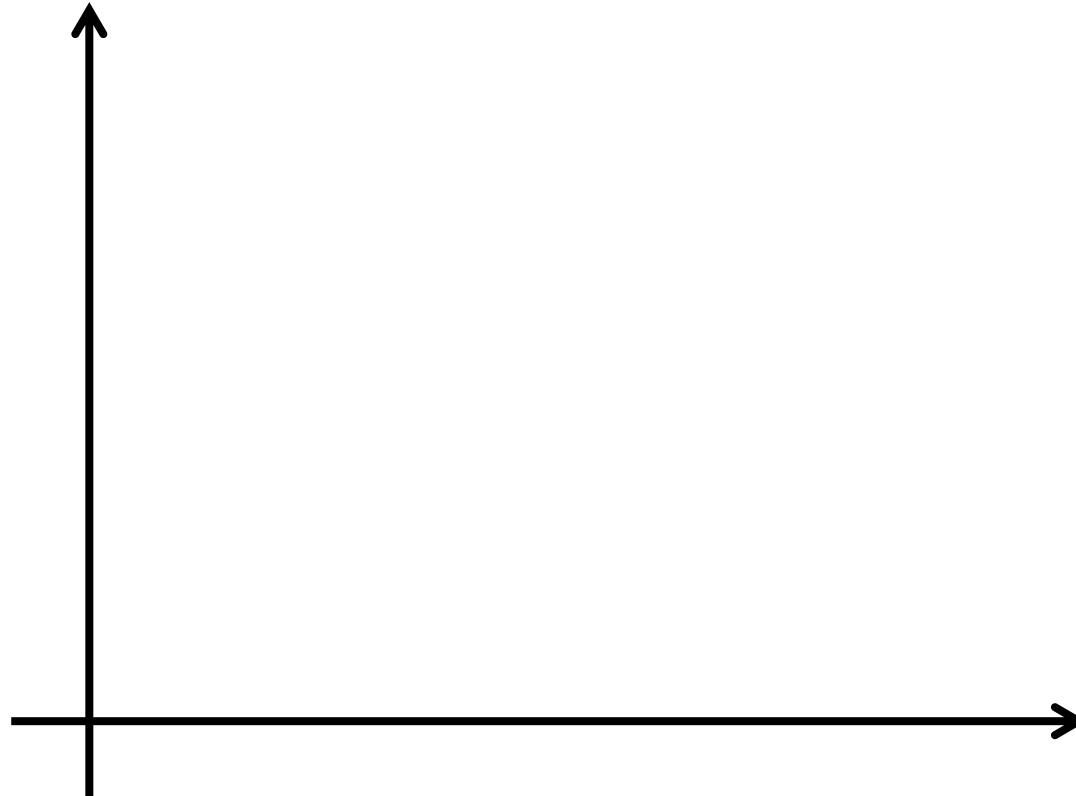
- Data may vary differently along different directions

PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative

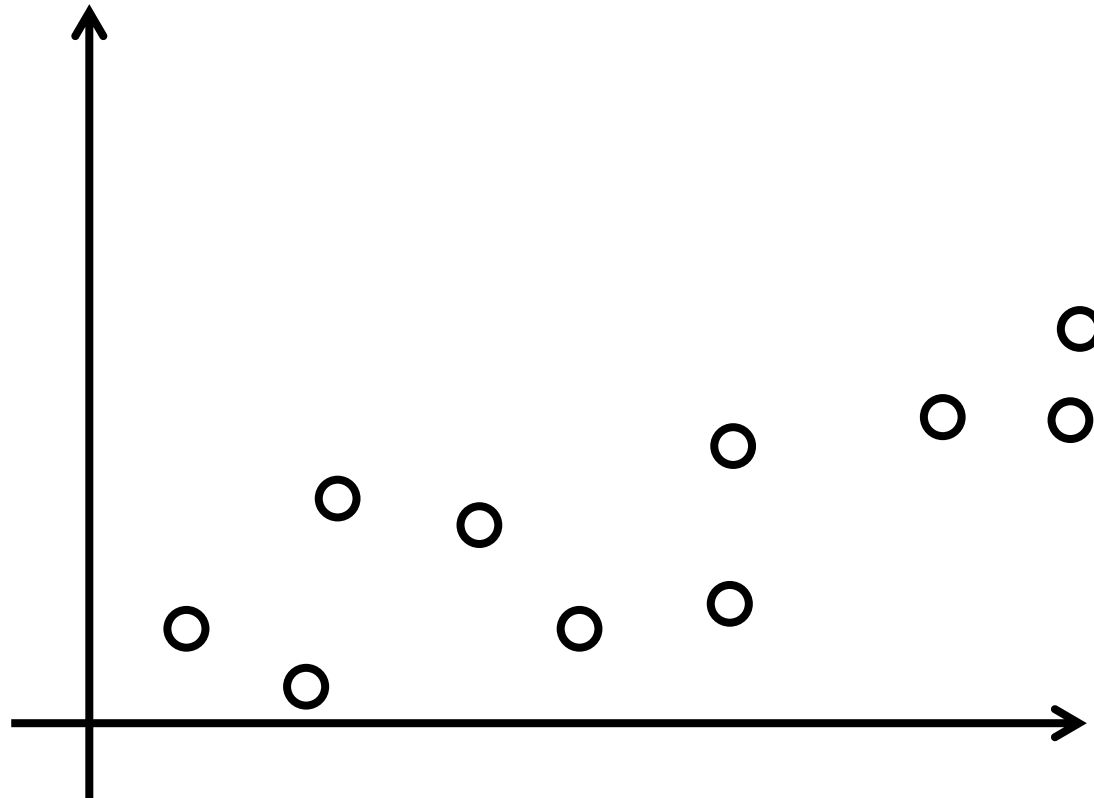
PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative



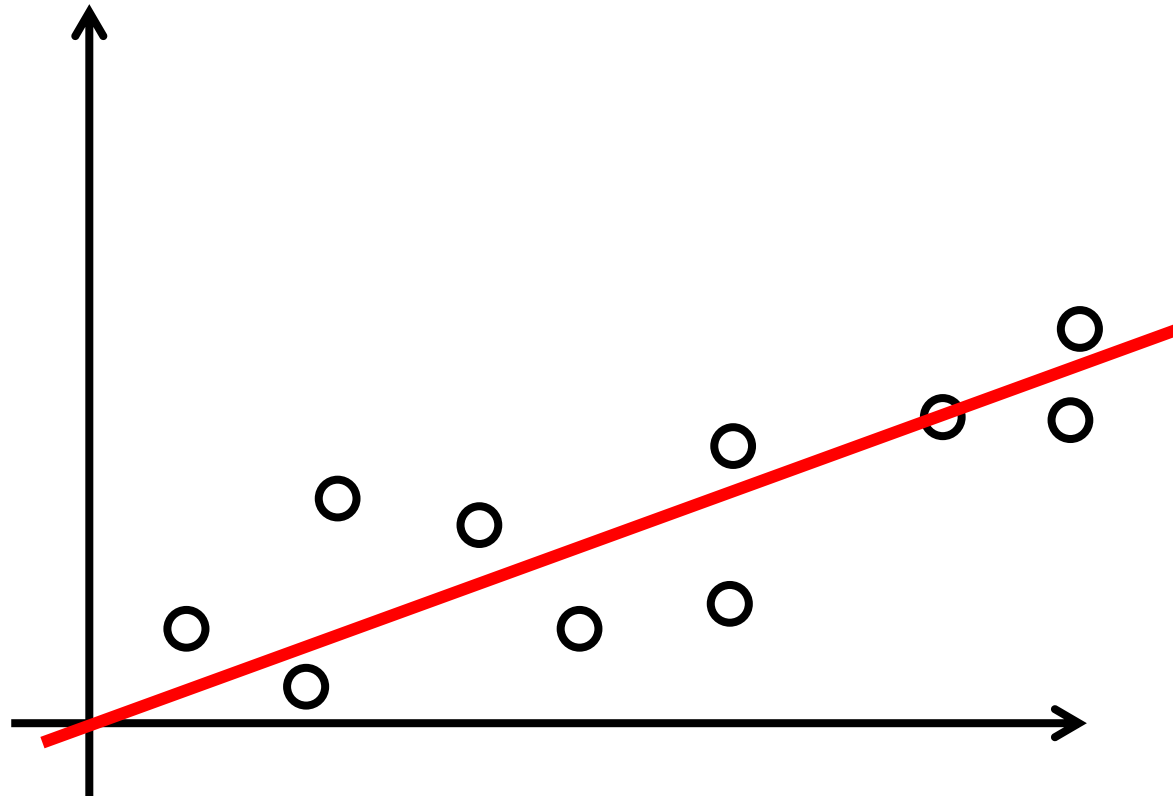
PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative



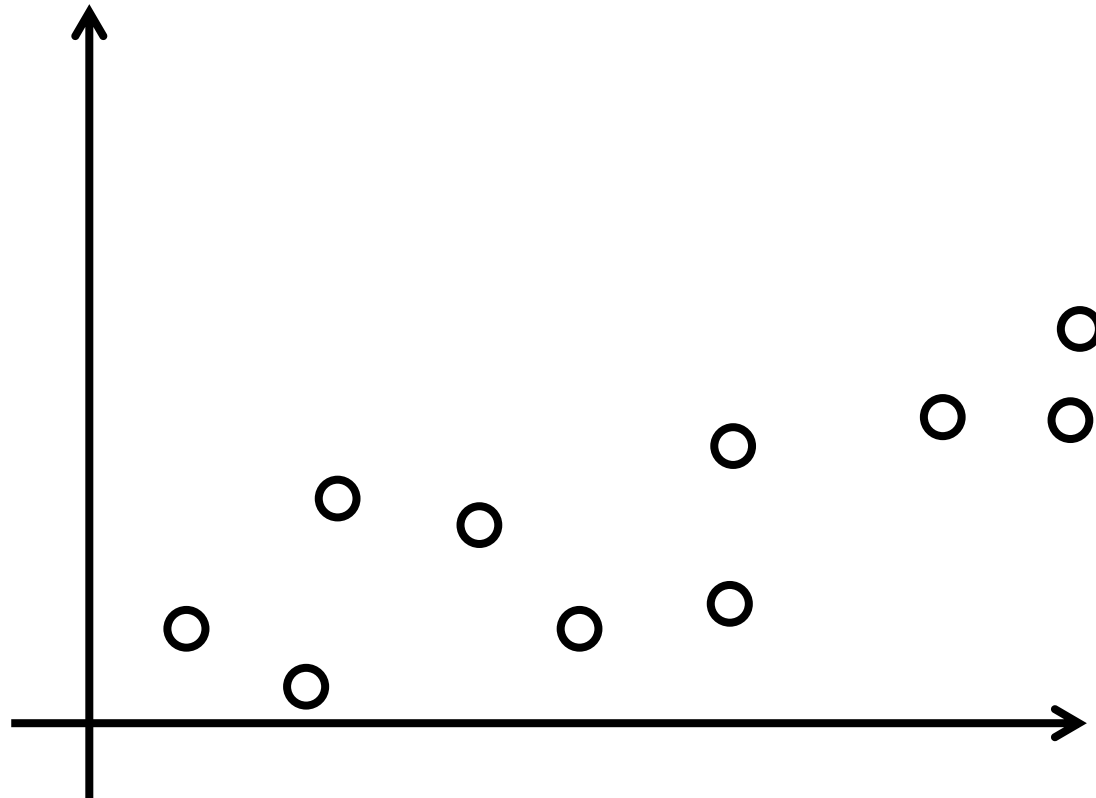
PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative



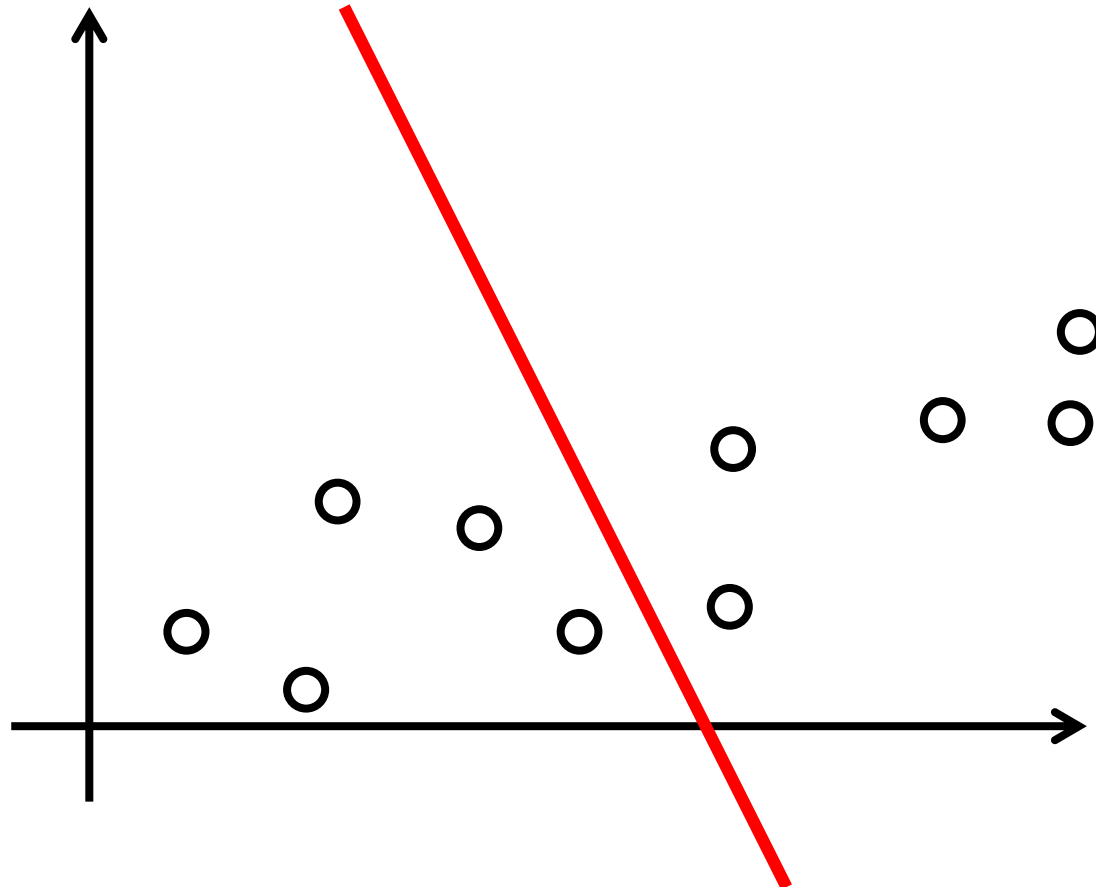
PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative



PCA as maximizing capture of variance

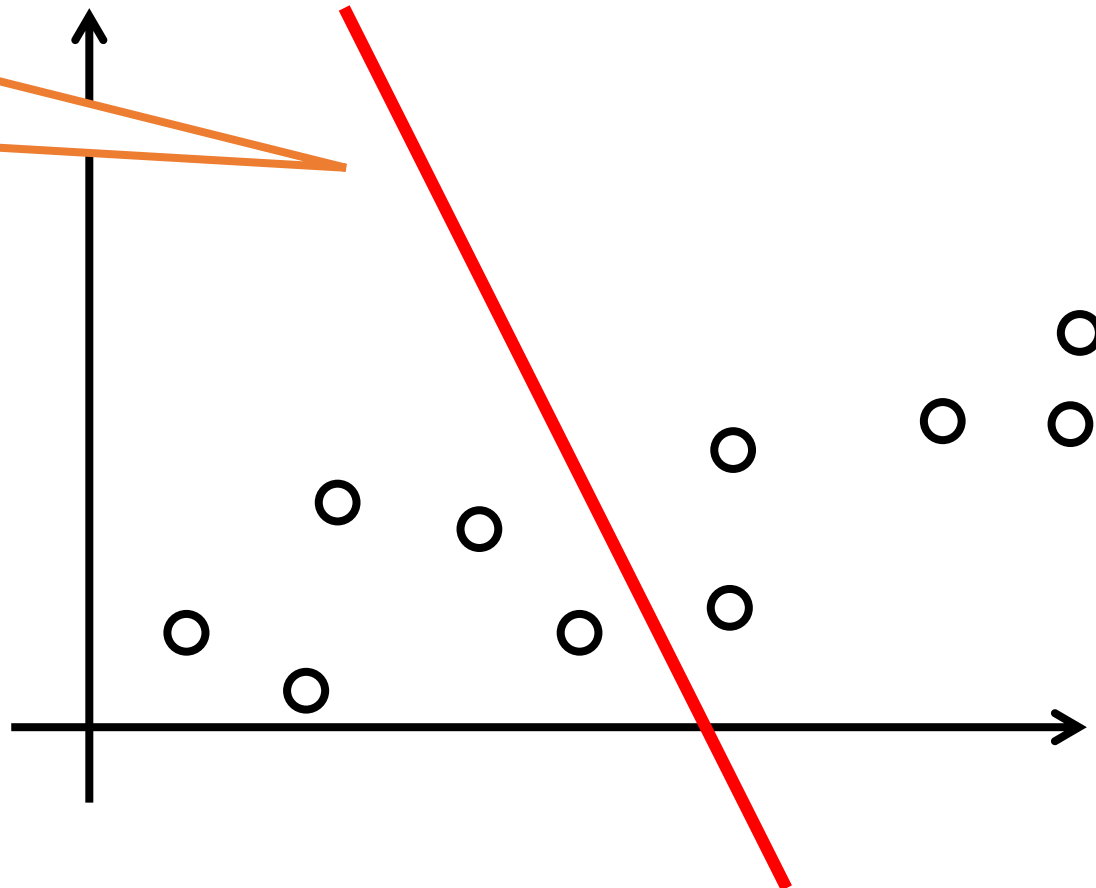
- Data may vary differently along different directions
- Directions with more spread useful and informative



PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative

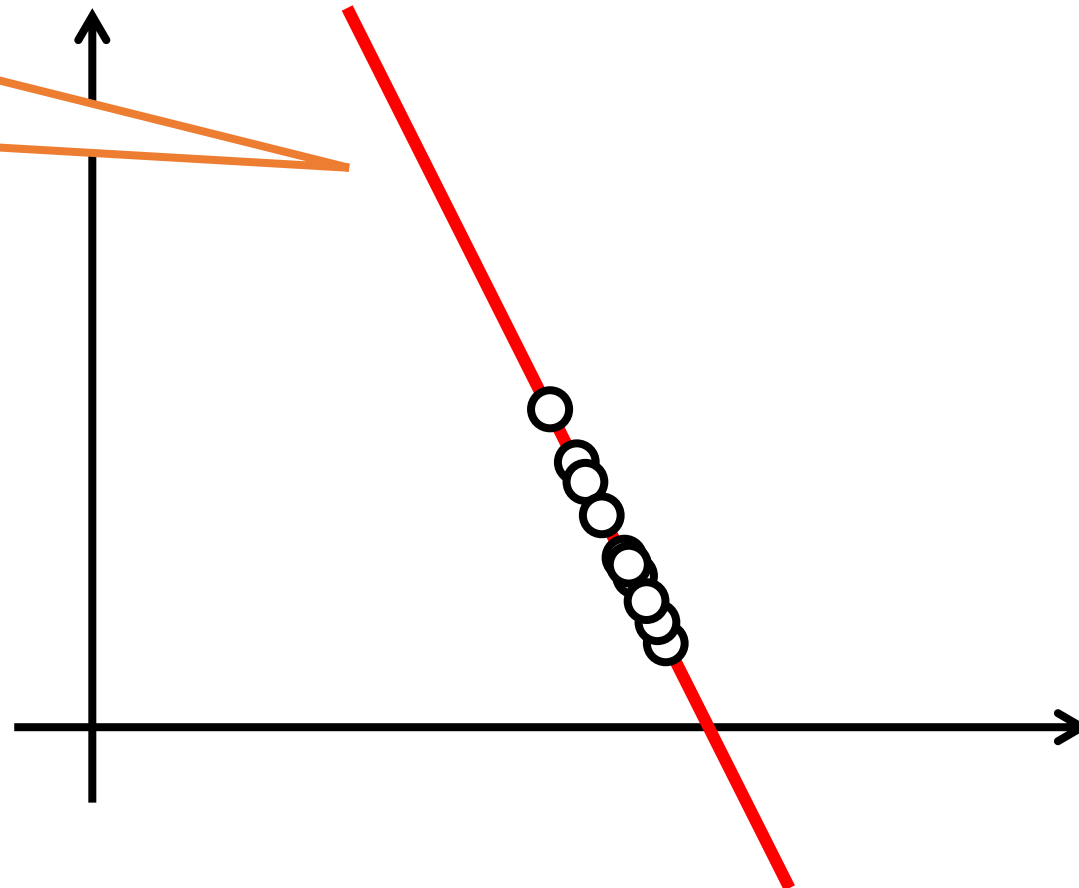
Projecting onto this line throws away a lot of information about the data



PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative

Projecting onto this line throws away a lot of information about the data

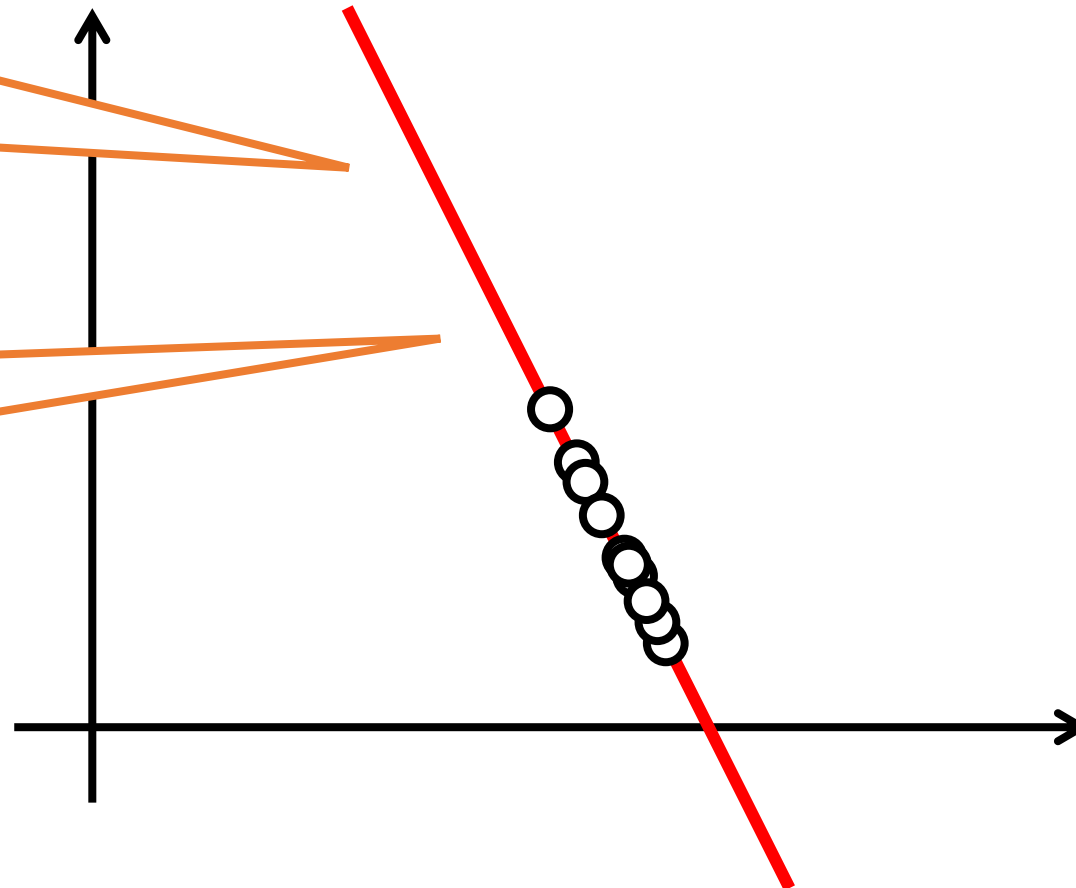


PCA as maximizing capture of variance

- Data may vary differently along different directions
- Directions with more spread useful and informative

Projecting onto this line throws away a lot of information about the data

Save space, but introduces large amount of noise!



PCA as maximizing capture of variance

- Given: n data points $\mathbf{x}^1, \dots, \mathbf{x}^n \in \mathbb{R}^d$ (assume mean centered)
- Suppose we approximate each of these points by the closest point on a line given by \mathbf{u}
- The approximations will be $\langle \mathbf{u}, \mathbf{x}^i \rangle \cdot \mathbf{u}$
- Means of these approximations will be $\mathbf{0}$ (data is mean centered)
- “Variance” of these approximations will be

$$\frac{1}{n} \sum_{i=1}^n \|\langle \mathbf{u}, \mathbf{x}^i \rangle \cdot \mathbf{u} - \mathbf{0}\|_2^2 = \frac{1}{n} \sum_{i=1}^n \|\mathbf{u} \mathbf{u}^\top \mathbf{x}^i\|_2^2 = \mathbf{u}^\top \mathbf{S} \mathbf{u}$$

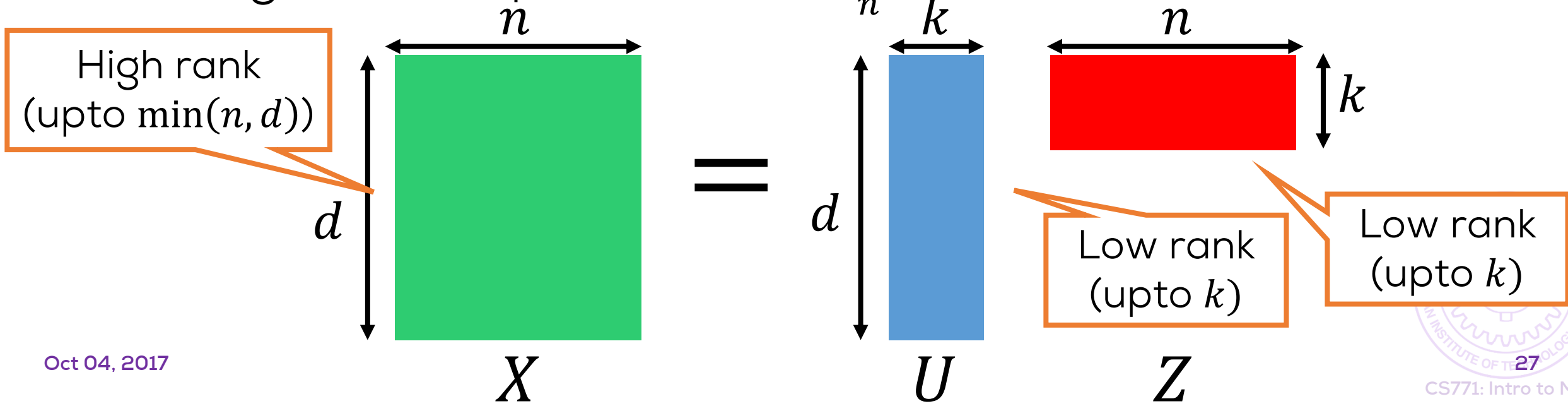
where $\mathbf{S} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}^i (\mathbf{x}^i)^\top$

- But PCA maximizes this!
- PCA discovers directions of maximum spread in data!



Some practical issues

- We saw how to recover a low-dim. representation of data
- PCA gave us a way to express $\mathbf{x}^i = U\mathbf{z}^i$
- For the entire data matrix $X = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n] \in \mathbb{R}^{d \times n}$, this means $X = UZ$, where $Z = [\mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^n] \in \mathbb{R}^{k \times n}$
- This gives a *low-rank* factorization of the matrix X
- Note: Eigen-decomposition on $S = \frac{1}{n}XX^\top \equiv$ PCA on X

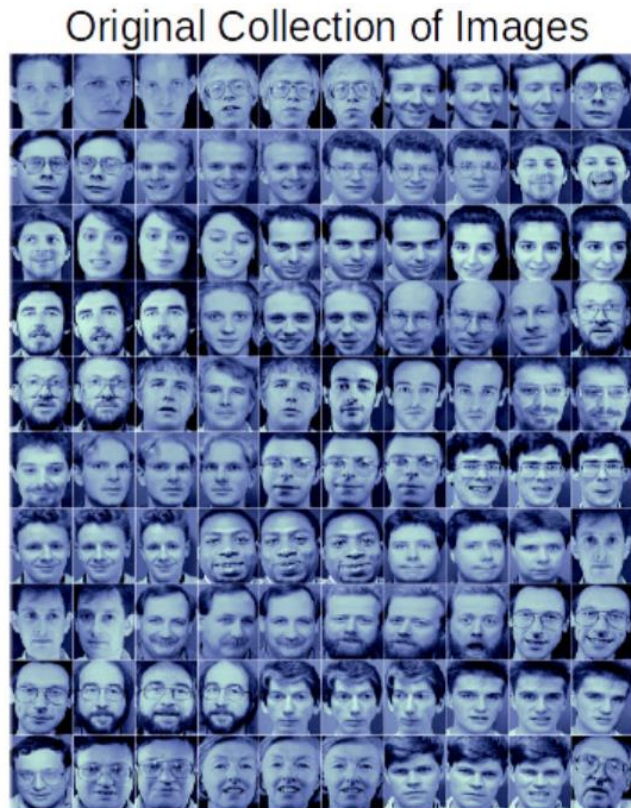


Some practical applications

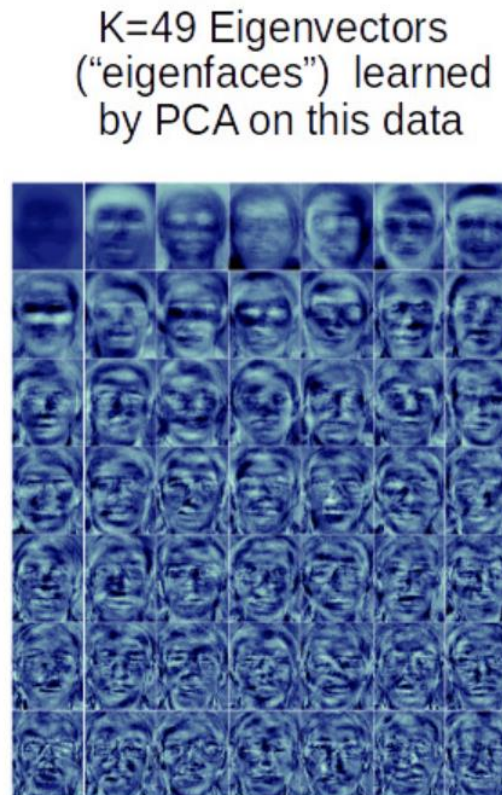
- The k vectors in U give us a useful *basis* for representing new data points!
- Time savings, space savings and noise removal
- Such bases (not necessarily orthonormal) often called dictionaries
- Dictionary Learning, Topic Modelling, other advanced methods for learning “simple” but latent structure in the data
- Excellent applications in image and signal processing, large-scale learning

Some practical applications

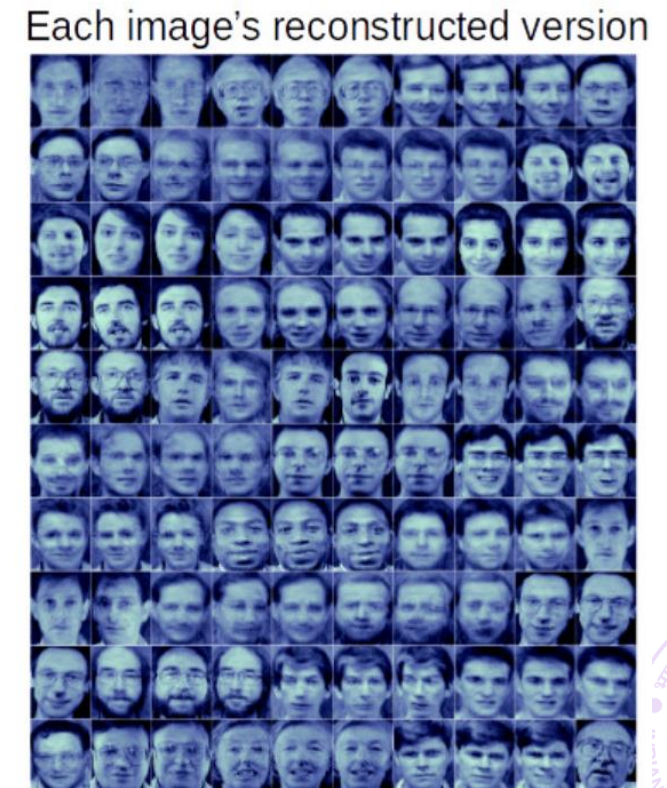
- Eigen-faces [Sirovich and Kirby, Turk and Pentland]
- Each face represented as just a 49-dim vector!



X



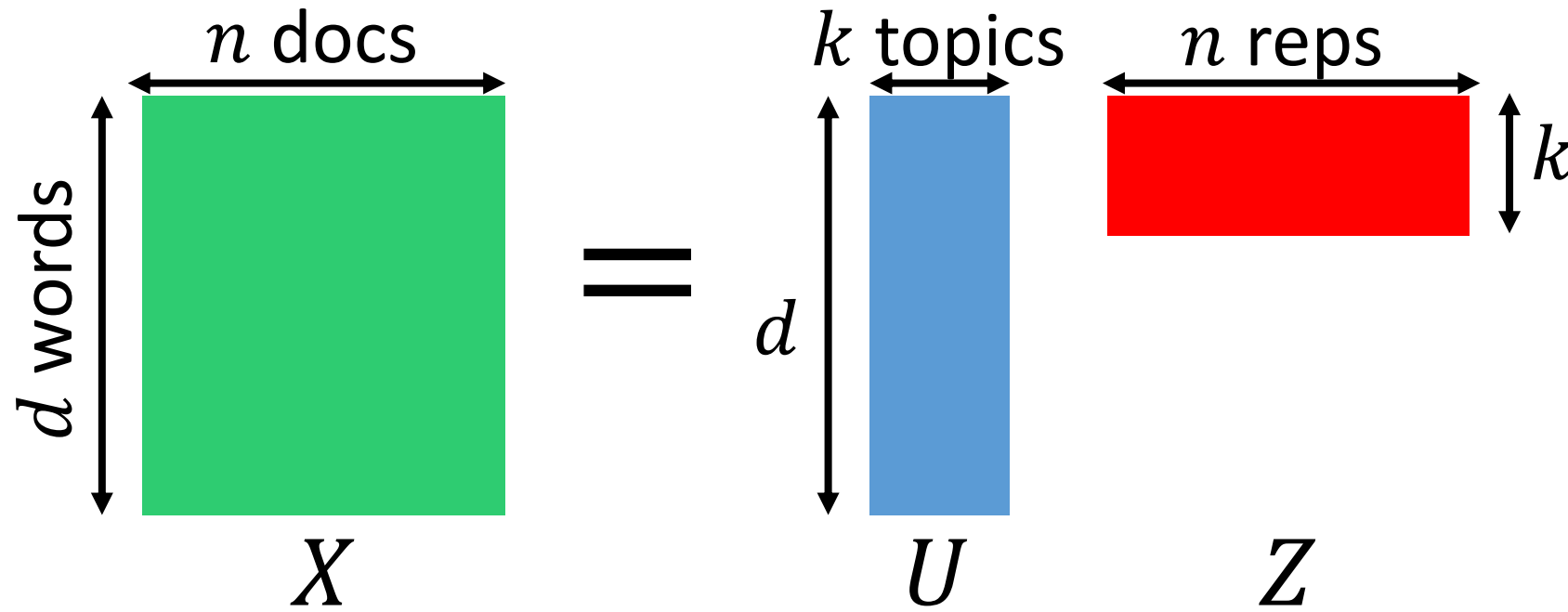
U



$UZ \approx X$

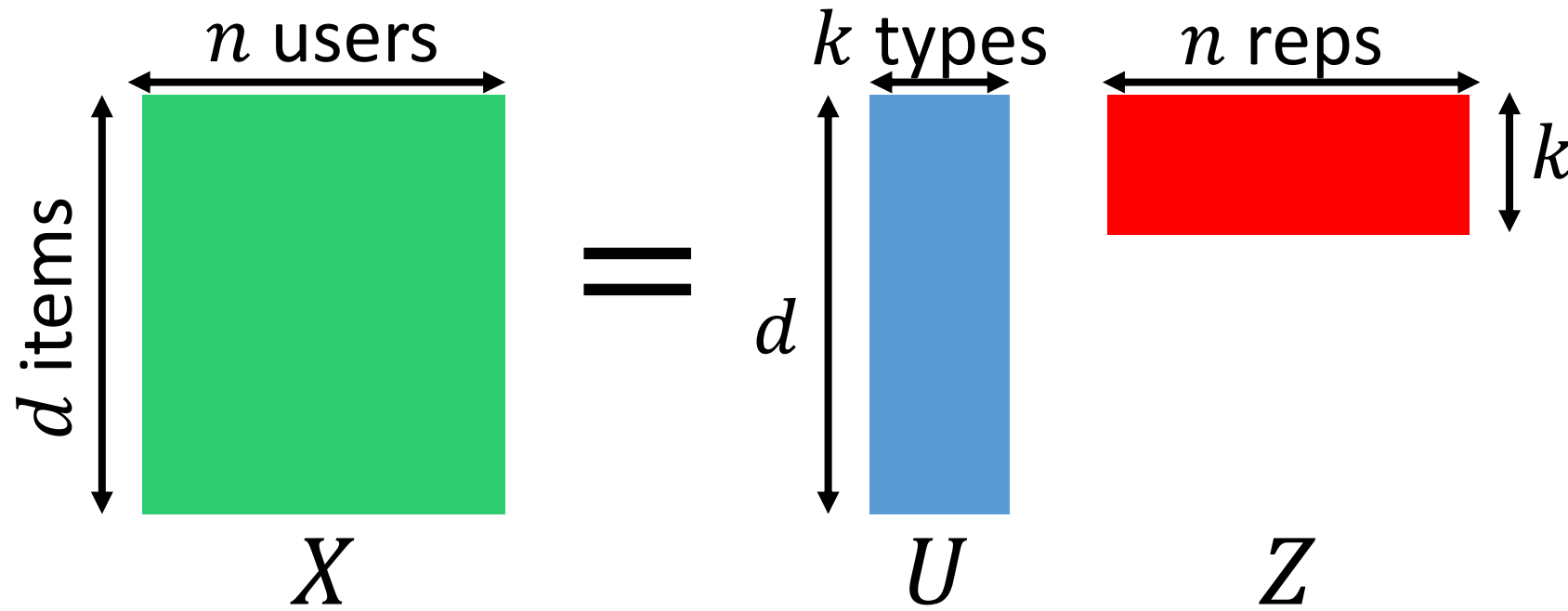
Some practical applications

- Latent Semantic Analysis
- Given a collection of n documents/articles (e.g. Wikipedia)
- Discover of k topics and a representation of each article in terms of these topics. E.g. topics can be sports, education, science etc.
- Each document represented as a bag of d words ($d \approx 10^6$)



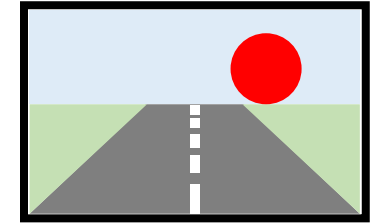
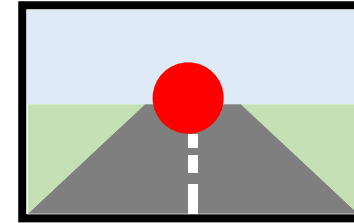
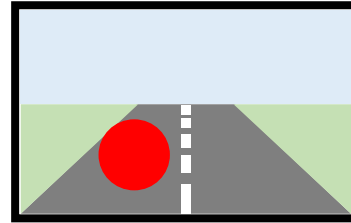
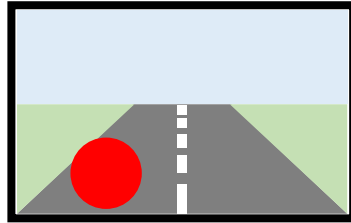
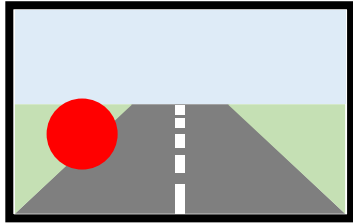
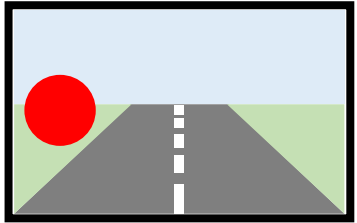
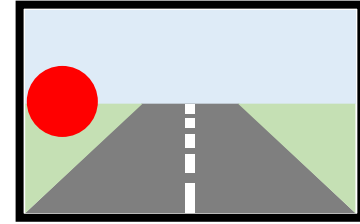
Some practical applications

- Recommendation systems (collaborative filtering)
- Given a collection of n users who have rated d movies each
- Discover of k user-types and a representation of each user in terms of these types.

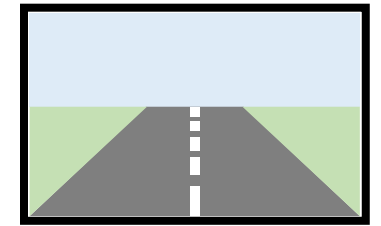
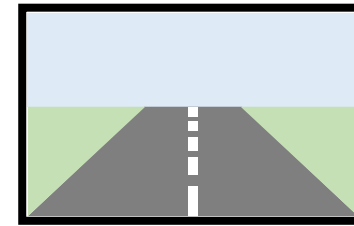
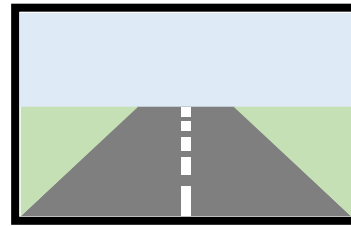
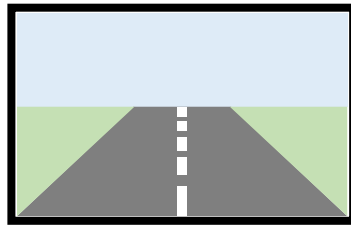
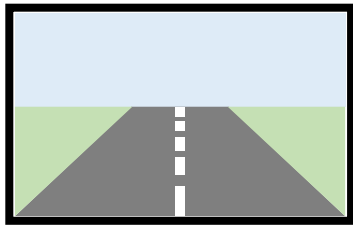
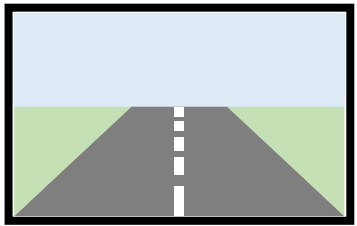


Some practical applications

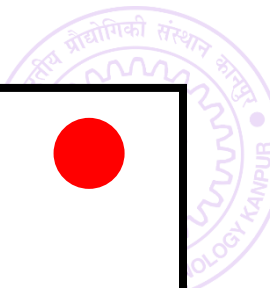
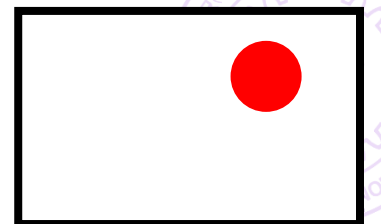
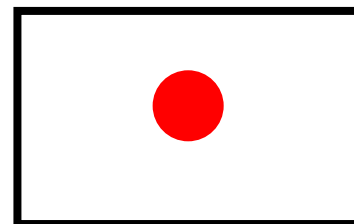
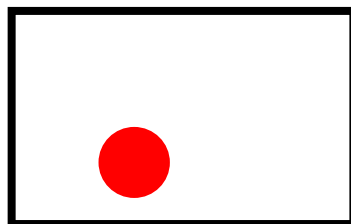
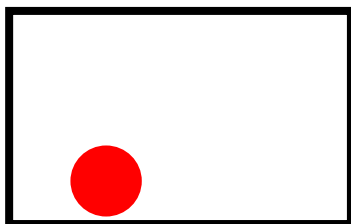
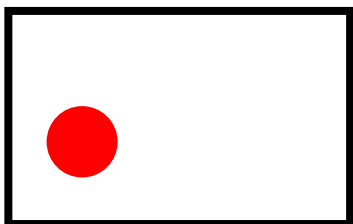
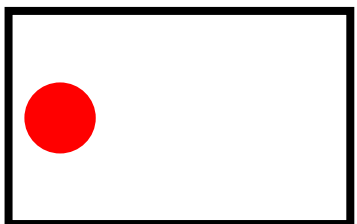
- Foreground/background separation



=



+



Some practical applications

- Make every frame a vector $\mathbf{x}^i \in \mathbb{R}^d$. n frames $X = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n] \in \mathbb{R}^{d \times n}$
- Background is a constant vector $\mathbf{b} \in \mathbb{R}^d$. Let $\mathbf{1} = [1, 1, 1, \dots, 1]^\top \in \mathbb{R}^n$
- Foreground treated as noise $\mathbf{f}^i \in \mathbb{R}^d$. Let $F = [\mathbf{f}^1, \mathbf{f}^2, \dots, \mathbf{f}^n] \in \mathbb{R}^{d \times n}$
- We have $X = \mathbf{b} \cdot \mathbf{1}^\top + F$
- Decompose X into low-rank components and recover F as noise



=



+



Please give your Feedback

<http://tinyurl.com/ml17-18afb>