Data Modelling Methods-V

CS771: Introduction to Machine Learning

Purushottam Kar

Outline of today's discussion

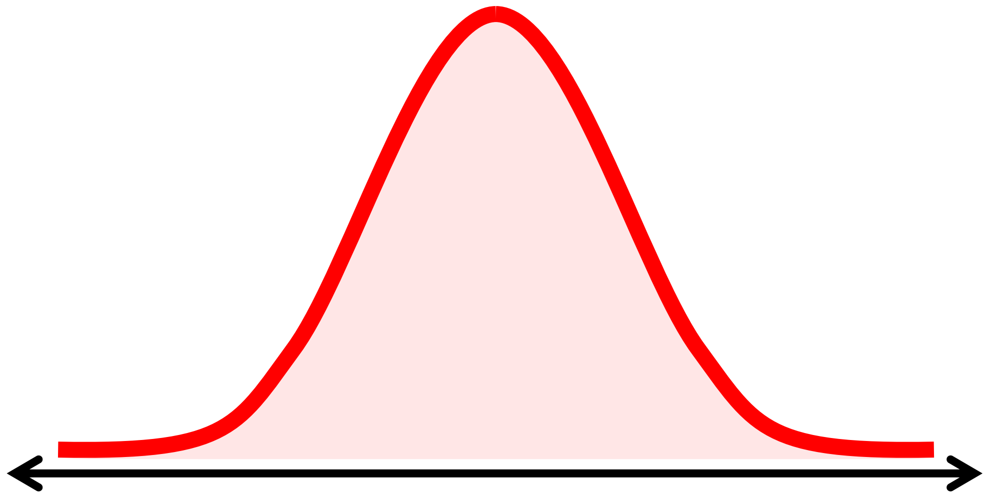
- Recap of PCA, PPCA
- Details of PCA and applications

Recap

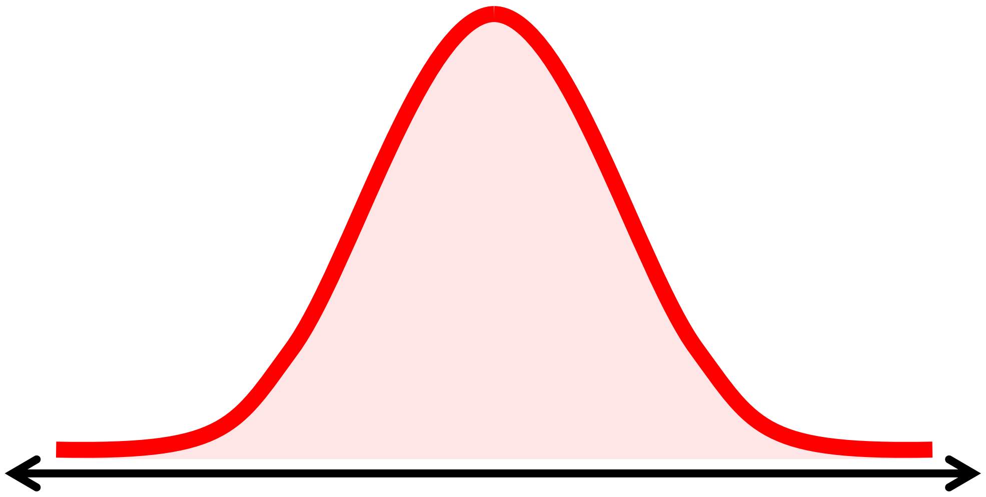
Modelling Low-dimensional structure in data

Low-dimensional Structure in Data

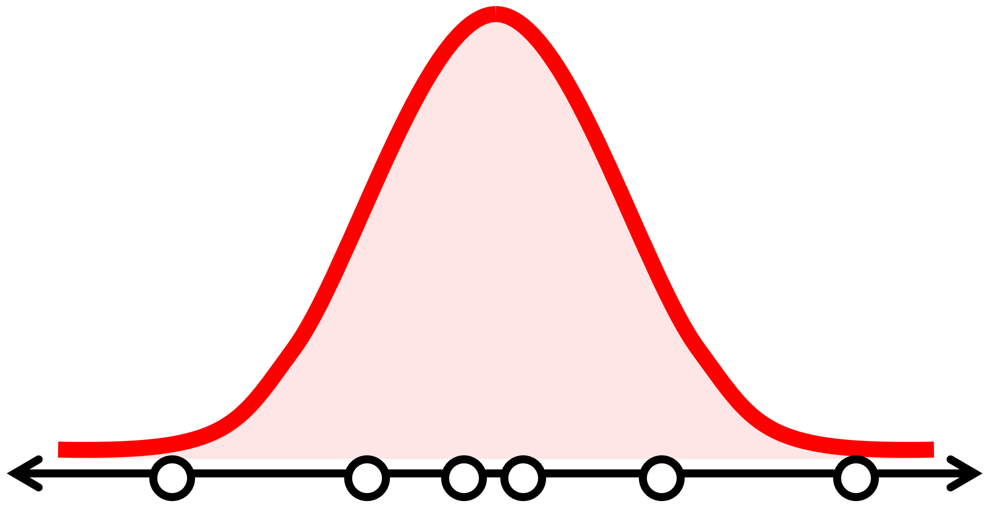
Low-dimensional Structure in Data



Low-dimensional Structure in Data

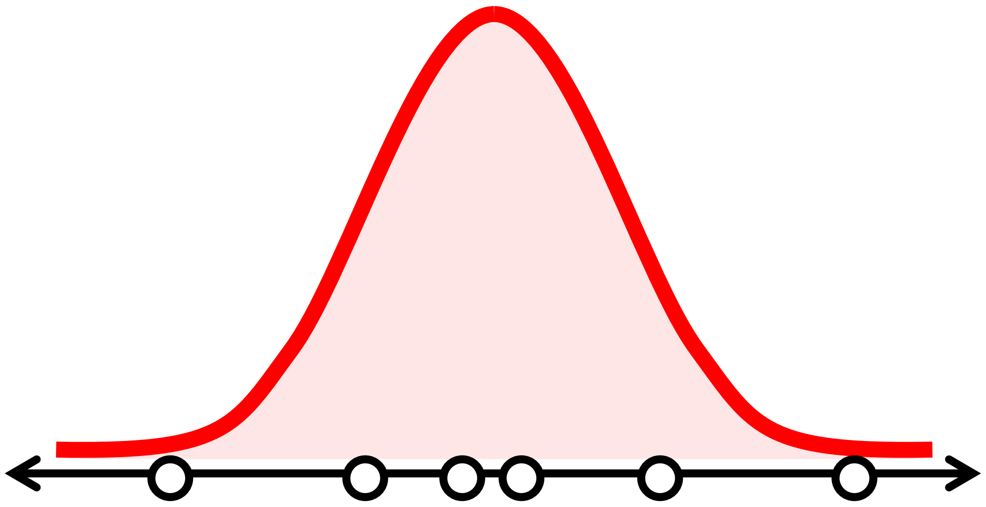


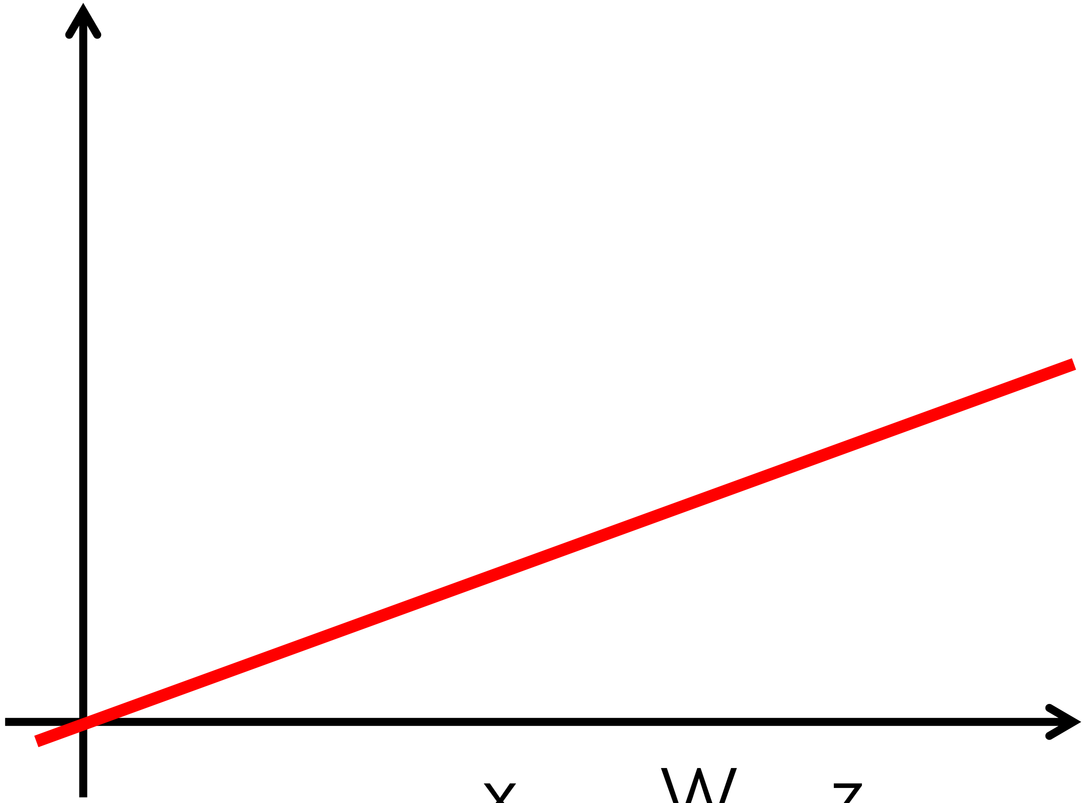
Low-dimensional Structure in Data



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

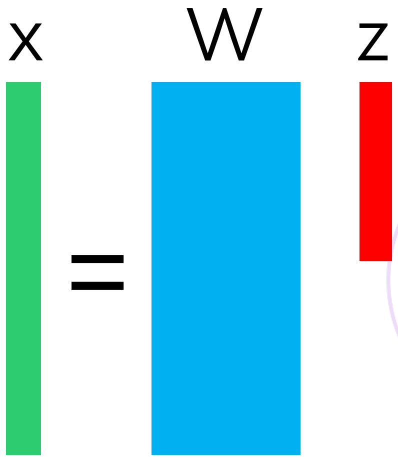
Low-dimensional Structure in Data



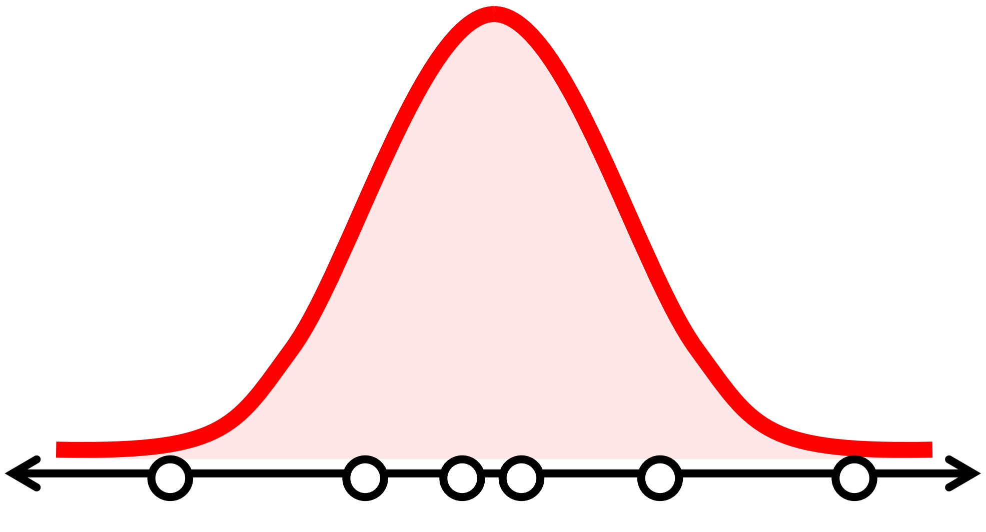


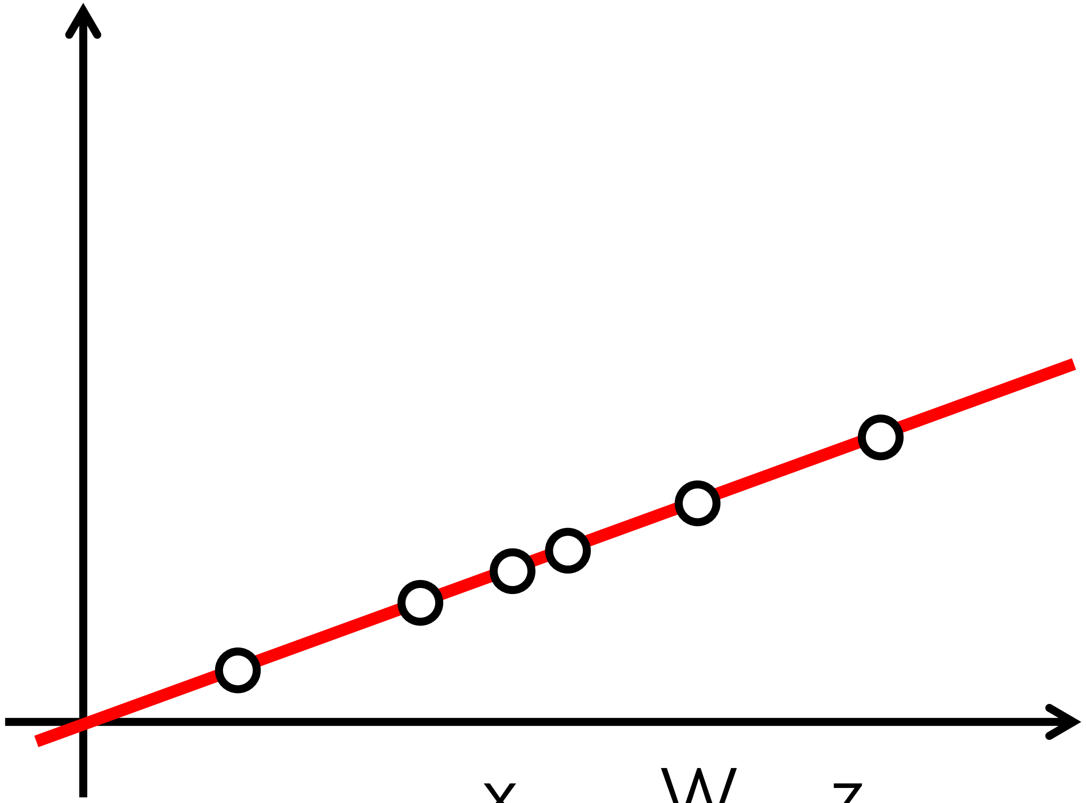
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

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



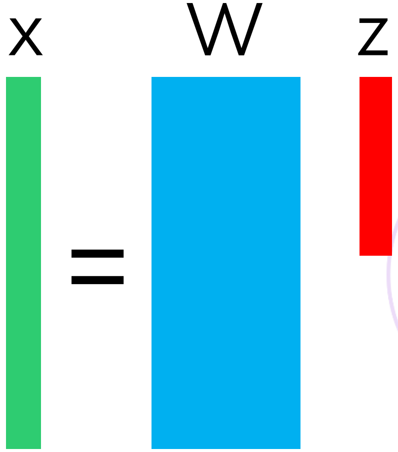
Low-dimensional Structure in Data



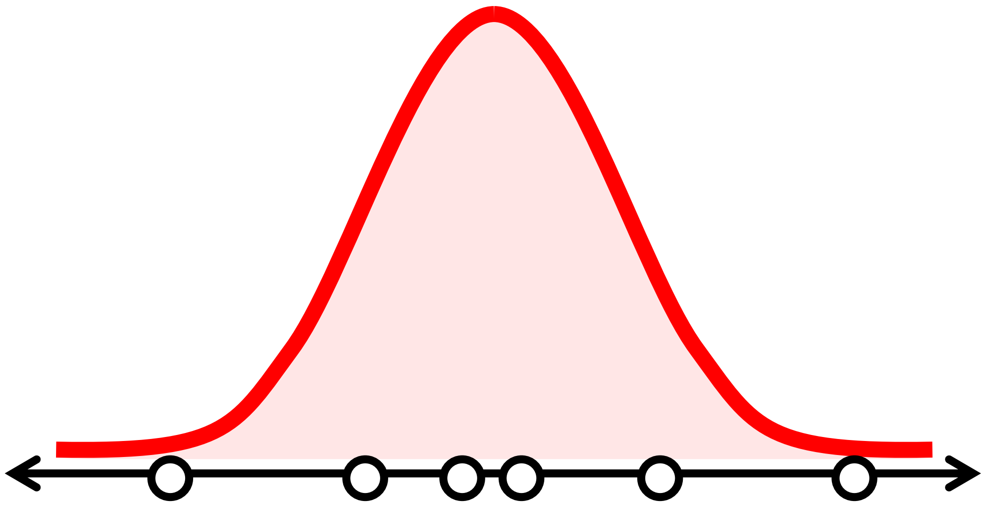


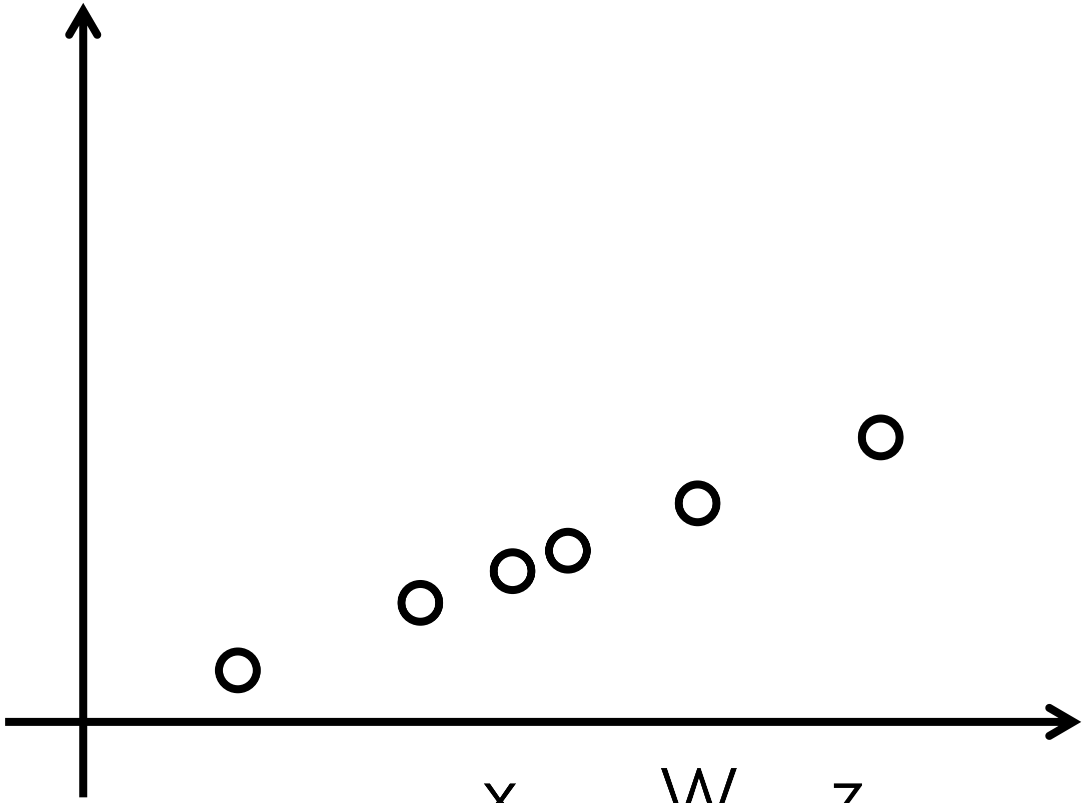
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

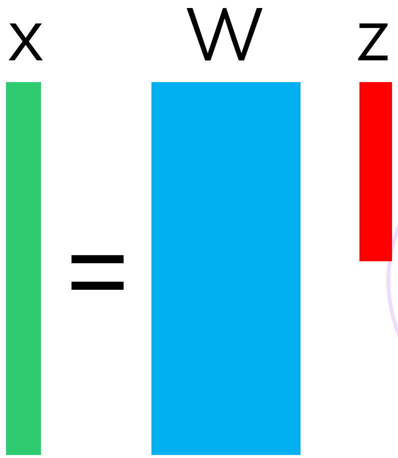
$$W \in \mathbb{R}^{d \times k}$$



Low-dimensional Structure in Data



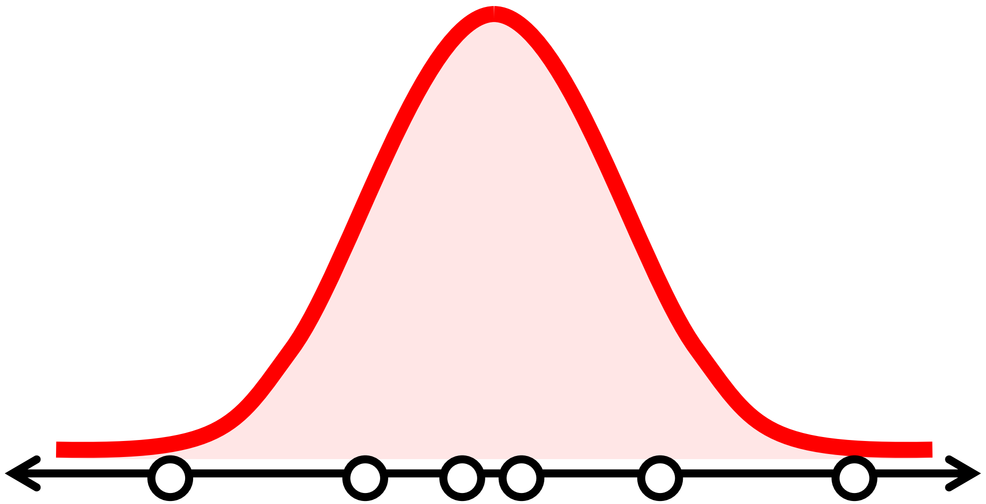


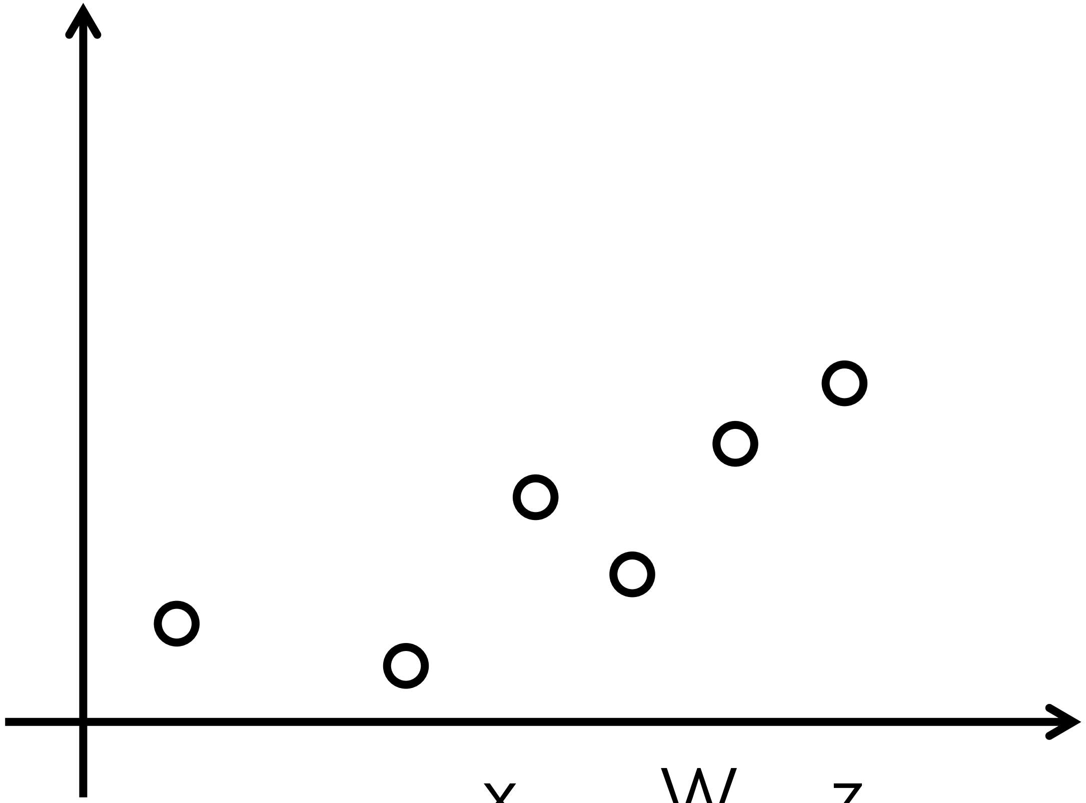


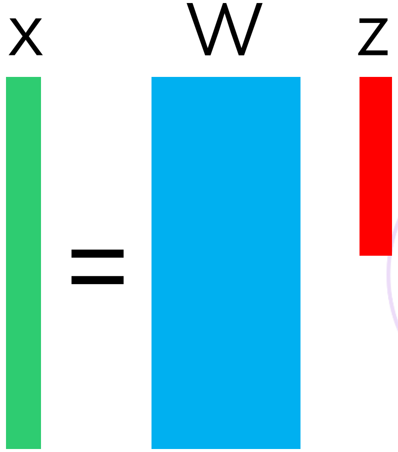
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

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

Low-dimensional Structure in Data





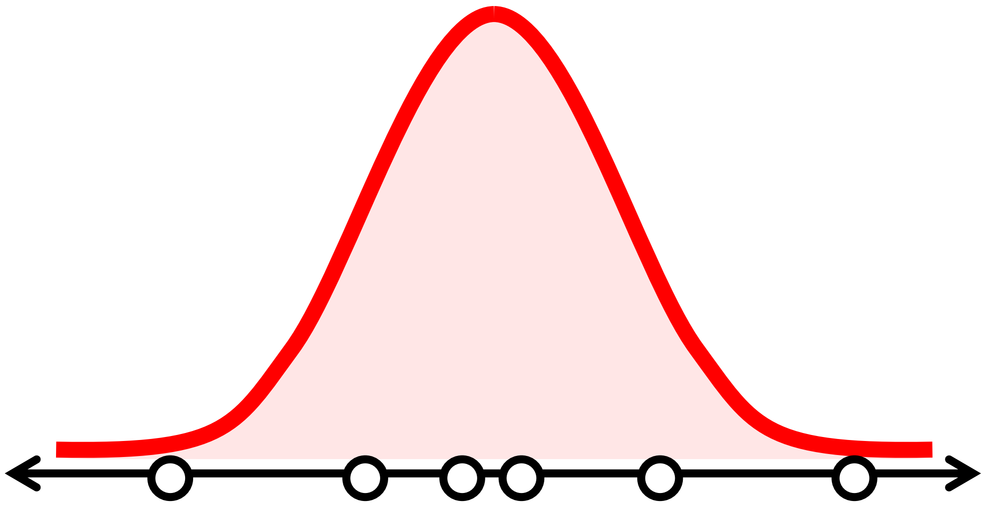


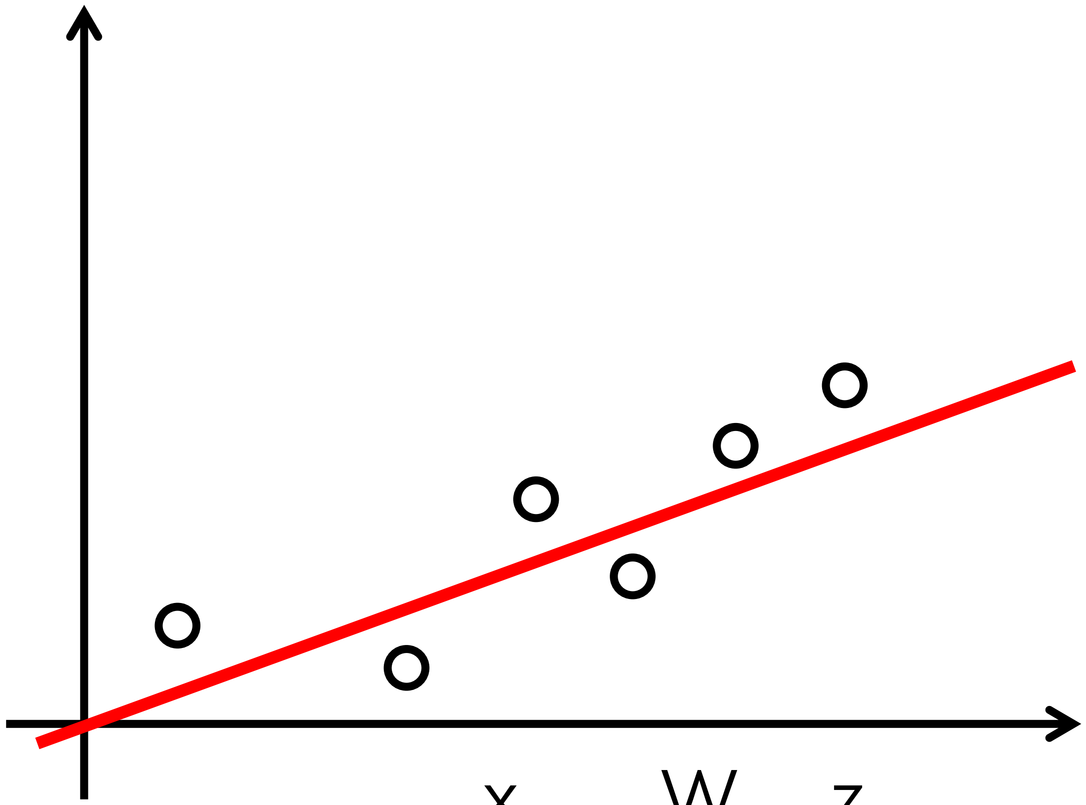
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

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

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

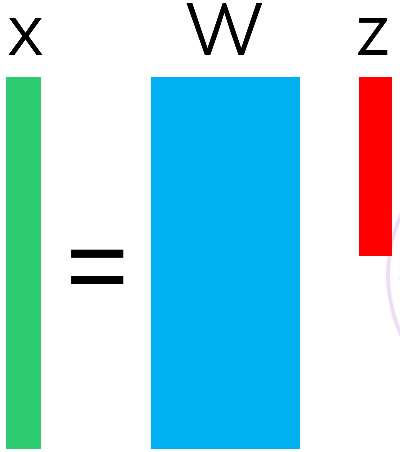
Low-dimensional Structure in Data



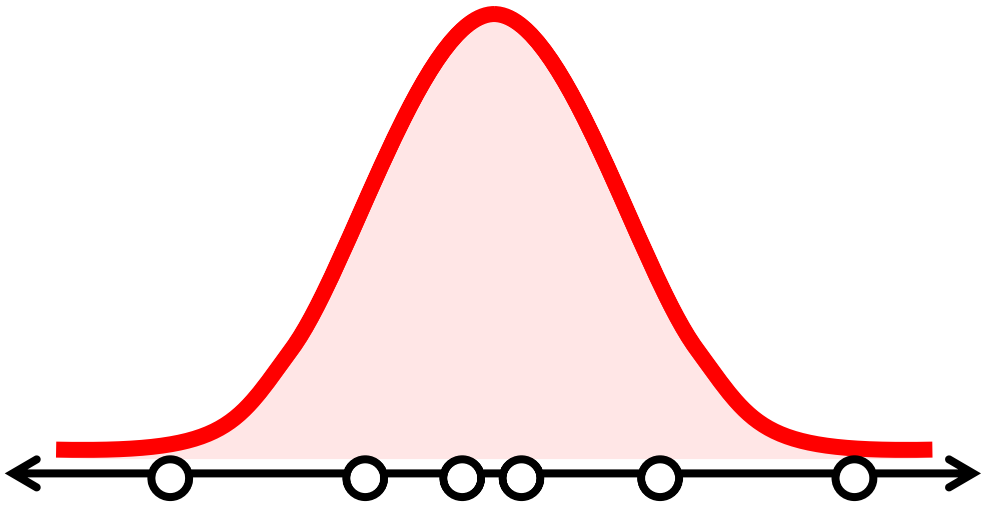


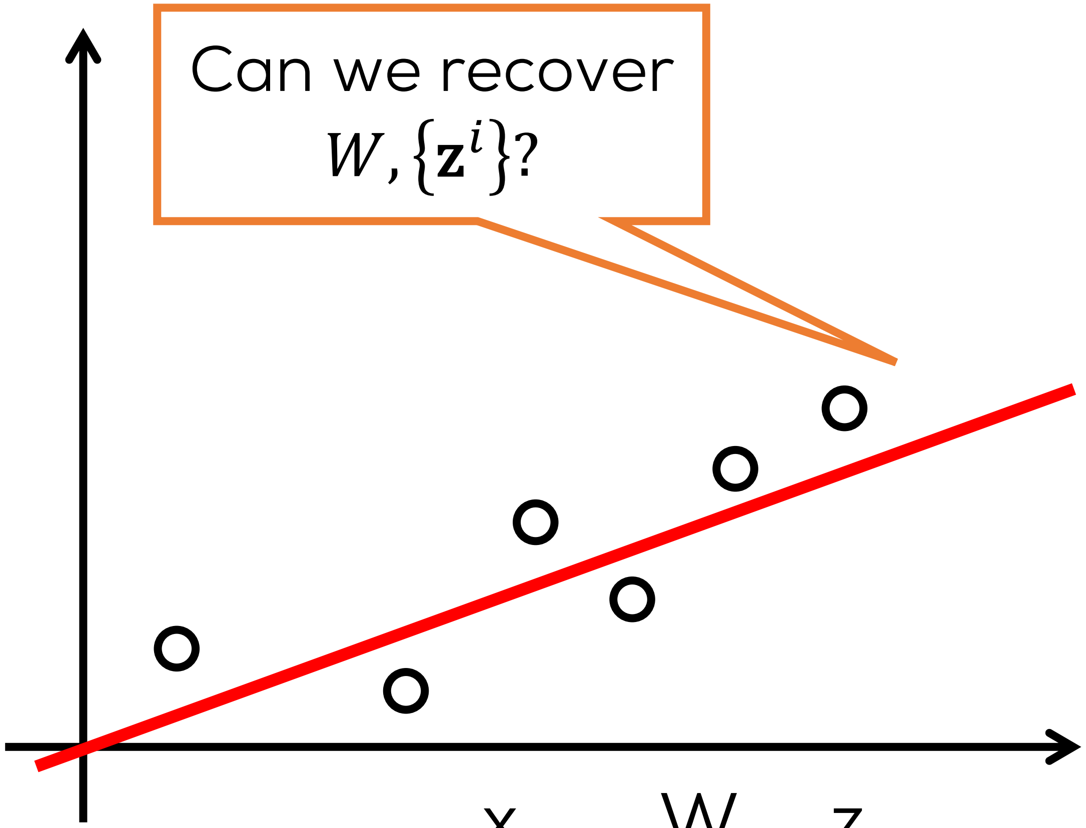
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i,\ \ \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

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



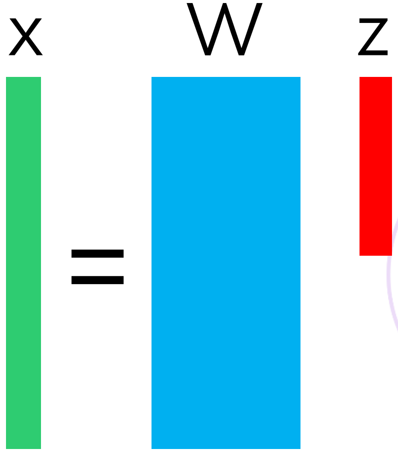
Low-dimensional Structure in Data



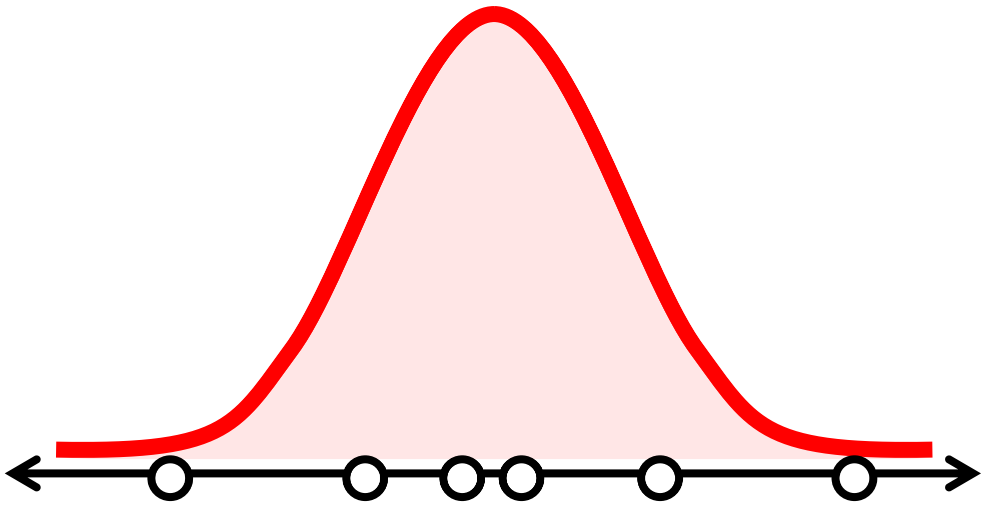


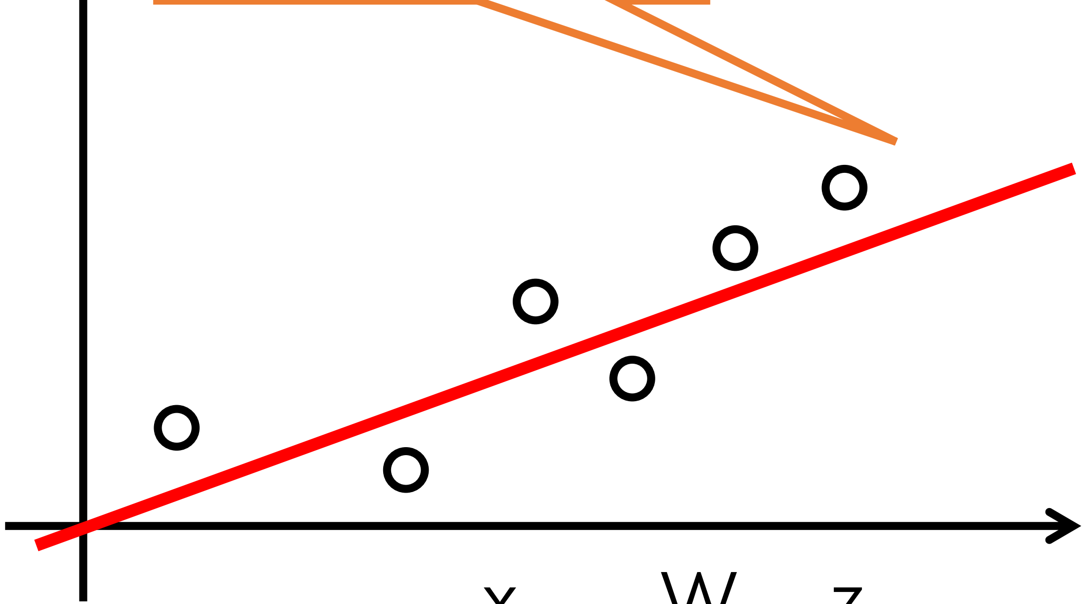
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i,\ \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

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



Low-dimensional Structure in Data



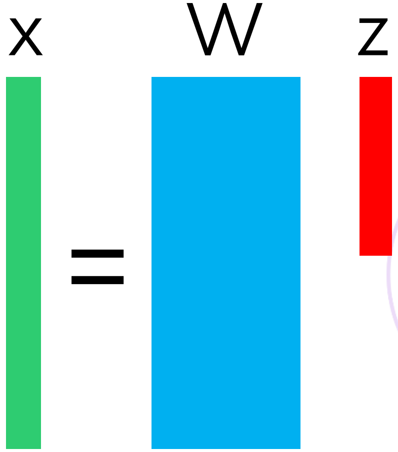


Can we recover $W, \{\mathbf{z}^i\}$?

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

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

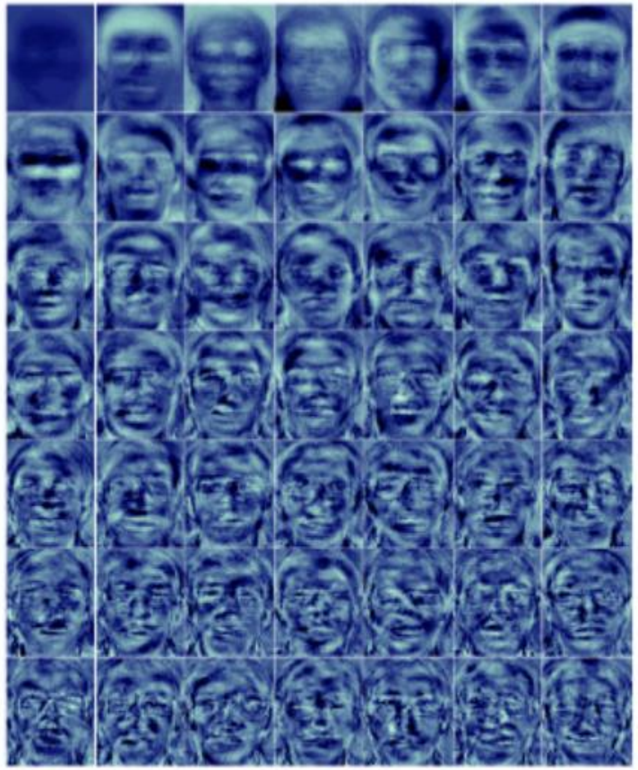
Credits: Piyush Rai, CS771, 2016-17-I

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



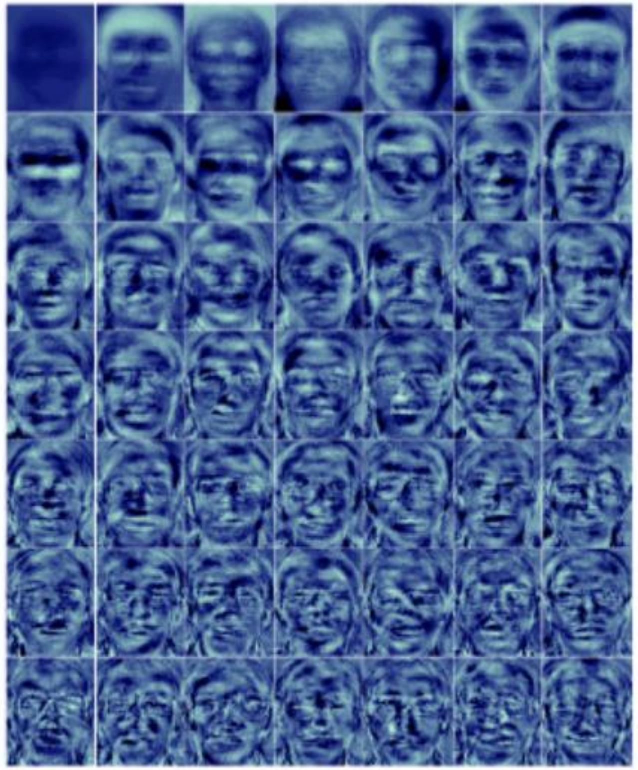
17-1

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

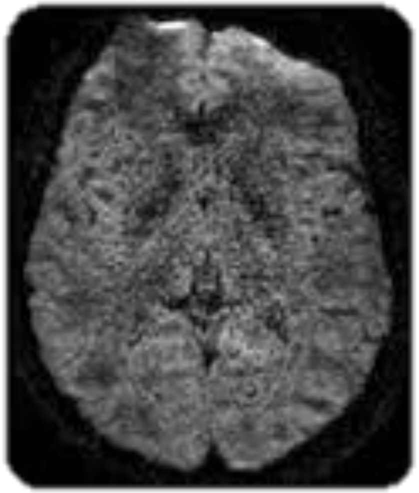


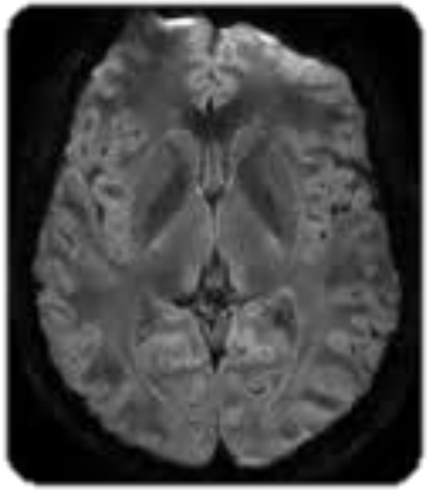
17-1

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)

 =  + 

http://personales.upv.es/jmanjon/
Netrapalli et al, Non-convex Robust PCA, NIPS 2014

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 $\mathbf{x}^i | \mathbf{z}^i \sim \mathcal{N}\left(W\mathbf{z}^i, \sigma^2 \cdot I_d\right)$
- 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 $\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \mathcal{N}(0, \Sigma)$ 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

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

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

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

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

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

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

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

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)

$$\widehat{W}_{\mathrm{MLE}} = \mathrm{U}_{\mathrm{k}}\sqrt{\Lambda_k}$$

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)
$$\widehat{W}_{\mathrm{MLE}} = \mathrm{U}_{\mathrm{k}}\sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \ldots, k$
   1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow$ 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 $\widehat{W}_{\mathrm{MLE}} = \left[\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j\right]_{j=1,\ldots,k}$

The PCA estimate

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

$$\widehat{W}_{\text{MLE}} = \text{U}_\text{k}\sqrt{\Lambda_k}$$

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 $\left(\hat{\lambda}_j, \hat{\mathbf{u}}_j\right) \leftarrow$ POWER–METHOD$\left(S^{j-1}\right)$
   2. Let $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j \left(\hat{\mathbf{u}}_j\right)^\top$
4. Return $\widehat{W}_{\text{MLE}} = \left[\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j\right]_{j=1,\dots,k}$

The peeling technique

The PCA estimate

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

$$\widehat{W}_{\mathrm{MLE}} = \mathrm{U_k}\sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d\times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \ldots, k$
   1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow$ 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 $\widehat{W}_{\mathrm{MLE}} = \left[\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j\right]_{j=1,\ldots,k}$

Overall $O(d^2k)$ time

The peeling technique

The PCA estimate

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

$$\widehat{W}_{\mathrm{MLE}} = \mathrm{U}_{\mathrm{k}}\sqrt{\Lambda_k}$$

PRINCIPAL COMPONENT ANALYSIS

1. Matrix $S \in \mathbb{R}^{d \times d}$
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \ldots, k$
    1. Let $(\hat{\lambda}_j, \hat{\mathbf{u}}_j) \leftarrow$ 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 $\widehat{W}_{\mathrm{MLE}} = \left[\sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j\right]_{j=1,\ldots,k}$

Overall $O(d^2 k)$ time

The peeling technique

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

The Power Method

THE POWER METHOD

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

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

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

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 $S \in \mathbb{R}^{d \times \mathrm{d}}$
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

The Power Method

THE POWER METHOD

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

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

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

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

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

PROBABILISTIC PCA

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

Recall that

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

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 $j = 1, \ldots, d$
   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 $\widehat{W}_{\text{MLE}} = \left[\sqrt{\hat{\lambda}_j - \hat{\sigma}^2_{\text{MLE}}} \cdot \hat{\mathbf{u}}\right]$

Recall that

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

How to recover z^i? Wait a bit.

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

PROBABILISTIC PCA

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

Recall that

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

Takes $O(d^3)$ time ☹

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

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

PROBABILISTIC PCA

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

Recall that

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

Takes $O(d^3)$ time ☹

Can we do better?

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

MLE Estimation

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

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

Need to find top k eigenvectors and all the eigenvalues

MLE Estimation

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

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

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

Need to find top k eigenvectors and all the eigenvalues

Exercise!

MLE Estimation

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

Need to find top k eigenvectors and top k eigenvalues

Exercise!

MLE Estimation

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n$ from $\mathcal{N}$ … log-likelihood is

$$\log \mathbb{P}[X \mid W, \sigma] = \frac{n}{2}\Big(d \log \ldots {}^{1}S)\Big)$$

  where $C = WW^\top + \sigma^2 \cdot I_d$, and $S =$ …

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

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

Need to find top k eigenvectors and top k eigenvalues

Exercise!

MLE Estimation

Improvement over last lecture's claim

- Given samples $\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n$ from $\mathcal{N}$ … likelihood is

$$\log \mathbb{P}[X \mid W, \sigma] = \frac{n}{2}\left(d \log \dots {}^{-1}S)\right)$$

where $C = WW^\top + \sigma^2 \cdot I_d$, and $S = \frac{}{n}$ …

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

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

Need to find top k eigenvectors and top k eigenvalues

Exercise!

FA Interpretations for PCA

Principal Component Analysis

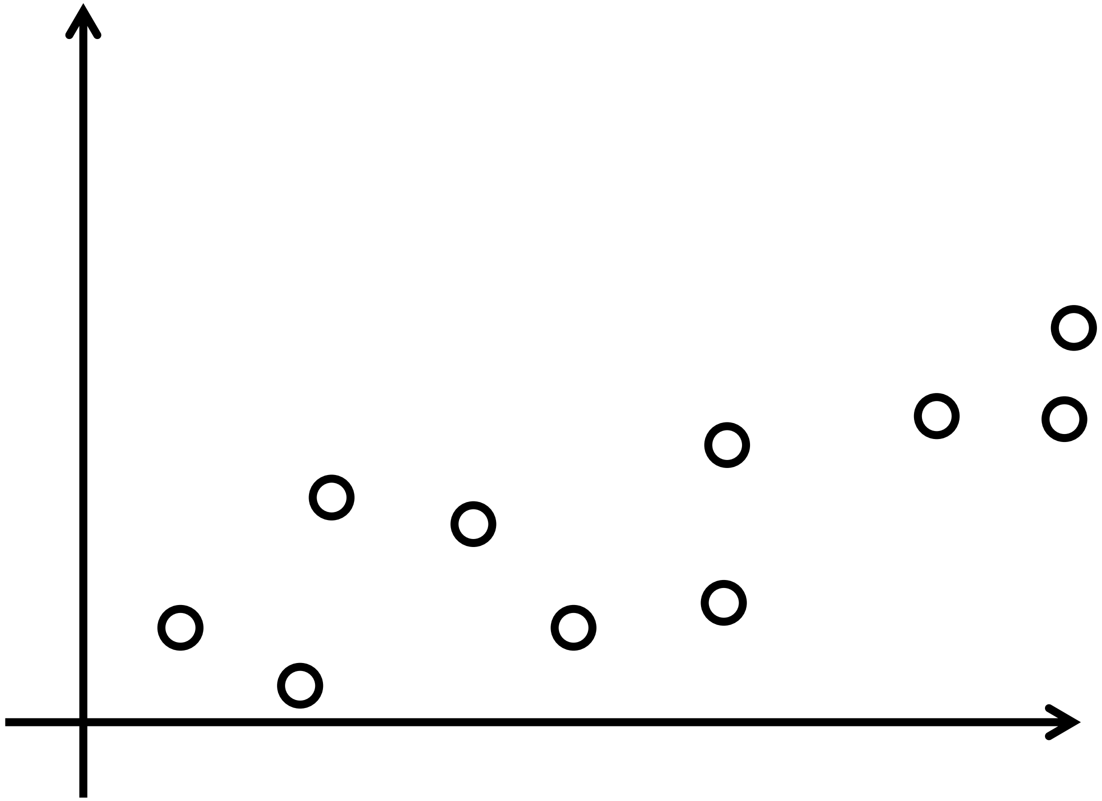
- Dates back more than a century [Pearson, 1901; Hotelling, 1930]
- Seeks to find the "closest" low-dim representation of data
- Also seeks to capture the maximum "variance" in the data

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

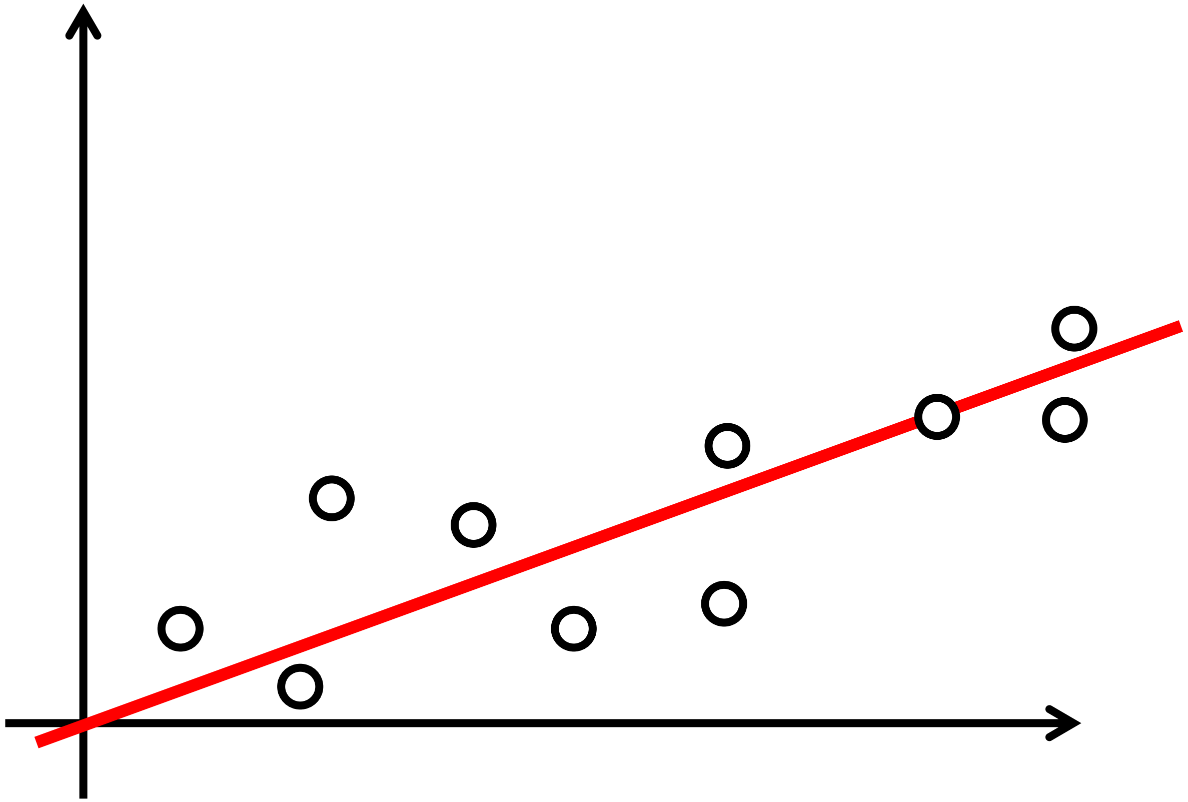
Principal Component Analysis

- 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



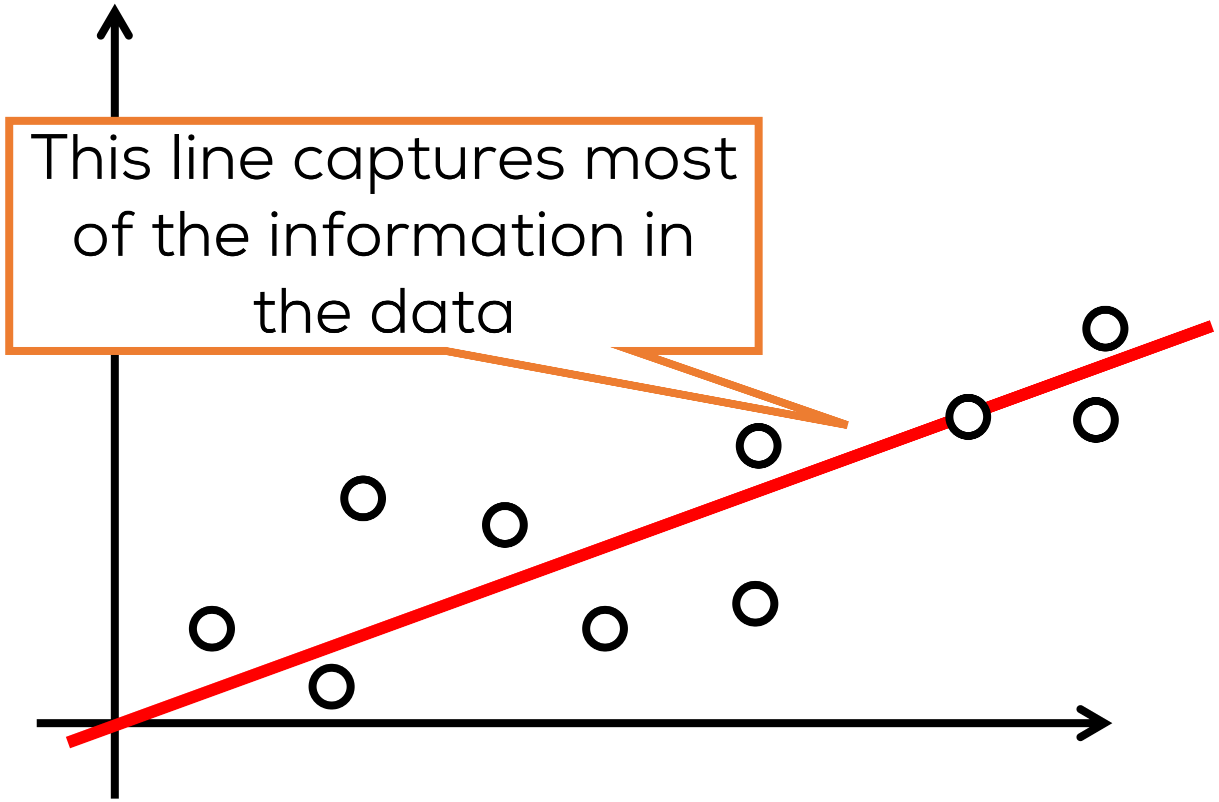
Principal Component Analysis

- 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



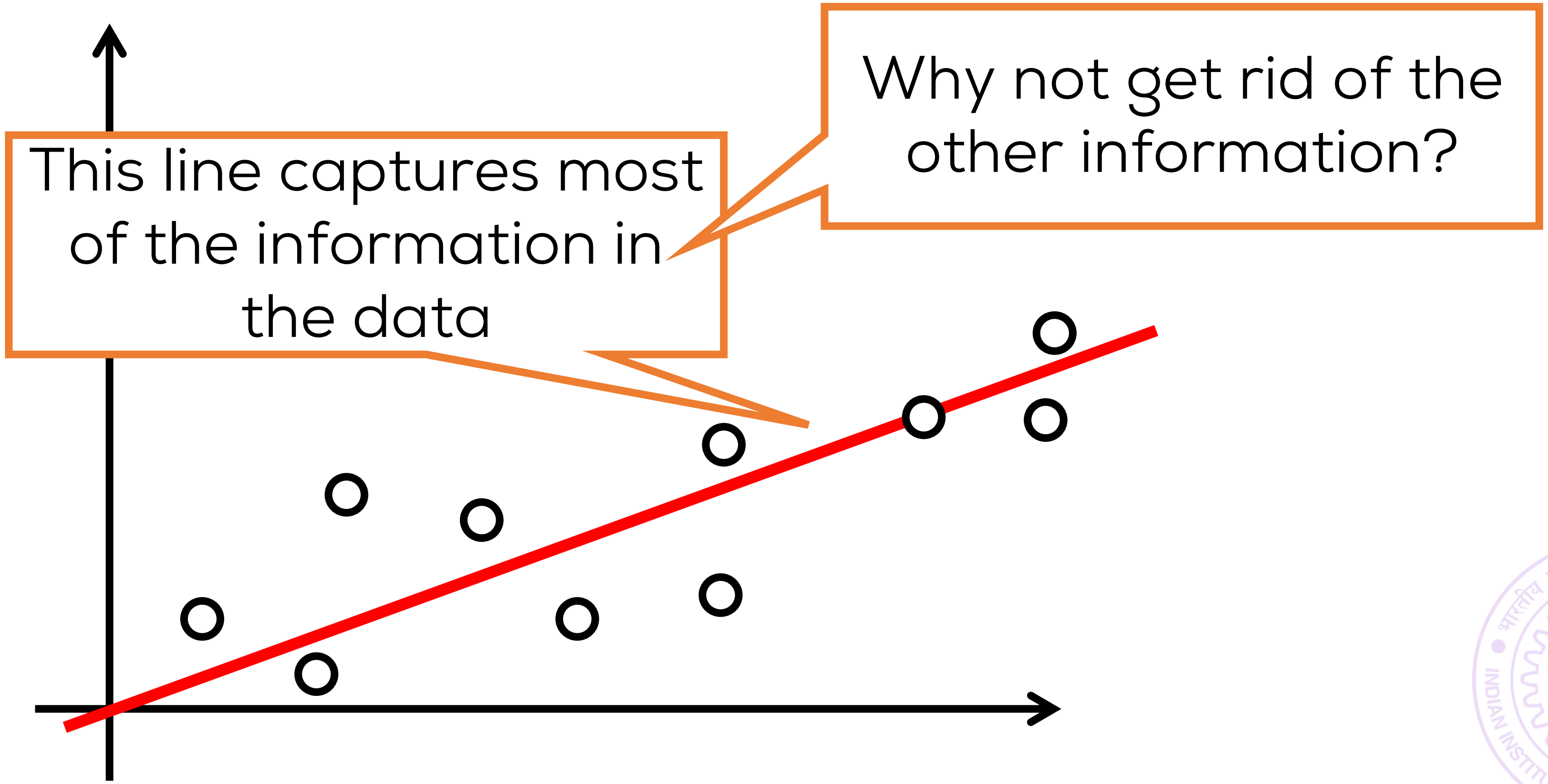
Principal Component Analysis

- 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



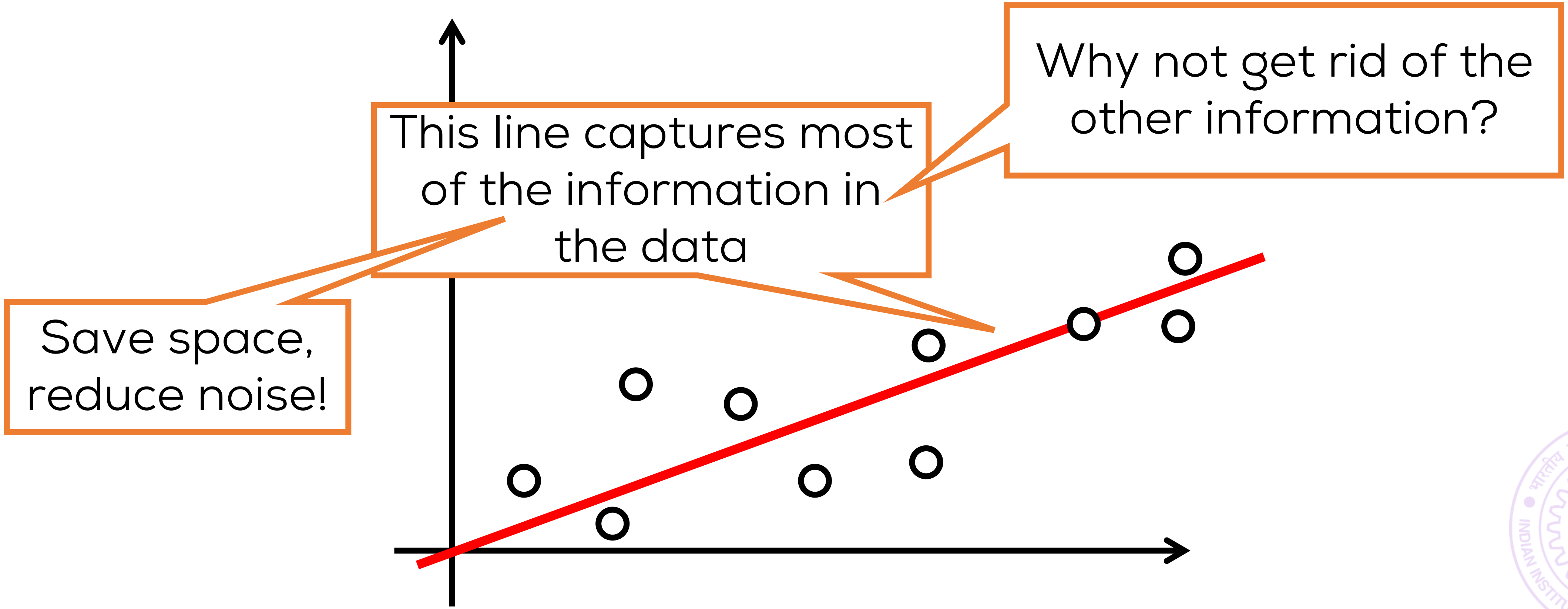
Principal Component Analysis

- 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



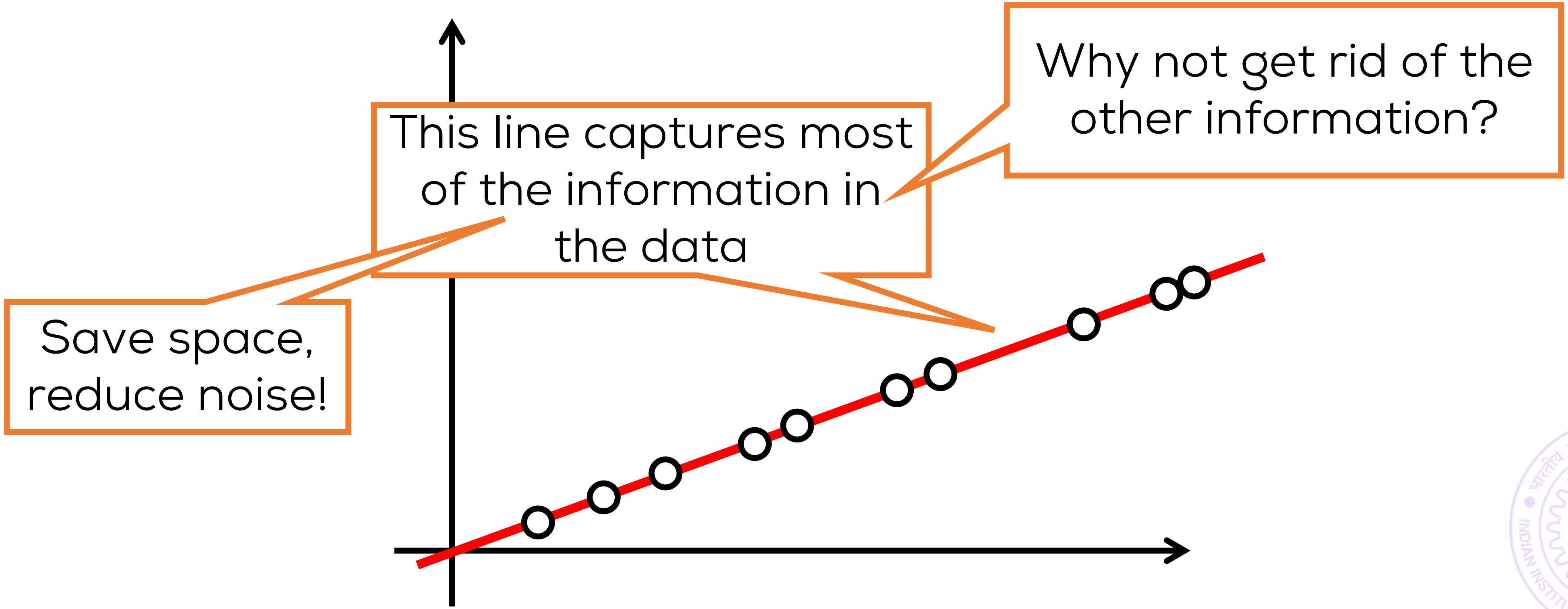
Principal Component Analysis

- 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 $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$
- 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 $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$ and line corresponding to $\mathbf{w} \in \mathbb{R}^d$, can find out how well the line "fits" the points

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

Exercise!

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

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

- Simple calculations show that the above is equivalent to

$$\underset{\|\mathbf{w}\|_2=1}{\arg\max}\, \mathbf{w}^\top S \mathbf{w}$$

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

Exercise!

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$
- 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

$$\underset{\|\mathbf{w}\|_2=1}{\arg\max}\, \mathbf{w}^\top S \mathbf{w}$$

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. $\boldsymbol{w} = \boldsymbol{u}^1$
- **Proof**: write $S = U\Lambda U^\top$, $U = [\mathbf{u}^1, \ldots, \mathbf{u}^d], \Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_d)$, $\lambda_1 \geq \lambda_2 \geq \cdots$
- Then $\mathbf{w}^\top S \mathbf{w} = \sum_{j=1}^d \lambda_j \left(\langle \mathbf{u}^j, \mathbf{w} \rangle\right)^2$. 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 $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$
- 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 $\mathbf{w}^1, \mathbf{w}^2, \ldots, \mathbf{w}^k \in \mathbb{R}^d$ (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 $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$
- 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

$$\underset{W^\top W = I_k}{\arg\min} \sum_{i=1}^{n} \left\| \mathbf{x}^i - WW^\top \mathbf{x}^i \right\|_2^2 \equiv \underset{W^\top W = I_k}{\arg\max}\, \mathrm{tr}(W^\top S W)$$

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

Exercise!

Exercise!

PCA as minimizing reconstruction error

- **Proof**: by induction. Base case $k = 1$ already done
- Do a bit of rewriting

$$\underset{W^\top W = I_k}{\arg\max}\, \mathrm{tr}(W^\top S W) = \underset{\substack{\widetilde{W}^\top \widetilde{W} = I_{k-1} \\ \widetilde{W}^\top \boldsymbol{v} = \mathbf{0}}}{\arg\max}\, \underset{\|\boldsymbol{v}\|_2 = 1}{\arg\max}\, \mathrm{tr}\left([\boldsymbol{v}\ \widetilde{W}]^\top S [\boldsymbol{v}\ \widetilde{W}] \right)$$

$$= \underset{\substack{\widetilde{W}^\top \widetilde{W} = I_{k-1} \\ \widetilde{W}^\top \boldsymbol{v} = \mathbf{0}}}{\arg\max} \left\{ \left\{ \underset{\|\boldsymbol{v}\|_2 = 1}{\arg\max}\, \boldsymbol{v}^\top S \boldsymbol{v} \right\} + \mathrm{tr}\left(\widetilde{W}^\top S \widetilde{W} \right) \right\}$$

$$= (\mathbf{u}^1)^\top S \mathbf{u}^1 + \underset{\substack{\widetilde{W}^\top \widetilde{W} = I_{k-1} \\ \widetilde{W}^\top \boldsymbol{u}^1 = \mathbf{0}}}{\arg\max}\, \mathrm{tr}\left(\widetilde{W}^\top S \widetilde{W} \right) = (\mathbf{u}^1)^\top S \mathbf{u}^1 + \underset{\widetilde{W}^\top \widetilde{W} = I_{k-1}}{\arg\max}\, \mathrm{tr}\left(\widetilde{W}^\top \tilde{S} \widetilde{W} \right)$$

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

The peeling technique

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

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$
- 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

$$\underset{W^\top W = I_k}{\arg\min} \sum_{i=1}^{n} \left\|\mathbf{x}^i - WW^\top\mathbf{x}^i\right\|_2^2 \equiv \underset{W^\top W = I_k}{\arg\max}\, \mathrm{tr}(W^\top S W)$$

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

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
- 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

$$\underset{W^\top W = I_k}{\arg\min} \sum_{i=1}^{n} \left\| \mathbf{x}^i - WW^\top \mathbf{x}^i \right\|_2^2 \equiv \underset{W^\top W = I_k}{\arg\max}\, \mathrm{tr}(W^\top S W)$$

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

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

Exercise!

Exercise!

PCA as minimizing reconstruction error

- Given: n data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$
- 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 $WW^\top\mathbf{x}^i = W\mathbf{z}^{\mathrm{i}}$
- Multiply both sides by $W^\top$ and use the fact that $W^\top W = I_k$

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

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 $W^\top W = I_k$
- This is why we got $W = U_k$ since we also have $U_k^\top U_k = I_k$
- Both FA/PML are valid. They just shift normalization constants
- FA view: $W_{\text{FA}} = U_k$ and $\mathbf{z}^{\mathrm{i}}_{\text{FA}} = W_{\text{FA}}^\top \mathbf{x}^i$
- PML view: $W_{\text{PML}} = U_k\sqrt{\Lambda_k}$ and $\mathbf{z}^{\mathrm{i}}_{\text{PML}} = \Lambda_k^{-1} W_{\text{PML}}^\top \mathbf{x}^i$
- The PML $\mathbf{z}^{\mathrm{i}}$ are "normalized" since PML models $\mathbf{z}^{\mathrm{i}} \sim \mathcal{N}(\mathbf{0}, I_k)$
- The FA W is "normalized" as they are constrained to be so
- Note that $W_{\text{FA}}\mathbf{z}^{\mathrm{i}}_{\text{FA}} = W_{\text{PML}}\,\mathbf{z}^{\mathrm{i}}_{\text{PML}} = U_k U_k^\top \mathbf{x}^i$

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

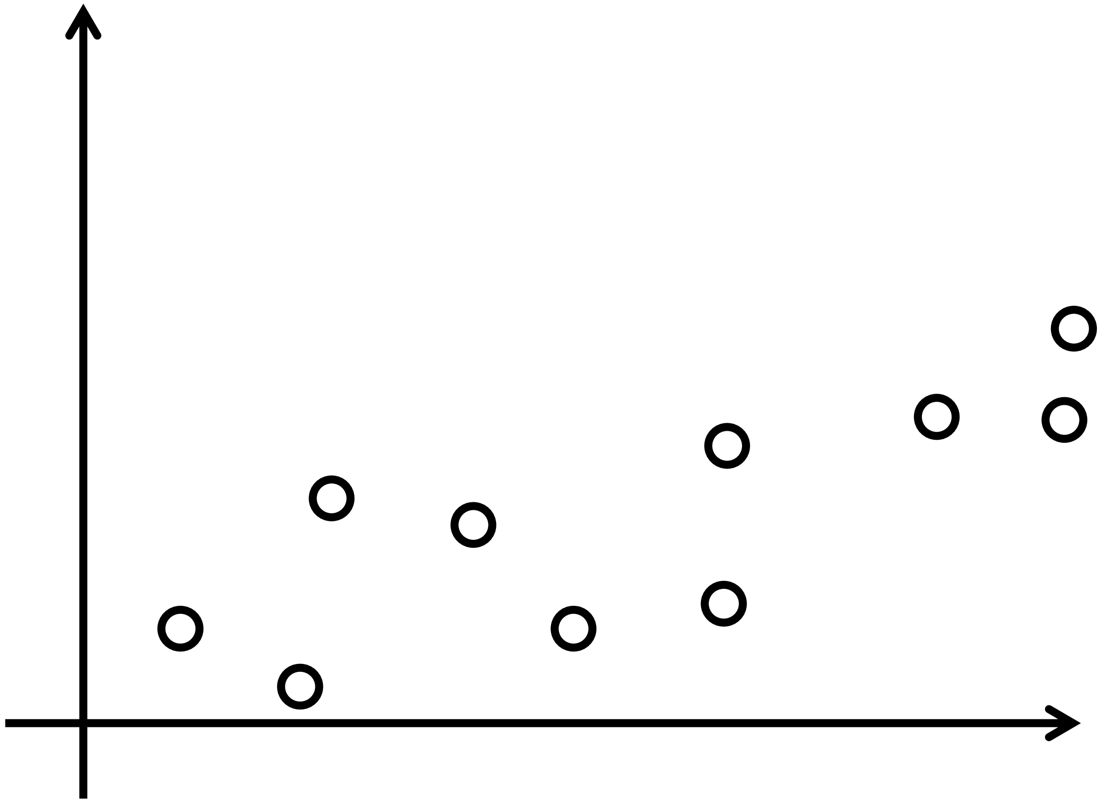
- Data may vary differently along different directions
- Directions with more spread useful and informative

PCA as maximizing capture of variance

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

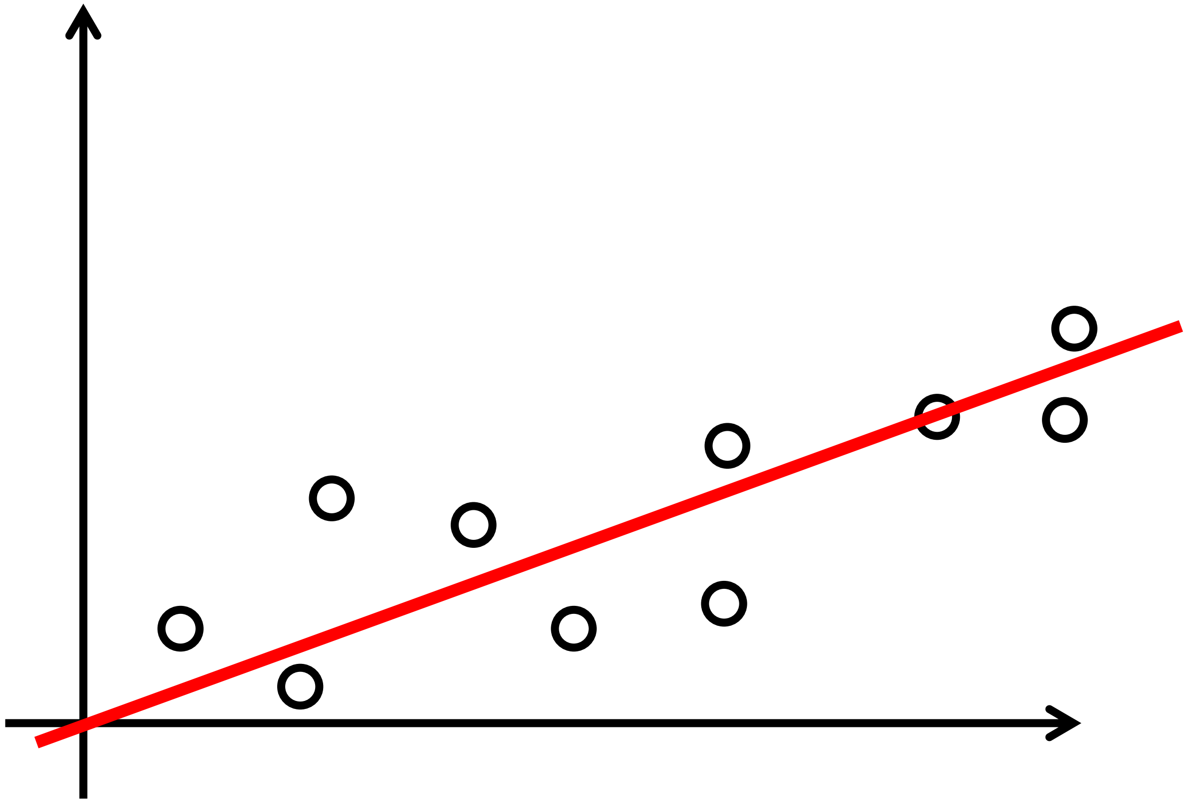
PCA as maximizing capture of variance

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



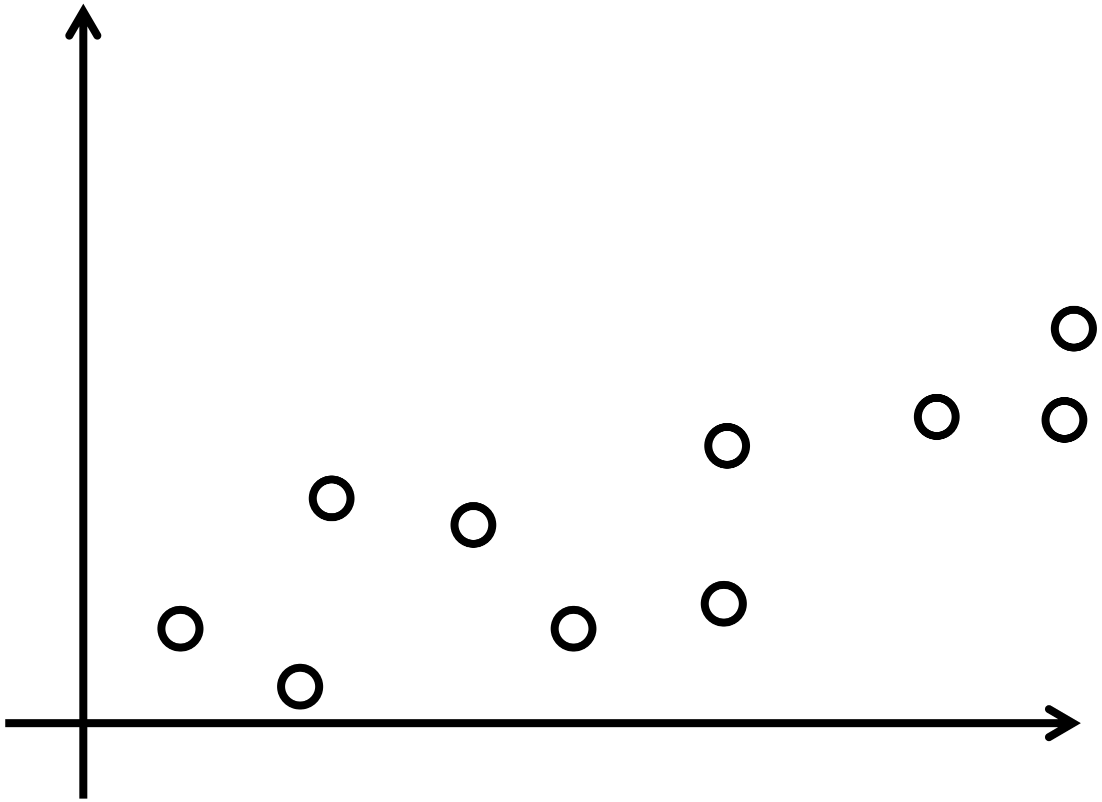
PCA as maximizing capture of variance

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



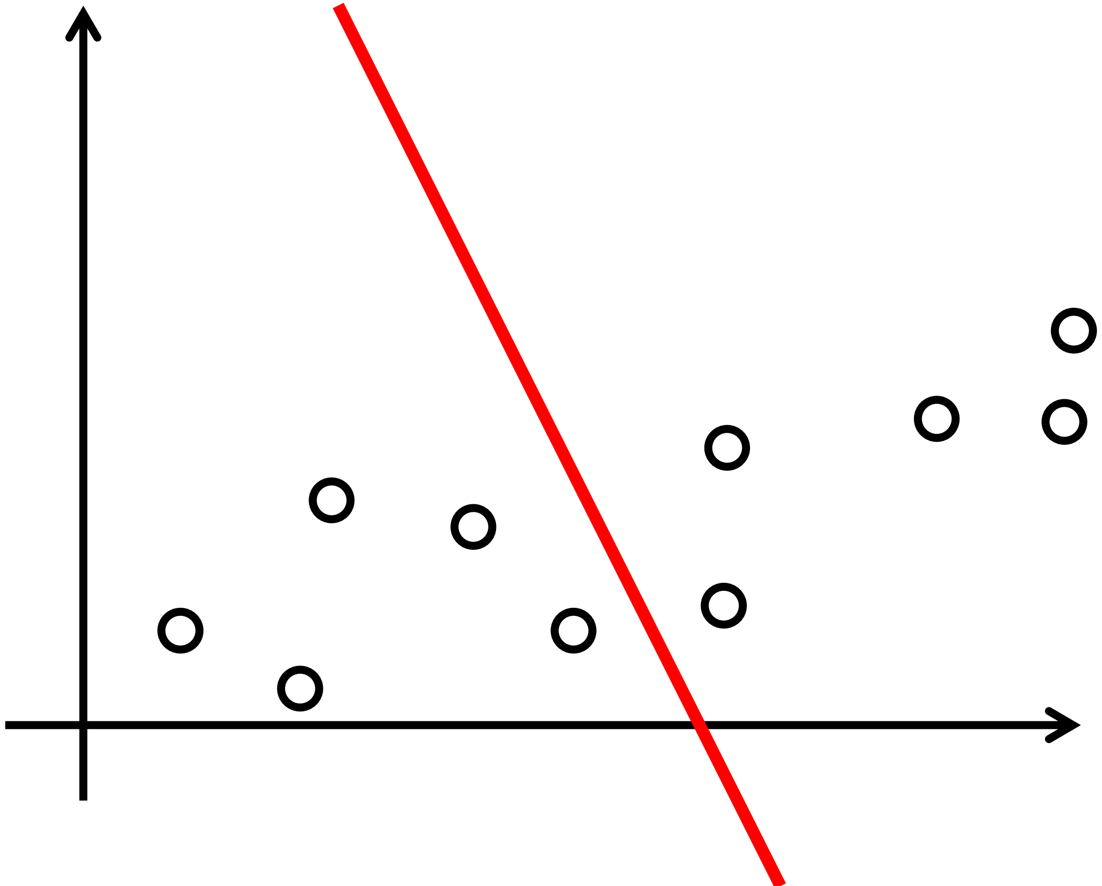
PCA as maximizing capture of variance

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



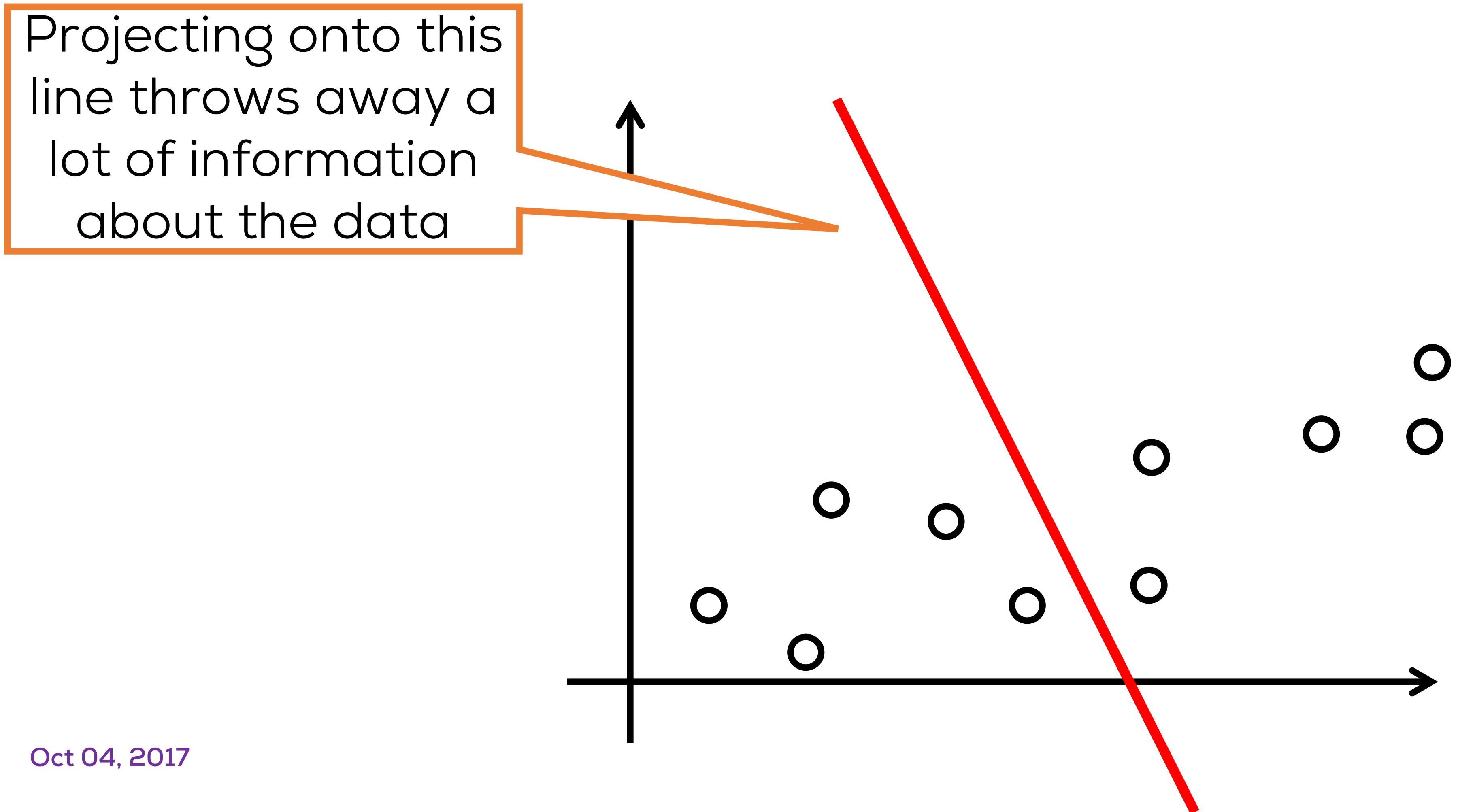
PCA as maximizing capture of variance

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



PCA as maximizing capture of variance

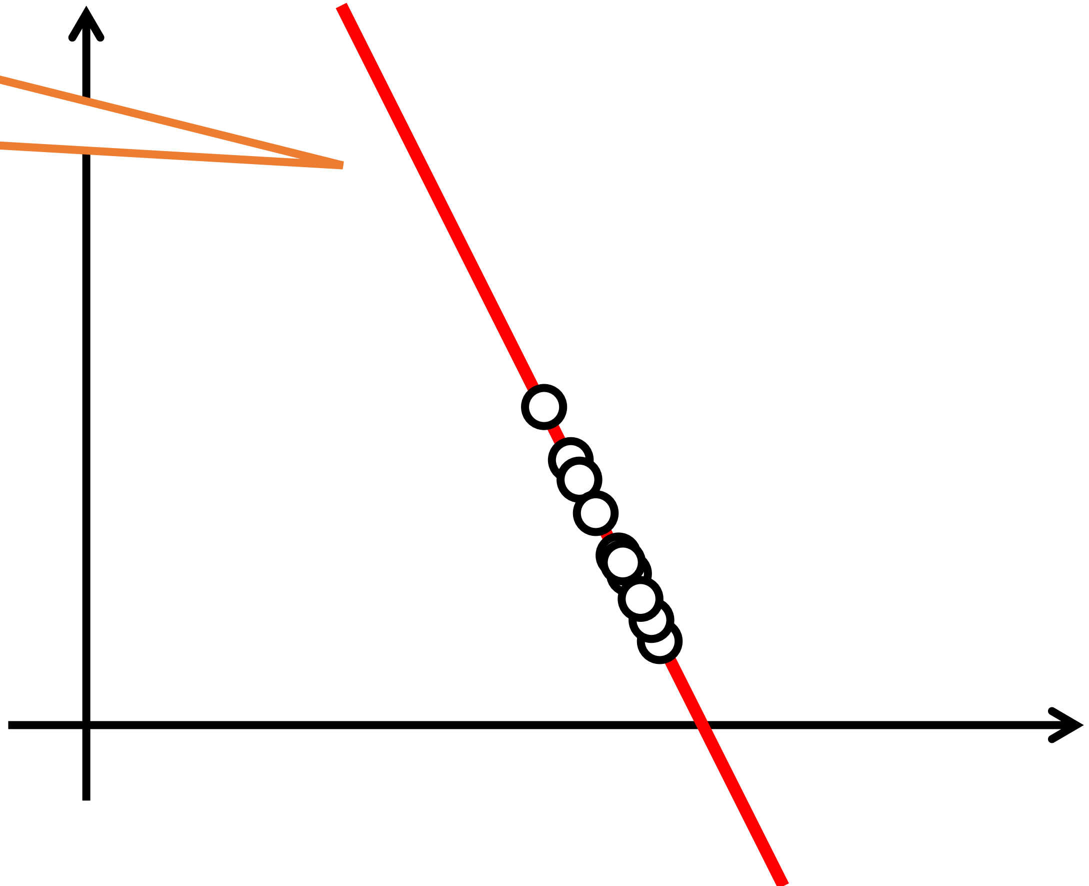
- Data may vary differently along different directions
- Directions with more spread useful and informative



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

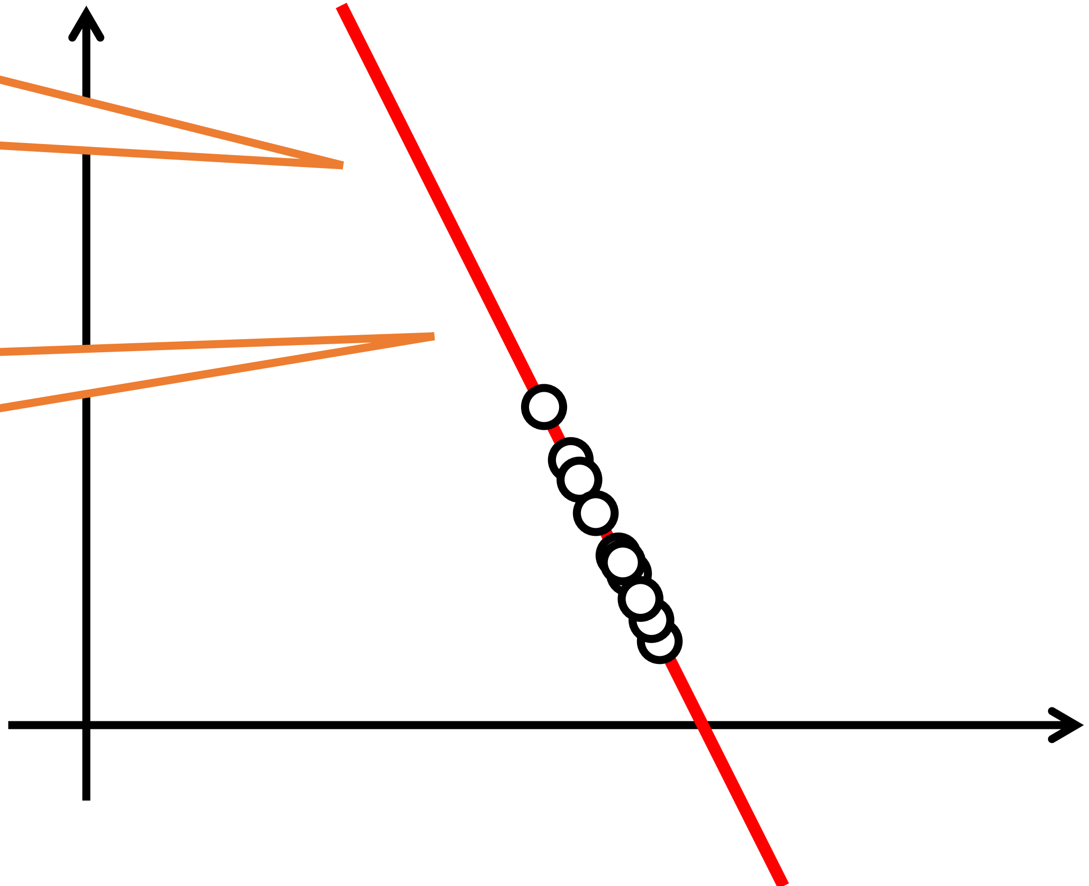


PCA as maximizing capture of variance

- 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

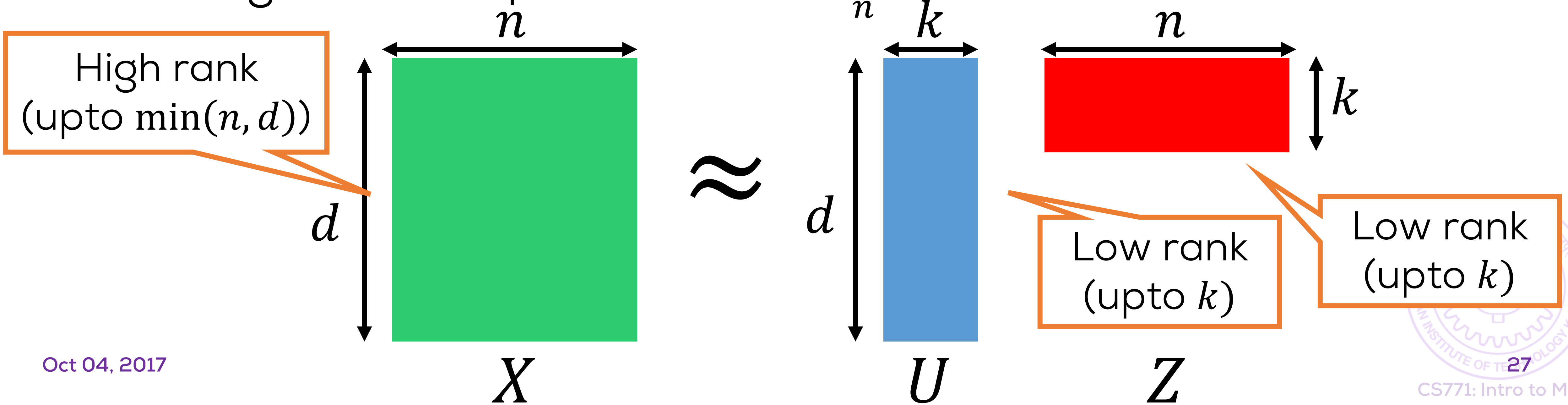
- Given: n data points $\mathbf{x}^1, \ldots, \mathbf{x}^n \in \mathbb{R}^d$ (assume mean centered)
- Suppose we approximate each of these points by the closest point on a line given by $\mathbf{u}$
- The approximations will be $\langle \mathbf{u}, \mathbf{x}^i \rangle \cdot \mathbf{u}$
- Means of these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

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

  where $S = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}^i \left(\mathbf{x}^i\right)^\top$
- 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 $\mathbf{x}^{\mathrm{i}} \approx U\mathbf{z}^{i}$
- For the entire data matrix $X = [\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^{\mathrm{n}}] \in \mathbb{R}^{d \times n}$, this means $X = UZ$, where $Z = [\mathbf{z}^1, \mathbf{z}^2, \dots, \mathbf{z}^{\mathrm{n}}] \in \mathbb{R}^{k \times n}$
- This gives a *low-rank* factorization of the matrix X
- Note: Eigen-decomposition on $S = \frac{1}{n} X X^{\top} \equiv$ 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

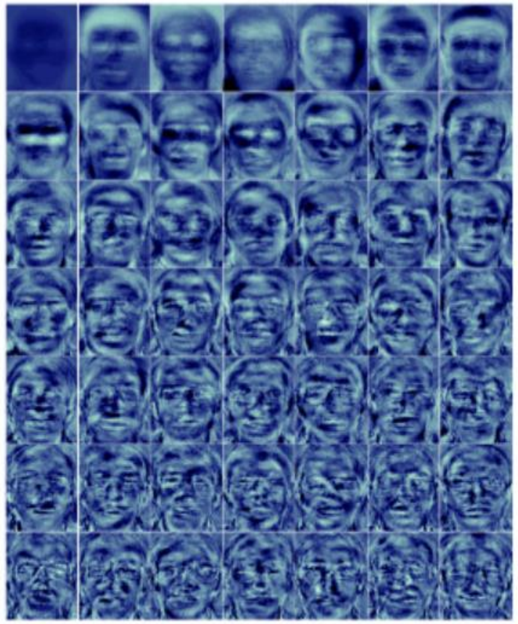
Some practical applications

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

Original Collection of Images

X

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



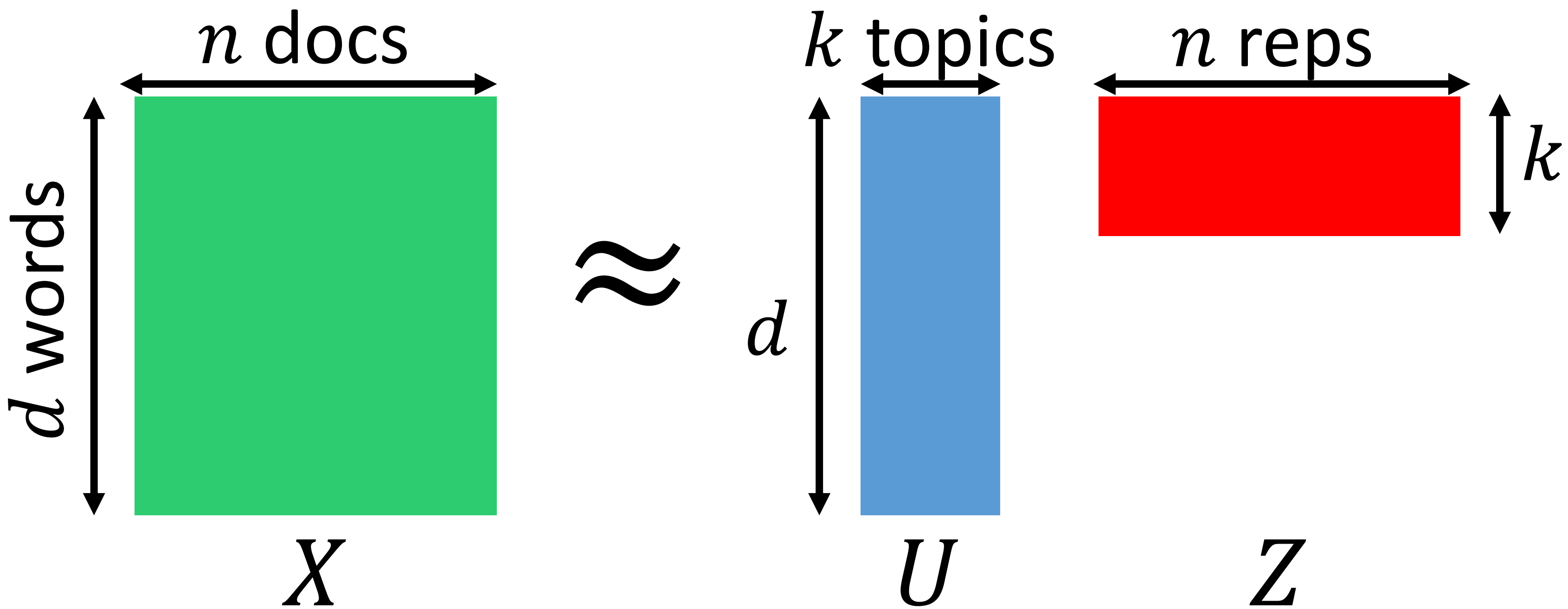
U

Each image's reconstructed version

$UZ \approx X$

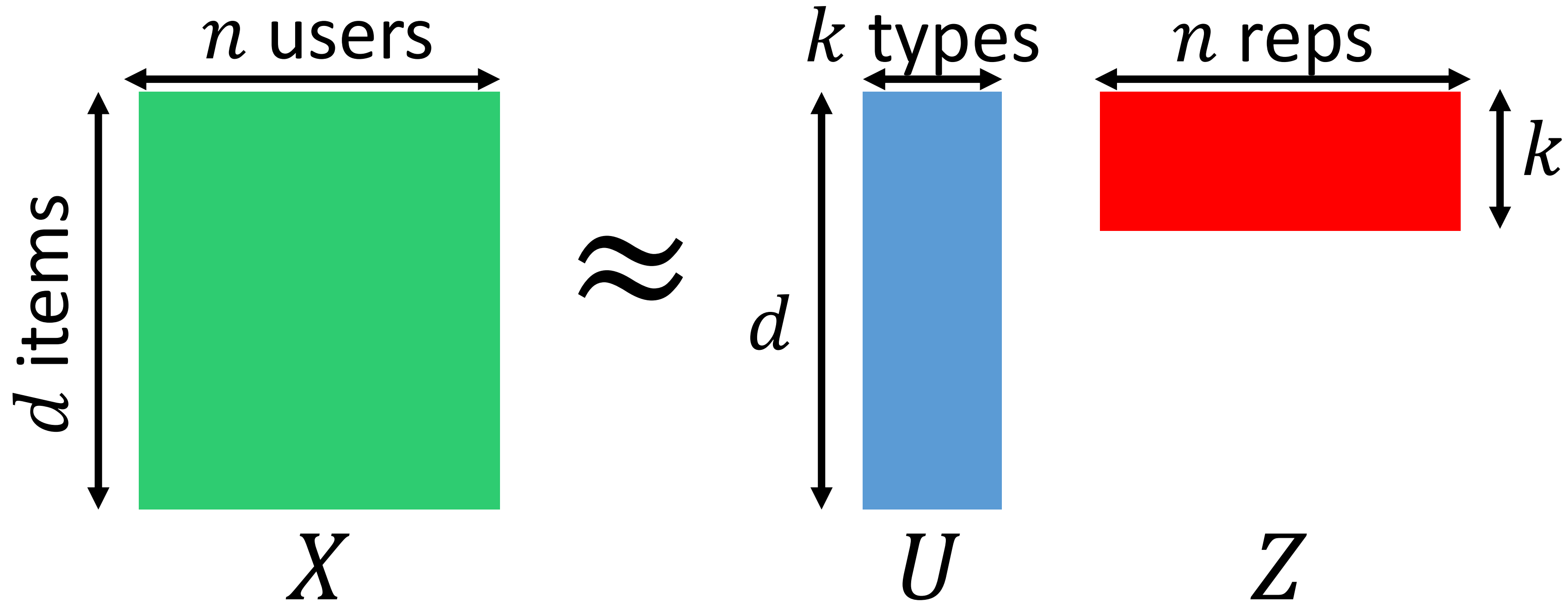
Some practical applications

- 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 ($d \approx 10^6$)

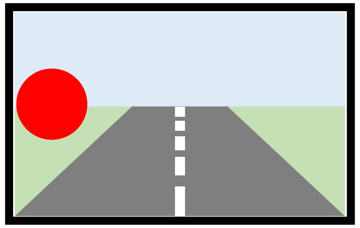


Some practical applications

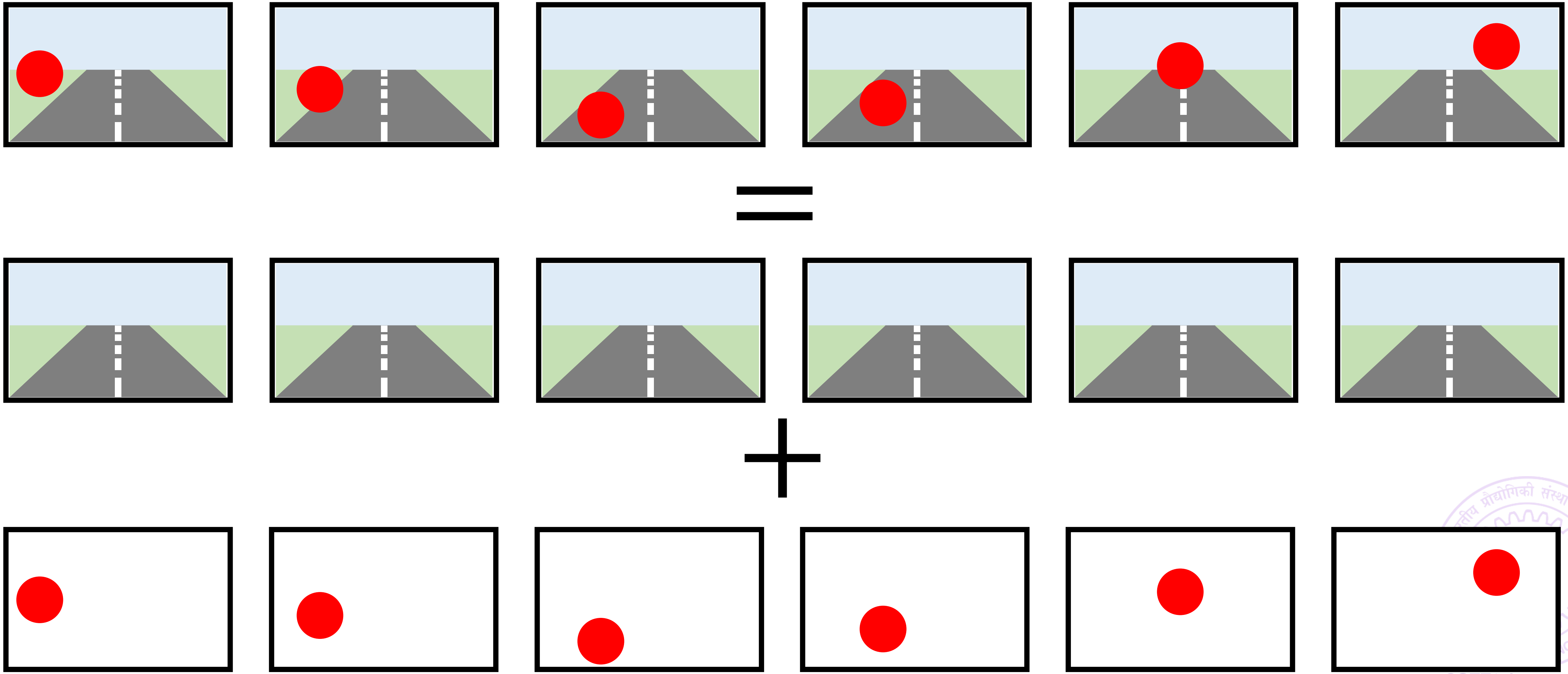
- Recommendation systems (collaborative filtering)
- Given a collection of n users who have rated d movies each
- Discover of k user-types and a representation of each user in terms of these types.



Some practical applications



- Foreground/background separation



Some practical applications

- Make every frame a vector $\mathbf{x}^i \in \mathbb{R}^d$. n frames $X = [\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^n] \in \mathbb{R}^{d \times n}$
- Background is a constant vector $\mathbf{b} \in \mathbb{R}^d$. Let $\mathbf{1} = [1,1,1, \ldots 1]^\top \in \mathbb{R}^n$
- Foreground treated as noise $\mathbf{f}^i \in \mathbb{R}^d$. Let $F = [\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^n] \in \mathbb{R}^{d \times n}$
- 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