Data Modelling Methods-IV

CS771: Introduction to Machine Learning

Purushottam Kar

Mid Semester Examination

- September 21st, 2017 (Thursday) 1300-1500 hrs
- Venue L18, 19, L20 (all OROS)
- Syllabus: till whatever we covered on Wednesday + maybe one question from today's lecture
- Open notes (handwritten only)
- No printed/photocopied material
- No laptops, i-pads, mobile phones (switched off)
- Please bring a notepad with you for rough work
- Please bring a pencil/eraser with you – we will not provide these
- Answers will have to be written on the question paper itself

Outline of today's discussion

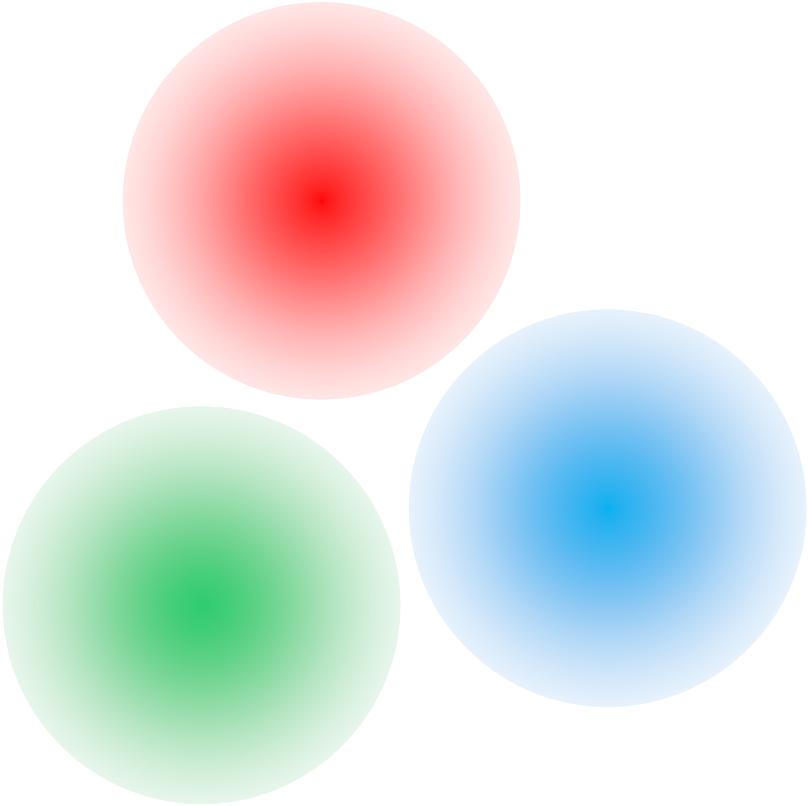
DIMENSIONALITY REDUCTION TECHNIQUES

- Study an appropriate generative model for low-dim. Data
- Study the MLE for zero-noise condition (PCA)
- Study the MLE for noisy conditions (PPCA)
- See an efficient "power method" to solve the MLE

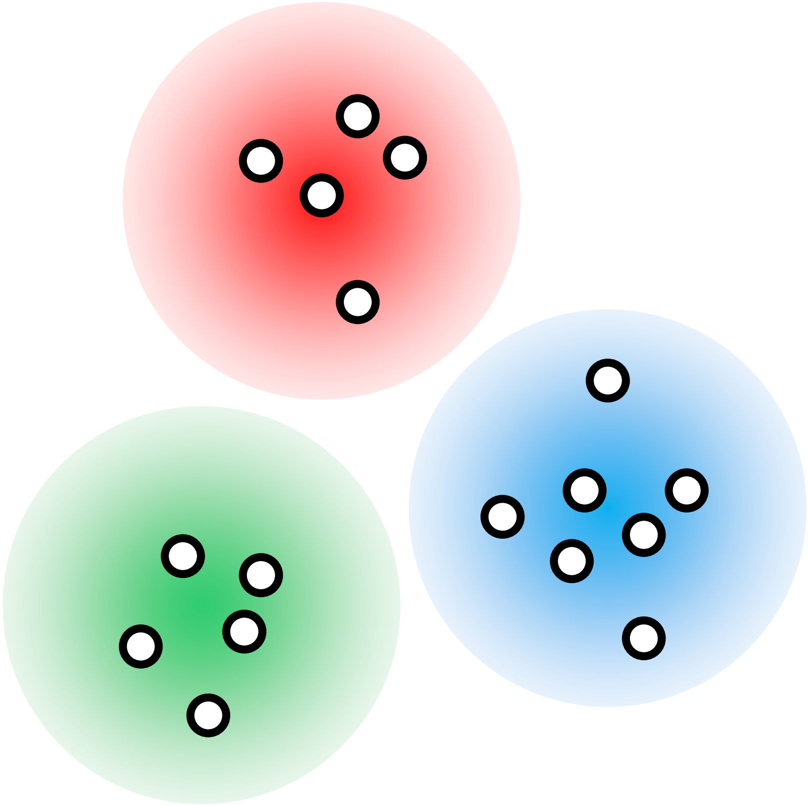
AFTER MID-SEMS

- See a "soft"-assignment approach to solving the MLE
- See how the "hard" assignment rule can be used to solve PCA
- The One EM to Rule them All, Kernels, Deep learning, RecSys

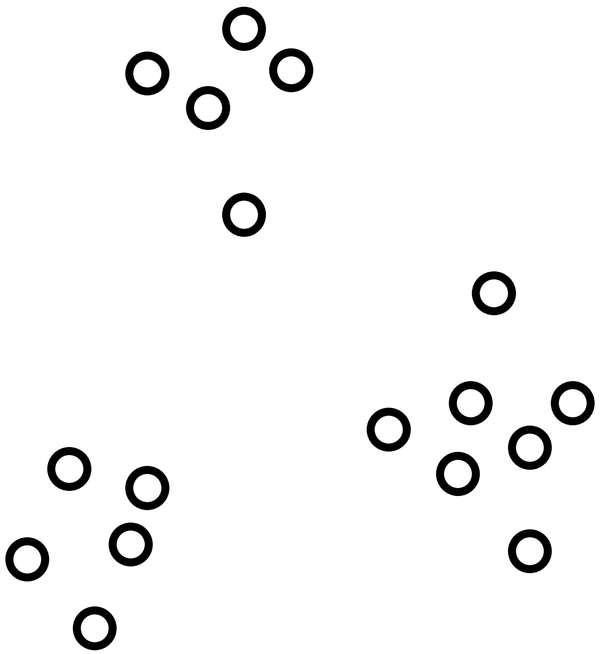
Recap



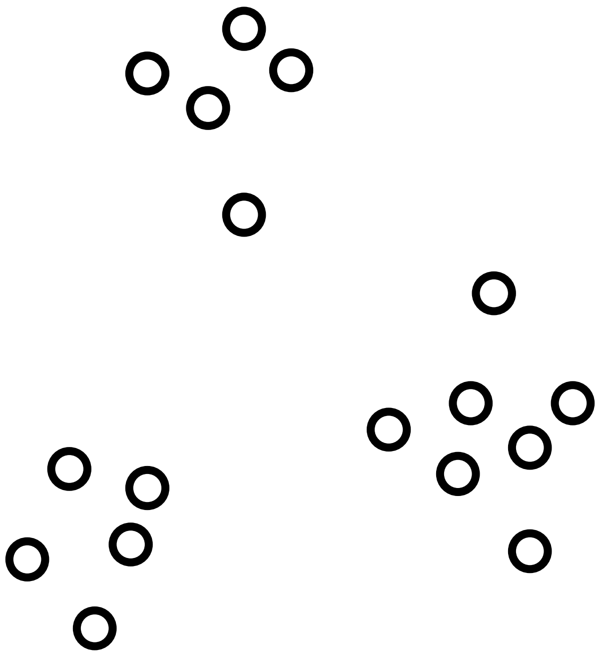
Recap



Recap

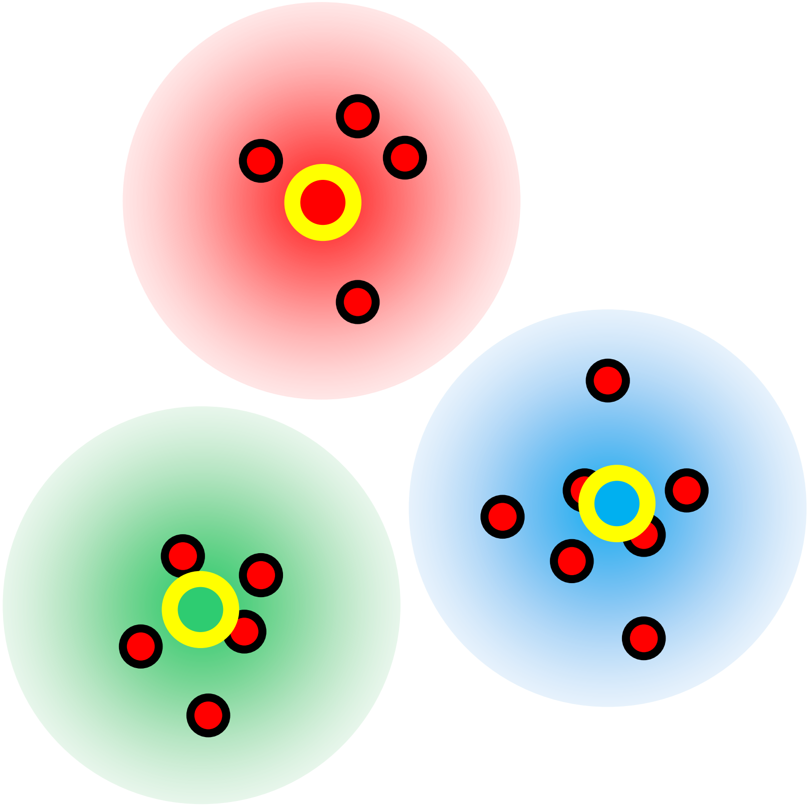


Recap



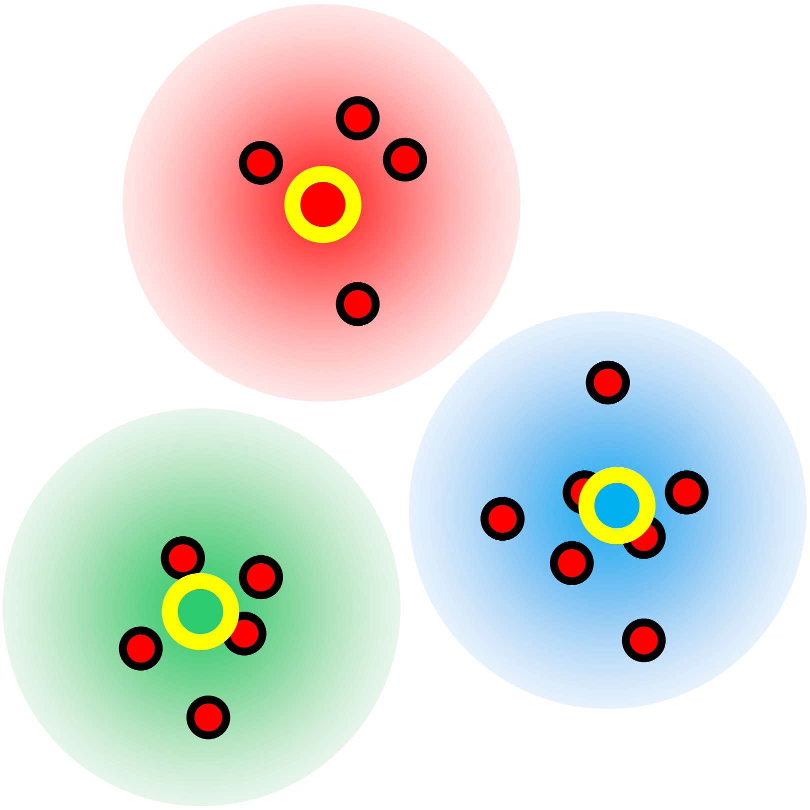
Hard assignment – k-means
Soft assignment – soft k-means

Recap



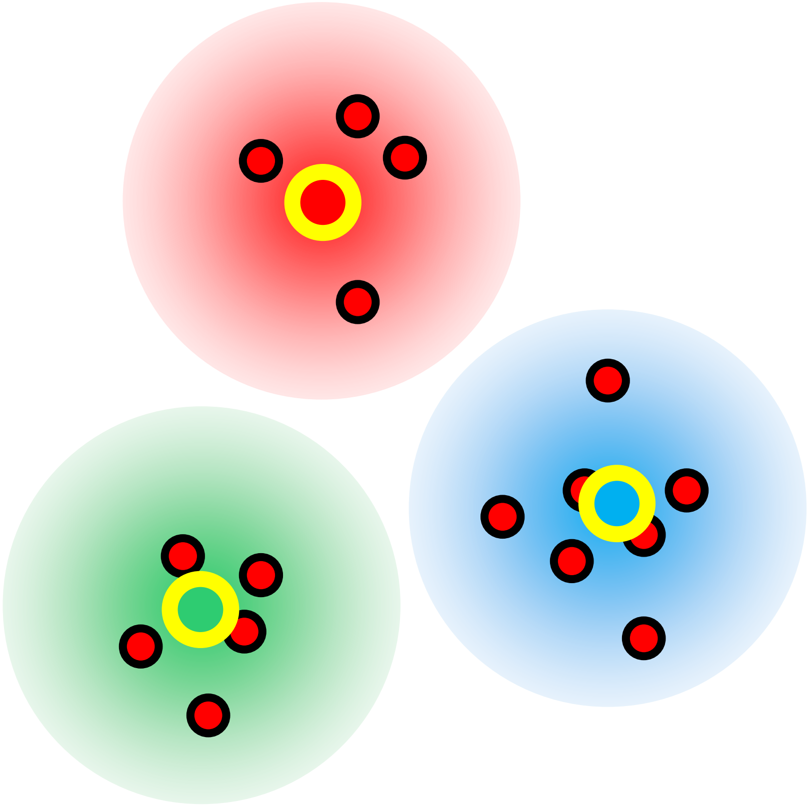
Hard assignment – k-means
Soft assignment – soft k-means

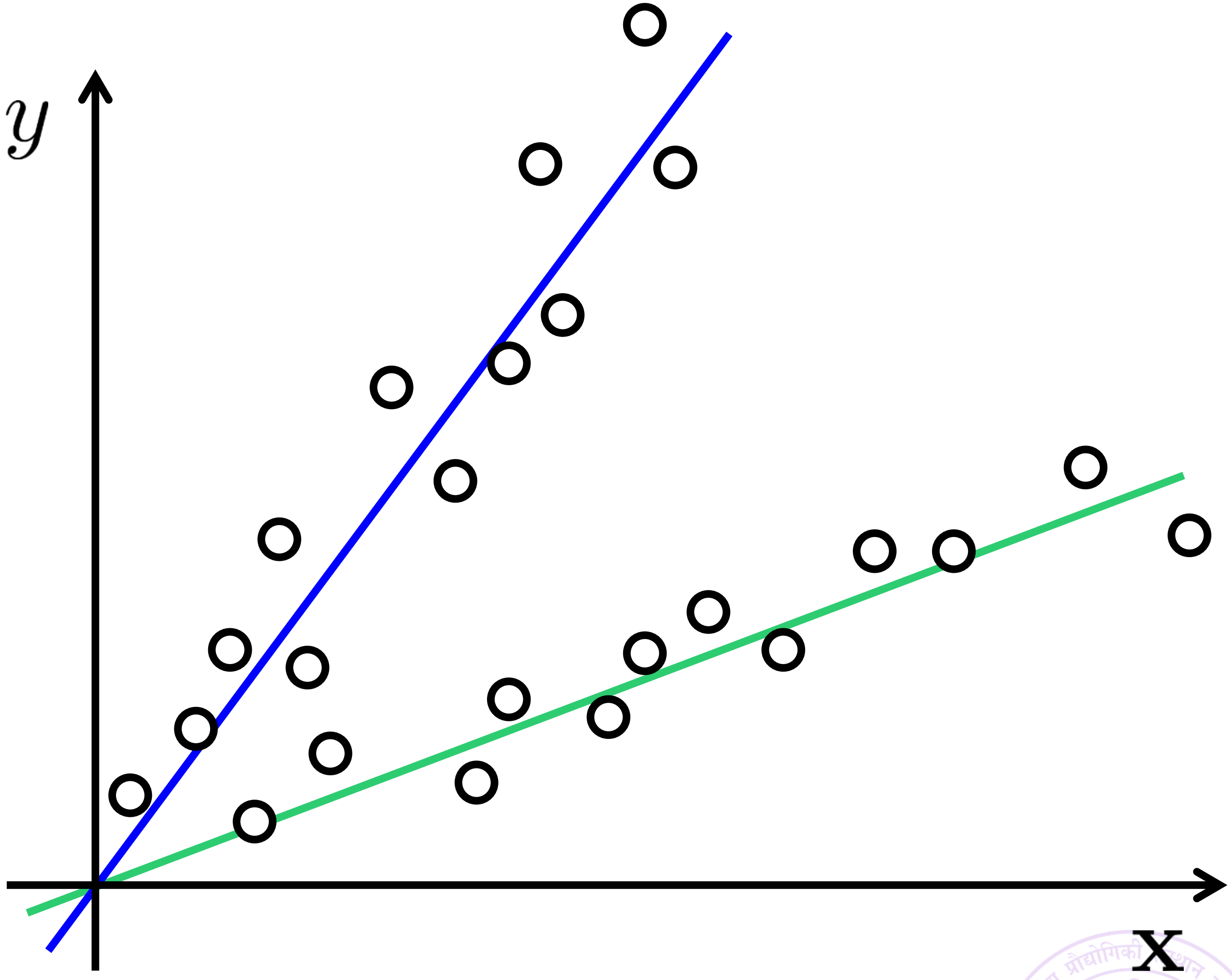
Recap



Hard assignment – k-means
Soft assignment – soft k-means

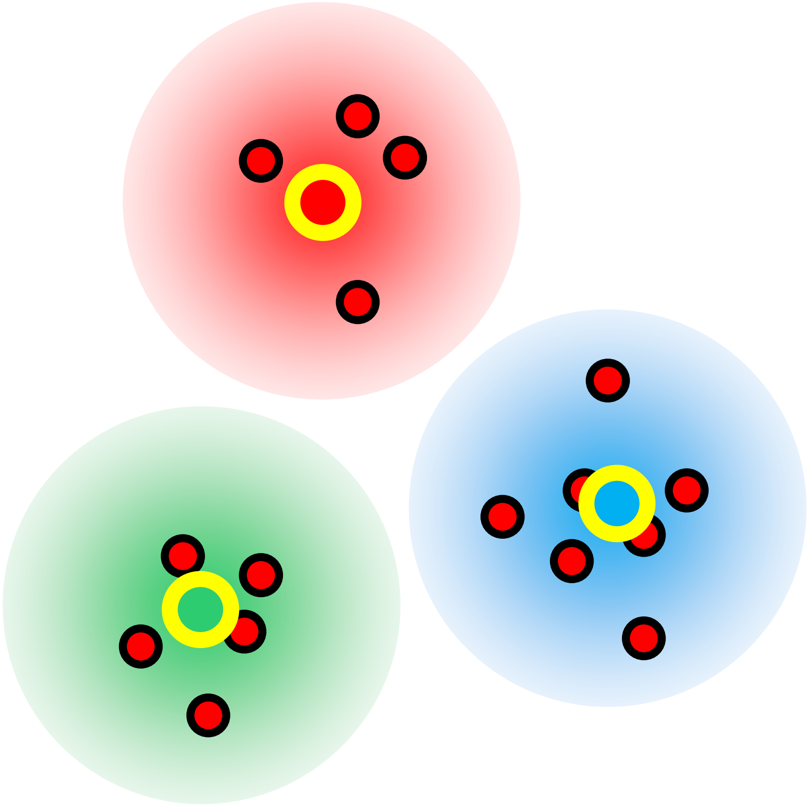
Recap

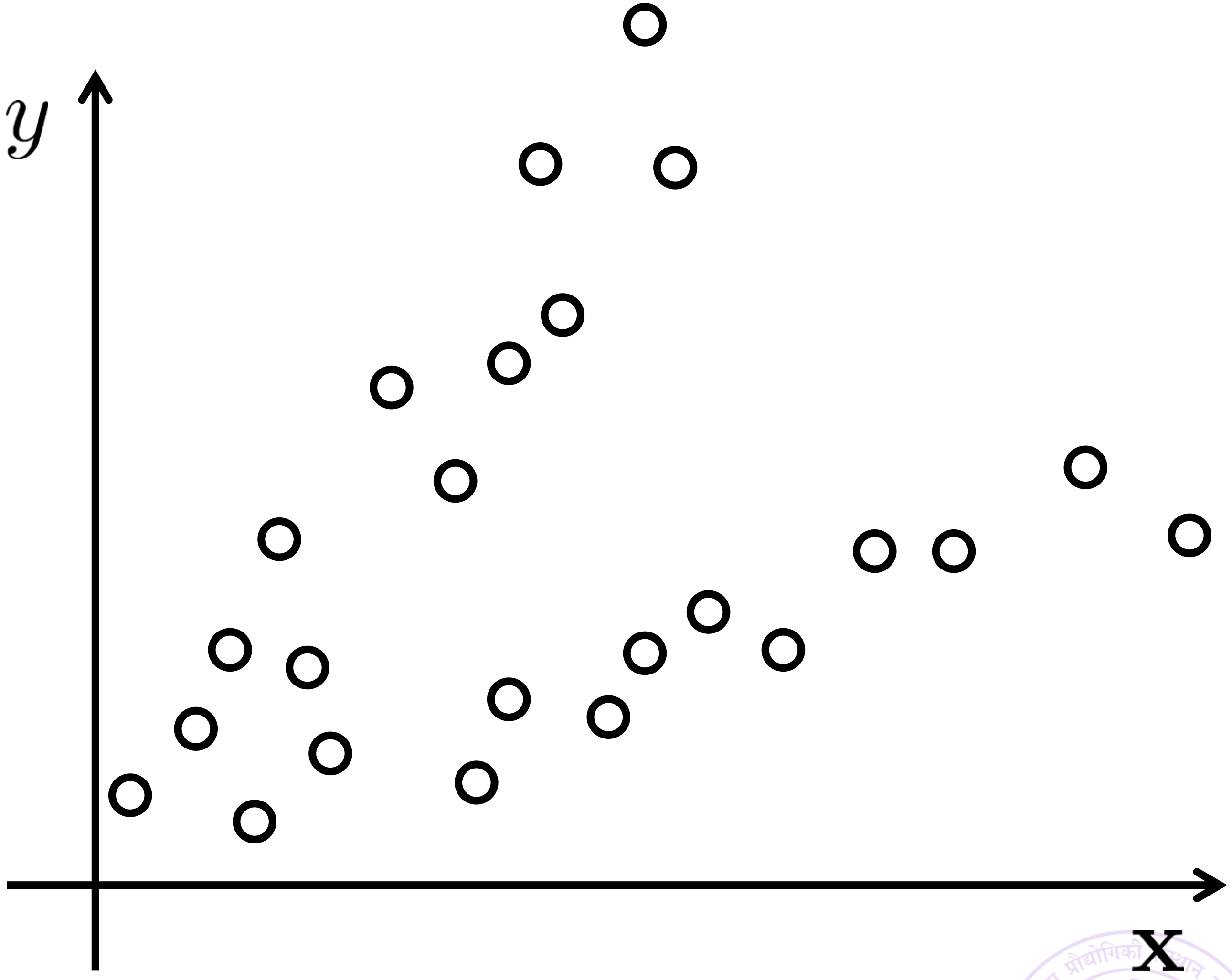




Hard assignment – k-means
Soft assignment – soft k-means

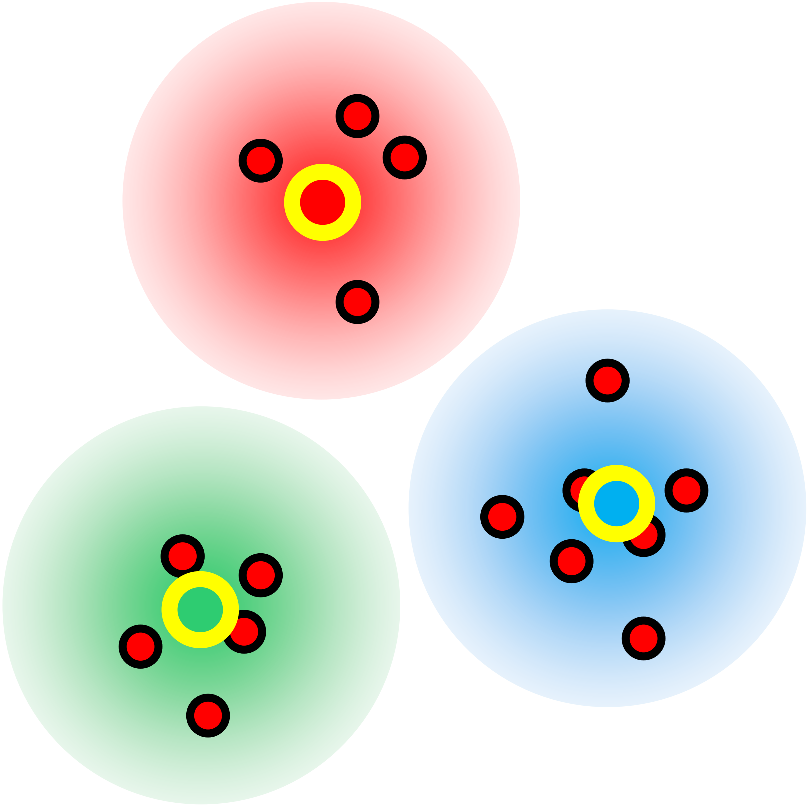
Recap

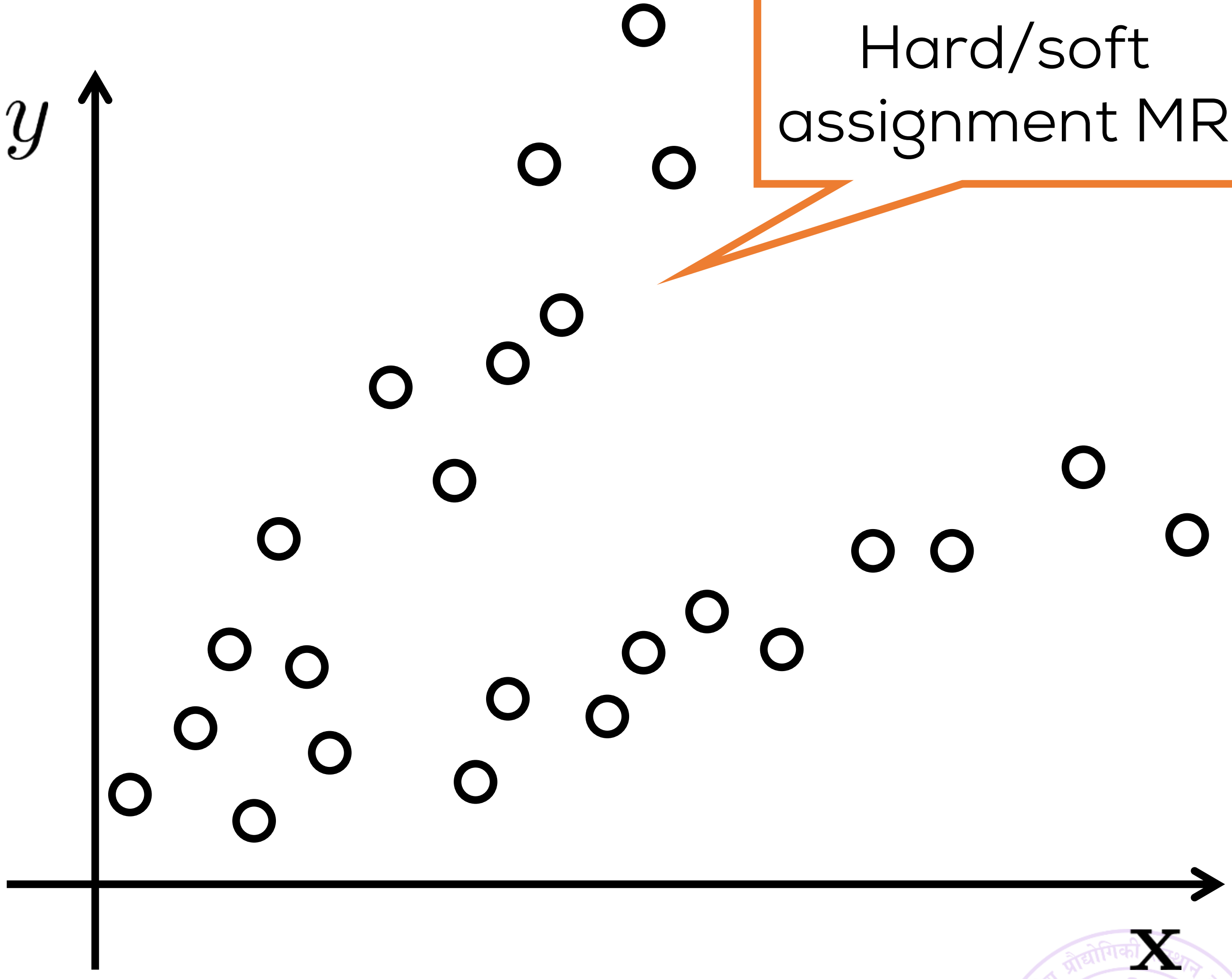




Hard assignment – k-means
Soft assignment – soft k-means

Recap

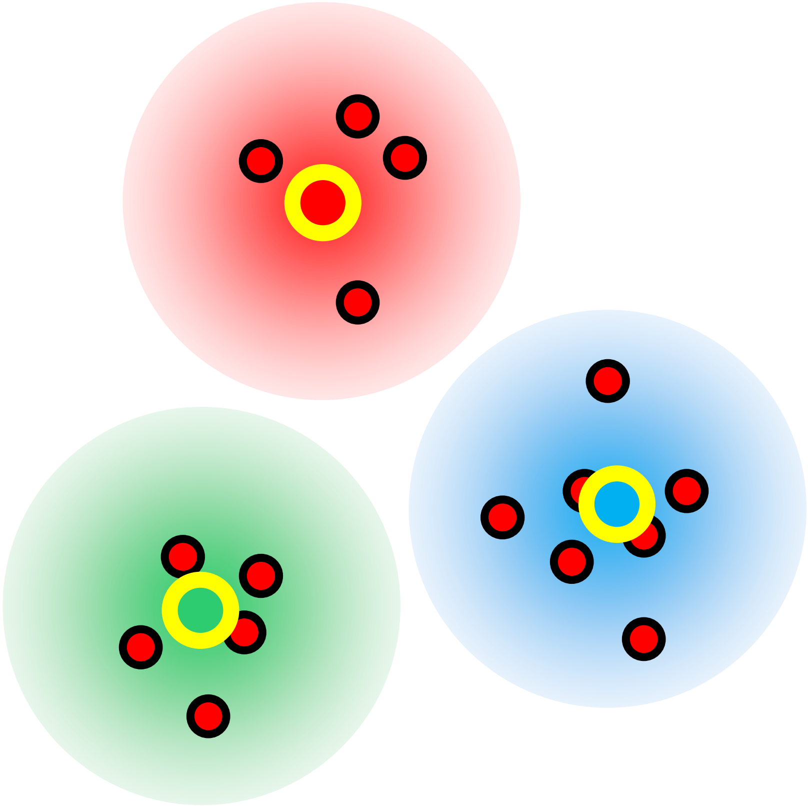


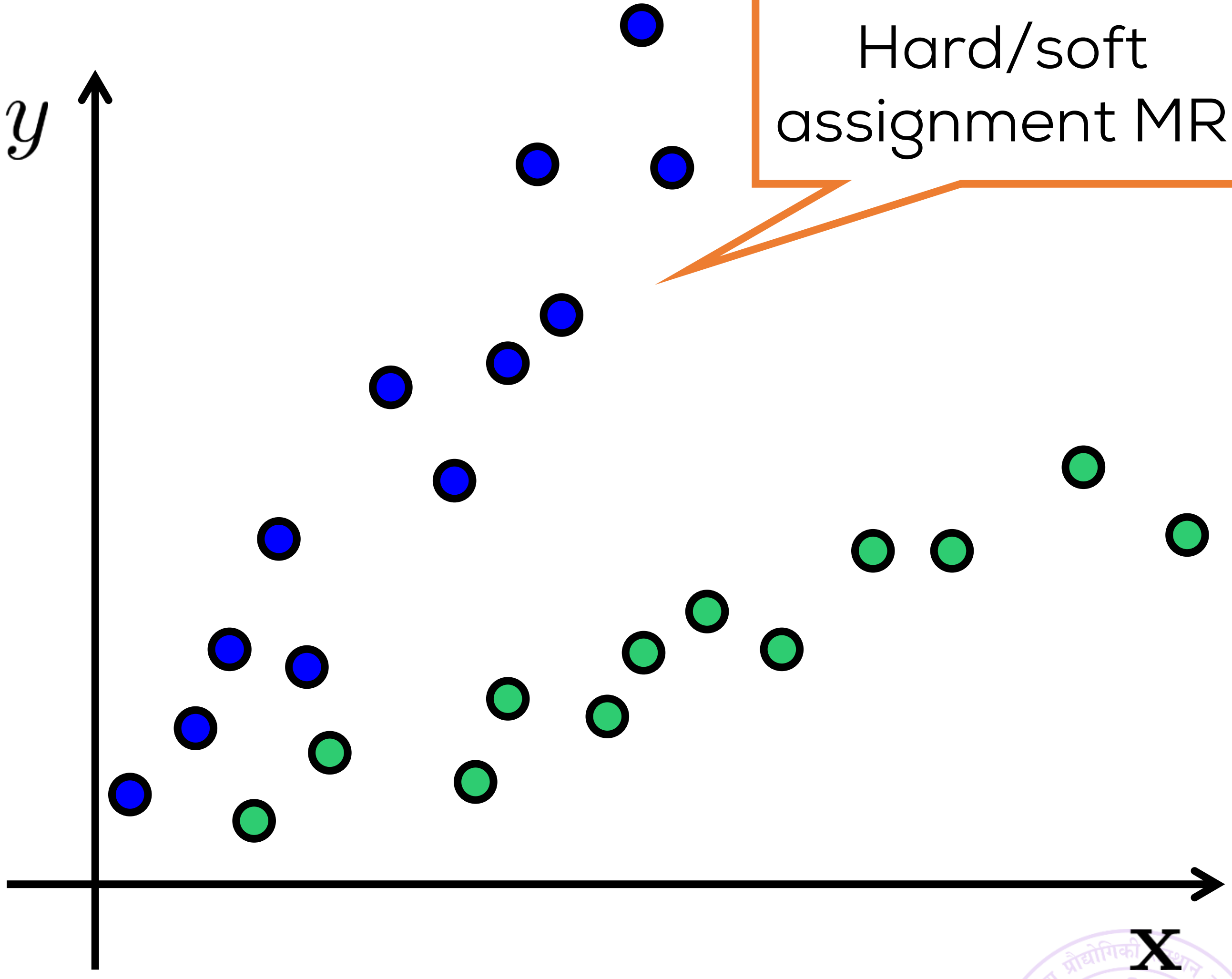


Hard/soft assignment MR

Hard assignment – k-means
Soft assignment – soft k-means

Recap

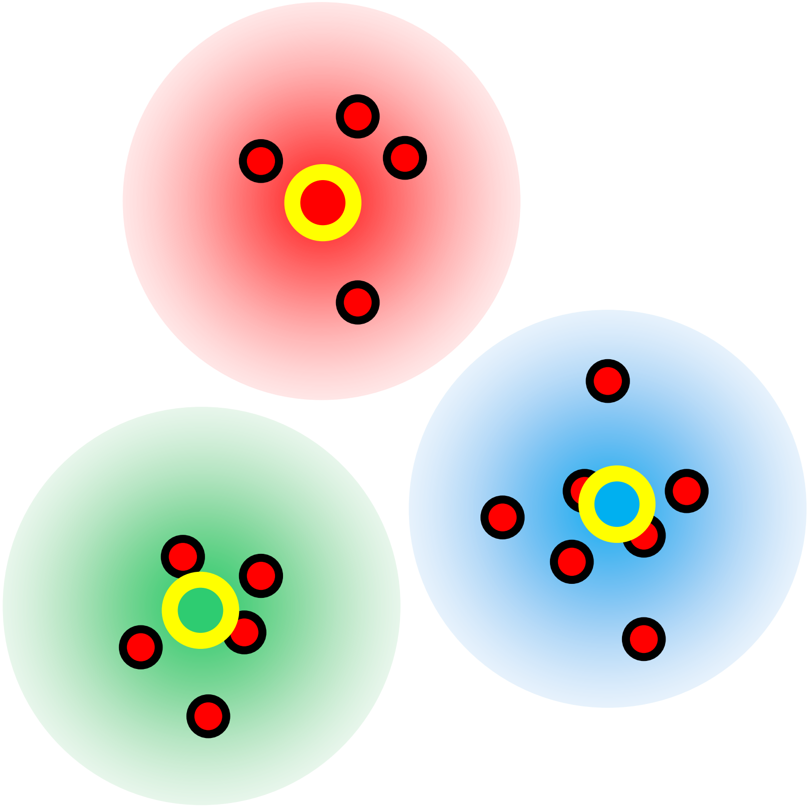


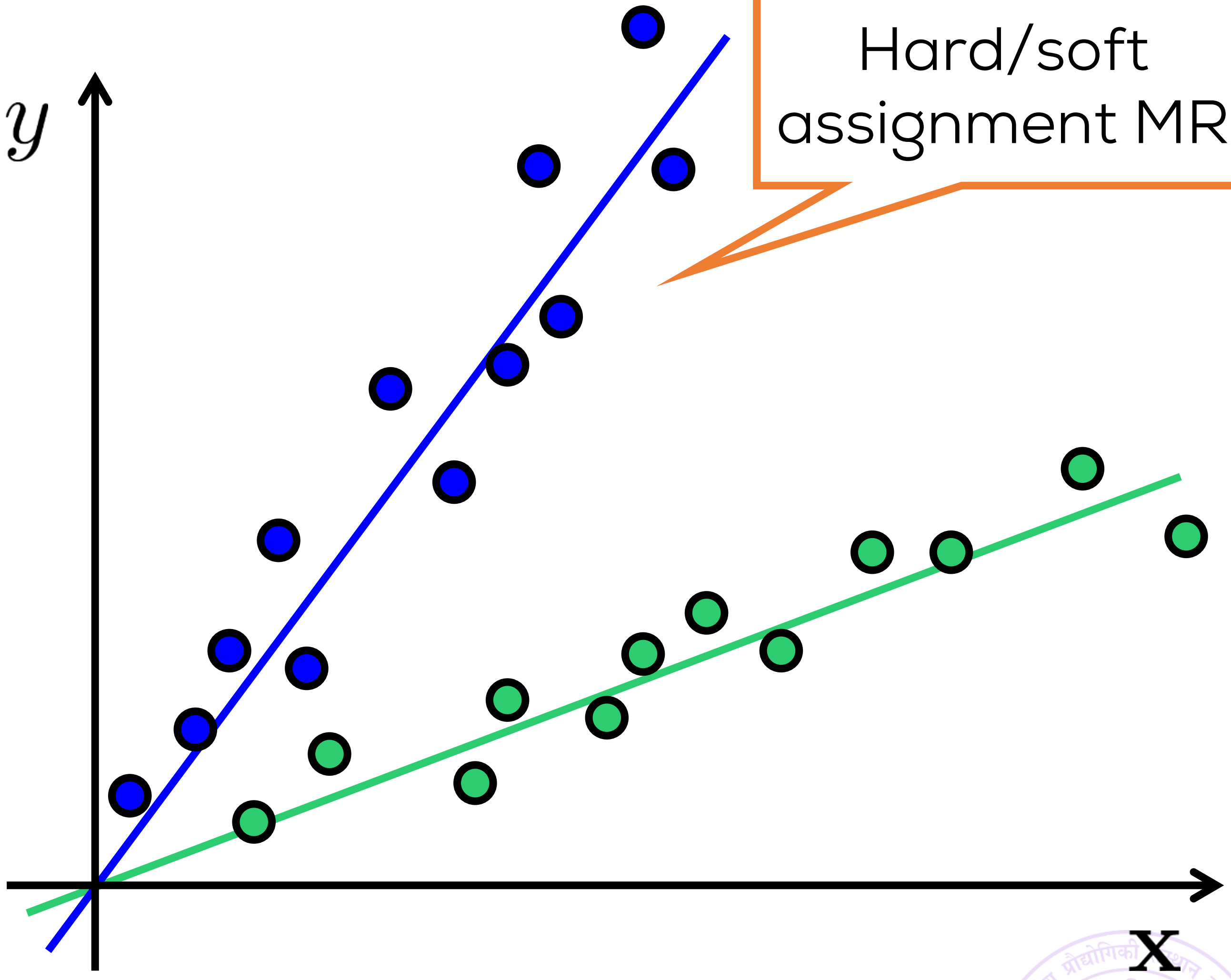


Hard/soft assignment MR

Hard assignment – k-means
Soft assignment – soft k-means

Recap

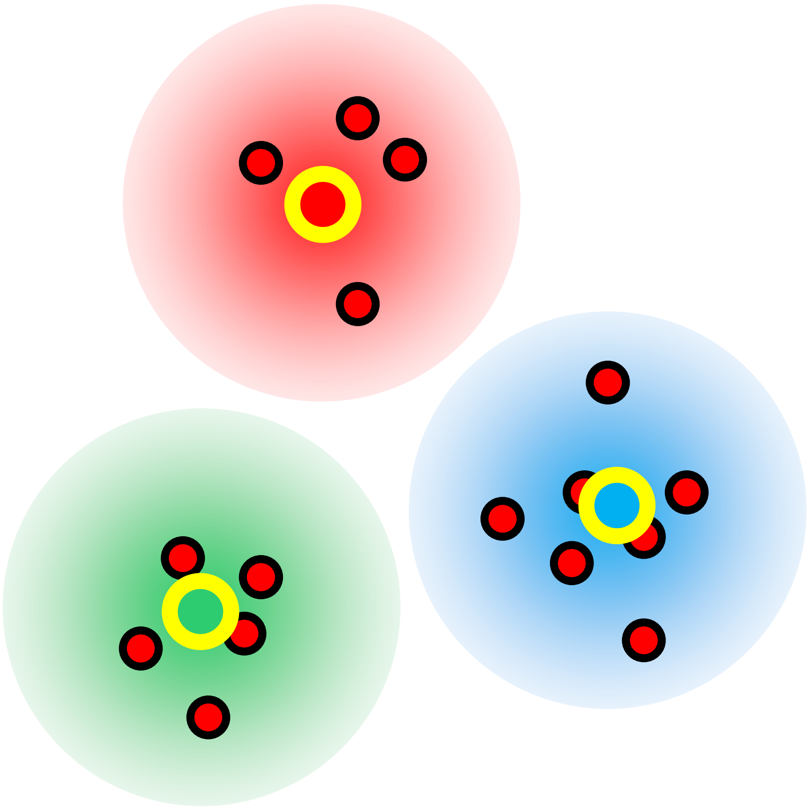


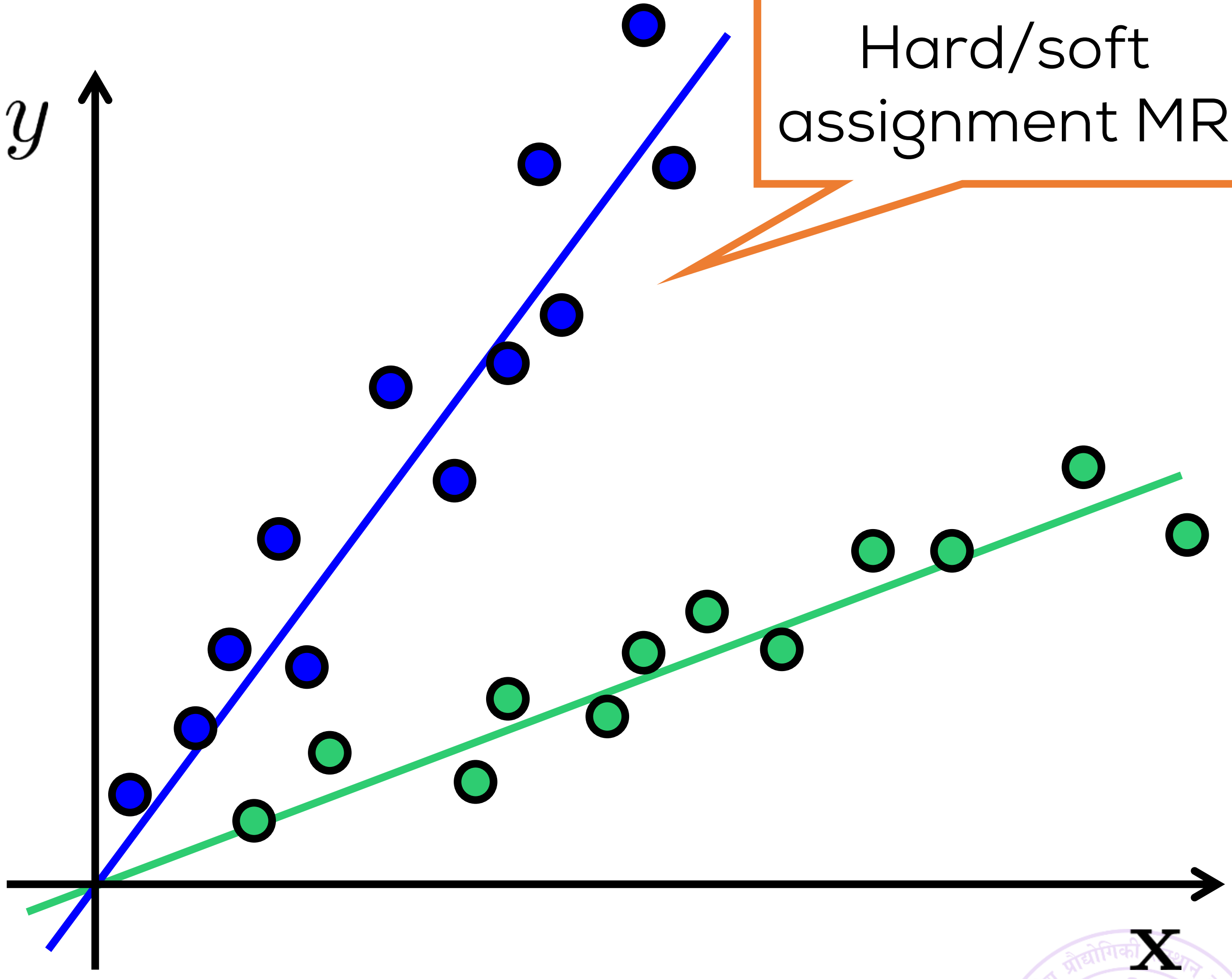


Hard/soft assignment MR

Hard assignment – k-means
Soft assignment – soft k-means

Recap





Hard/soft assignment MR

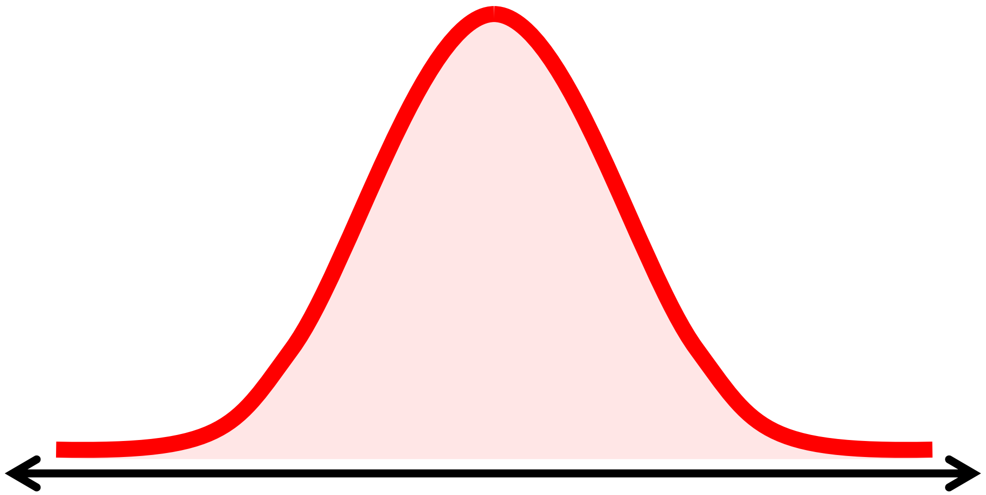
Hard assignment – k-means
Soft assignment – soft k-means

Discovering hidden structure in data

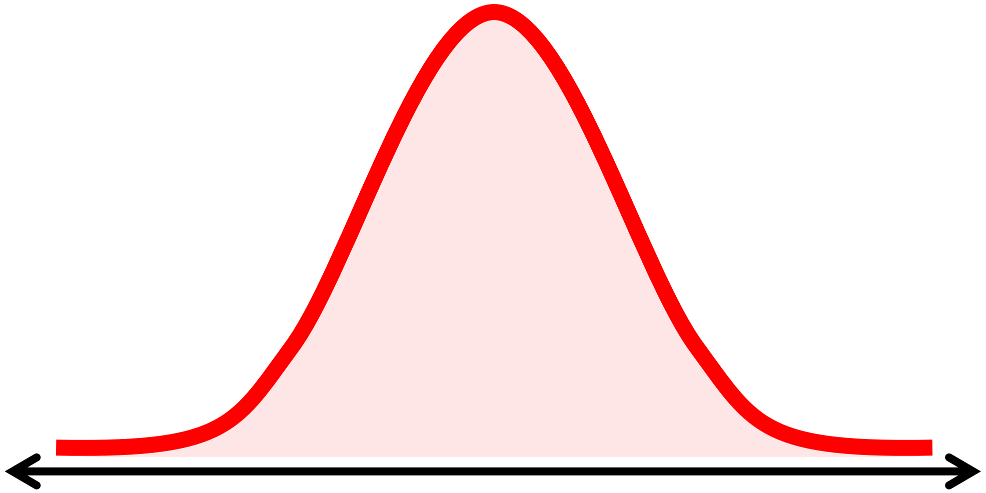
Low-dimensional Structure in Data

Low-dimensional Structure in Data

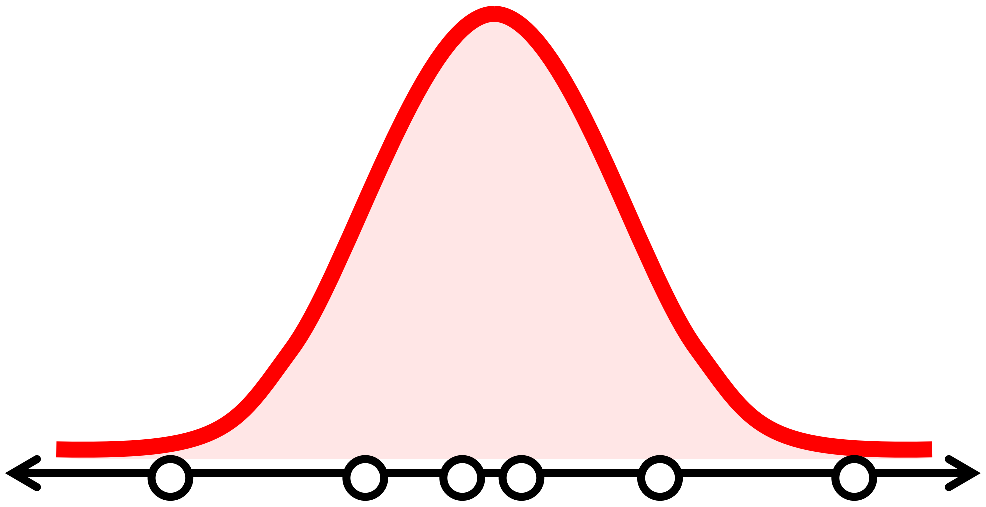
Low-dimensional Structure in Data



Low-dimensional Structure in Data

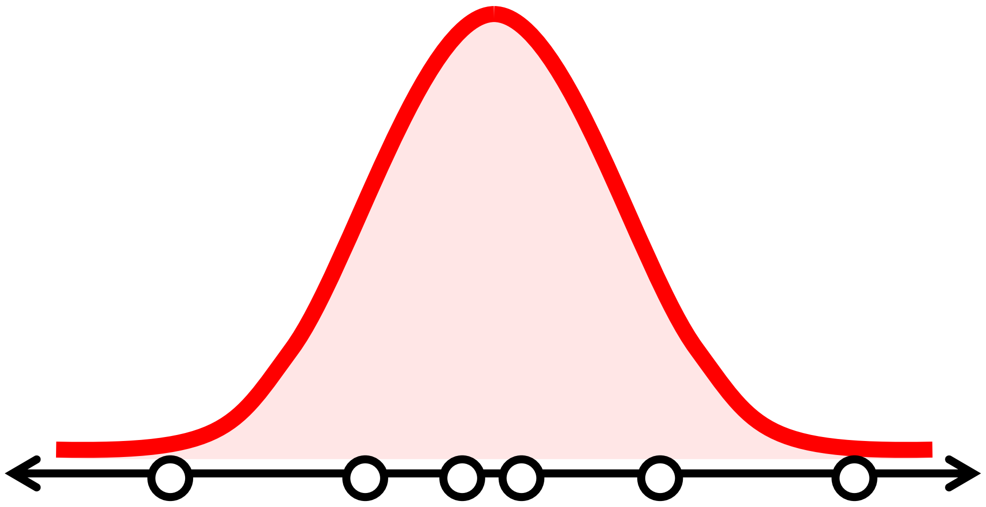


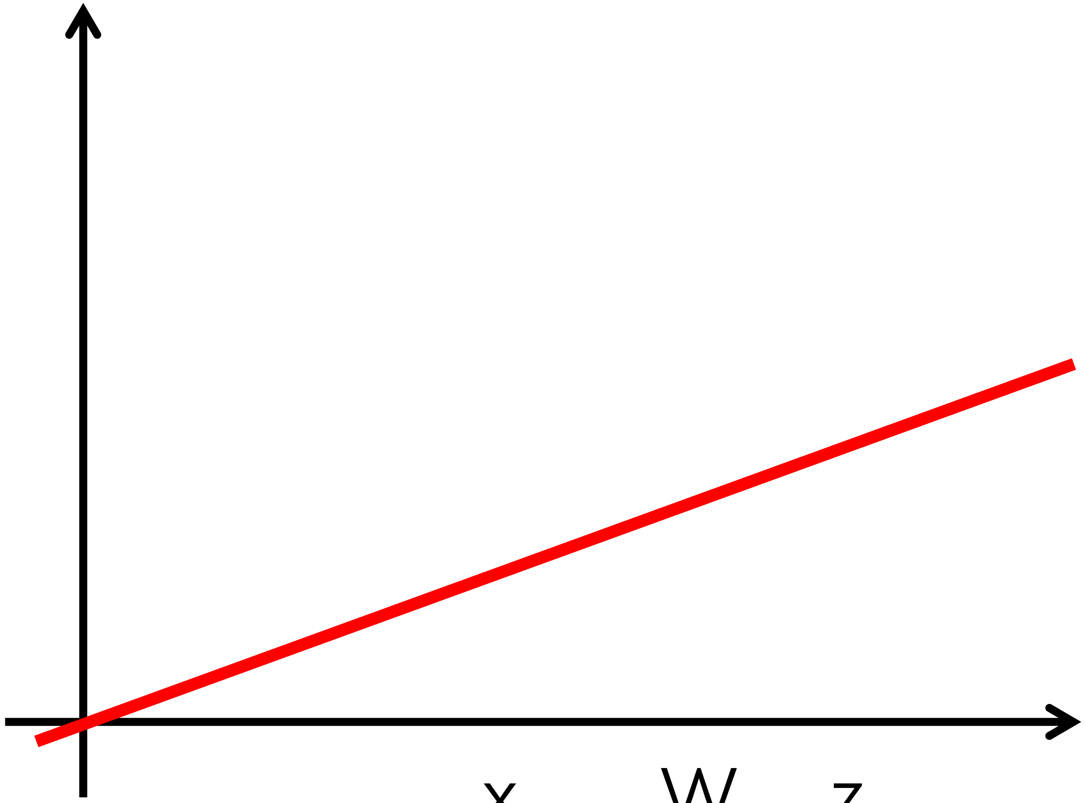
Low-dimensional Structure in Data

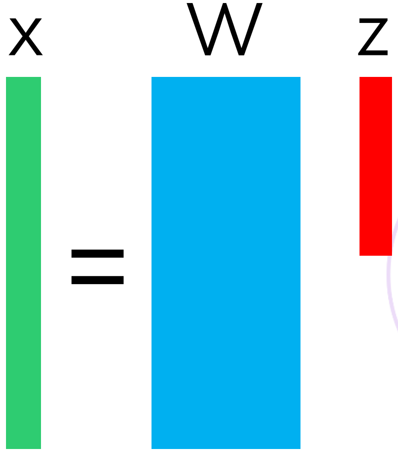


$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

Low-dimensional Structure in Data



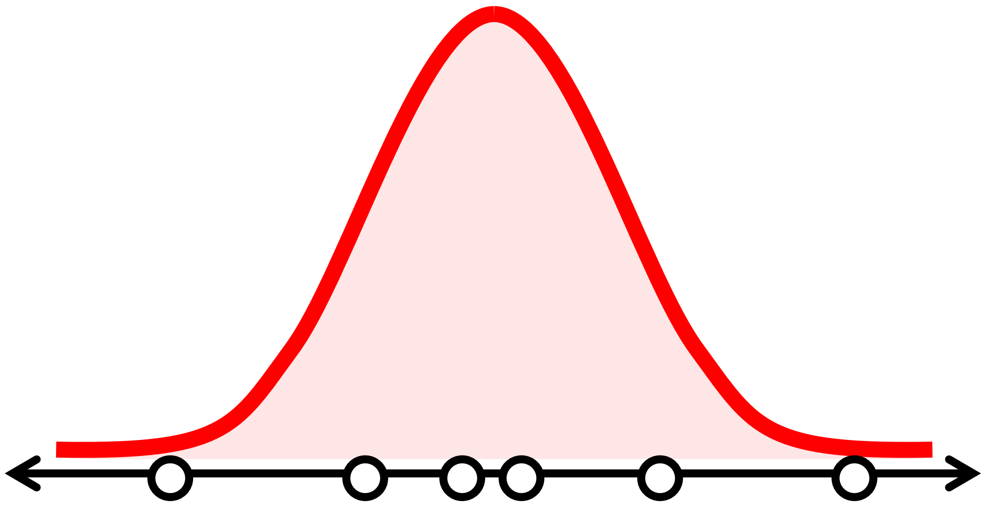


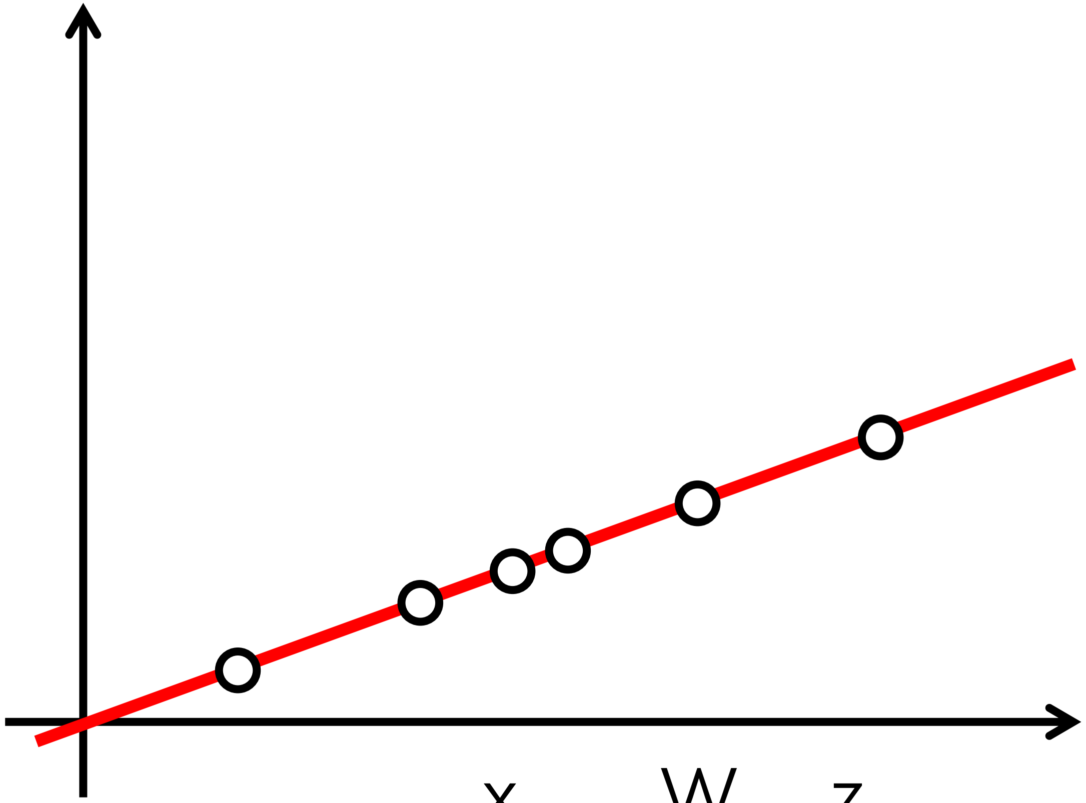


$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

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

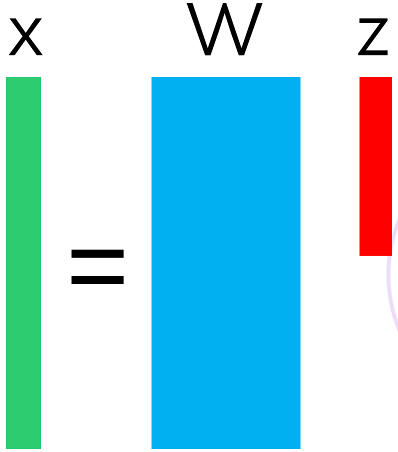
Low-dimensional Structure in Data



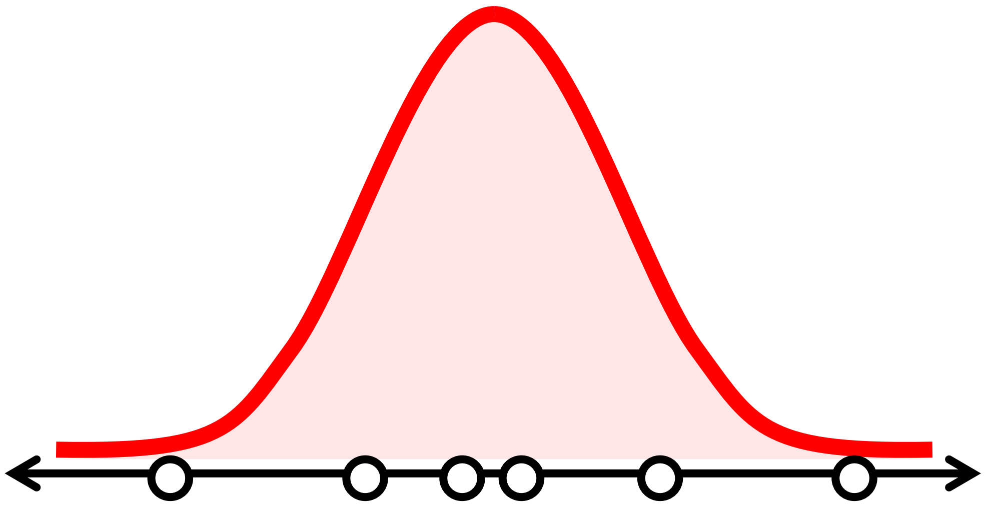


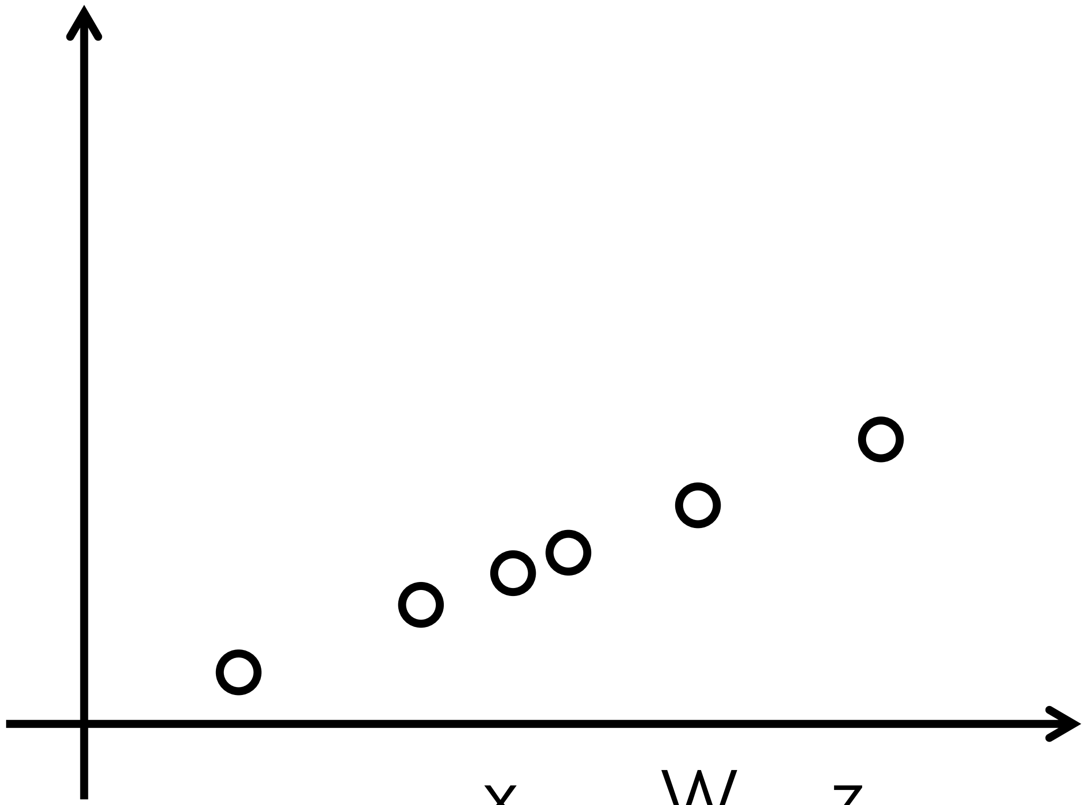
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

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



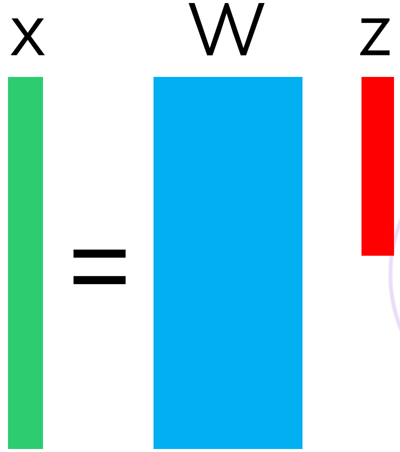
Low-dimensional Structure in Data



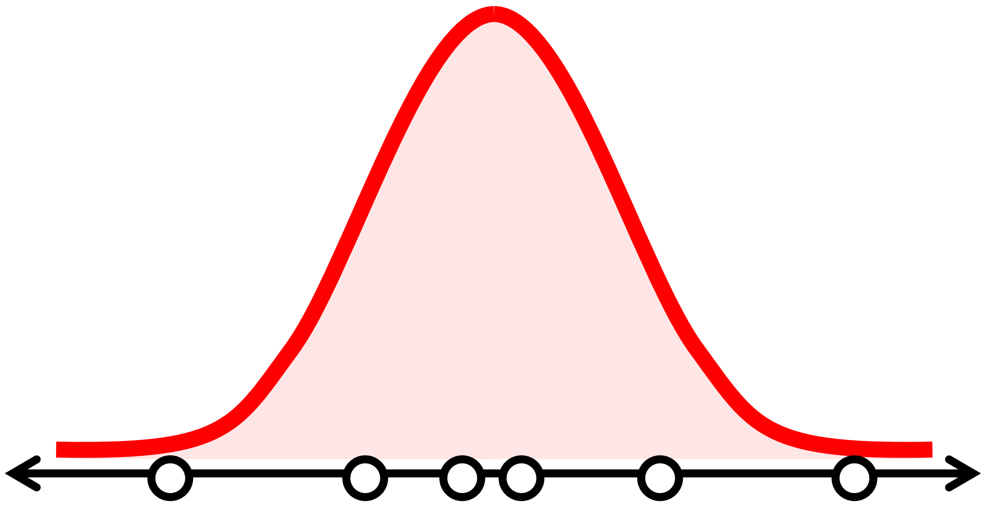


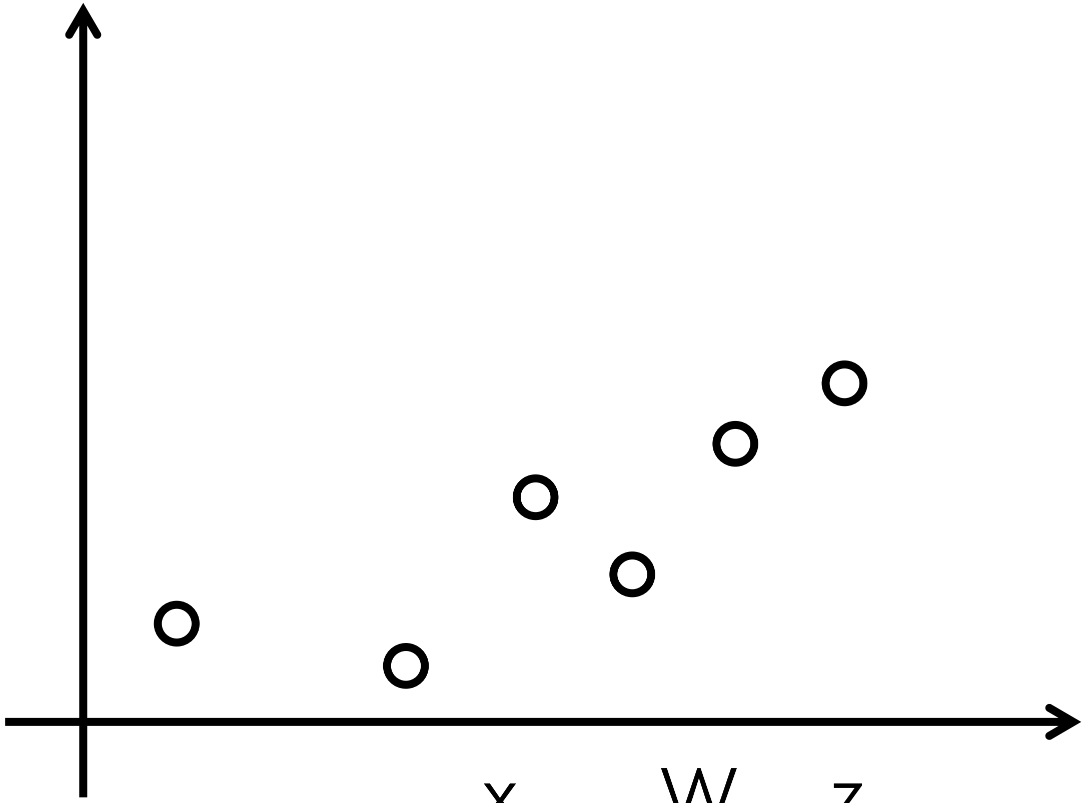
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k) \qquad \mathbf{x}^i = W\mathbf{z}^i$$

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



Low-dimensional Structure in Data

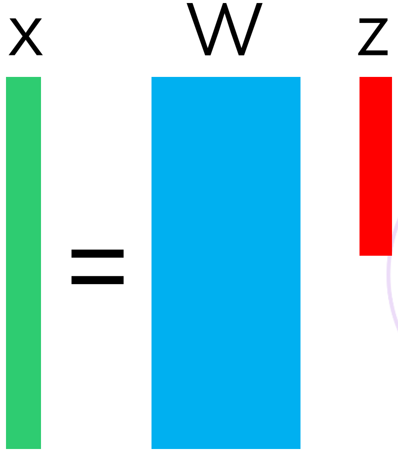




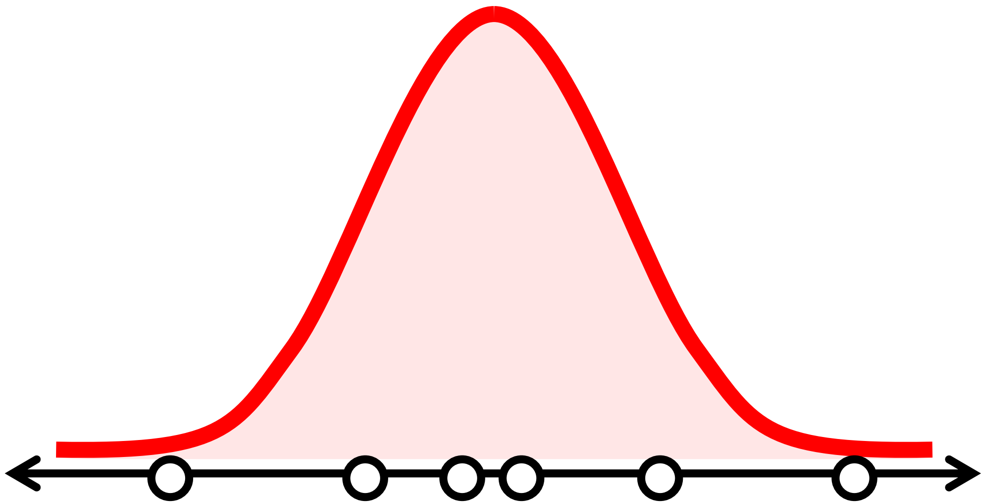
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

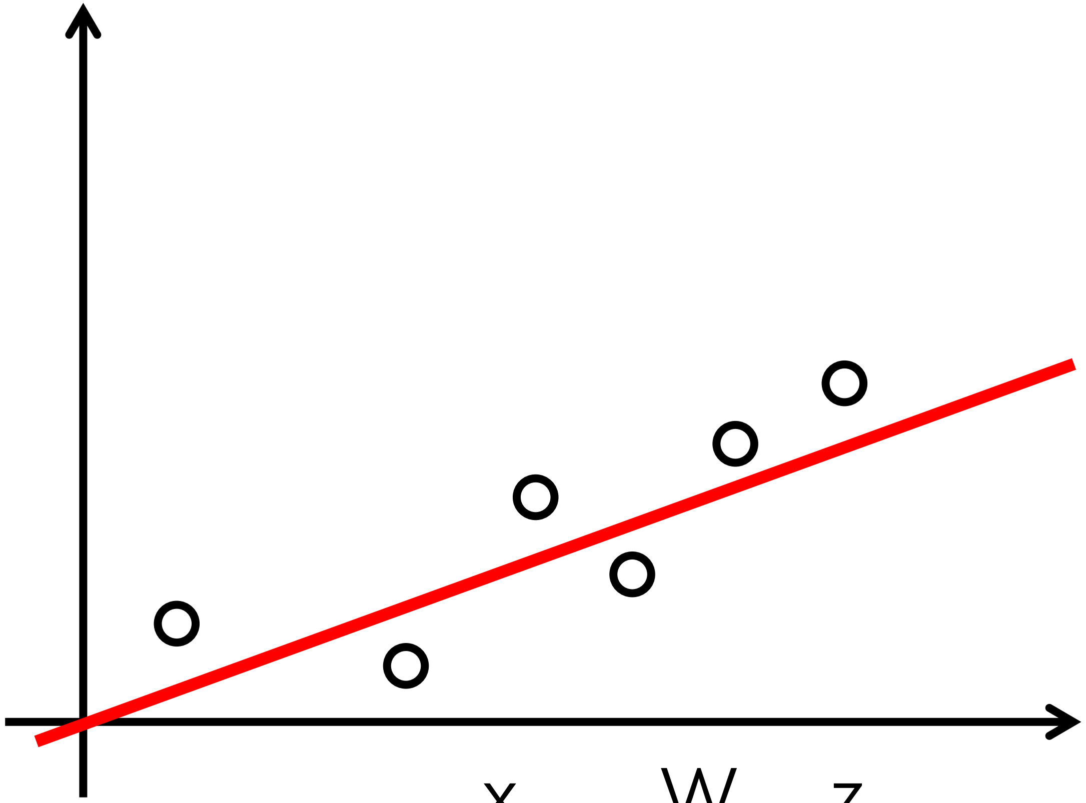
$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i,\ \ \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

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



Low-dimensional Structure in Data

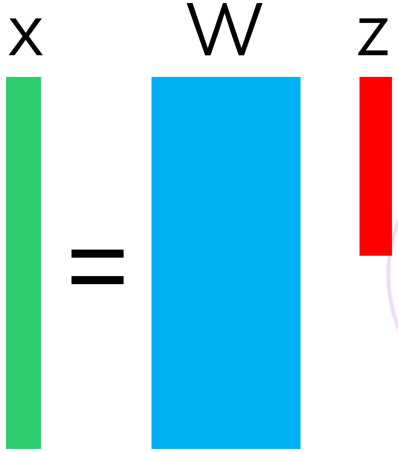




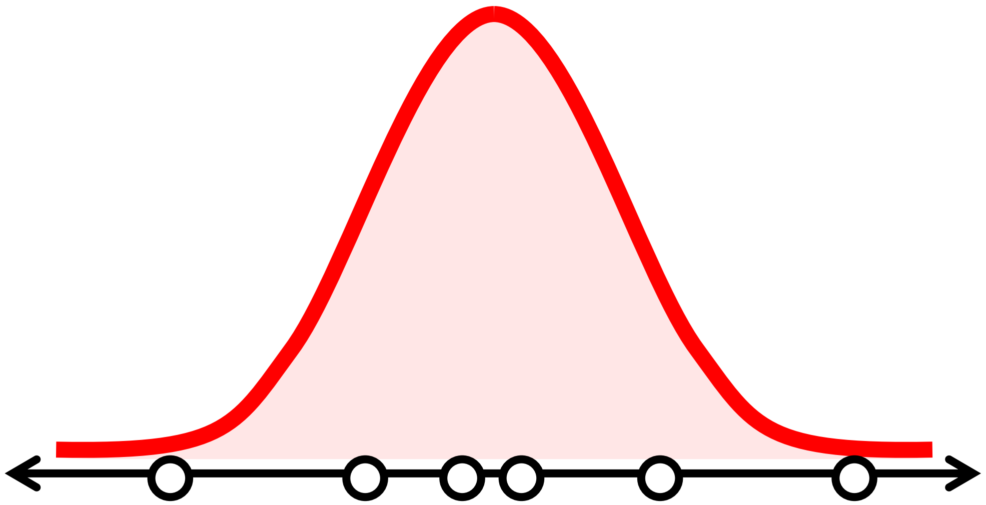
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

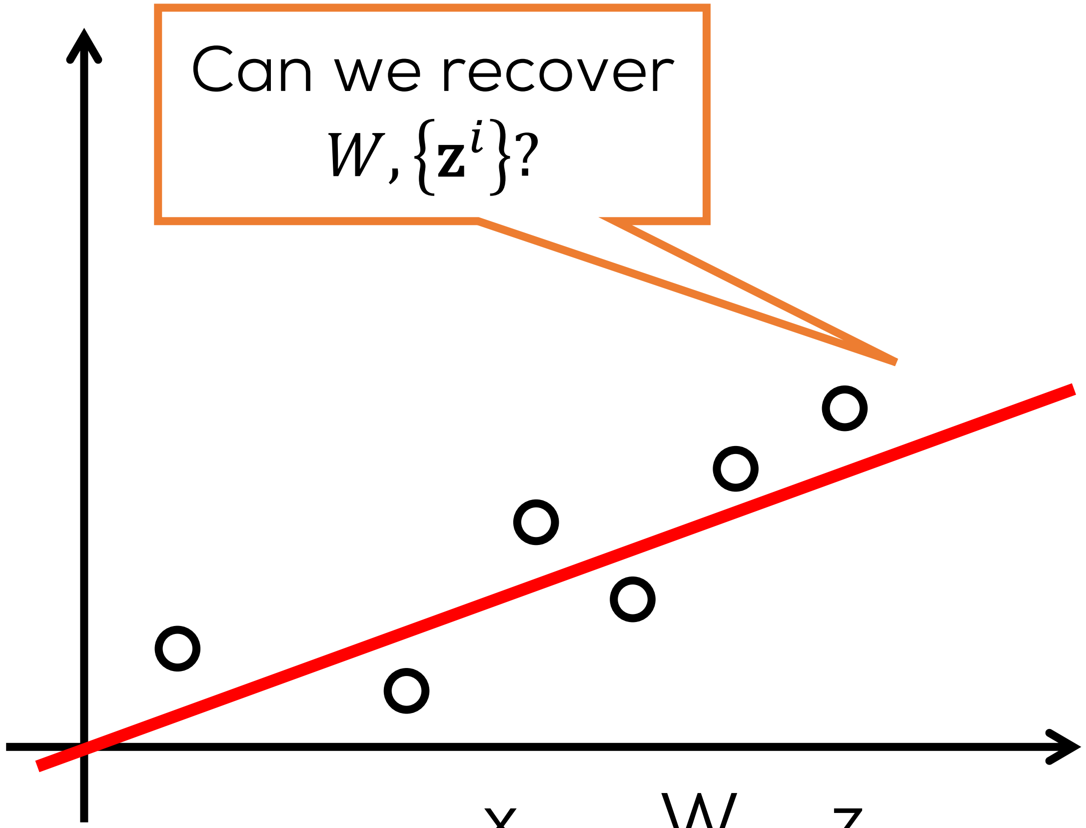
$$\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i,\ \ \boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d)$$

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



Low-dimensional Structure in Data

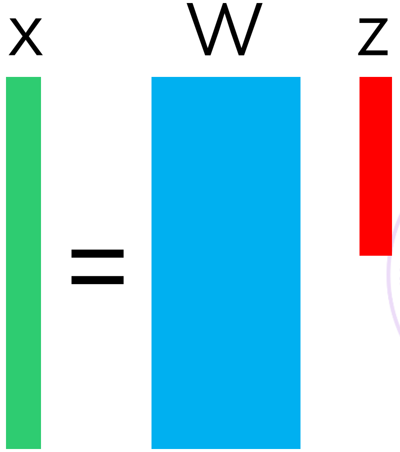




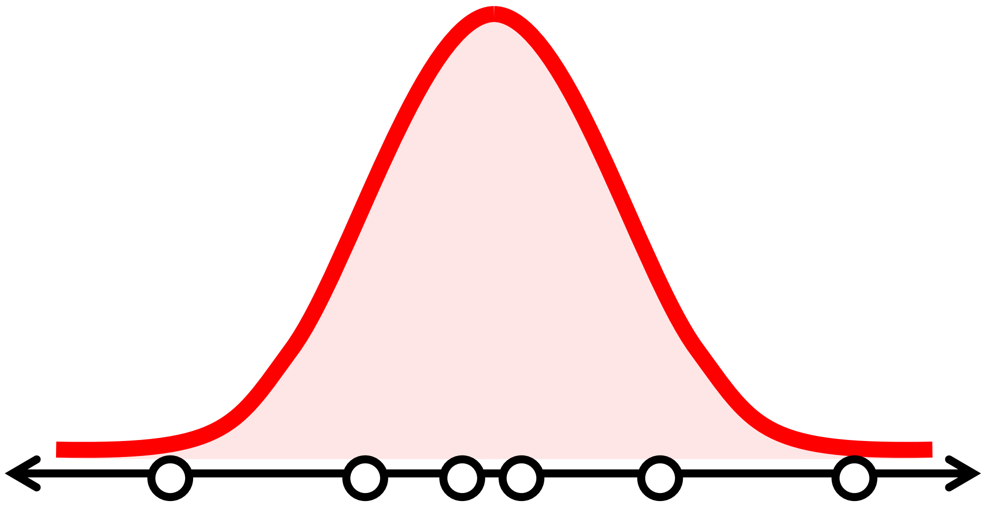
$$\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$$

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

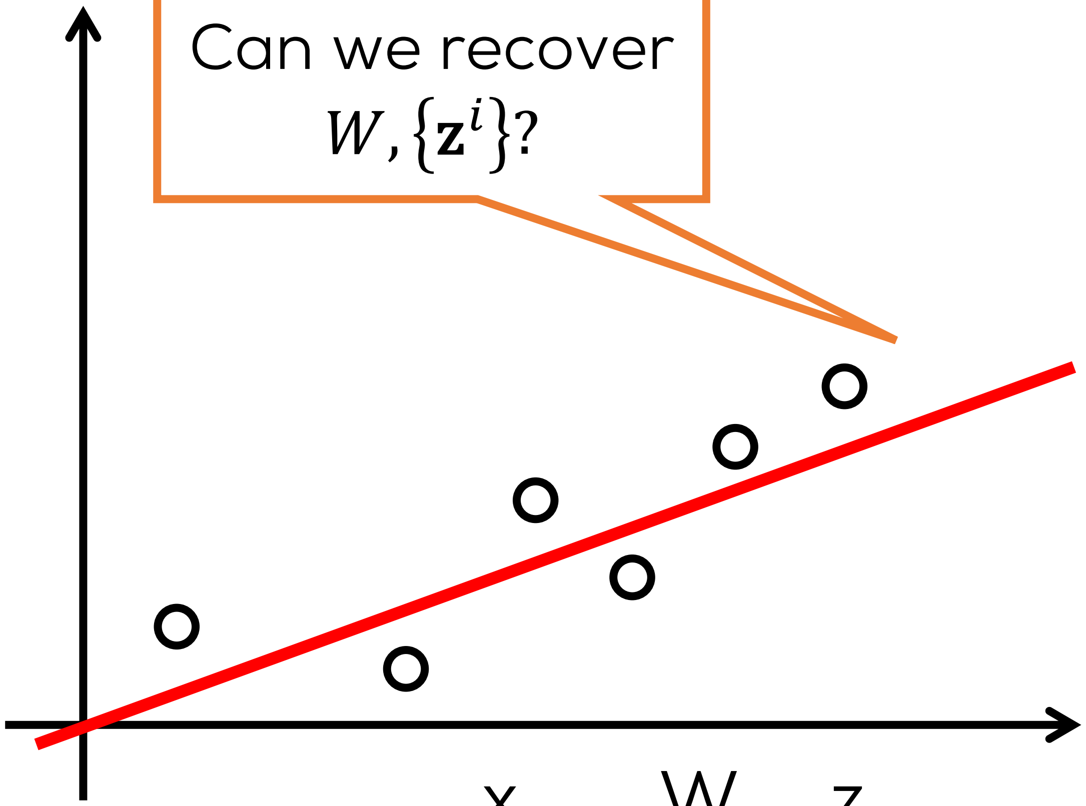
$$W \in \mathbb{R}^{d \times k}$$



Low-dimensional Structure in Data



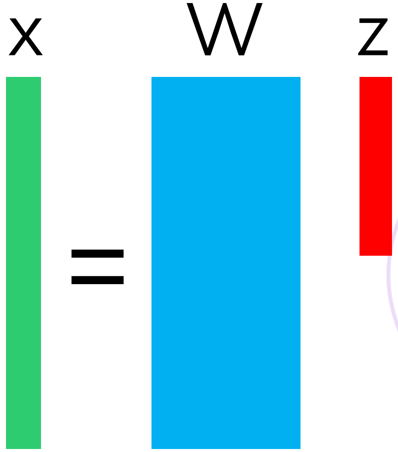
Can we recover $W, \{\mathbf{z}^i\}$?



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

Dictionary/Factor Loading matrix

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



Isn't this exactly Linear Regression?

- No, subtle differences exist
- If we write things in the same notation, then

Linear Regression

- $\mathbf{z}^i \in \mathbb{R}^k$,
- $y^i = \langle \mathbf{w}^*, \mathbf{z}^i \rangle + \epsilon^i$
- $\mathbf{w}^* \in \mathbb{R}^k$
- $\epsilon^i \sim \mathcal{N}(0, \sigma^2) \in \mathbb{R}$

- Observed data $\mathbf{x}^i = (\mathbf{z}^i, y^i) \in \mathbb{R}^{k+1}$

Low-rank Modelling

- $\mathbf{z}^i \in \mathbb{R}^k$,
- $\mathbf{y}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i$
- $W \in \mathbb{R}^{d \times k}$
- $\boldsymbol{\epsilon}^i \sim \mathcal{N}(\mathbf{0}, \sigma^2 \cdot I_d) \in \mathbb{R}^d$

- Observed data $\mathbf{x}^i = \mathbf{y}^i \in \mathbb{R}^d$

- In linear regression, $\mathbf{z}^i$ is visible, in low-rank data it is latent!

Applications

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

Applications

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

Original Collection of Images

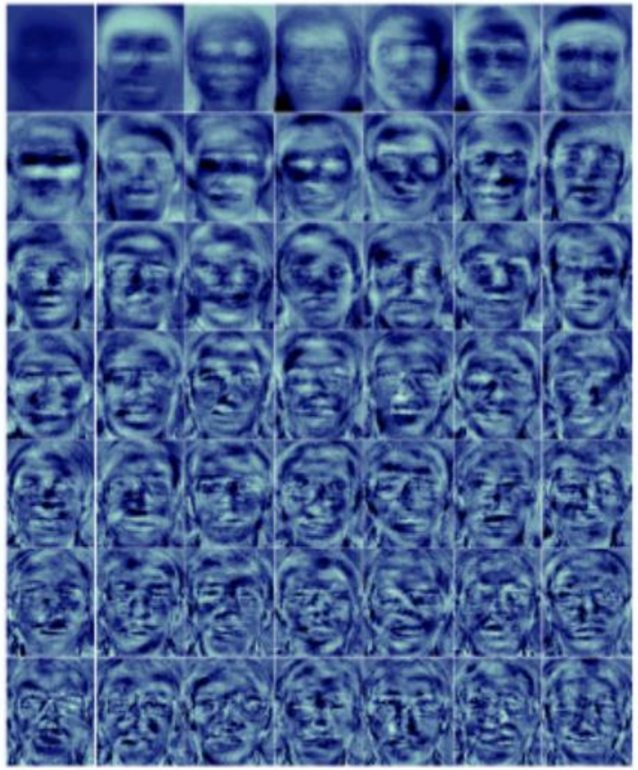
Credits: Piyush Rai, CS771, 2016-17-I

Applications

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

Original Collection of Images

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

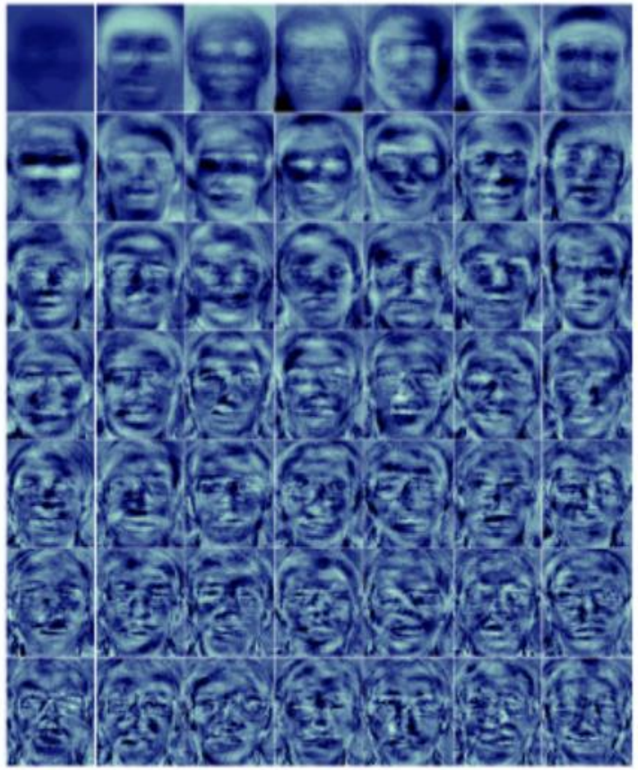


Applications

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

Original Collection of Images

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



17-1

Each image's reconstructed version

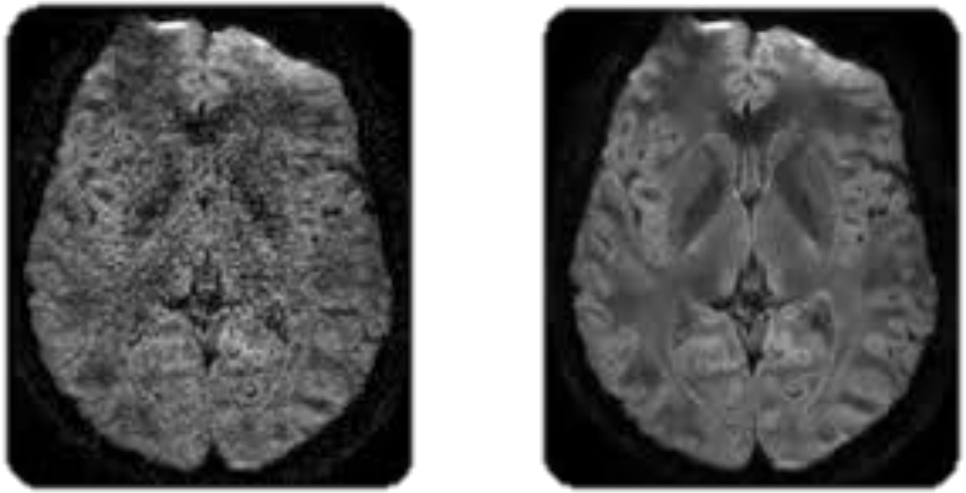
Applications

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

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

Applications

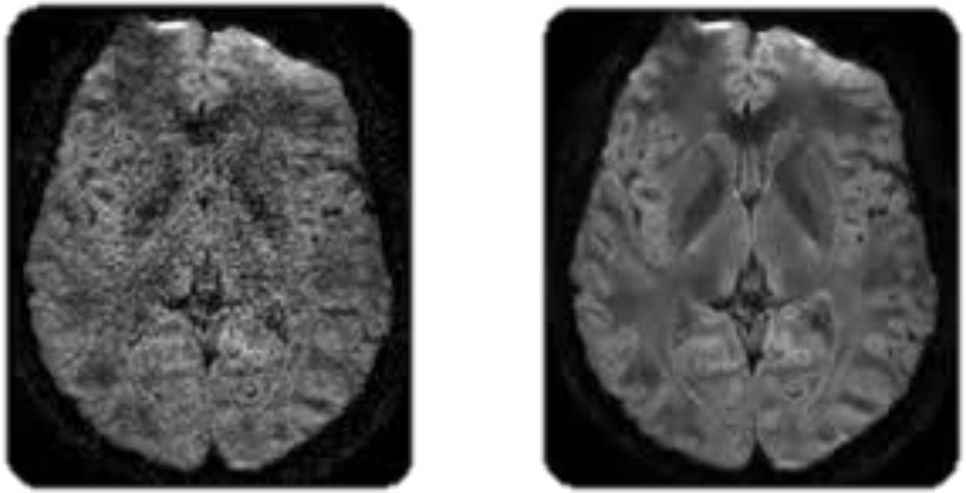
- Noise removal: $\mathbf{z}^i$ contains all the useful info, rest is noise



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

Applications

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



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

Modelling Low-rank Data

- As discussed, $\mathbf{z}^i \sim \mathcal{N}(\mathbf{0}, I_k)$
- As discussed $\mathbf{x}^i | \mathbf{z}^i \sim \mathcal{N}\left(W\mathbf{z}^i, \sigma^2 \cdot I_d\right)$
- Not the only possible choice - others possible – Factor Analysis
- Things are not that bad here

$$\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \int_{\mathbf{z}} \mathbb{P}\left[\mathbf{x}^i \mid \mathbf{z}, \sigma, W\right] \cdot \mathbb{P}[\mathbf{z}]\, d\mathbf{z} = \mathcal{N}(0, \sigma^2 \cdot I_d + WW^\top)$$

- Note: $\mathbb{P}[\mathbf{z} \mid \sigma, W] = \mathbb{P}[\mathbf{z}]$ by our definition
- Hmm … so $\mathbb{P}\left[\mathbf{x}^i \mid \sigma, W\right] = \mathcal{N}(0, \Sigma)$ where $\Sigma = \sigma^2 \cdot I_d + WW^\top$
- But I know how to estimate Σ give many samples of $\mathbf{x}$

[**MUR**] Chapter 4, [**BIS**] Chapter 2

Approach 1

Direct Estimation

Direct Estimation

- If we have $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$, then given many (many) samples $\mathbf{x}^i$

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

- So done ???
- Yeah … No…
- How do we extract σ, W from $\hat{\Sigma}$? (Remember $\Sigma = \sigma^2 \cdot I_d + WW^\top$)
- More importantly, Σ has d^2 parameters in it ($\Sigma \in \mathbb{R}^{d \times d}$)
- To estimate it reliably, will need $n \approx d^2$ samples … too much
- Moreover, there are actually only $\approx dk + 1$ parameters (W $\in \mathbb{R}^{d \times k}$ and $\sigma \in \mathbb{R}$). Should need only $n \approx dk$ samples

Approach 2

MLE Estimation for σ and W

Elementary Matrix Algebra

The Singular Value Decomposition Theorem

- Every real matrix $M \in \mathbb{R}^{m \times n}$ can be decomposed as
$$M = U \Lambda V^\top$$
- $U = [\mathbf{u}^1, \ldots, \mathbf{u}^m] \in \mathbb{R}^{m \times m}$ is an orthonormal matrix $UU^\top = U^\top U = I$
- $\langle \mathbf{u}^i, \mathbf{u}^j \rangle = 1$ if $i = j$, 0 otherwise
- Columns of U are the *left singular vectors* of M
- $V = [\mathbf{v}^1, \ldots, \mathbf{v}^m] \in \mathbb{R}^{n \times n}$ is an orthonormal matrix $VV^\top = V^\top V = I$
- Columns of V are the *right singular vectors* of M
- $\Lambda = \mathrm{diag}\left(\lambda_1, \ldots, \lambda_{\min(m,n)}\right) \in \mathbb{R}_+^{m \times n}$ is a diagonal matrix
- We order $\lambda_1 \geq \lambda_2 \geq \cdots$
- Diagonal entries of Λ are the *singular values* of M

λ_1	0	0	0
0	λ_2	0	0
0	0	λ_3	0

Elementary Matrix Algebra

The Singular Value Decomposition Theorem

- Every real matrix $M \in \mathbb{R}^{m\times n}$ can be decomposed as

$$M = U\Lambda V^\top = \sum_{i=1}^{\min(m,n)} \lambda_i \cdot \mathbf{u}^i \left(\mathbf{v}^i\right)^\top$$

- For all $i = 1, \ldots \min(m, n)$, $M\mathbf{v}^i = \lambda_i \mathbf{u}^i$
- Why? Because $\langle \mathbf{v}^i, \mathbf{v}^j \rangle = 1$ if $i = j$, 0 otherwise, will be very useful
- U forms a basis for $\mathbb{R}^m$
- Every vector $\mathbf{x} \in \mathbb{R}^m$ can be written as $\mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{u}^i$
- α_i can be (uniquely) found as $\alpha_i = \langle \mathbf{x}, \mathbf{u}^i \rangle$
- V similarly forms a basis for $\mathbb{R}^n$

Elementary Matrix Algebra

- If the matrix $M \in \mathbb{R}^{m \times m}$ is symmetric (and hence square) then we can instead write the matrix as

$$M = U \Lambda U^\top = \sum_{i=1}^{m} \lambda_i \cdot \mathbf{u}^i (\mathbf{u}^i)^\top$$

- Columns of U are the *eigenvectors* of M
- Diagonal entries of Λ are the *eigenvalues* of M (they are real)
- If all eigenvalues are $\geq \mathbf{0}$, then the matrix is called positive semi-definite (PSD)
- $\sigma^2 \cdot I$ is PSD, matrices of the form $M = XX^\top$ or $X^\top X$ are PSD
- If A, B are PSD then $A + B$ is PSD too!

MLE Estimation

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

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

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

- $\operatorname{tr}(M) = \sum_i M_{i,i}$
- For any $A, B \in \mathbb{R}^{m\times n}$, $\operatorname{tr}(A^\top B) = \sum_{i,j} A_{i,j}B_{i,j} = \operatorname{tr}(B^\top A)$
- Let $S = U\Lambda U^\top$ be the eigen-decomposition of S
- Alternately, let $X = U\sqrt{\Lambda}V^\top$ be the singular decomposition of X
- $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_d)$, $\lambda_i \geq 0$ (Why?), $\sqrt{\Lambda} = \operatorname{diag}\left(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_d}\right)$ (notation)
- In general if $A = \operatorname{diag}(a_1, \ldots, a_d)$, then $\sqrt{A} = \operatorname{diag}\left(\sqrt{a_1}, \ldots, \sqrt{a_d}\right)$

MLE Estimation

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

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

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
- $\hat{\sigma}_{\mathrm{MLE}} = \frac{1}{d-k}\sum_{j=k+1}^{d} \lambda_j$
- Remember, we order $\lambda_1 \geq \lambda_2 \geq \cdots$
- $\widehat{W}_{\mathrm{MLE}} = \mathrm{U_k}\sqrt{\Lambda_k - \hat{\sigma}^2{}_{\mathrm{MLE}} \cdot I}$
- where $U_k = [u^1, \ldots, u^k]$ and $\Lambda_k = [\lambda_1, \ldots, \lambda_k]$
- Top k eigenvalues and eigenvectors

Probabilistic Principal Component Analysis (Tipping and Bishop, 1999)

Principal Component Analysis

Noiseless dimensionality reduction

The PCA estimate

- Let $\sigma = 0$, then the MLE looks like (no need to estimate σ)
$$\widehat{W}_{\mathrm{MLE}} = \mathrm{U}_{\mathrm{k}}\sqrt{\Lambda_k}$$
- So we need to find the k leading eigenvalues/vectors of S
- Recall $S = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^i \left(\mathbf{x}^i\right)^{\top}$
- In general it takes $O(d^3)$ time to find all d eigenvectors/values
- Much faster method to find top k in $O(d^2k)$ time

The PCA estimate

Beautiful FA interpretation – next time!!

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

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

- So we need to find the k leading eigenvalues/vectors of S
- Recall $S = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}^i \left(\mathbf{x}^i\right)^{\top}$
- In general it takes $O(d^3)$ time to find all d eigenvectors/values
- Much faster method to find top k in $O(d^2 k)$ time

The Power Method

- Let $S = U\Lambda U^\top = \sum_{i=1}^{d} \lambda_i \mathbf{u}^i (\mathbf{u}^i)^\top$ and $\lambda_1 > \lambda_2 \geq \cdots$ (strict separation)
- The above condition can be relaxed to handle cases $\lambda_1 = \lambda_2$
- But makes life more complicated
- Key idea: U forms a basis for $\mathbb{R}^d$, every $\mathbf{x} \in \mathbb{R}^d$ is $\mathbf{x} = \sum_{i=1}^{d} \alpha_i \mathbf{u}^i$
- Assume that we have a vector $\mathbf{x}$ so that $\alpha_i = \frac{1}{\sqrt{d}}$ for all $i \in [d]$
- Then what is the vector $S\mathbf{x}$?
- Since $\langle \mathbf{u}^i, \mathbf{u}^j \rangle = 1$ if $i = j$, 0 otherwise

$$S\mathbf{x} = \sum_{i=1}^{d} \alpha_i \lambda_i \cdot \mathbf{u}^i$$

Notice $\alpha_1 \lambda_1 > \alpha_i \lambda_i$ for all $i \neq 1$

Amplifies component along leading eigenvector

The Power Method

- Continuing this way, we can show that

$$S^t\mathbf{x} = \underbrace{SSSSSS}_{t}\ \mathbf{x} = \sum_{i=1}^{d} \alpha_i \lambda_i^t \cdot \mathbf{u}^i$$

- Even if $\lambda_1 = 1.01$ and $\lambda_2 = 1.005$, after t=1000 iterations, $\lambda_1^t > 20000$ whereas $\lambda_2 < 150$. Tiny differences get amplified greatly!!
- Even if $\lambda_1 = 0.995$ and $\lambda_2 = 0.99$, after t=1000 iterations, $\lambda_1^t > 0.005$ whereas $\lambda_2 < 0.00005$. The difference is still amplified!!
- No need to have α_i equal. Even if α_1 is smaller than other α_i, soon we will have $\alpha_1\lambda_1^t$ much much larger than $\alpha_i\lambda_i^t$ for $i \neq 1$.
- The only thing we need to be careful about is to not have $\alpha_1 = 0$. The above procedure fails if $\alpha_1 = 0$

The Power Method

THE POWER METHOD

1. Matrix S
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

Ensures $\alpha_1 \neq 0$ with high probability

THE POWER METHOD

1. Matrix S
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

Ensures $\alpha_1 \neq 0$ with high probability

Takes only $O(d^2)$ time

THE POWER METHOD

1. Matrix S
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
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 $\|S\mathbf{x}^T\|_2$

Ensures $\alpha_1 \neq 0$ with high probability

Takes only $O(d^2)$ time

Eigenvectors are all unit norm

The Power Method

THE POWER METHOD

1. Matrix S
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 $\|S\mathbf{x}^T\|_2$

Ensures $\alpha_1 \neq 0$ with high probability

Takes only $O(d^2)$ time

Eigenvectors are all unit norm

Why is this appropriate ??

The Power Method

THE POWER METHOD

1. Matrix S
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 $\|S\mathbf{x}^T\|_2$

Ensures $\alpha_1 \neq 0$ with high probability

Takes only $O(d^2)$ time

Eigenvectors are all unit norm

Why is this appropriate ??

Overall $O(d^2)$ time

Principal Component Analysis

THE PCA METHOD

1. Matrix S
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \ldots, k$
   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 $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

Principal Component Analysis

THE PCA METHOD

1. Matrix S
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \ldots, k$
   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 $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

The peeling method

Principal Component Analysis

THE PCA METHOD

1. Matrix S
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 $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

The peeling method

$$S = \lambda_1 \mathbf{u}^1 (\mathbf{u}^1)^\top + \lambda_2 \mathbf{u}^2 (\mathbf{u}^2)^\top + \lambda_3 \mathbf{u}^3 (\mathbf{u}^3)^\top + \lambda_4 \mathbf{u}^4 (\mathbf{u}^4)^\top$$

Principal Component Analysis

THE PCA METHOD

1. Matrix S
2. Initialize $S^0 \leftarrow S$
3. For $j = 1, \ldots, k$
   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. Return $\widehat{W}_{\text{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

The peeling method

$$S = \lambda_1 \mathbf{u}^1 (\mathbf{u}^1)^\top + \lambda_2 \mathbf{u}^2 (\mathbf{u}^2)^\top + \lambda_3 \mathbf{u}^3 (\mathbf{u}^3)^\top + \lambda_4 \mathbf{u}^4 (\mathbf{u}^4)^\top$$

Principal Component Analysis

THE PCA METHOD

1. Matrix S
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 $\widehat{W}_{\text{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

The peeling method

$$S = \lambda_1 \mathbf{u}^1 (\mathbf{u}^1)^\top + \lambda_2 \mathbf{u}^2 (\mathbf{u}^2)^\top + \lambda_3 \mathbf{u}^3 (\mathbf{u}^3)^\top + \lambda_4 \mathbf{u}^4 (\mathbf{u}^4)^\top$$

Principal Component Analysis

THE PCA METHOD

1. Matrix S
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 $\hat{W}_{\text{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

The peeling method

$$S = \lambda_1 \mathbf{u}^1 (\mathbf{u}^1)^\top + \lambda_2 \mathbf{u}^2 (\mathbf{u}^2)^\top + \lambda_3 \mathbf{u}^3 (\mathbf{u}^3)^\top + \lambda_4 \mathbf{u}^4 (\mathbf{u}^4)^\top$$

Principal Component Analysis

THE PCA METHOD

1. Matrix S
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 $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

The peeling method

Some residue might still be left due to inaccurate estimation of $\lambda_i, \mathbf{u}^i$ but usually small

$$S = \lambda_1 \mathbf{u}^1 (\mathbf{u}^1)^\top + \lambda_2 \mathbf{u}^2 (\mathbf{u}^2)^\top + \lambda_3 \mathbf{u}^3 (\mathbf{u}^3)^\top + \lambda_4 \mathbf{u}^4 (\mathbf{u}^4)^\top$$

Principal Component Analysis

THE PCA METHOD

1. Matrix S
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 $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j} \cdot \hat{\mathbf{u}}_j$

Overall $O(d^2 k)$ time

The peeling method

Some residue might still be left due to inaccurate estimation of $\lambda_i, \mathbf{u}^i$ but usually small

$$S = \lambda_1 \mathbf{u}^1 (\mathbf{u}^1)^\top + \lambda_2 \mathbf{u}^2 (\mathbf{u}^2)^\top + \lambda_3 \mathbf{u}^3 (\mathbf{u}^3)^\top + \lambda_4 \mathbf{u}^4 (\mathbf{u}^4)^\top$$

Probabilistic Principal Component Analysis

THE PPCA METHOD

1. Matrix S
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 $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j - \hat{\sigma}_{\mathrm{MLE}}} \cdot \hat{\mathbf{u}}_j$

Probabilistic Principal Component Analysis

THE PPCA METHOD

1. Matrix S
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 $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Let $\hat{\sigma}_{\mathrm{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^{d} \hat{\lambda}_j$
5. Return $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j - \hat{\sigma}_{\mathrm{MLE}}} \cdot \hat{\mathbf{u}}_j$

Recall that

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

Probabilistic Principal Component Analysis

THE PPCA METHOD

1. Matrix S
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 $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Let $\hat{\sigma}_{\mathrm{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^{d} \hat{\lambda}_j$
5. Return $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j - \hat{\sigma}_{\mathrm{MLE}}} \cdot \hat{\mathbf{u}}_j$

Recall that

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

Takes $O(d^3)$ time ☹

Probabilistic Principal Component Analysis

THE PPCA METHOD

1. Matrix S
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 $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Let $\hat{\sigma}_{\mathrm{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^{d} \hat{\lambda}_j$
5. Return $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j - \hat{\sigma}_{\mathrm{MLE}}} \cdot \hat{\mathbf{u}}_j$

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?

Probabilistic Principal Component Analysis

THE PPCA METHOD

1. Matrix S
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 $S^j \leftarrow S^{j-1} - \hat{\lambda}_j \cdot \hat{\mathbf{u}}_j (\hat{\mathbf{u}}_j)^\top$
4. Let $\hat{\sigma}_{\mathrm{MLE}} = \frac{1}{d-k} \sum_{j=k+1}^{d} \hat{\lambda}_j$
5. Return $\widehat{W}_{\mathrm{MLE}} = \sum_{j=1}^{k} \sqrt{\hat{\lambda}_j - \hat{\sigma}_{\mathrm{MLE}}} \cdot \hat{\mathbf{u}}_j$

Recall that

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

Takes $O(d^3)$ time ☹

After we reconvene

Can we do better?

A Few Thoughts

- Many extensions possible
  - Factor analysis $\boldsymbol{\epsilon}^i \sim \mathcal{N}(0, \Sigma_x)$
  - Non-centered data $\mathbf{z}^i \sim \mathcal{N}(\boldsymbol{\mu}_z, \Sigma_z)$
  - Non-centered noise $\mathbf{x}^i = W\mathbf{z}^i + \boldsymbol{\epsilon}^i$, $\boldsymbol{\epsilon}^i \sim \mathcal{N}(\boldsymbol{\mu}_\epsilon, \Sigma_x)$
- Handle missing data, as can most generative models (GMM etc)
  - $\mathbf{x}^i = [\mathbf{x}^i_{\text{obs}}, \mathbf{x}^i_{\text{miss}}]$, $\mathbb{P}[\mathbf{x}^i] = \mathbb{P}[\mathbf{x}^i_{\text{miss}} \mid \mathbf{x}^i_{\text{obs}}] \cdot \mathbb{P}[\mathbf{x}^i_{\text{obs}}]$
- Mixture of PPCA? Mixture of GMMs?
- Sequential models: Kalman filters, Hidden Markov models
- Hierarchical models

A Few Thoughts

- PPCA, PCA do not do well on data with non-linear structure

A Few Thoughts

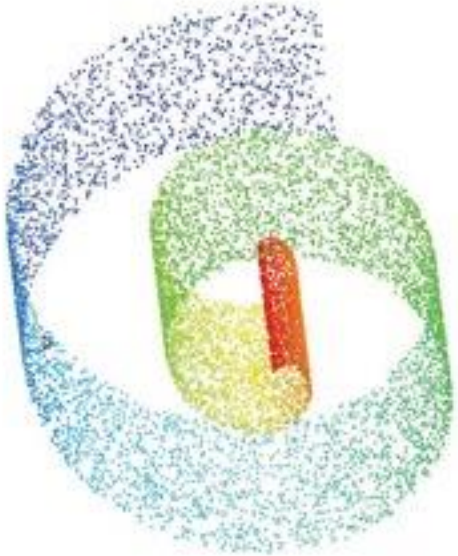
- PPCA, PCA do not do well on data with non-linear structure

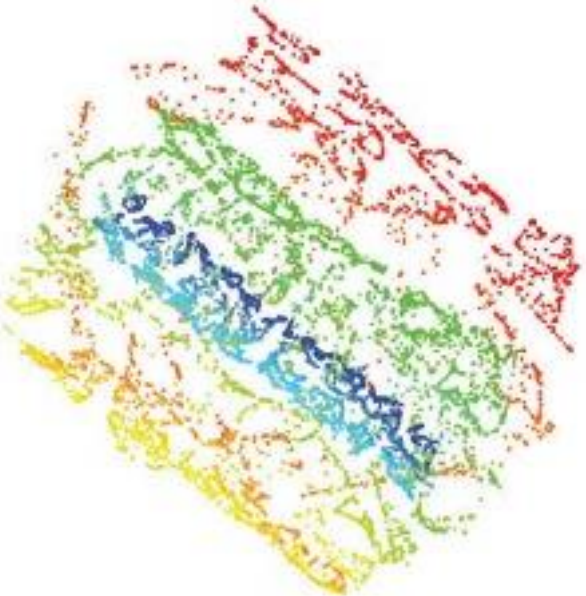
“Swiss Roll” data

A Few Thoughts

- PPCA, PCA do not do well on data with non-linear structure

"Swiss Roll" data



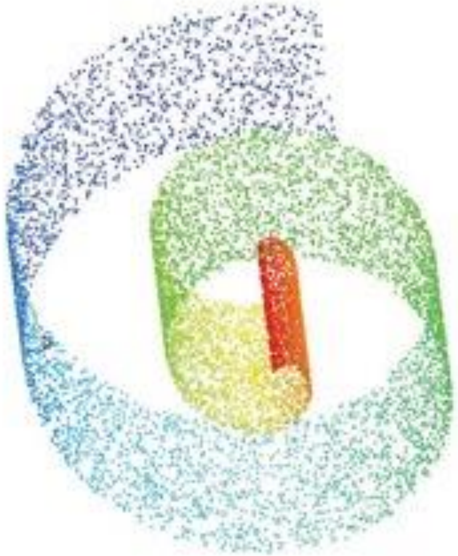


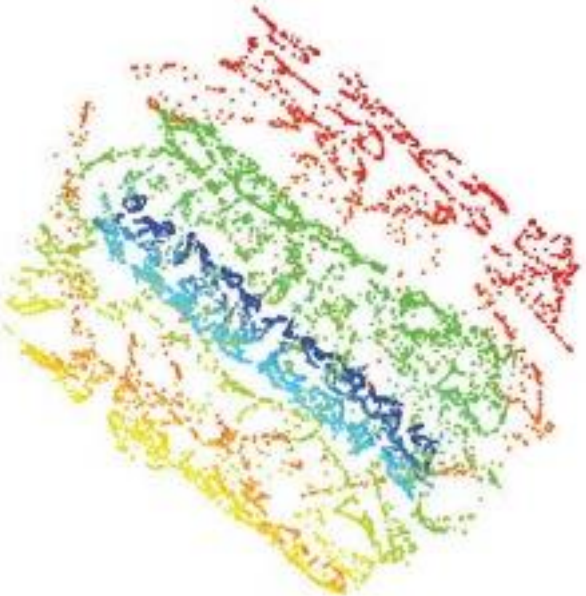
What PCA/PPCA will give us

A Few Thoughts

- PPCA, PCA do not do well on data with non-linear structure

“Swiss Roll” data





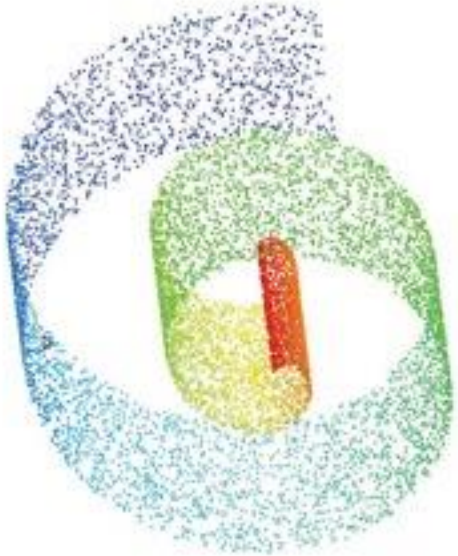
What we really want

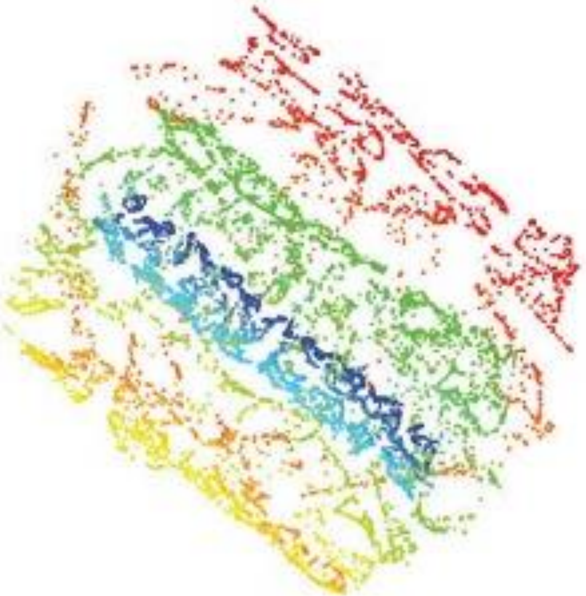
What PCA/PPCA will give us

A Few Thoughts

- PPCA, PCA do not do well on data with non-linear structure

“Swiss Roll” data





Non-linear dimensionality reduction Kernel PCA, Autoencoders

What we really want

What PCA/PPCA will give us

Please give your Feedback

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