EE5907: Pattern Recognition
(Second Half)

&

EE5026: Machine Learning for Data Analytics

Dr. WANG Si

Email: si.wang@nus.edu.sg

Office: E1a-04-01

Outlines

- Pattern Representation Learning
  - Unsupervised Representation Learning (week 7)
  - Supervised Representation Learning (week 8)
  - Unified Framework: Graph Embedding (week 8)
- Patter Recognition Models
  - Clustering and Applications (week 9)
  - Gaussian Mixture Model and Boosting (week 10)
  - Support Vector Machines (week 11)
- Deep Learning (week 12)
- Revision and Q&A (week 13)

Continuous Assessments

(20%) (40%)

- ❑ EE5907 CA2 & EE5026 CA1: 9 Oct. 2025 - 14 Nov. 2025 1800 SGT (Thu. Week 8 - Fri. Week 13)

  Policy on late submission:

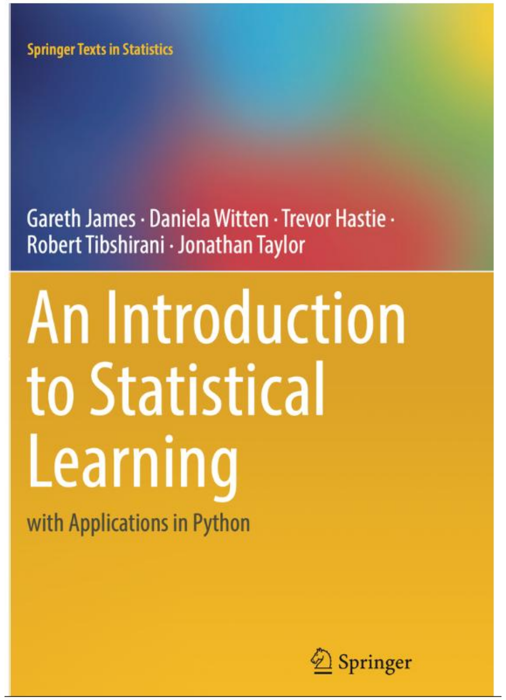
  - Up to 12 hours late: 80% of your earned points
  - Up to 1 day late: 50% of your earned points
  - Up to 2 days late: 30% of your earned points
  - More than 2 days late: No marks (0%)

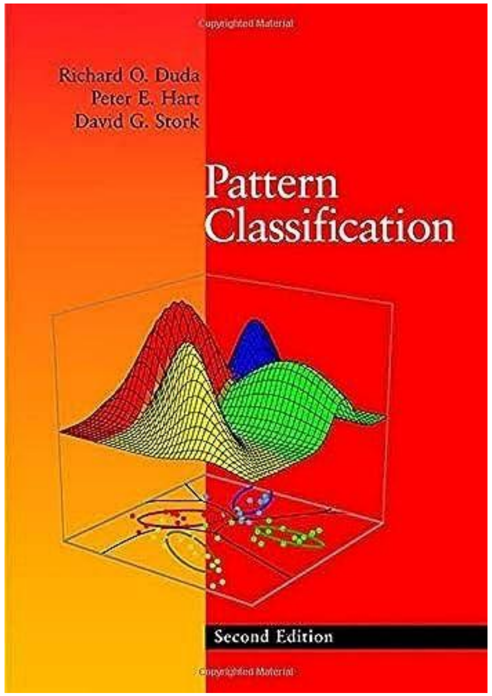
- ❑ EE5907 & EE5026 Final Exam (60%): Closed book with one-page double-sided A4-sized cheat sheet

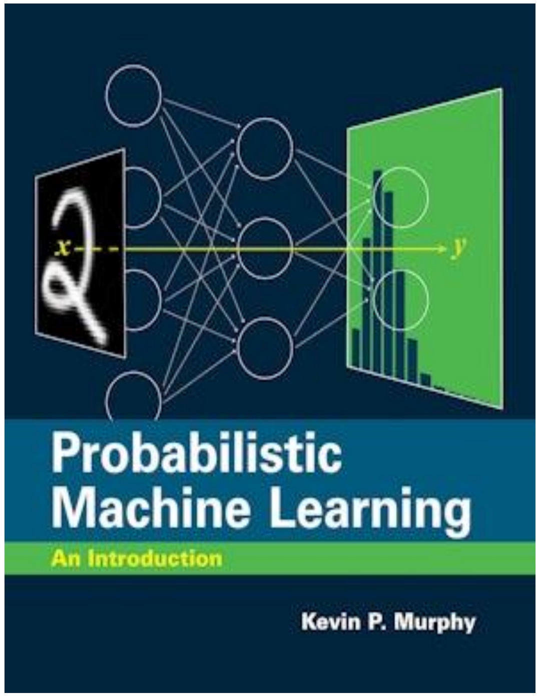
Textbooks and References

(no fixed textbook)

- Books
  - Gareth, J., et al. "**An Introduction to Statistical Learning: with Applications in Python**." (2023)
  - Richard O. Duda, et al. “**Pattern Classification.**” (2001, Wiley)
  - Murphy, Kevin P., “**Probabilistic Machine Learning: An Introduction.**” (2022, MIT press)







Pattern Representation Learning

- We will discuss:
  - What is pattern representation learning
  - Principal Component Analysis (PCA)
  - Non-negative Matrix Factorization (NMF)
- At the end of this lecture, you should be able to:
  - Know what pattern representation learning is
  - Understand rationales behind PCA and NMF
  - Perform PCA and NMF

What are we doing with Pattern Recognition?

Raw Data

⇩

Data Preprocessing

⇩

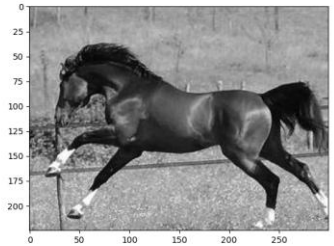
Extract Informative Pattern Representation

⇩

PR Model (after training)

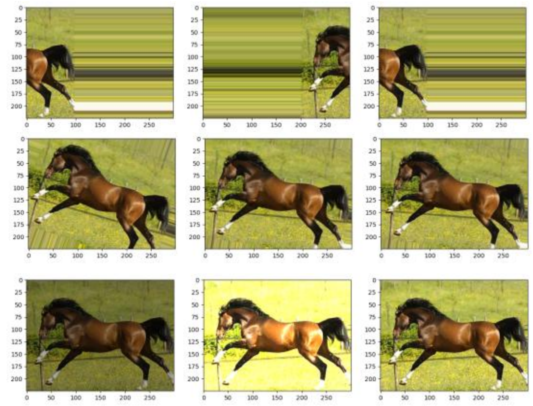
⇩

Recognition Results



Grayscale

Normalization



Augmentation

	Face	Neck	Legs
Horse	Long	Long	Long
Cat	Round	Short	Short

Unsupervised
- K-means
- Hierarchical clustering
- Gaussian Mixture Model

Supervised
- Boosting
- Support Vector Machine

Deep Neural Network

↓ Cluster assignment

↓ Classification / Regression

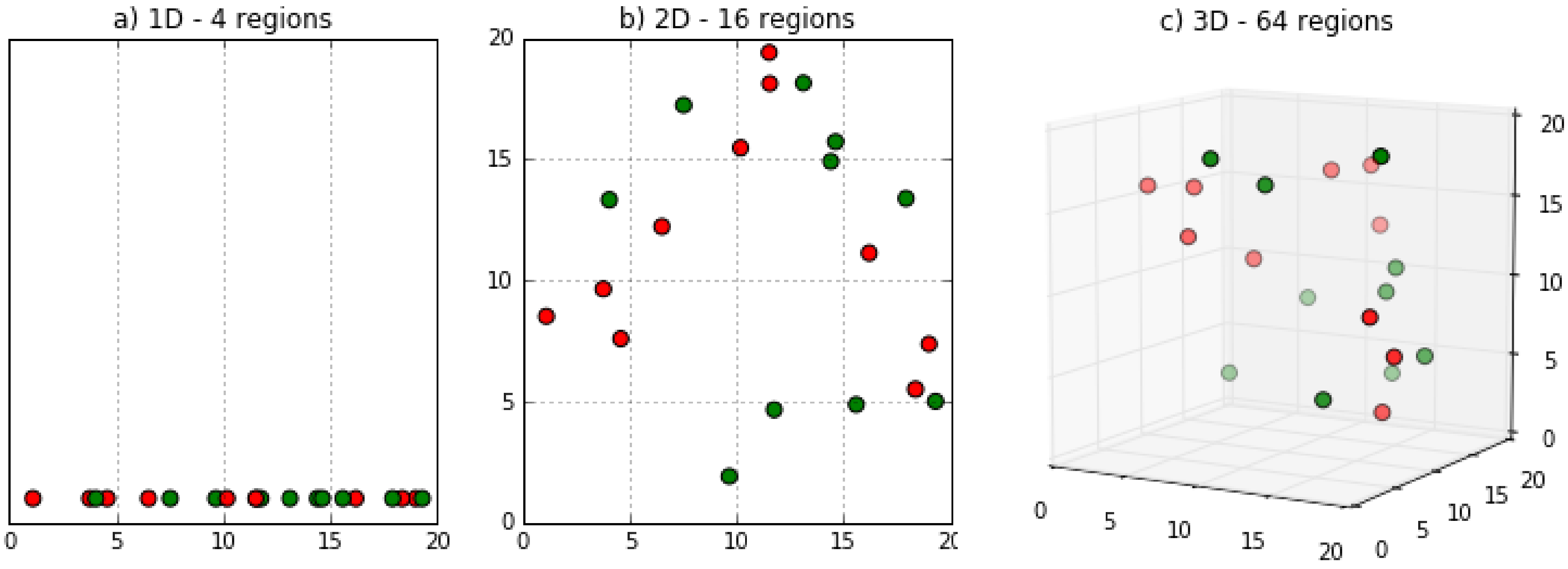
Representation Learning / Extraction

- Representation learning refers to transforming the raw/preprocessed data into another (possibly lower-dimensional) space.
- Such that, in the new space, one can perform pattern recognition more easily.
- Criterion for good representation in different problem settings:
  - **Unsupervised**: minimize information loss (no class information)
  - **Supervised**: maximize discrimination (with class information)

	Face	Neck	Legs
Horse	Long	Long	Long
Cat	Round	Short	Short

Why Feature Extraction?

- Many traditional pattern recognition techniques may not be effective for high-dimensional data
  - Curse of dimensionality



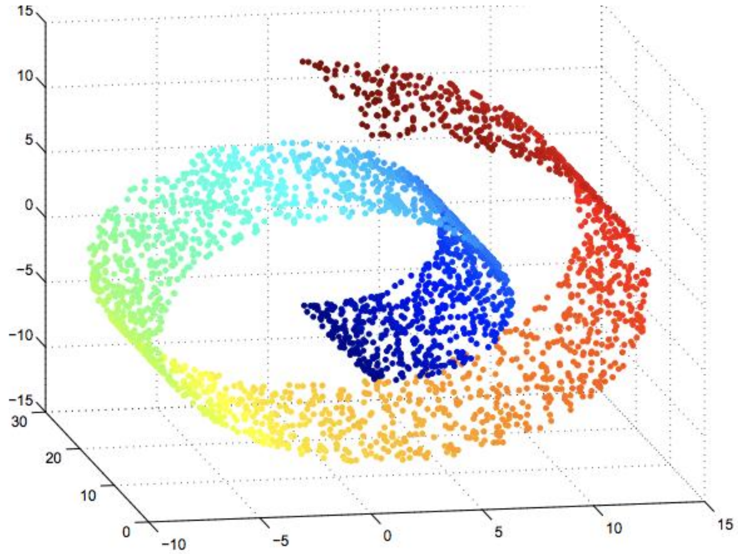
Why Feature Extraction?

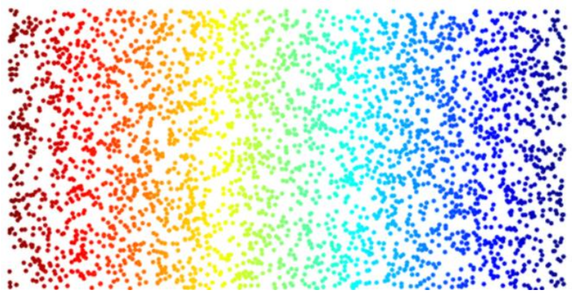
- Patterns may have small intrinsic dimension

400x400x3=480000 dimensions

Intrinsic dimensions

(50-100 dimensions)





Why Feature Extraction?

- Visualization: projecting high-dimensional data onto 2D or 3D planes

- Data compression: efficient storage and retrieval

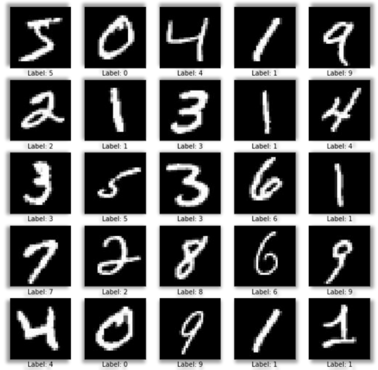
- Noise removal: positive effect on testing accuracy

Applications of Feature Extraction

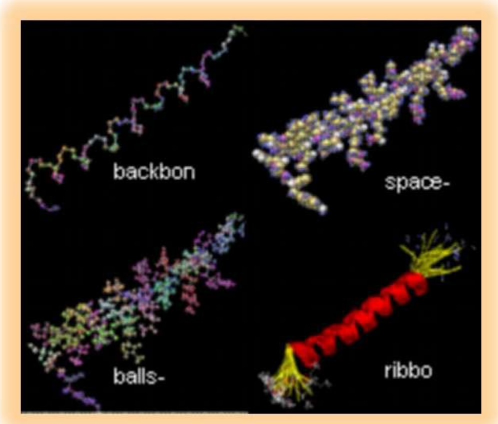
- Face recognition
- Handwritten digit recognition
- Text mining
- Image retrieval
- Protein classification

Face Images

Text



Handwritten digit



Proteins

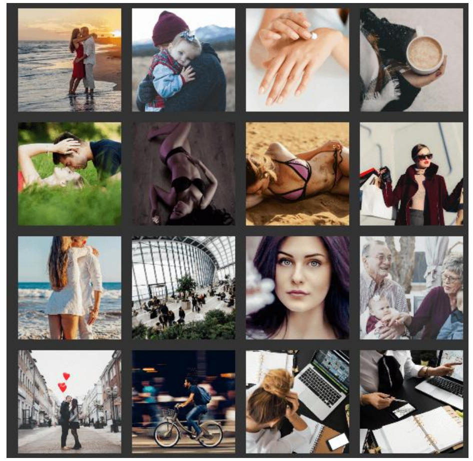


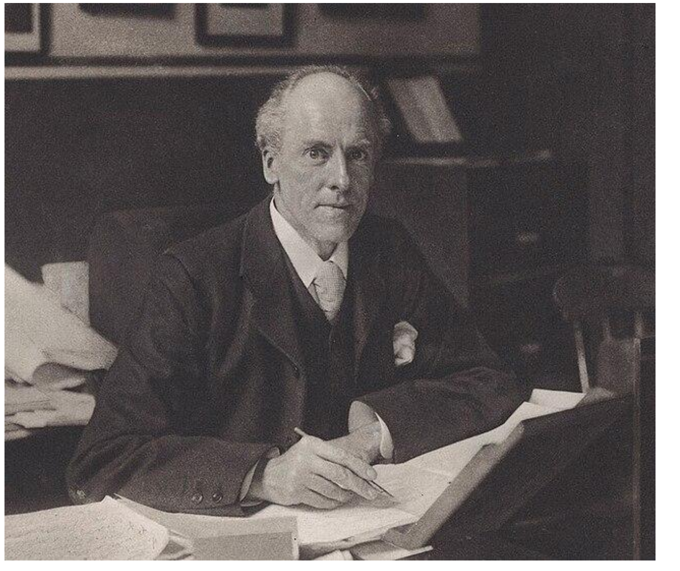
Image database

Unsupervised Feature Learning

Principal Component Analysis

Principal Component Analysis (PCA)

- Probably the most widely-used and well-known multivariate analysis method.
- Introduced by Pearson (1901)
- First applied in ecology by Goodall (1954) under the name “factor analysis”.



Karl Pearson

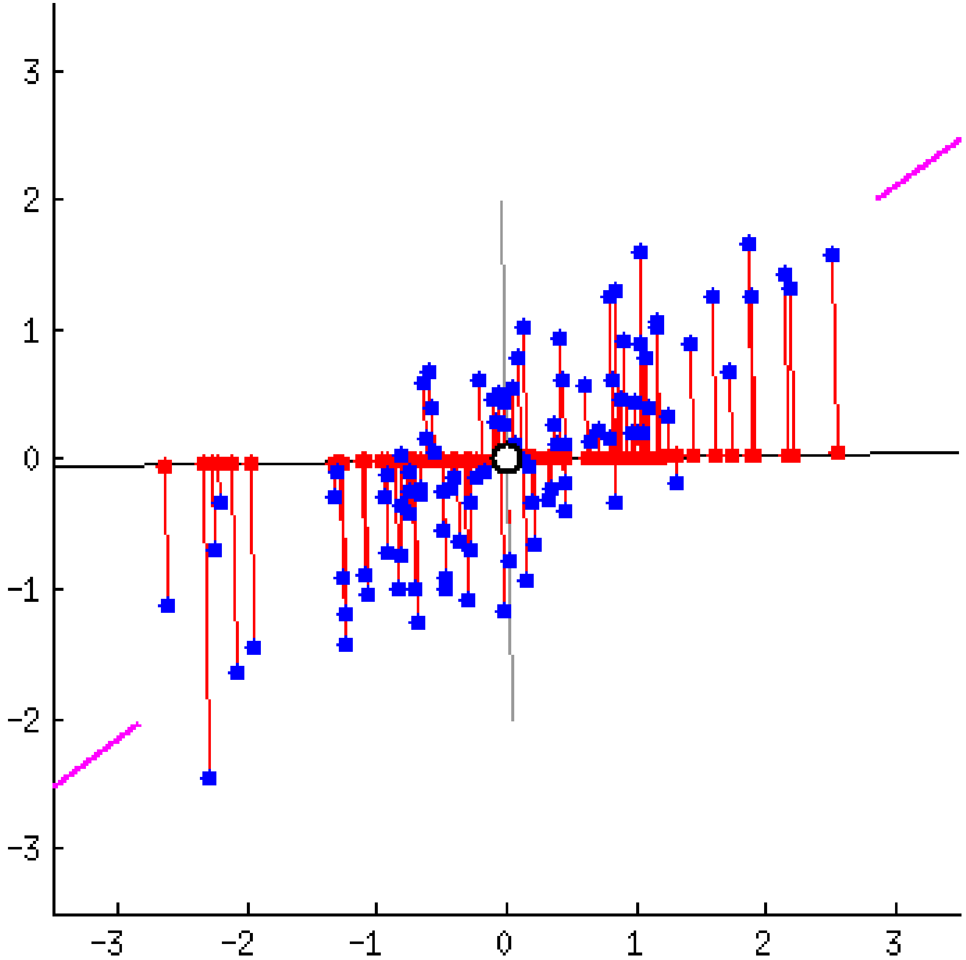
Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space" Philosophical Magazine **2** (11): 559–572.

What is Principal Component Analysis?

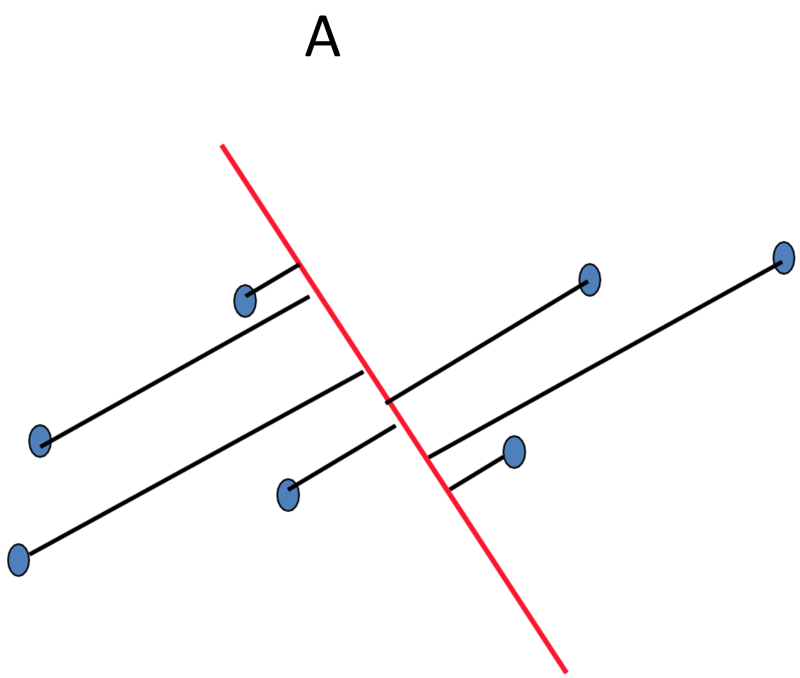
- Principal component analysis (PCA)
  - Reduce the dimensionality of a collection of observations by projecting onto a new smaller set of variables, called principal components (PCs), while preserving as much information as possible.

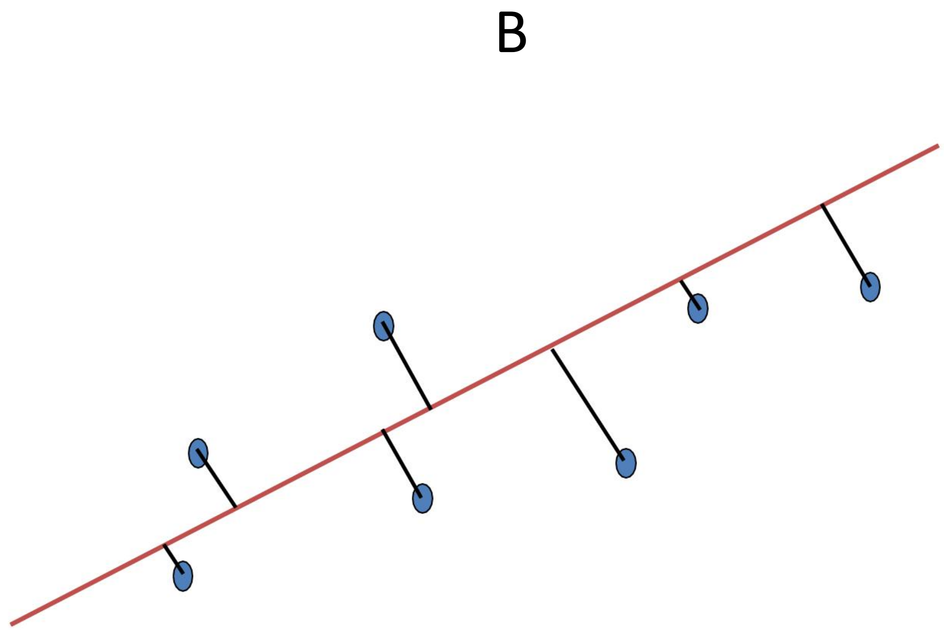
- What is PC?
  - Capture the big (principal) variability in the data
  - Ordered by captured variations from largest to smallest (e.g., 1^{st} PC captures the largest variability)
  - PCs are uncorrelated (orthogonal to each other)

Geometric Picture of Principal Components



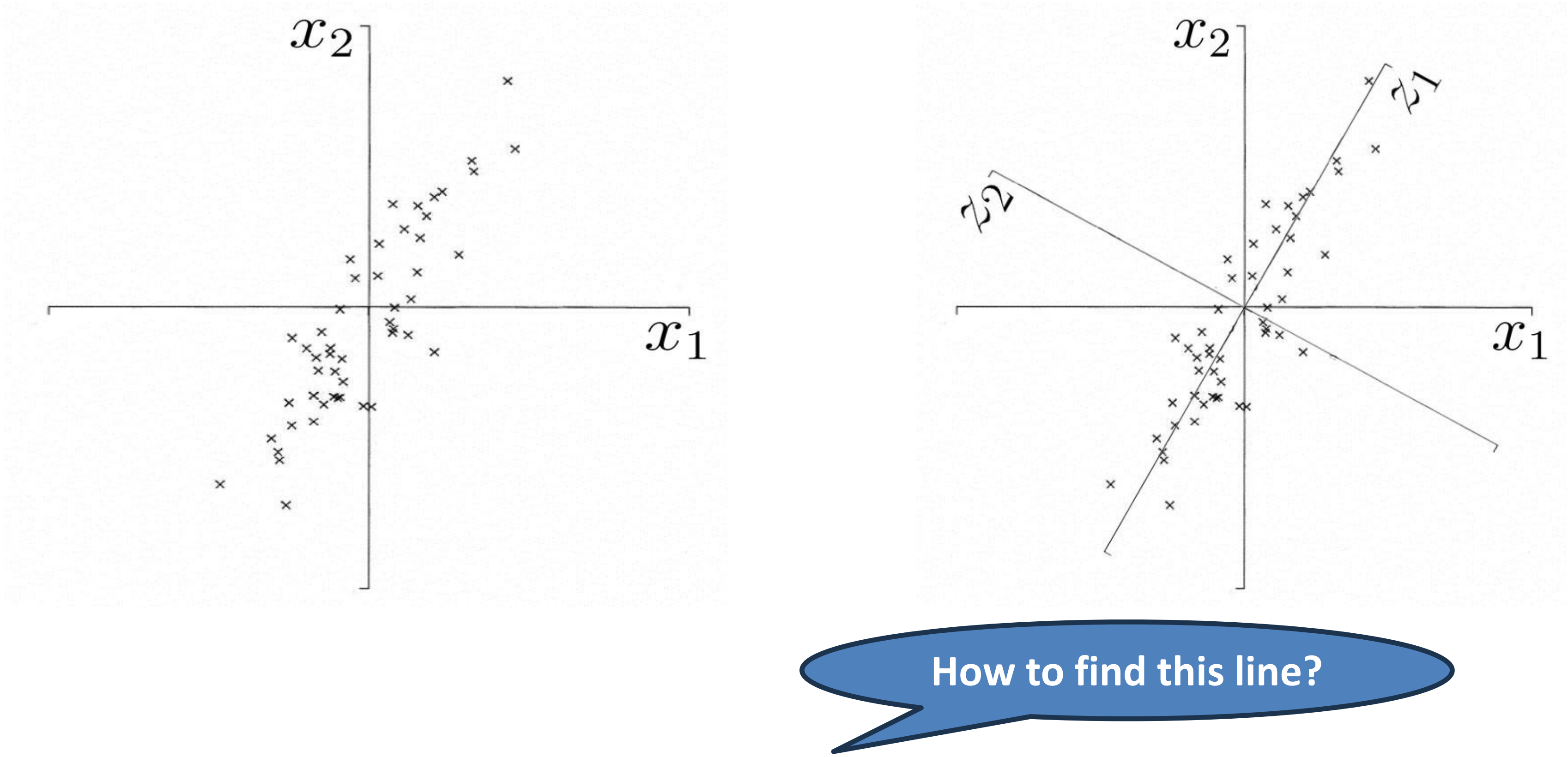
Geometric Picture of Principal Components





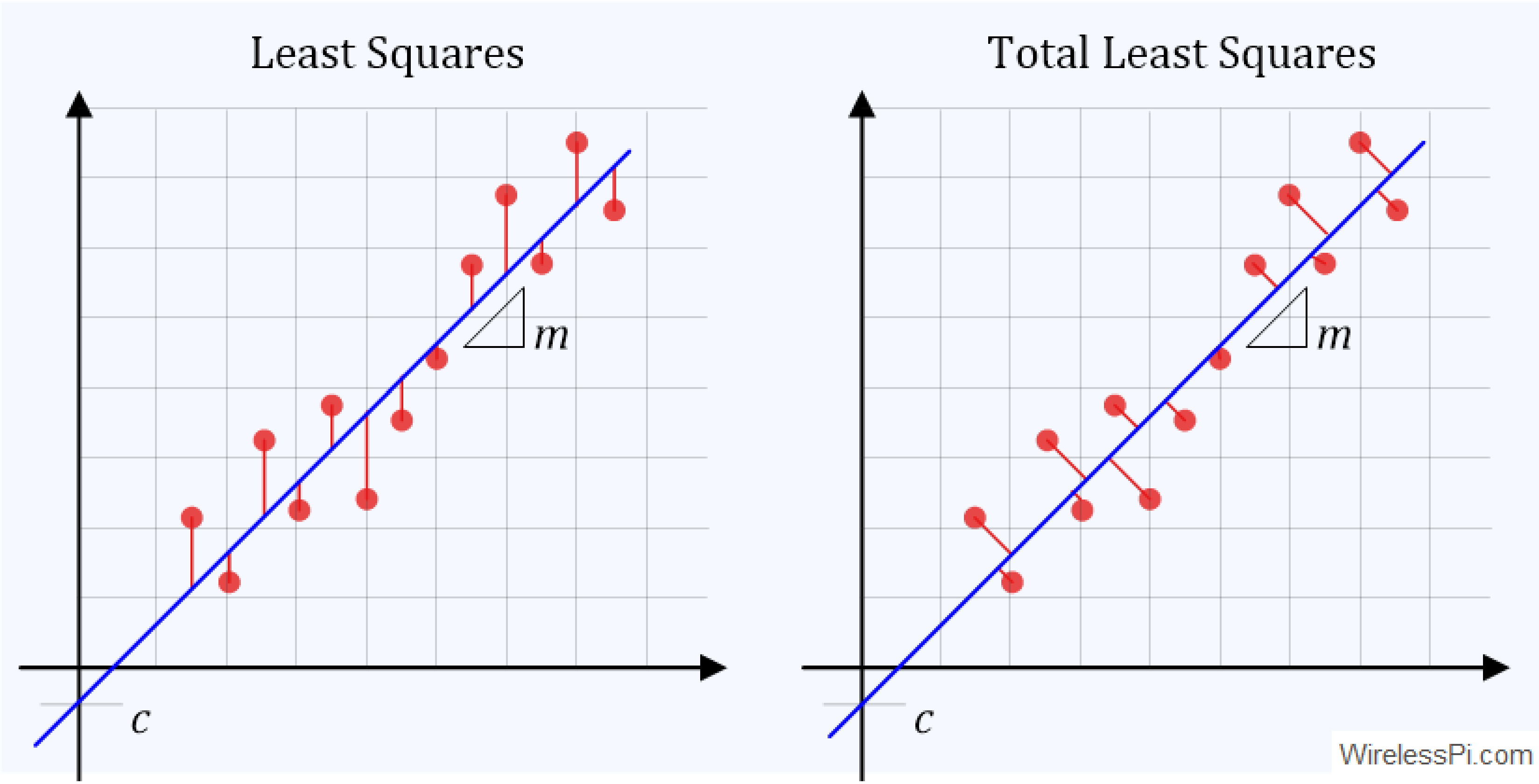
Which one is the 1st PC?

Geometric Picture of Principal Components



- The 1st PC z_1 is parallel to the line minimum-distance fitted to the data points
- The 2nd PC z_2 is parallel to the line minimum-distance fitted to the data points after z_1 has been taken into account

Geometric Picture of Principal Components



https://people.duke.edu/~hpgavin/SystemID/CourseNotes/TotalLeastSquares.pdf

Algebraic Definition of PCs

Given a sample set of n observations on a vector of d variables

$$\{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$$

Define the first principal component (PC_1) to be the normalized linear projection vector $\mathbf{z}_1 = [z_{1,1}, \ldots, z_{d,1}]^T$, such that

$$y_1 = \mathbf{z}_1^T \mathbf{x}$$

var[y_1] is maximum, where y_1 is the projection along 1^{st} PC axis

Algebraic Derivation of PCs

To find $\mathbf{z}_1$ that maximizes $\mathrm{var}[y_1]$, subject to $\mathbf{z}_1^T\mathbf{z}_1 = 1$

$$\operatorname*{argmax}_{\mathbf{z}_1} \mathrm{var}[y_1]\,, s.t., \mathbf{z}_1^T\mathbf{z}_1 = 1 \qquad \bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

$$\mathrm{var}[y_1] = \mathbb{E}[(y_1 - \overline{y_1})^2] = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{z}_1^T\mathbf{x}_i - \mathbf{z}_1^T\bar{\mathbf{x}})^2 = \mathbf{z}_1^T\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right)}_{\text{Covariance matrix } \mathbf{S}}\mathbf{z}_1$$

$$\begin{bmatrix} \mathrm{Cov}(X_1, X_1) & \cdots & \mathrm{Cov}(X_1, X_d) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_d, X_1) & \cdots & \mathrm{Cov}(X_d, X_d) \end{bmatrix}$$

Cov(*A*,*B*)= $\mathbb{E}[(A - \mathbb{E}[A])\big(B - \mathbb{E}(B)\big)]$

$$f = \mathbf{z}_1^T\mathbf{S}\mathbf{z}_1,\ g = \mathbf{z}_1^T\mathbf{z}_1 - 1$$

Let λ be a Lagrange multiplier,

$$\nabla_{\mathbf{z}_1}(\mathbf{z}_1^T\mathbf{S}\mathbf{z}_1) = \lambda\nabla_{\mathbf{z}_1}(\mathbf{z}_1^T\mathbf{z}_1 - 1)$$

$$\mathbf{S}\mathbf{z}_1 = \lambda\mathbf{z}_1$$

Lagrange Multipliers:
Given *f* to be optimized, subject to the constraint *g*=0, then, solve: $\nabla f = \lambda\nabla g$

Eigenvector:
direction unchanged after applying a linear transformation **S**

Therefore, $\mathbf{z}_1$ is an eigenvector of **S**, corresponding to the largest eigenvalue $\lambda = \lambda_1$

Algebraic Derivation of PCs

$$\underset{\mathbf{z}_1}{\operatorname{argmax}}\, \mathrm{var}[y_1]\,, s.t.\,, \mathbf{z}_1^T\mathbf{z}_1 = 1 \qquad \bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$$

$$\mathrm{var}[y_1] = \mathbb{E}[(y_1 - \overline{y_1})^2] = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{z}_1^T\mathbf{x}_i - \mathbf{z}_1^T\bar{\mathbf{x}})^2 = \mathbf{z}_1^T\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T\right)}_{\text{Covariance matrix } \mathbf{S}}\mathbf{z}_1$$

$$f = \mathbf{z}_1^T\mathbf{S}\mathbf{z}_1,\ g = \mathbf{z}_1^T\mathbf{z}_1 - 1$$

Let λ be a Lagrange multiplier,

$$\nabla_{\mathbf{z}_1}(\mathbf{z}_1^T\mathbf{S}\mathbf{z}_1) = \lambda\nabla_{\mathbf{z}_1}(\mathbf{z}_1^T\mathbf{z}_1 - 1)$$

$$\mathbf{S}\mathbf{z}_1 = \lambda\mathbf{z}_1$$

Therefore, $\mathbf{z}_1$ is an eigenvector of **S**, corresponding to the <u>largest</u> eigenvalue $\lambda = \lambda_1$

$$\mathrm{var}[y_1] = \mathbf{z}_1^T\mathbf{S}\mathbf{z}_1 = \mathbf{z}_1^T\lambda\mathbf{z}_1 = \lambda$$

Therefore, λ is largest eigenvalue in order to maximize $\mathrm{var}[y_1]$

<u>Lagrange Multipliers</u>:
Given *f* to be optimized, subject to the constraint *g*=0, then, solve: $\nabla f = \lambda\nabla g$

<u>Eigenvector:</u>
direction unchanged after applying a linear transformation ***S***

Algebraic Derivation of PCs

Similarly, $\mathbf{z}_2$ is also an eigenvector of **S**,

whose eigenvalue $\lambda = \lambda_2$ is the second largest

Are obtained PCs orthogonal?

In general,

$$\mathrm{var}[y_k] = \mathbf{z}_k^T \mathbf{S} \mathbf{z}_k = \lambda_k$$

- The k^{th} largest eigenvalue of **S** is the variance of the k^{th} PC projections.
- The k^{th} PC $\mathbf{z}_k$ captures the k^{th} greatest variation in the samples

Algebraic Derivation of PCs

Proof: Obtained PCs are orthogonal to each other

$$\boldsymbol{S}\boldsymbol{z}_1 = \lambda_1 \boldsymbol{z}_1, \qquad \boldsymbol{S}\boldsymbol{z}_2 = \lambda_2 \boldsymbol{z}_2, \dots \dots$$

$$\Downarrow \qquad\qquad \Downarrow$$

$$\boldsymbol{z}_2^T \boldsymbol{S}\boldsymbol{z}_1 = \lambda_1 \boldsymbol{z}_2^T \boldsymbol{z}_1 \qquad \boldsymbol{z}_1^T \boldsymbol{S}\boldsymbol{z}_2 = \lambda_2 \boldsymbol{z}_1^T \boldsymbol{z}_2$$

$$(\boldsymbol{z}_1^T \boldsymbol{S}\boldsymbol{z}_2)^T = \lambda_2 (\boldsymbol{z}_1^T \boldsymbol{z}_2)^T$$

$$\boldsymbol{z}_2^T \boldsymbol{S}^T \boldsymbol{z}_1 = \lambda_2 \boldsymbol{z}_2^T \boldsymbol{z}_1$$

$$\Downarrow \; \boldsymbol{S}^T = \boldsymbol{S}$$

$$\underbrace{\boldsymbol{z}_2^T \boldsymbol{S} \boldsymbol{z}_1}\, = \lambda_2 \boldsymbol{z}_2^T \boldsymbol{z}_1$$

$$\lambda_1 \boldsymbol{z}_2^T \boldsymbol{z}_1 = \lambda_2 \boldsymbol{z}_2^T \boldsymbol{z}_1 \Rightarrow \boldsymbol{z}_2^T \boldsymbol{z}_1 = 0$$

Compute PCs

- **Main steps for computing PCs and Projections on PCs**
  - Calculate the covariance matrix $\mathbf{S}$.
  - Compute its eigenvectors: $\mathbf{z}_i$
  - The first p eigenvectors $\{\mathbf{z}_i\}_{i=1}^{p}$ form the p PCs
  - The transformation matrix $\mathbf{Z}$ consists of the p PCs:

$(\mathbf{S} - \lambda \mathbf{I}_d)\mathbf{z}_i = 0$ ①

Solve $|\mathbf{S} - \lambda \mathbf{I}_d| = 0$, Get λ

Substitute each λ to ①, Get $\mathbf{z}_i$

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_p],$$

$$\mathbf{y} = \mathbf{Z}^T\mathbf{x}$$

$A_{d\times d} = [c_1, c_2 \ldots c_d]$

$a_1c_1 + a_2c_2 + \ldots a_dc_d = 0$

Examples: Compute PCs

Given the following dataset, compute the first PC and its corresponding projection for each data sample.

Observations	Feature 1	Feature 2
1	1	2
2	3	4
3	5	6
4	7	8

Examples: Compute PCs

n=4, d=2

Observations	Feature 1	Feature 2
1	1	2
2	3	4
3	5	6
4	7	8

Step 1: Compute Covariance Matrix **S**

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i = \frac{1}{4}\left(\begin{bmatrix}1\\2\end{bmatrix}+\begin{bmatrix}3\\4\end{bmatrix}+\begin{bmatrix}5\\6\end{bmatrix}+\begin{bmatrix}7\\8\end{bmatrix}\right) = \begin{bmatrix}4\\5\end{bmatrix}$$

$$= \frac{1}{4}\left(\left(\begin{bmatrix}1\\2\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)\left(\begin{bmatrix}1\\2\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)^T + \left(\begin{bmatrix}3\\4\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)\left(\begin{bmatrix}3\\4\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)^T + \left(\begin{bmatrix}5\\6\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)\left(\begin{bmatrix}5\\6\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)^T + \left(\begin{bmatrix}7\\8\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)\left(\begin{bmatrix}7\\8\end{bmatrix} - \begin{bmatrix}4\\5\end{bmatrix}\right)^T\right) = \begin{bmatrix}5 & 5\\5 & 5\end{bmatrix}$$

Step 2: Compute eigenvectors $\mathbf{z}_i$

$$\det(\mathbf{S} - \lambda\mathbf{I}_2) = 0$$

$$\det\left(\begin{bmatrix}5 & 5\\5 & 5\end{bmatrix} - \lambda\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix}5-\lambda & 5\\5 & 5-\lambda\end{bmatrix}\right) = 0$$

$$(5-\lambda)(5-\lambda) - 5\times 5 = 0$$

$$\lambda_1 = 10, \lambda_2 = 0$$

For $\lambda_1 = 10$, $(\mathbf{S} - \lambda\mathbf{I}_d)\mathbf{z}_1 = 0$

$$\begin{bmatrix}-5 & 5\\5 & -5\end{bmatrix}\begin{bmatrix}z_{1,1}\\z_{2,1}\end{bmatrix} = 0$$

$$z_{1,1} = z_{2,1}$$

Since $\mathbf{z}_1^T\mathbf{z}_1 = 1$, i.e., $z_{1,1}^2 + z_{2,1}^2 = 1$

$z_{1,1} = z_{2,1} = \pm\frac{1}{\sqrt{2}}$, $\mathbf{z}_1 = \pm\begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix}$, var[$y_1$]= $\lambda_1 = 10$

Step 3: Compute Projection y on $\mathbf{z}_1$

$$y_1 = \mathbf{z}_1^T\mathbf{x}_1 = \pm\left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]\begin{bmatrix}1\\2\end{bmatrix} = \pm\frac{3}{\sqrt{2}}, y_2 = \mathbf{z}_1^T\mathbf{x}_2, y_3 = \mathbf{z}_1^T\mathbf{x}_3, y_4 = \mathbf{z}_1^T\mathbf{x}_4$$

Practical Computation of PCA

- In reality, we compute the PCs via Singular Value Decomposition (SVD) on the centered data matrix.

$$\mathbf{S} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T$$

$$= \frac{1}{n}\mathbf{X}_{d\times n}\mathbf{X}_{d\times n}^T$$

$V \cdot D^T \cdot U^T$

$$= \frac{1}{n}\mathbf{U}_{d\times d}\mathbf{D}_{d\times n}\mathbf{V}_{n\times n}^T(\mathbf{U}_{d\times d}\mathbf{D}_{d\times n}\mathbf{V}_{n\times n}^T)^T$$

$$= \frac{1}{n}\mathbf{U}_{d\times d}\mathbf{D}_{d\times n}\mathbf{D}_{d\times n}^T\mathbf{U}_{d\times d}{}^T$$

Centered data matrix:

$$X_{d,n} = [(x_1 - \bar{x}), \ldots, (x_n - \bar{x})]$$

SVD of $\mathbf{X}_{d,n}$:

$$\mathbf{X}_{d\times n} = \mathbf{U}_{d\times d}\mathbf{D}_{d\times n}\mathbf{V}_{n\times n}^T$$

$$\mathbf{U}_{d\times d}\mathbf{U}_{d\times d}^T = \mathbf{U}_{d\times d}^T\mathbf{U}_{d\times d} = \mathbf{I}_d,$$
$$\mathbf{V}_{n\times n}\mathbf{V}_{n\times n}^T = \mathbf{V}_{n\times n}^T\mathbf{V}_{n\times n} = \mathbf{I}_n,$$

$$\mathbf{D}_{d\times n} = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_d & 0 & \cdots & 0 \end{bmatrix}$$

Then, the eigenvectors $\mathbf{z}_i$ of $\mathbf{S}$ are the columns of $\mathbf{U}$ and the eigenvalues λ_i are the diagonal elements of ${}^{\mathbf{DD}^T}/_n$

Practical Computation of PCA

Suppose, $\mathbf{U}_{d\times d} = [\mathbf{u}_1, \dots, \mathbf{u}_d]$, $\mathbf{D}_{d\times n} = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_d & 0 & \cdots & 0 \end{bmatrix}$, $\mathbf{DD}^T/_n = \begin{bmatrix} \frac{\sigma_1^2}{n} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\sigma_d^2}{n} \end{bmatrix}$

Prove: the eigenvectors $\mathbf{z}_i$ of $\mathbf{S}$ are the columns of $\mathbf{U}$ and the eigenvalues λ_i are the diagonal elements of $\mathbf{DD}^T/_n$

⇩

Prove: $\mathbf{Su}_i = \frac{\sigma_i^2}{n}\mathbf{u}_i$

$$\mathbf{Su}_i = (\frac{1}{n}\mathbf{U}_{d\times d}\mathbf{D}_{d\times n}\mathbf{D}_{d\times n}^T{\mathbf{U}_{d\times d}}^T)\mathbf{u}_i$$

$$= [\mathbf{u}_1, \dots, \mathbf{u}_d] \begin{bmatrix} \frac{\sigma_1^2}{n} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\sigma_d^2}{n} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_d^T \end{bmatrix} \mathbf{u}_i$$

i-th entry

$$= [\mathbf{u}_1, \dots, \mathbf{u}_d] \begin{bmatrix} \frac{\sigma_1^2}{n} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{\sigma_d^2}{n} \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = [\mathbf{u}_1, \dots, \mathbf{u}_d] \begin{bmatrix} 0 \\ \vdots \\ \frac{\sigma_i^2}{n} \\ \vdots \\ 0 \end{bmatrix} = \frac{\sigma_i^2}{n}\mathbf{u}_i$$

$$U_{d,d}U_{d,d}^T = I_d$$

$U_{d,d}$ is an orthonormal matrix

⇩

$$u_i^T u_i = 1, \quad u_j^T u_i = 0$$

Practical Computation of PCA

- The new reconstructed sample using p PCs in original space is:

$$\widehat{\mathbf{x}_{\boldsymbol{i}}} = \overline{\mathbf{x}} + \mathbf{U}_{\boldsymbol{d}\times\boldsymbol{p}}\mathbf{U}_{\boldsymbol{d}\times\boldsymbol{p}}^{\boldsymbol{T}}(\mathbf{x}_{\boldsymbol{i}} - \overline{\mathbf{x}})$$

Step 1: project the original sample $\mathbf{x}_i$ onto the p PCs

$$\mathbf{y} = \mathbf{U}_{d\times p}^{T}(\mathbf{x}_i - \overline{\mathbf{x}})$$

Step 2: map **y** back to the original d-dimensional space

$$\widehat{\mathbf{x}_i^c} = \mathbf{U}_{d\times p}\mathbf{y} = \mathbf{U}_{d\times p}\mathbf{U}_{d\times p}^{T}(\mathbf{x}_i - \overline{\mathbf{x}})$$

$$\widehat{\mathbf{x}_i} = \widehat{\mathbf{x}_i^c} + \overline{\mathbf{x}} = \mathbf{U}_{d\times p}\mathbf{U}_{d\times p}^{T}(\mathbf{x}_i - \overline{\mathbf{x}}) + \overline{\mathbf{x}}$$

Visualize PCs

Eigenface images

Raw input

KxK

$ave(r, g, b)$

grayscale

Flatten

1 2 ... K^2

Single feature vector

d=K^2

n images, subtract mean

$\mathbf{X}_{d \times n}$

SVD

$\mathbf{U}_{d \times p}$

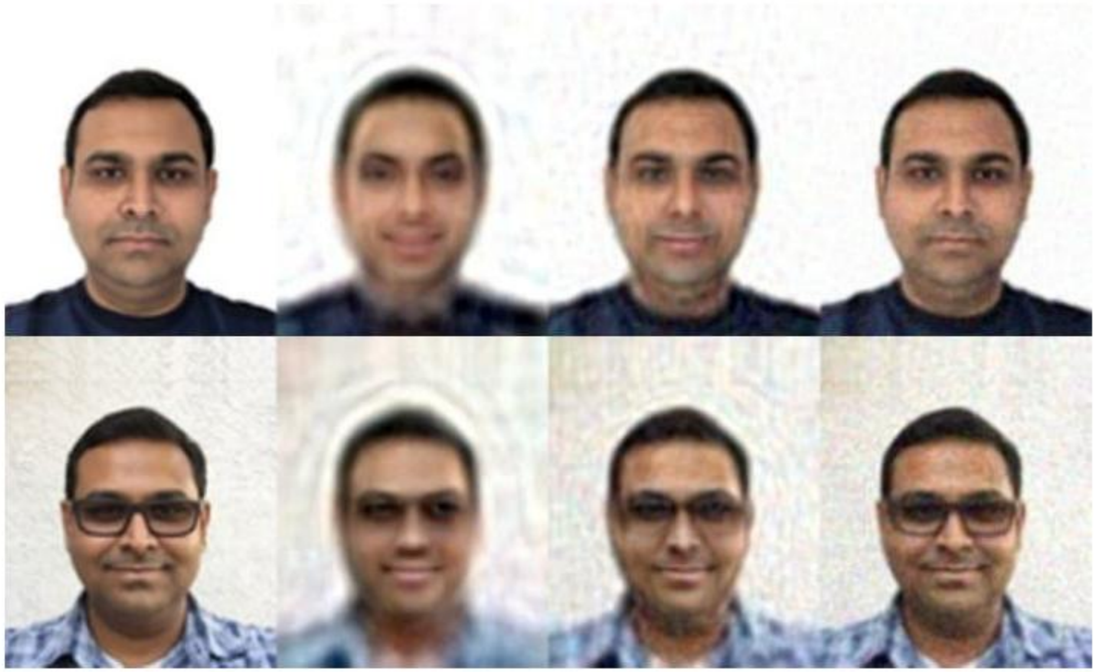
Reshape each eigenvector $\mathbf{u}_i$ *into KxK*

p eigenfaces

Reconstruction with PCs

Image Dimension:
100x100x3

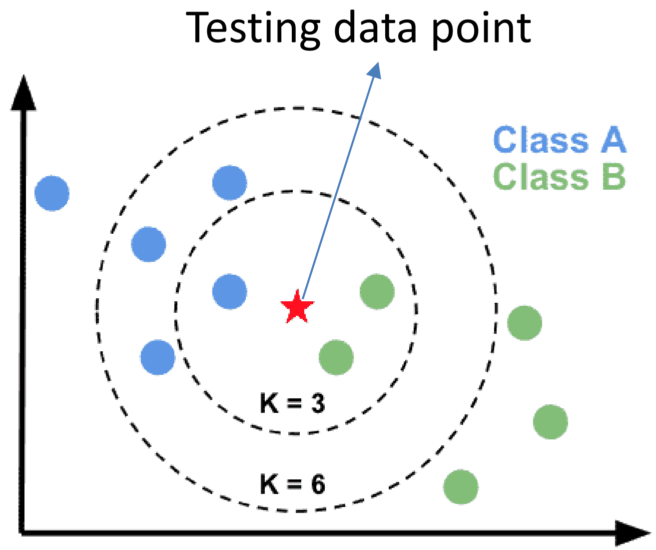
Original | 250 Eigenfaces | 1000 Eigenfaces | 4000 Eigenfaces



$$\widehat{\mathbf{x}}_i = \overline{\mathbf{x}} + \mathbf{U}_{d\times p}\mathbf{U}_{d\times p}^T(\mathbf{x}_i - \overline{\mathbf{x}})$$

PCA and Classification

- Classification with PCA
  - Compute PCs using training dataset
  - Project both training and testing data into the obtained PCs space
  - For each testing data point, use K-Nearest-Neighbor (KNN) for classification
  - Issue: accuracy is sensitive to the number of PCs



How to determine the number of PCs?

To choose p based on percentage of variation to retain, we can use the following criterion (smallest p):

$$\frac{\sum_1^p \lambda_i}{\sum_1^d \lambda_i} \geq Threshold\ (e.g., 0.95)$$

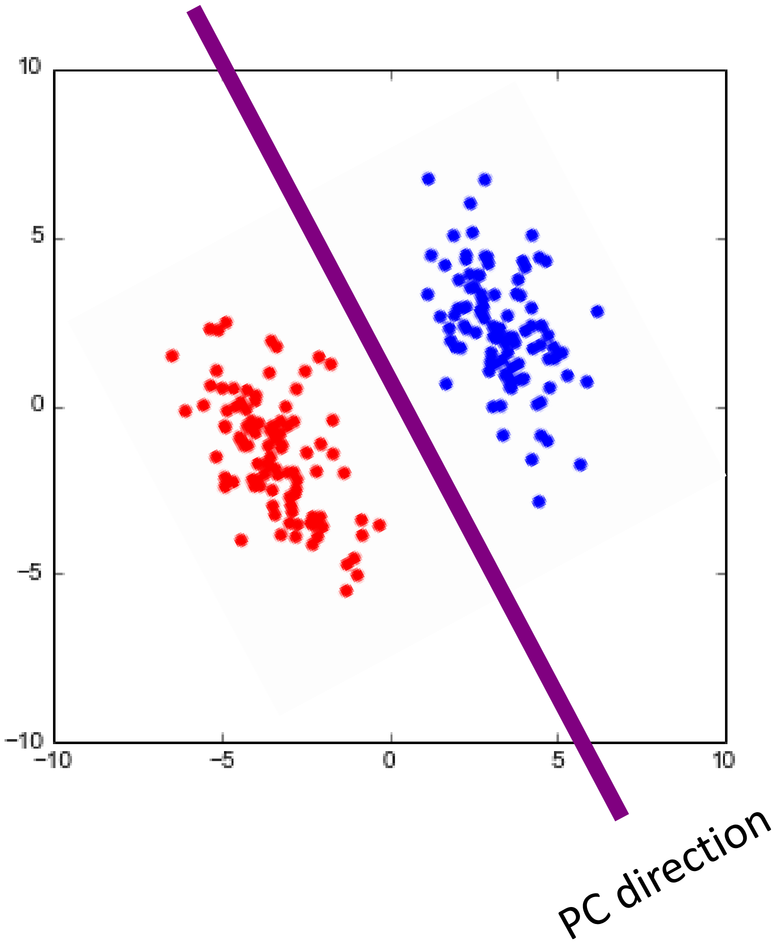
PCA Remarks

- PCA
  - finds orthonormal basis for data
  - Sorts dimensions in order of “importance”
  - Discard low significance dimensions

- Uses:
  - Get compact description
  - Ignore noise
  - Improve classification (hopefully)

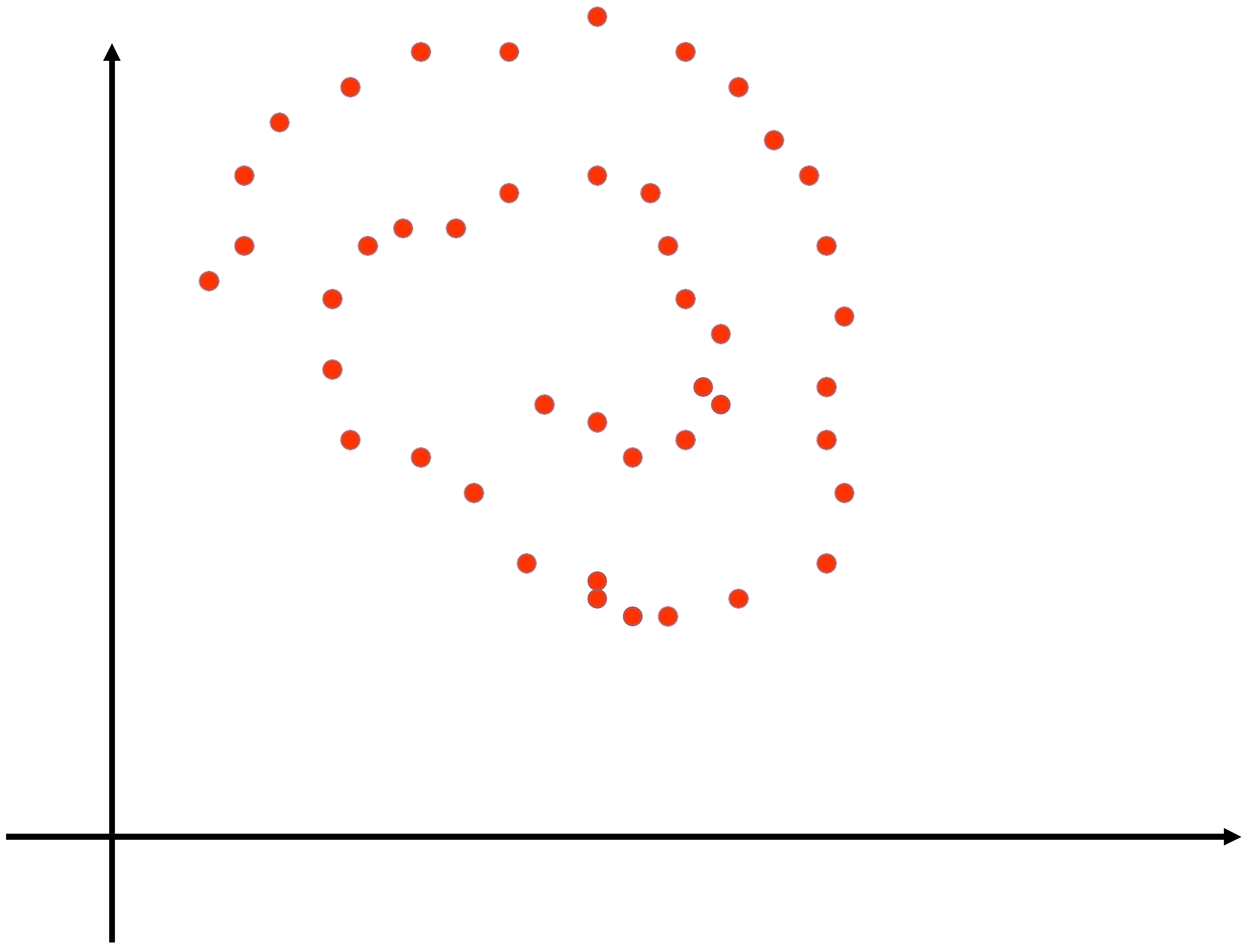
PCA Limitations

- PCA is not always an optimal feature extraction for classification
  - PCA is based on the sample covariance, irrespective of class-membership
  - The projection axes chosen by PCA might not provide good discrimination power



PCA Limitations

- PCA cannot capture NON-LINEAR structure



Note: Curvilinear Component Analysis can solve this case. Study this work if you are interested.

PCA Limitations

- Lack of interpretability

  - Each PC projection is a linear combination of **all** input features

$$y_k = \mathbf{z}_k^T \mathbf{x}$$

What does each PC actually represent?

e.g., Face Recognition

Not localized to a single feature such as “mouth”, “eyebrows”

e.g., Topic Modelling from Text

A PC projection is a linear combination of occurrence of many words

$$\text{PC=}\begin{bmatrix}0.4\\-0.3\\0.2\\-0.1\end{bmatrix}\begin{array}{l}\longrightarrow \text{“war”}\\ \longrightarrow \text{“education”}\\ \longrightarrow \text{“government”}\\ \longrightarrow \text{“poem”}\end{array}$$

PCA Limitations

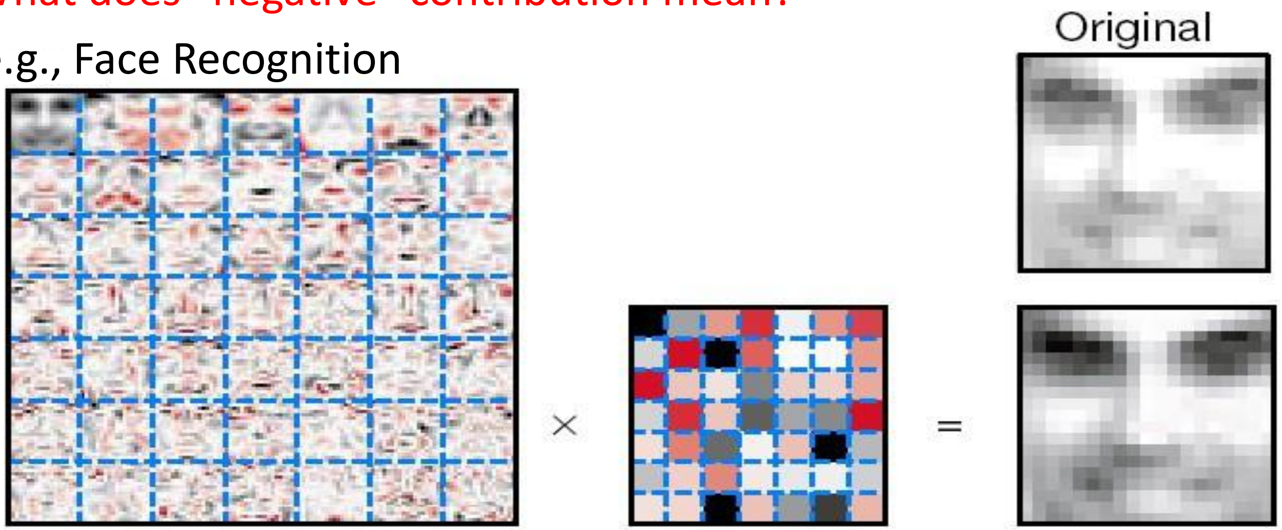
- Lack of interpretability

$$\mathbf{x} = a_1 \boldsymbol{z}_1 + \cdots + a_p \boldsymbol{z}_p$$

- Both PCs and coefficients can be positive or negative

What does “negative” contribution mean?

e.g., Face Recognition



PCs (Eigenfaces) Coefficients Reconstructed

Every vector can be expressed as the linear combination of basis vectors

$$\begin{bmatrix}2\\6\end{bmatrix} = 2\begin{bmatrix}1\\0\end{bmatrix} + 6\begin{bmatrix}0\\1\end{bmatrix}$$
$$= 3\begin{bmatrix}1\\4\end{bmatrix} + (-1)\begin{bmatrix}1\\6\end{bmatrix}$$

what does “subtracting an eigenface” mean?

e.g., Topic Modelling from Text

$$PC=\begin{bmatrix}0.4\\-0.3\\0.2\\-0.1\end{bmatrix}\begin{array}{l}\rightarrow \text{“war”}\\ \rightarrow \text{“education”}\\ \rightarrow \text{“government”}\\ \rightarrow \text{“poem”}\end{array}$$

what does a **“negative occurrence” of a word** mean?

Unsupervised Feature Learning

Non-negative Matrix Factorization

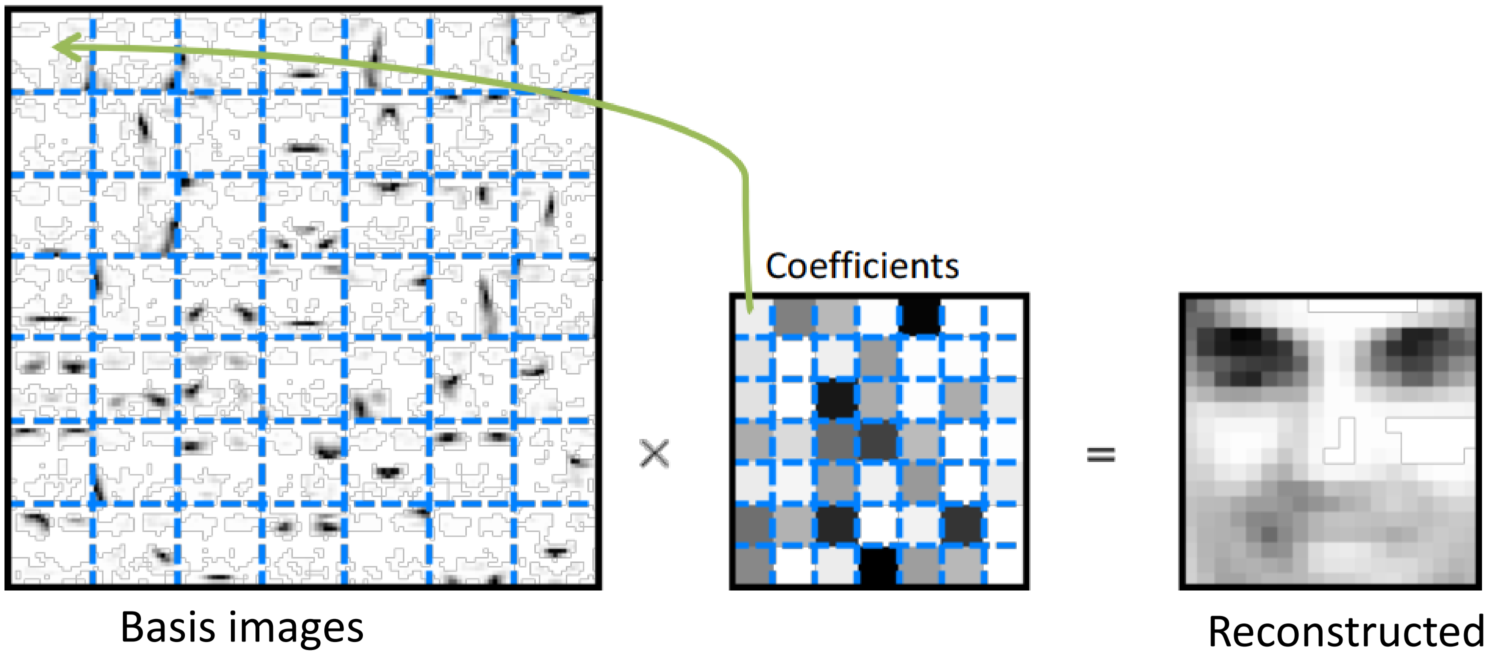
(non-negative coefficients and basis vectors)

NMF: Motivation

- Non-negative coefficients and basis vectors

  -More intuitive for non-negative datasets (e.g., images, word counts in documents, etc.)

  -Leads to basis vectors that represent parts



NMF: Mathematical Formulation

- Matrix factorization: **V**≈**WH** All entries in **V**, **W**, **H**≥0
  - **V**: (d×n) Input data matrix, where *d* is the number of input features, *n* is the number of samples. $\mathbf{V}_{d\times n} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$
  - **W**: (d×r) Basis matrix, where *r* is the number of basis vectors.

$$\mathbf{W}_{d\times r} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_r]$$

  - **H**: (r×n) Coefficient matrix, each column of **H** is called encoding.

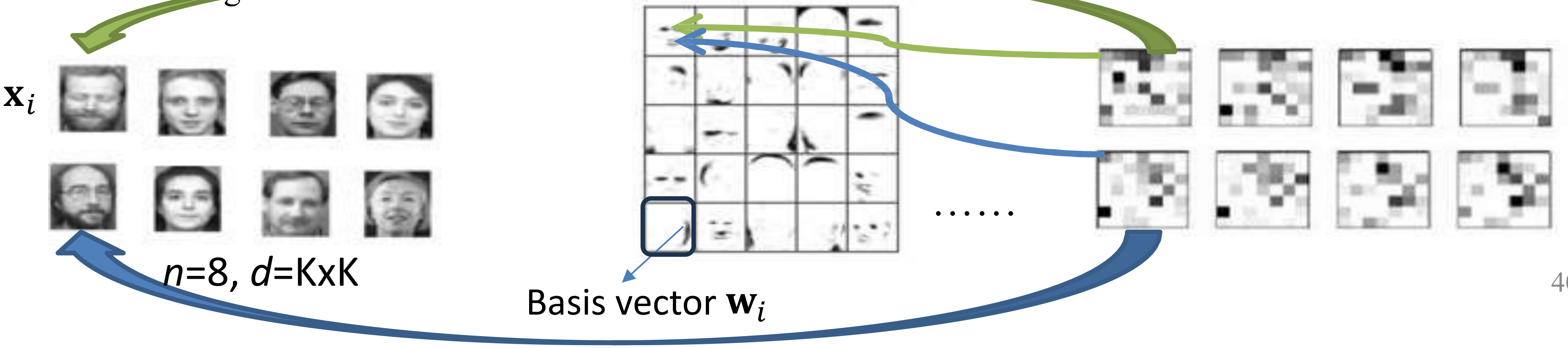
$$\mathbf{H}_{r\times n} = \begin{array}{c} \mathbf{h}_1 \\ \begin{bmatrix} \boxed{\begin{matrix} h_{1,1} \\ \vdots \\ h_{r,1} \end{matrix}} & \begin{matrix} \cdots \\ \ddots \\ \cdots \end{matrix} & \begin{matrix} h_{1,n} \\ \vdots \\ h_{r,n} \end{matrix} \end{bmatrix} \end{array}$$

$$\mathbf{x}_i = h_{1,i}\mathbf{w}_1 + h_{2,i}\mathbf{w}_2 + \cdots + h_{r,i}\mathbf{w}_r$$
$$= \mathbf{W}\mathbf{h}_i$$

V: Each column is a flattened vectorized image

W: Each column is a **non-negative** basis image

H: Each column is a **non-negative** coefficient vector



NMF: Objective Function

$$\min_{\mathbf{W},\mathbf{H}} \frac{1}{2} \boxed{\|\mathbf{V} - \mathbf{WH}\|^2}, \text{ s.t. } \mathbf{W} \geq 0, \mathbf{H} \geq 0$$

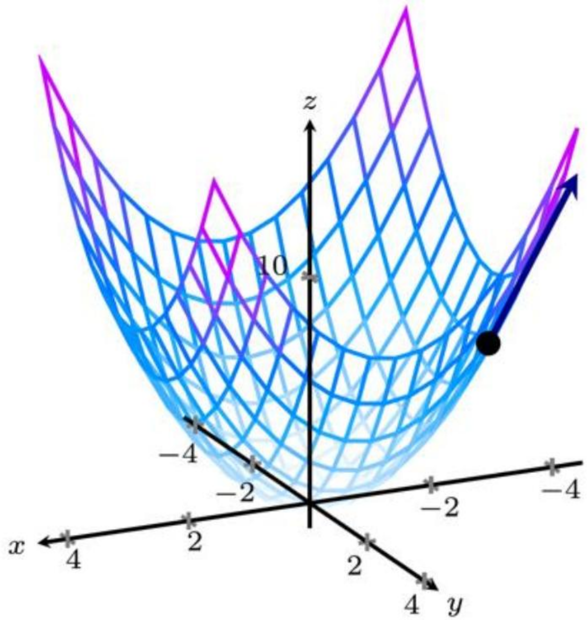
$$\sum_{i,j} (\mathbf{V}_{i,j} - [\mathbf{WH}]_{i,j})^2$$

No closed form solution!

Gradient Descent!

Recall Goal: $\operatorname*{argmin}_{\theta} J(\theta)$

Initialize θ_0 and learning rate η
while *true* **do**
    $\theta_{k+1} \leftarrow \theta_k - \eta \boxed{\nabla_\theta J(\theta_k)}$
    if *converge* **then**
        return θ_{k+1}
    end
end



NMF: Gradient Descent Solution

$$\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{V}-\mathbf{W}\mathbf{H}\|^2, \text{ s.t. } \mathbf{W} \ge 0, \mathbf{H} \ge 0$$

No closed form solution!

Gradient Descent!

$$J(\mathbf{W},\mathbf{H}) = \frac{1}{2}\|\mathbf{V}-\mathbf{W}\mathbf{H}\|^2 = \frac{1}{2}\sum_{i,j}(\mathbf{V}_{i,j} - \sum_{l=1}^{r}\mathbf{W}_{i,l}\,\mathbf{H}_{l,j})^2$$

$$\nabla_{\mathbf{W}_{p,q}} J(\mathbf{W},\mathbf{H}) = \sum_{i,j}(\mathbf{V}_{i,j} - \sum_{l=1}^{r}\mathbf{W}_{i,l}\,\mathbf{H}_{l,j})(-\frac{\partial}{\partial \mathbf{W}_{p,q}}\sum_{l=1}^{r}\mathbf{W}_{i,l}\,\mathbf{H}_{l,j})$$

Only when $i=p$, $l=q$, $\mathbf{W}_{i,l}$ depends on $\mathbf{W}_{p,q}$

$$\nabla_{\mathbf{X}} f = \frac{df(\mathbf{X})}{d\mathbf{X}} = \begin{bmatrix} \frac{\partial f}{\partial x_{1,1}} & \cdots & \frac{\partial f}{\partial x_{1,K}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial x_{d,1}} & \cdots & \frac{\partial f}{\partial x_{d,K}} \end{bmatrix}$$

$$= -\sum_{j}(\mathbf{V}_{p,j} - \sum_{l=1}^{r}\mathbf{W}_{p,l}\,\mathbf{H}_{l,j})\mathbf{H}_{q,j}$$

$$= \sum_j((\mathbf{W}\mathbf{H})_{p,j} - \mathbf{V}_{p,j})\mathbf{H}_{q,j} = \sum_j(\mathbf{W}\mathbf{H}-\mathbf{V})_{p,j}\mathbf{H}^T_{j,q}$$

$$= \left((\mathbf{W}\mathbf{H}-\mathbf{V})\mathbf{H}^{\mathrm{T}}\right)_{p,q}$$

$$\nabla_{\mathbf{W}} J(\mathbf{W},\mathbf{H}) = (\mathbf{W}\mathbf{H}-\mathbf{V})\mathbf{H}^{\mathrm{T}}$$

Similarly, $\nabla_{\mathbf{H}} J(\mathbf{W},\mathbf{H}) = \mathbf{W}^{\mathrm{T}}(\mathbf{W}\mathbf{H}-\mathbf{V})$

NMF: Gradient Descent Solution

Procedures:

Step 1. Initialize non-negative matrices **W, H**

Step 2. Iteratively update **W, H** through gradient descent until convergence

$$\mathbf{H}_{new} \leftarrow \mathbf{H} - \eta_{\mathbf{H}} \mathbf{W}^{\mathrm{T}} (\mathbf{W}\mathbf{H} - \mathbf{V})$$

$$\mathbf{W}_{new} \leftarrow \mathbf{W} - \eta_{\mathbf{W}} (\mathbf{W}\mathbf{H}_{new} - \mathbf{V}) \mathbf{H}_{new}{}^{\mathrm{T}}$$

How to ensure non-negativity?

Method 1:

After each update, clip any negative values

$$\mathbf{W} \leftarrow \max(\mathbf{W}, \mathbf{0}), \ \mathbf{H} \leftarrow \max(\mathbf{H}, \mathbf{0})$$

Method 2:

Set $\eta_{\mathbf{H}_{p,q}} = \frac{\mathbf{H}_{p,q}}{[\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H}]_{p,q}}, \ \eta_{\mathbf{W}_{p,q}} = \frac{\mathbf{W}_{p,q}}{[\mathbf{W}\mathbf{H}_{new}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}},$

$$(\mathbf{H}_{new})_{p,q} \leftarrow \mathbf{H}_{p,q} \frac{[\mathbf{W}^{\mathrm{T}}\mathbf{V}]_{p,q}}{[\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H}]_{p,q}}, \ (\mathbf{W}_{new})_{p,q} \leftarrow \mathbf{W}_{p,q} \frac{[\mathbf{V}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}}{[\mathbf{W}\mathbf{H}_{new}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}}.$$

Multiplicative Update

Proposed by D. Lee and H. Seung (NIPS 2000)

NMF: Multiplicative Update

Procedures:

Step 1. Initialize non-negative matrices **W, H**

Step 2. Iteratively update **W, H** until convergence

$$(\mathbf{H}_{new})_{p,q} \leftarrow \mathbf{H}_{p,q} \frac{[\mathbf{W}^{\mathrm{T}}\mathbf{V}]_{p,q}}{[\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H}]_{p,q}}$$

$$(\mathbf{W}_{new})_{p,q} \leftarrow \mathbf{W}_{p,q} \frac{[\mathbf{V}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}}{[\mathbf{W}\mathbf{H}_{new}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}}$$

Global Minimum?

Examples: Compute W, H

Suppose r = 2,

W is initialized to $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$,

H is initialized to $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

Observations	Feature 1	Feature 2
1	1	2
2	3	4
3	5	6
4	7	8

What are **H** and **W** after implementing 1 iteration of multiplicative update?
What is the corresponding reconstructed data matrix **V**?

Examples: Compute W, H

Suppose r = 2,

W is initialized to $\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$,

H is initialized to $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

n=4, d=2

Observations	Feature 1	Feature 2
1	1	2
2	3	4
3	5	6
4	7	8

1st Iteration:

$$(\mathbf{H}_{new})_{p,q} \leftarrow \mathbf{H}_{p,q} \frac{[\mathbf{W}^{\mathrm{T}}\mathbf{V}]_{p,q}}{[\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{H}]_{p,q}}$$

$$\mathbf{H}_{new} \leftarrow \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \times \frac{\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}}{\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}} = \begin{bmatrix} 0.56 & 1.22 & 1.89 & 2.56 \\ 0.44 & 1.11 & 1.78 & 2.44 \end{bmatrix}$$

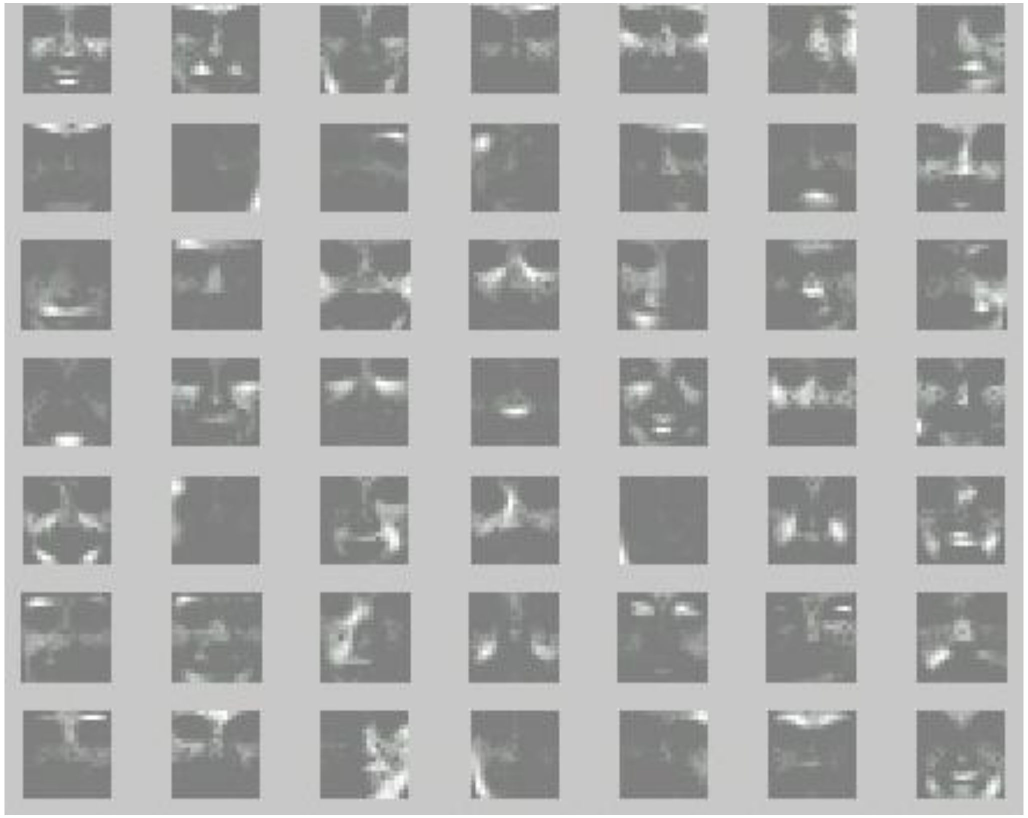
$$\mathbf{V}_{reconst} = \mathbf{W}_{new}\,\mathbf{H}_{new} = \begin{bmatrix} 1.33 & 3.17 & 5.01 & 6.85 \\ 1.68 & 3.84 & 5.99 & 8.15 \end{bmatrix}$$

$$(\mathbf{W}_{new})_{p,q} \leftarrow \mathbf{W}_{p,q} \frac{[\mathbf{V}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}}{[\mathbf{W}\mathbf{H}_{new}\mathbf{H}_{new}{}^{\mathrm{T}}]_{p,q}}$$

$$\mathbf{W}_{new} \leftarrow \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \times \frac{\begin{bmatrix} 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{bmatrix}\begin{bmatrix} 0.56 & 0.44 \\ 1.22 & 1.11 \\ 1.89 & 1.78 \\ 2.56 & 2.44 \end{bmatrix}}{\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 0.56 & 1.22 & 1.89 & 2.56 \\ 0.44 & 1.11 & 1.78 & 2.44 \end{bmatrix}\begin{bmatrix} 0.56 & 0.44 \\ 1.22 & 1.11 \\ 1.89 & 1.78 \\ 2.56 & 2.44 \end{bmatrix}} = \begin{bmatrix} 0.92 & 1.84 \\ 2.16 & 1.08 \end{bmatrix}$$

Observations	Basis 1	Basis 2
1		
2		
3		
4		

Visualize Basis Images



NMF basis images

Raw input

KxK

$ave(r, g, b)$

grayscale

Flatten

1 2 ... K^2

Single feature vector

d=K^2

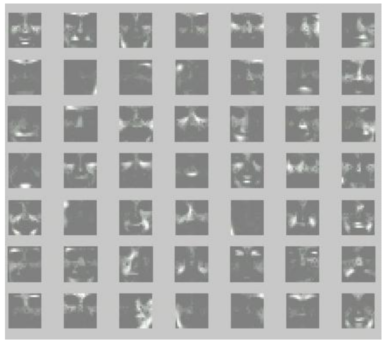
n images

$V_{d \times n}$

NMF: $Multiplicative\ Update$

$\mathbf{W}_{d \times r}$

Reshape each basis vector into KxK



r basis images

Reconstruction from W

Suppose we want to reconstruct the *i*-th image among the *n* samples

$$\widehat{x_i} = \mathbf{WH}_{(:,i)}$$

Original

Image Dimension: 549x767

r=10

r=20

r=40

r=80

r=100

r=150

r=200

r=250

Discussions

- For a new image, how to obtain the reconstruction coefficients?

- How to use NMF for classification?

- How to choose *r*

Summary

	PCA	NMF
Representation	Holistic	Part-based
Basis Vector	(PCs) Globalized; Orthogonal; Sign ambiguity	(**W**) Localized; Nonnegative
Features	(PC projections) Sign ambiguity	(**H**) Nonnegative
Optimization	Global Optimum (Eigenvalue Problem)	Local Optimum (Multiplicative Update)

Papers to Read and Self-Study

- D. Lee and H. Seung. Algorithms for Non-negative Matrix Factorization NIPS (2000).

- ICA (Independent Component Analysis): http://en.wikipedia.org/wiki/Independent_component_analysis

- CCA (Canonical Correlation Analysis): http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.101.6359&rep=rep1&type=pdf