Artificial Learning Models

Lecture 08 : Dimensionality reduction

By : Dr. Lamri SAYAD

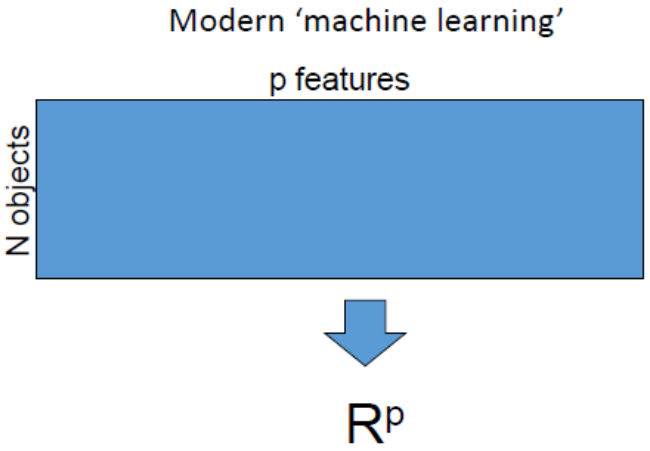
2024

Agenda

- Introduction
- Dimensionality reduction

Introduction

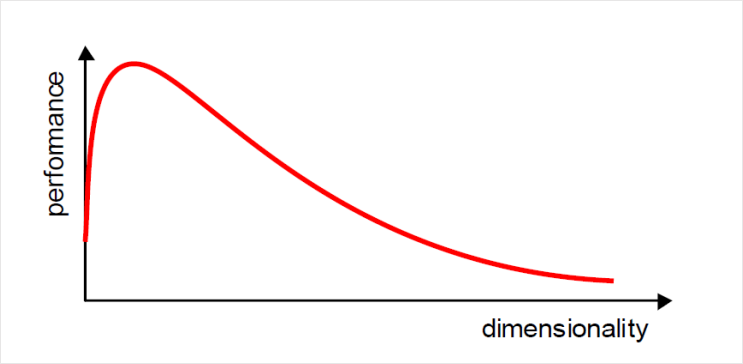
Introduction
Modern data



BIG DATA: N >> 1
WIDE DATA: p>>N
REAL-WORLD BIG DATA: p>>N>>1 (most frequently)

Introduction

- In many machine learning problems, Datasets have a high dimensionality D (huge number of features).
- **Examples** :
  - Images: for an 1800 x 1200 color image, $D \cong 6.5$ million.
  - Signals: for a 1-second acoustic voice signal sampled at 5kHz, $D = 5{,}000$.
  - Genomics: thousands of genes, millions of DNA polymorphisms.
- Increasing the number of features will not always improve the accuracy of the model.



Introduction

- In practice, the inclusion of more features might actually lead to worse performance.
- The number of training examples required increases exponentially with dimensionality **D** (i.e., k^D).
- **Problems**:
  - There is typically insufficient training data to learn a probabilistic model in such a high-dimensional space.
  - Training a ML model with such high dimension dataset requires lot of computational resources and will consume lot of time.
  - Model complexity
- **Solution** : Reduce the number of features ⟹ Dimensionality reduction

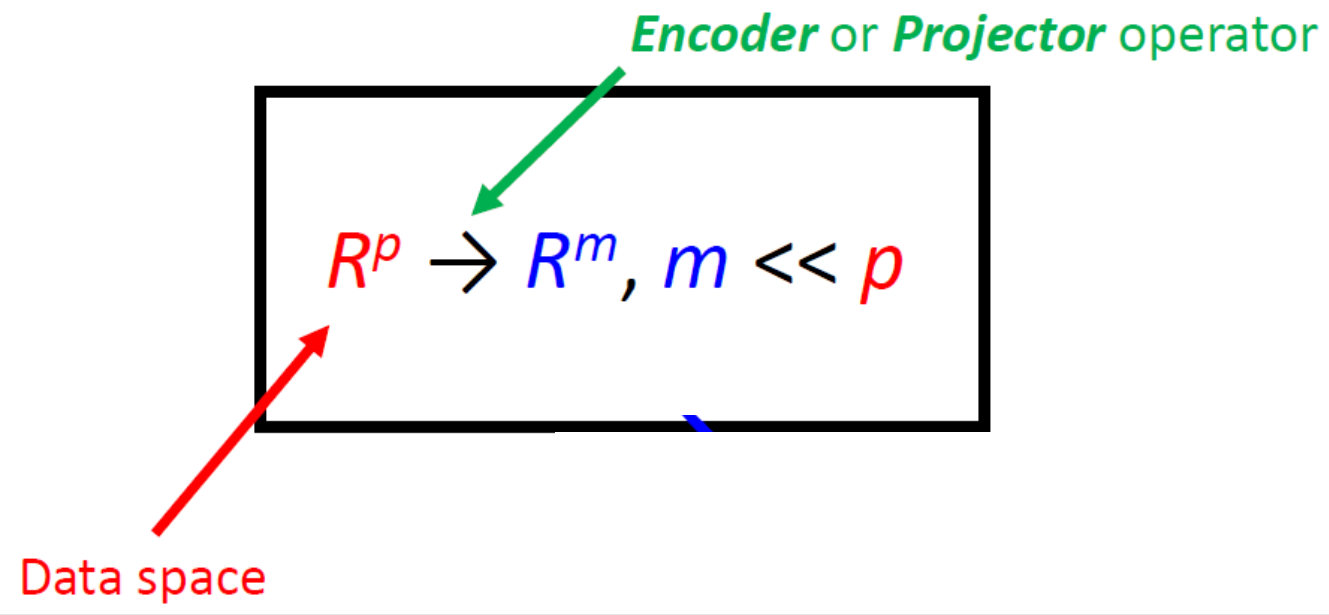
Why reduce dimensions?

1. Reduces time complexity: Less computation
2. Reduces space complexity: Less parameters
3. Saves the cost of observing the feature
4. Simpler models are more robust on small datasets
5. More interpretable; simpler explanation
6. Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

Dimensionality reduction

What is dimensionality reduction?

- Dimensionality reduction attempts to distill higher dimensional data down to a smaller number of dimensions, while preserving as much of the variance in the data as possible.



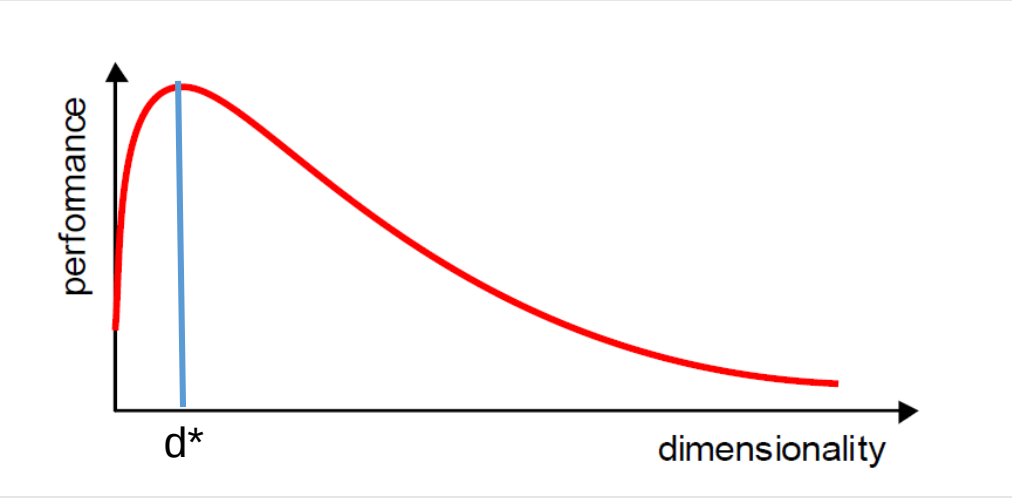
Ambient (total) and Intrinsic dimensionality of data

- *p*= ambient dimensionality (number of variables after data preprocessing)
- Intrinsic dimensionality (ID): ‘how many variables are needed to generate a good approximation of the data’
- *m* should be close to intrinsic dimensionality

$$R^p \rightarrow R^m,\ m << p$$

Dimensionality reduction : Objective

- What is the objective?
  - Choose an optimum set of features d* of lower dimensionality to improve classification accuracy.



Dimensionality Reduction: Methods classification

- Methods are commonly divided into
  - linear
  - and nonlinear approaches.
- Approaches can also be divided into
  - feature selection
  - feature extraction.

Linear dimensionality reduction

- Linearly project n-dimensional data onto a k-dimensional space
  - K **<< n**
- There are infinitely many k-dimensional subspaces we project the data onto.
- Which one to choose ???

Linear dimensionality reduction

- Best k-dimensional subspace for projection depends on task
  - Classification: maximize separation among classes
    - Example: linear discriminant analysis (LDA)
  - Regression: maximize correlation between projected data and response variable
    - Example: partial least squares (PLS)
  - Unsupervised: retain as much data variance as possible
    - Example: principal component analysis (PCA)

Dimensionality Reduction :
Feature selection vs feature extraction

- **Feature selection** :
  - focuses on the most informative variables, where 'informative' is with respect to the problem to be solved
  - Choosing $k<d$ important features, ignoring the remaining $d - k$.
  - *Algorithms* : Subset selection algorithms
- **Feature extraction (construction)** :
  - create a set of fewer variables (features), each of which would be a function (linear or non-linear) of the initial variables
  - Project the original x_i , i =1,...,d dimensions to new $k<d$ dimensions, z_j , j =1,...,k
  - *Algorithms* : Principal components analysis (PCA), linear discriminant analysis (LDA), factor analysis (FA)

Dimensionality Reduction :
Feature selection vs feature extraction

Feature extraction: computes a new set of features from the original features through some transformation f() .

f() could be linear or non-linear

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix} \xrightarrow{f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

K << D

Feature selection: chooses a subset of the original features.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_K} \end{bmatrix}$$

K << D

Dimensionality Reduction :
feature extraction

- Linear transformations are particularly attractive because they are simpler to compute and analytically tractable.
- Given x ϵ RD, find an K x D matrix T such that:

$$y = Tx \in R^K \text{ where } K \ll D$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix} \xrightarrow{T,\ f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_K \end{bmatrix}$$

This is a projection transformation from D dimensions to K dimensions.

Each new feature y_i is a linear combination of the original features x_i

Dimensionality Reduction :
feature extraction

- From a mathematical point of view, finding an optimum mapping $\mathbf{y}=f(\mathbf{x})$ can be formulated as an optimization problem (i.e., minimize or maximize an **objective** criterion).

- Commonly used objective criteria:

  - Minimize Information Loss: projection in the lower-dimensional space preserves as much **information** in the data as possible.

  - Maximize Discriminatory Information: projection in the lower-dimensional space increases **class separability**.

Dimensionality Reduction :
feature extraction

- Popular linear feature extraction methods:
  - Principal Components Analysis (PCA): Seeks a projection that **minimizes information loss**.
  - Linear Discriminant Analysis (LDA): Seeks a projection that **maximizes discriminatory information.**

- Many other methods:
  - Making features as independent as possible (Independent Component Analysis).
  - Retaining interesting directions (Projection Pursuit).
  - Embedding to lower dimensional manifolds (Isomap, Locally Linear Embedding).

Dimensionality Reduction :
vector representation

- A vector **x** ∈ R^D can be represented by D components:

$$\mathbf{x}: \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix}$$

- Assuming the standard base <v_1, v_2, …, v_D> (i.e., unit vectors in each dimension), x_i can be obtained by projecting **x** along the v_i direction:

$$x_i = \frac{\mathbf{x}^T v_i}{v_i^T v_i} = \mathbf{x}^T v_i$$

x_i are called projection coefficients

- **x** can be "reconstructed" from its projection coefficients:

$$\mathbf{x} = \sum_{i=1}^{D} x_i v_i = x_1 v_1 + x_2 v_2 + \ldots + x_D v_D$$

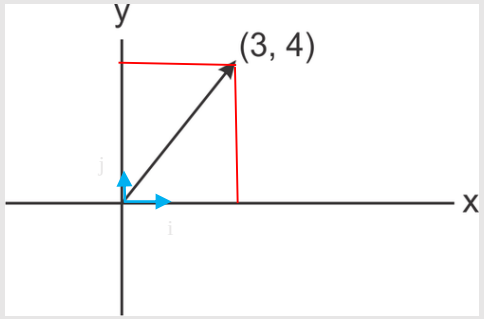
x_i are called expansion coefficients in this context

- In summary, the components of a vector are associated with a specific base; if the base is changed, the components will change too (essentially a change of coordinate systems).

Dimensionality Reduction :
vector representation

- **Example** assuming D=2:

$$\mathbf{x}: \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$$



- Assuming the standard base <v_1=i, v_2=j>, x_i can be obtained by projecting x along the direction of v_i:

$$x_1 = \mathbf{x}^T i = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 3$$

$$x_2 = \mathbf{x}^T j = \begin{bmatrix} 3 & 4 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 4$$

- **x** can be "reconstructed" from its projection coefficients as follows:

$$\mathbf{x} = 3i + 4j$$

Principle Component Analysis
PCA

PCA

- PCA stands for principal components analysis
- Unsupervised learning algorithm
- PCA is a dimensionality reduction technique
- Require feature scaling
- Filter noisy dataset
- Improve model performance
- Non-parametric statistic technique

PCA

- Find a set of linear transformations (projection) of the original variables which could describe most of the variance using a relatively fewer number of variables.
- It is usual to keep only the first few principal components that may contain 95% or more of the variance of the original data set.
- PCA is useful when there are too many independent variables and they show high correlation.

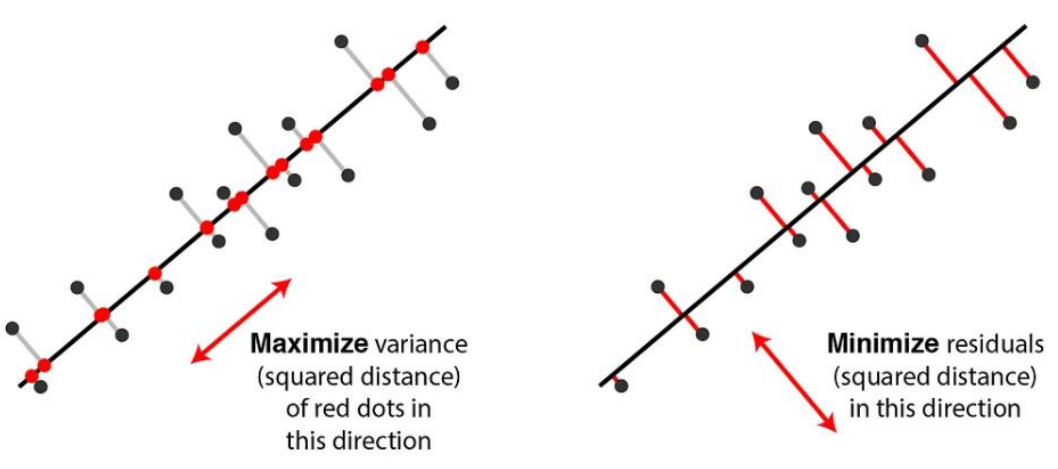
PCA

How does PCA work?

- PCA is a powerful tool for reducing the dimensionality of data while preserving its essential information. The goal is to get the best principle components that capture maximum variance.
- The projection of $\boldsymbol{x}$ on the direction of $\boldsymbol{w}$ is: $z = \boldsymbol{w}^T\boldsymbol{x}$
- Find $\boldsymbol{w}$ such that Var(z) is maximized

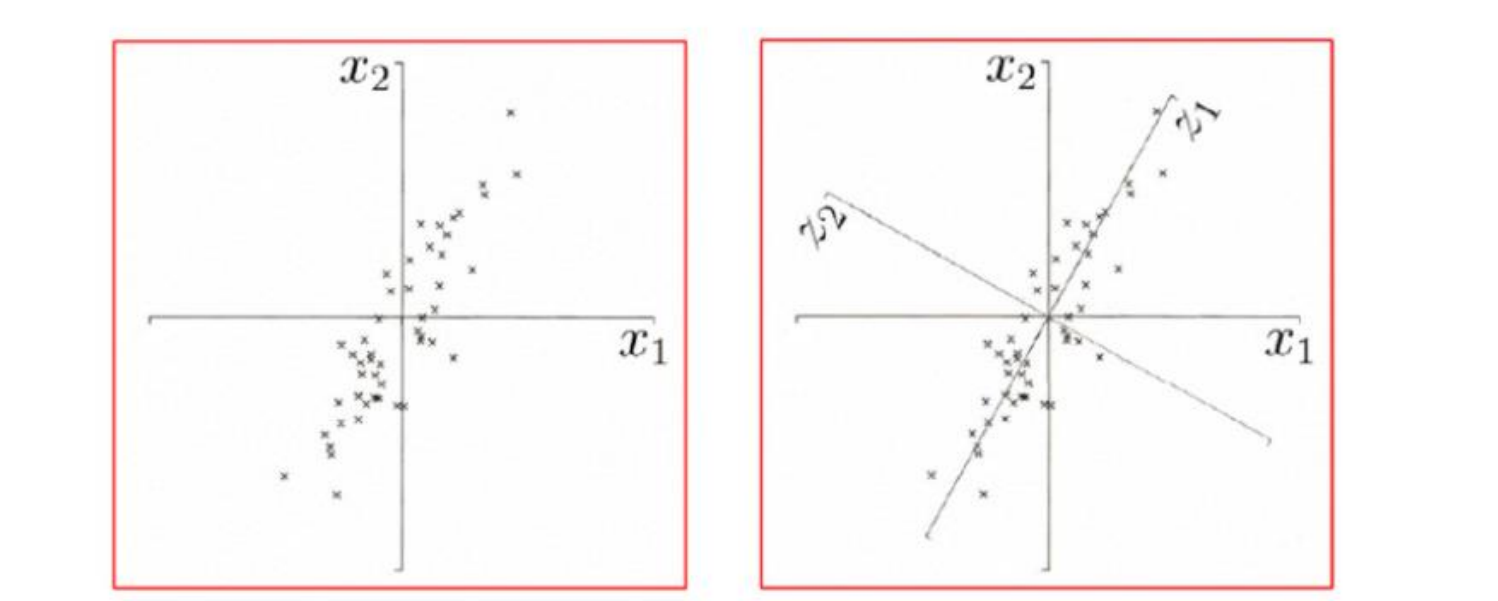
$$\begin{aligned}\text{Var}(z) &= \text{Var}(\boldsymbol{w}^T\boldsymbol{x}) = E[(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu})^2] \\ &= E[(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu})(\boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{\mu})] \\ &= E[\boldsymbol{w}^T(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T\boldsymbol{w}] \\ &= \boldsymbol{w}^T E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]\boldsymbol{w} = \boldsymbol{w}^T \sum \boldsymbol{w}\end{aligned}$$

where $\text{Var}(\boldsymbol{x}) = E[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T] = \sum$



PCA

Geometric picture of principal components (PCs)



PCA

How does PCA work?

- **Standardization**: Scale the features to have a mean of 0 and a standard deviation of 1 (optional but recommended).
- **Compute Covariance Matrix**: This captures how the features vary together.
- **Compute Eigenvectors and Eigenvalues**: Eigenvectors represent the directions of greatest variance, and eigenvalues tell you how much variance each eigenvector explains.
- **Select Principal Components**: Choose the top *k* eigenvectors based on their corresponding eigenvalues to form the new feature subspace.
- **Projection**: Project the original data onto the new feature subspace formed by the selected eigenvectors. Or project the original data points onto the new principal component axes.

PCA
main ideas

- Any $x \in R^D$ can be written as a linear combination of an orthonormal set of D basis vectors $<v_1, v_2, \ldots, v_D>$, $v_i \in R^D$ (e.g., using the standard base):

$$v_i^T v_j = \begin{cases} 1 & if \ i = j \\ 0 & otherwise \end{cases} \qquad \mathbf{x} = \sum_{i=1}^{D} x_i v_i = x_1 v_1 + x_2 v_2 + \ldots + x_D v_D \qquad where \quad x_i = \frac{\mathbf{x}^T v_i}{v_i^T v_i} = \mathbf{x}^T v_i$$

$$\mathbf{x}: \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_D \end{bmatrix}$$

- PCA seeks to approximate x in a subspace of R^D using a new set of K<<D basis vectors $<u_1, u_2, \ldots, u_K>$, $u_i \in R^D$:

$$\hat{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K \qquad where \quad y_i = \frac{\mathbf{x}^T u_i}{u_i^T u_i} = \mathbf{x}^T u_i$$

$$u_i^T u_j = \begin{cases} 1 & if \ i = j \\ 0 & otherwise \end{cases}$$

such that $\|X - \hat{X}\|$ is minimized!

$$\hat{\mathbf{x}}: \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

PCA
main ideas

- The "optimal" set of basis vectors $<u_1, u_2, \ldots, u_K>$ can be found as follows (we'll explain the details later):

  1) Find the eigenvectors u_i of the covariance matrix Σ_x of the (training) data (i.e., typically, D distinct eigenvectors)

     $\Sigma_x u_i = \lambda_i u_i$ (reminder: u_i form an orthogonal basis)

  2) Choose the K "largest" eigenvectors u_i (i.e., corresponding to the K "largest" eigenvalues λ_i)

$<u_1, u_2, \ldots, u_K>$ form to the "optimal" basis!

- We refer to the "largest" eigenvectors u_i as principal components.

PCA
Steps

- Suppose we are given $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ (D x 1) vectors

D: # of features

M: # data

Step 1: compute sample mean

$$\bar{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}\mathbf{x}_i$$

Step 2: subtract sample mean (i.e., center the data at zero)

$$\Phi_i = \mathbf{x}_i - \bar{\mathbf{x}}$$

Step 3: compute the sample covariance matrix Σ_x

$$\Sigma_{\mathbf{x}} = \frac{1}{M}\sum_{i=1}^{M}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = \frac{1}{M}AA^T$$

where $A=[\Phi_1\ \Phi_2\ \ldots\ \Phi_M]$

i.e., the columns of A are the Φ_i

(D x M matrix)

Step 4: compute the eigenvalues/eigenvectors of Σ_x

$\Sigma_x u_i = \lambda_i u_i$ where we assume $\lambda_1 > \lambda_2 > \ldots > \lambda_N$

PCA
Steps

- Since Σ_x is symmetric, $\langle u_1, u_2, \ldots, u_D \rangle$ form an orthogonal basis in R^D; therefore, we can represent any $\mathbf{x} \in R^D$ as:

$$\mathbf{x} - \bar{\mathbf{x}} = \sum_{i=1}^{D} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_D u_D$$

$$y_i = \frac{(\mathbf{x} - \bar{\mathbf{x}})^T u_i}{u_i^T u_i} = (\mathbf{x} - \bar{\mathbf{x}})^T u_i \qquad if \quad \| u_i \| = 1$$

i.e., this is just a "change" of basis!

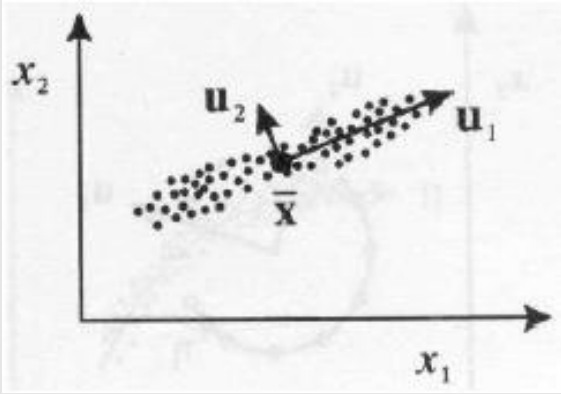
$$\mathbf{x} - \bar{\mathbf{x}}: \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ x_D \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ y_D \end{bmatrix}$$

- **Step 5:** dimensionality reduction step – approximate using only the first K eigenvectors (K<<D) (i.e., corresponding to the K largest eigenvalues where K is a parameter

$$\mathbf{x} - \bar{\mathbf{x}} = \sum_{i=1}^{D} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_D u_D \quad \Rightarrow \quad \hat{\mathbf{x}} - \bar{\mathbf{x}} = \sum_{i=1}^{K} y_i u_i = y_1 u_1 + y_2 u_2 + \ldots + y_K u_K$$

PCA
Interpretation of PCA

- PCA chooses the eigenvectors corresponding to the largest eigenvalues.
- The eigenvalues correspond to the variance of the data along the eigenvector directions.
- Therefore, PCA projects the data along the directions where the data varies most.
- PCA preserves as much information in the data by preserving as much variance in the data.



u_1: direction of max variance
u_2: orthogonal to u_1

PCA
Example

- Compute the PCA of the following dataset:

(1,2),(3,3),(3,5),(5,4),(5,6),(6,5),(8,7),(9,8)

- Compute the sample covariance matrix:

$$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t$$

$$\Sigma_x = \begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix}$$

- The eigenvalues can be computed by finding the roots of the characteristic polynomial:

$$\Sigma_x v = \lambda v \Rightarrow |\Sigma_x - \lambda I| = 0$$
$$\Rightarrow \begin{vmatrix} 6.25 - \lambda & 4.25 \\ 4.25 & 3.5 - \lambda \end{vmatrix} = 0$$
$$\Rightarrow \boldsymbol{\lambda_1 = 9.34; \lambda_2 = 0.41}$$

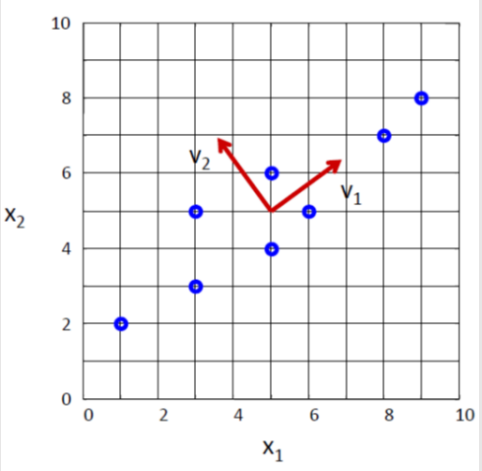
PCA
Example

- The eigenvectors are the solutions of:

$$\Sigma_{\mathbf{x}} u_i = \lambda_i u_i$$

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} \lambda_1 v_{11} \\ \lambda_1 v_{12} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{11} \\ v_{12} \end{bmatrix} = \begin{bmatrix} 0.81 \\ 0.59 \end{bmatrix}$$

$$\begin{bmatrix} 6.25 & 4.25 \\ 4.25 & 3.5 \end{bmatrix} \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} \lambda_2 v_{21} \\ \lambda_2 v_{22} \end{bmatrix} \Rightarrow \begin{bmatrix} v_{21} \\ v_{22} \end{bmatrix} = \begin{bmatrix} -0.59 \\ 0.81 \end{bmatrix}$$



Note: if u_i is a solution, then cu_i is also a solution where $c \neq 0$.

Eigenvectors are typically normalized to have unit-length:

$$\hat{v}_i = \frac{v_i}{\| v_i \|}$$

PCA
How Many PCs (value of K)?

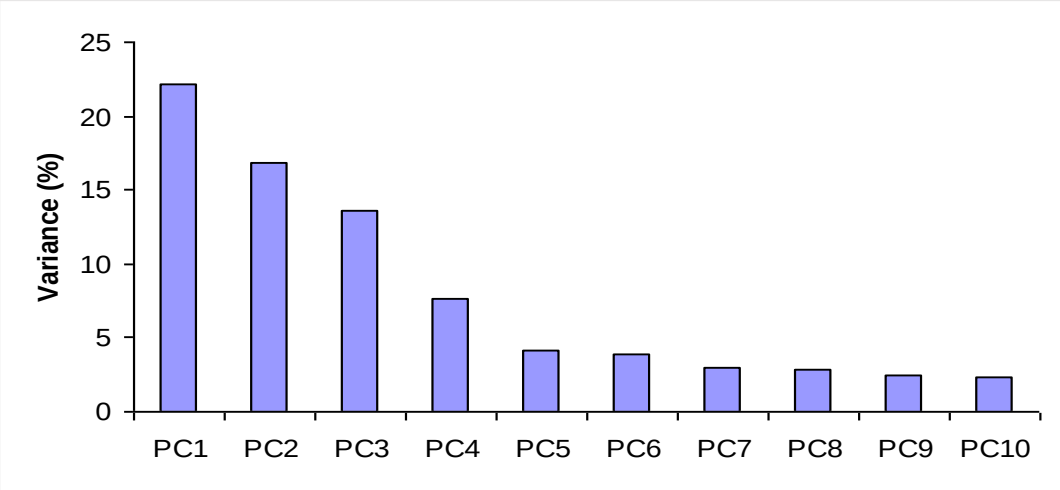
- For n original dimensions, correlation matrix is nxn, and has up to n eigenvectors. So n PCs.
- Proportion of Variance (PoV) explained

$$\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

  when λ_i are sorted in descending order
- Typically, stop at PoV>0.9
- Scree graph plots of PoV vs *k*, stop at “elbow”

PCA
How Many PCs?

Can *ignore* the components of lesser significance.



You do lose some information, but if the eigenvalues are small, you don't lose much

- n dimensions in original data
- calculate n eigenvectors and eigenvalues
- choose only the first p eigenvectors, based on their eigenvalues
- final data set has only p dimensions

PCA in sklearn

- ```
  import numpy as np
  from sklearn.decomposition import PCA

  X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
  pca = PCA(n_components=2)
  pca.fit(X)
  print(pca.explained_variance_ratio_)
  print(pca.singular_values_)
  ```
  ```