Machine Learning (CE 40717)

Fall 2024

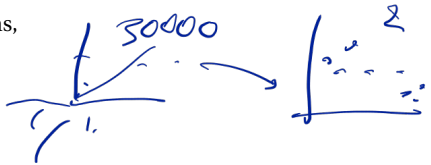
Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

November 23, 2025

Clustering vs Dimensionality Reduction

- **Clustering**
  - A method for summarizing a complex real-valued data point using a single categorical variable.
- **Dimensionality Reduction**
  - A different approach for simplifying complex, high-dimensional data.
  - Represents each data point with a lower-dimensional real-valued vector.
  - In dimensionality reduction:
    - Given data points in d dimensions,
    - Transform them into data points in $r < d$ dimensions,
    - With minimal loss of information.

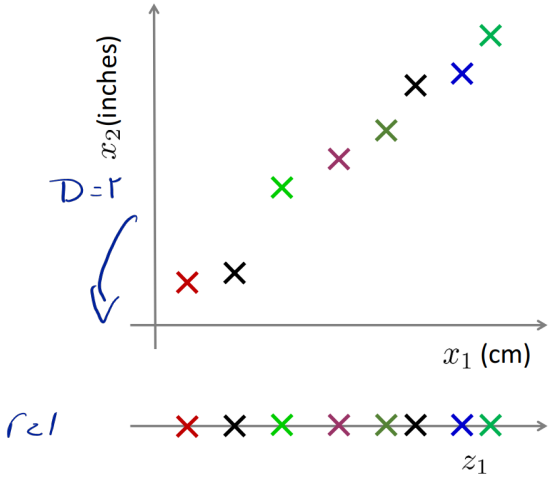


Dimensionality Reduction

Feature Extraction

- **Goal:** Represent high-dimensional data using fewer dimensions, for example:
  - *Compression*: reduced storage requirements and faster retrieval
  - *Visualization*: enabling 2D plotting
- **A good dimensionality reduction method can be defined in different ways**
  - Ability to reproduce the original data
  - Preserving discriminative features
  - Preserving neighborhood structure

Data Compression



Reducing data from 2D to 1D

$$\mathbf{x}^{(1)} \longrightarrow z^{(1)}$$

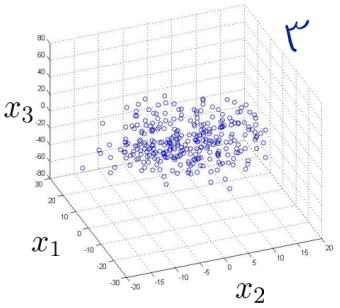
$$\mathbf{x}^{(2)} \longrightarrow z^{(2)}$$

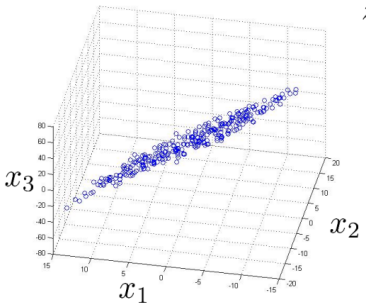
$$\vdots$$

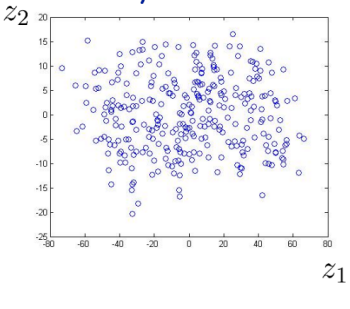
$$\mathbf{x}^{(m)} \longrightarrow z^{(m)}$$

Data Compression

Reduce data from 3D to 2D

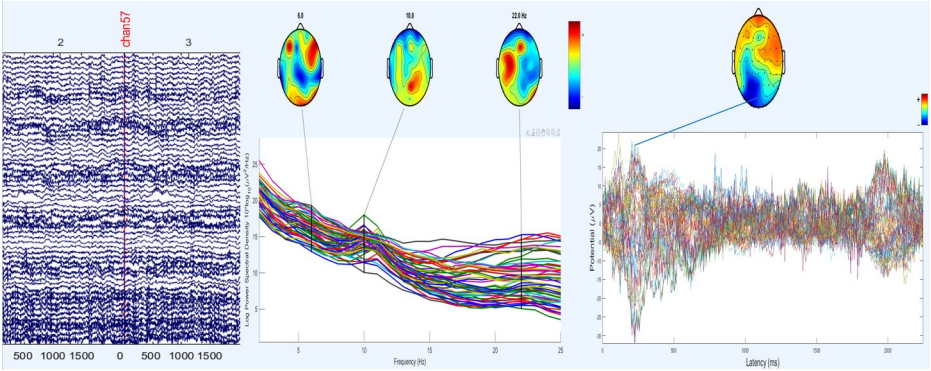






High-Dimensional Data

- High-dimensional data contains many features.
- Example: EEG signals recorded from the brain using 56 channels and 3000 time points per trial.



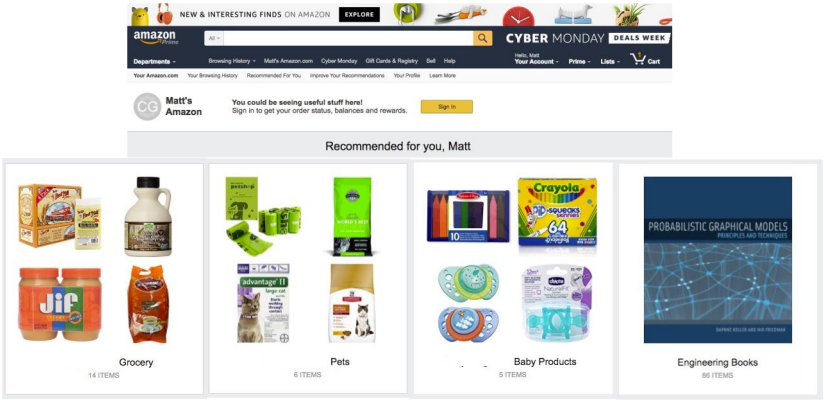
High-Dimensional Data

- Social media generates high-dimensional data.

Adopted from machinelearninggeek.com

High-Dimensional Data

- Customer purchase data.



Curse of Dimensionality

Why are more features bad?

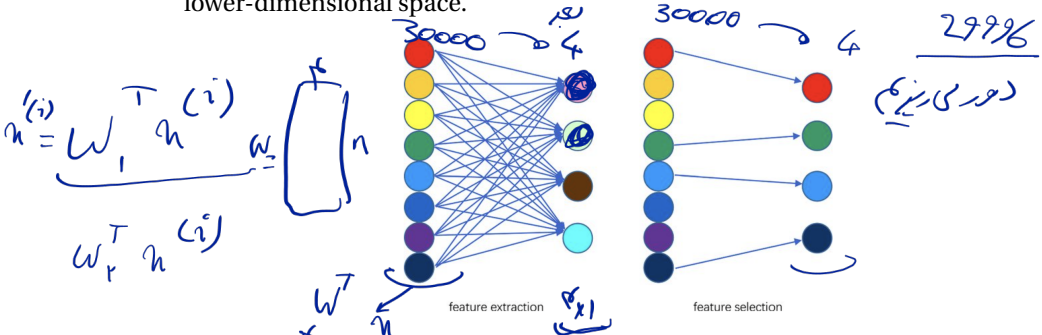
- Redundant features (e.g., not all words are useful for classifying a document) introduce more noise than signal.
- High-dimensional data is difficult to interpret and visualize.
- Storing and processing high-dimensional data is computationally expensive.
- The complexity of the decision rule tends to grow with the number of features. As the VC dimension increases, learning complex rules becomes statistically more challenging.

Dimensionality Reduction Benefits

- **Visualization**
  - Project high-dimensional data into 2D or 3D space.
- **Helps avoid overfitting**
  - Reduces noise by removing irrelevant features.
  - Can improve accuracy by eliminating noisy dimensions.
- **More efficient use of resources**
  - Saves time, memory, and computational power.

Dimensionality Reduction Techniques

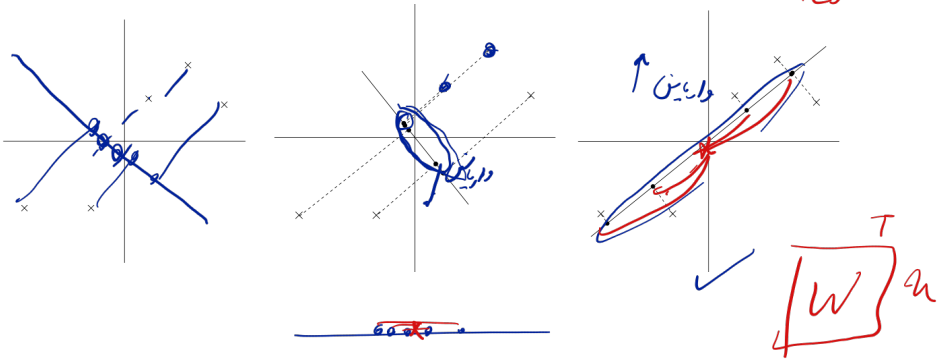
- **Feature Selection**
  - Select a subset of the original feature set.
- **Feature Extraction**
  - Apply a linear or nonlinear transformation to map the original feature space into a lower-dimensional space.



Adopted from www.link.springer.com

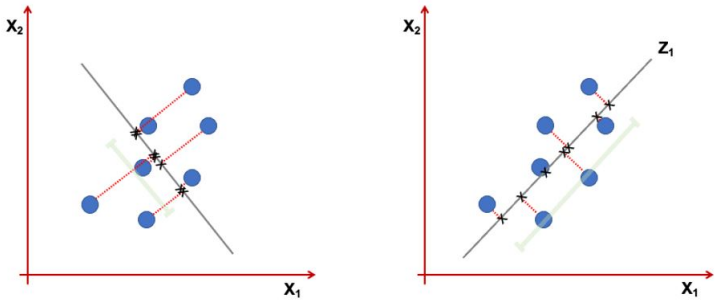
Which Projection Is Better?

- Maximize the retention of **important information** while reducing dimensionality.
- But what counts as **important information**?



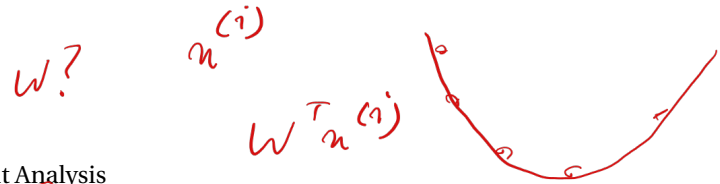
Purpose: Variance of Data

- Maximize the retention of **important information** while reducing dimensionality.
- **Information**: variance of the projected data.



Adopted from www.bookdown.org

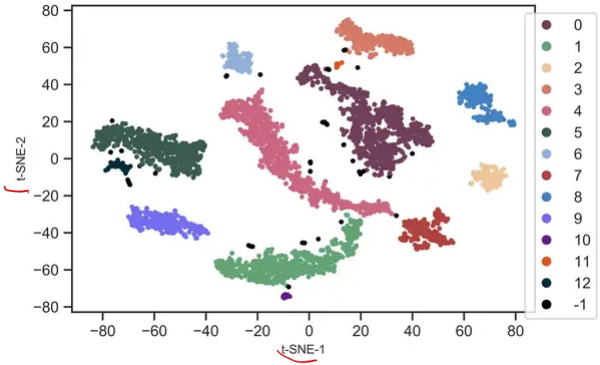
PCA and Manifolds



- **Linear Projection**
  - PCA: Principal Component Analysis
  - Reduces dimensionality while preserving the variance of the data.
- **Embedding / Manifold Learning**
  - Techniques such as MDS (multidimensional scaling), IsoMap, t-SNE, and UMAP
  - Aim to preserve point-to-point distances and/or local neighborhood structure.

Purpose: Local Geometric Neighborhood

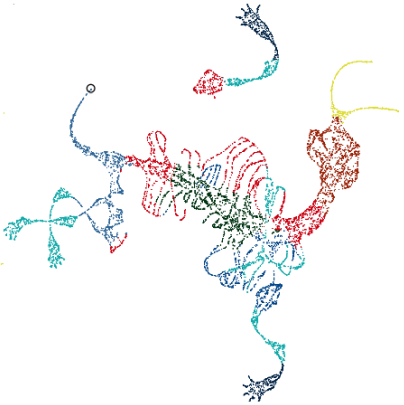
- **Information**: preserve the local geometric neighborhood.



Adopted from www.reneshbedre.com

Purpose: Local and Global Geometric Neighborhood

- **Information**: preserve both local and global geometric neighborhoods.



Adopted from www.pair-code.github.io

Background

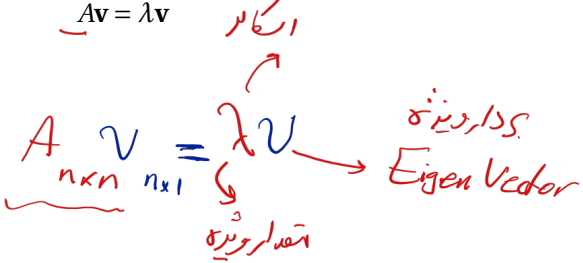
- Before jumping into the PCA algorithm, we should be familiar with the following concepts:
  - Eigenvalues and eigenvectors
  - Sample covariance matrix
  - Lagrange multipliers

What Are Eigenvalues and Eigenvectors?

- **Eigenvector**: A non-zero vector that is scaled only by a scalar factor when a linear transformation is applied.
- **Eigenvalue**: The scalar factor by which an eigenvector is scaled.
- **Equation** for an $n \times n$ matrix:

$$A\mathbf{v} = \lambda\mathbf{v}$$

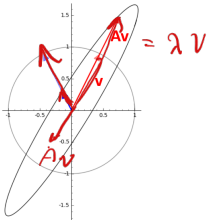
- Where:
  - A: A square matrix
  - $\mathbf{v}$: Eigenvector
  - λ: Eigenvalue



Geometrical Interpretation

- Eigenvectors point in the same (or opposite) direction after the transformation.
  - Eigenvectors do not change direction under a linear transformation.
- Eigenvalues represent how much the vector is stretched or compressed.
  - Eigenvalues indicate the scaling applied to the vector.

$$A = \begin{pmatrix} 1 & \frac{1}{3} \\ \frac{4}{3} & 1 \end{pmatrix} \quad n \times n \quad v \quad n \times 1 \quad = \quad u \quad n \times 1$$



Adopted from www.mathformachines.com

How to Find Eigenvalues and Eigenvectors?

- We know that

$$A\mathbf{v} = \lambda\mathbf{v}$$

- So,

$$A\mathbf{v} - \lambda\mathbf{v} = \mathbf{0}$$

$$(A - \lambda I)\mathbf{v} = \mathbf{0}$$

- Since $\mathbf{v}$ cannot be the zero vector, we must have:

$$\det(A - \lambda I) = 0$$

- Solve for λ.
- Substitute the obtained values of λ back into the equation

$$A\mathbf{v} = \lambda\mathbf{v}$$

to find the corresponding eigenvectors $\mathbf{v}$.

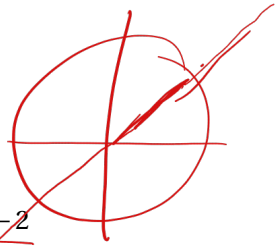
Numerical Example

- Assume $A = \begin{pmatrix} 4 & -5 \\ 2 & -3 \end{pmatrix}$
- $A - \lambda I = ?$

Numerical Example

- Assume $A = \begin{pmatrix} 4 & -5 \\ 2 & -3 \end{pmatrix}$
- $A - \lambda I = \begin{pmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{pmatrix}$
- Determinant $(A - \lambda I) = (4-\lambda)(-3-\lambda) + 10 = \lambda^2 - \lambda - 2$

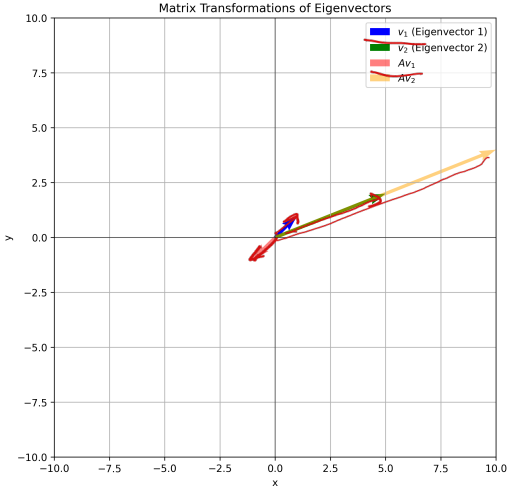
Numerical Example



- Assume $A = \begin{pmatrix} 4 & -5 \\ 2 & -3 \end{pmatrix}$
- $A - \lambda I = \begin{pmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{pmatrix}$
- Determinant $(A - \lambda I) = (4-\lambda)(-3-\lambda) + 10 = \lambda^2 - \lambda - 2$
- $\lambda = -1$ or $\lambda = 2$
- $\lambda_1 = -1$: $(A - \lambda_1 I)v_1 = \begin{bmatrix} 5 & -5 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$
- $\lambda_2 = 2$: $(A - \lambda_2 I)v_2 = \begin{bmatrix} 2 & -5 \\ 2 & -5 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow v_2 = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

Visualization

- $Av = \lambda v$



What Is Covariance?

- Covariance measures how much two random variables vary together.
- $$\mathrm{Cov}(X,Y) = \mathbb{E}[(X-\mathbb{E}[X])(Y-\mathbb{E}[Y])] = \mathbb{E}[(Y-\mathbb{E}[Y])(X-\mathbb{E}[X])] = \mathrm{Cov}(Y,X)$$
- Therefore, covariance is symmetric.
- Example: height and weight of individuals.

$Cov(X,Y) = E[XY]$

$E(X) = 0$

$E(Y) = 0$

What Is a Covariance Matrix?

- A **covariance matrix** generalizes the concept of covariance to multiple features.
- For a random vector $\mathbf{F} = [F_1, F_2, \ldots, F_d]$:

$$\Sigma = \begin{pmatrix} \mathrm{Var}(F_1) & \mathrm{Cov}(F_1, F_2) & \cdots & \mathrm{Cov}(F_1, F_d) \\ \mathrm{Cov}(F_2, F_1) & \mathrm{Var}(F_2) & \cdots & \mathrm{Cov}(F_2, F_d) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(F_d, F_1) & \mathrm{Cov}(F_d, F_2) & \cdots & \mathrm{Var}(F_d) \end{pmatrix}$$

- The diagonal elements represent variances, and the off-diagonal elements represent covariances.

Covariance Matrix Example

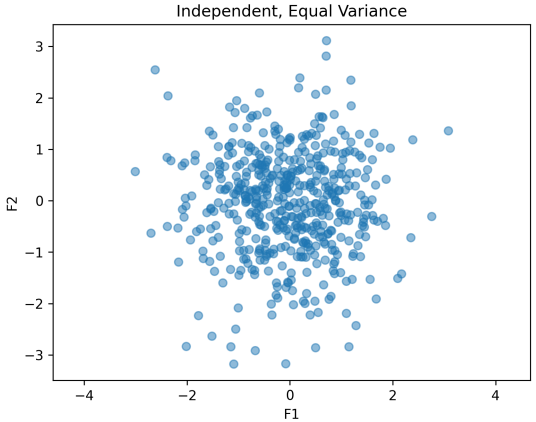
- Suppose we have a covariance matrix for two features:

$$\Sigma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ b & d \end{pmatrix}$$

- Why is $b = c$?
- What is the relationship between a, b, and d?

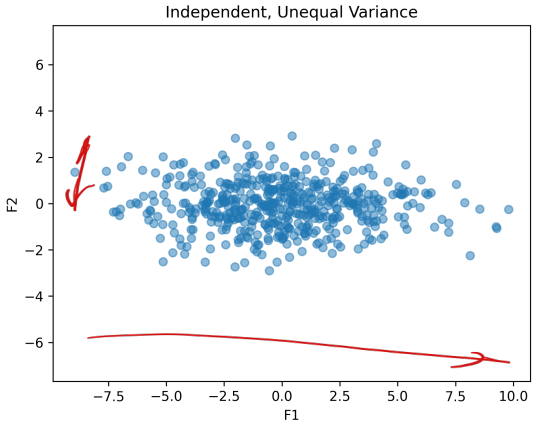
Covariance Matrix Example

- If $\Sigma = \begin{pmatrix} a & 0 \\ 0 & a \end{pmatrix}$, then:



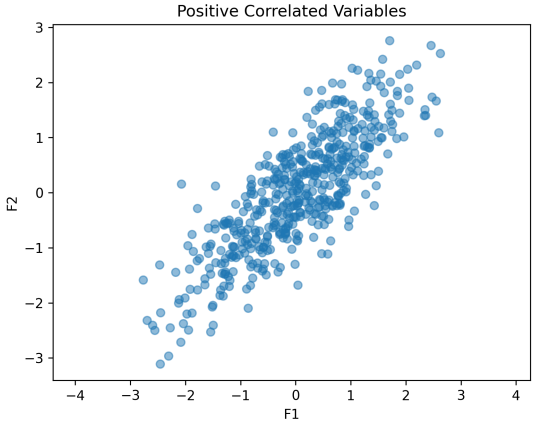
Covariance Matrix Example

- If $\Sigma = \begin{pmatrix} a & 0 \\ 0 & d \end{pmatrix}$ and $a > d$, then:



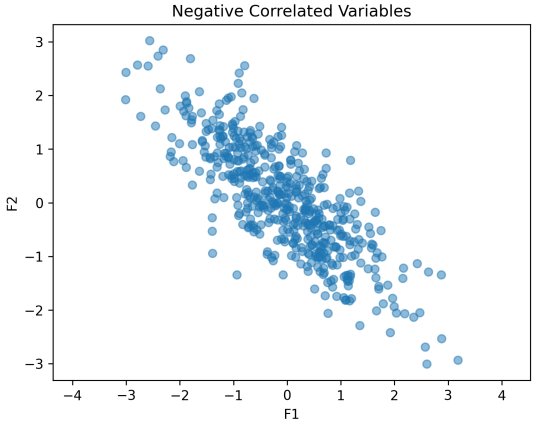
Covariance Matrix Example

- If $\Sigma = \begin{pmatrix} a & b \\ b & d \end{pmatrix}$, $a > d$, and $b > 0$, then:

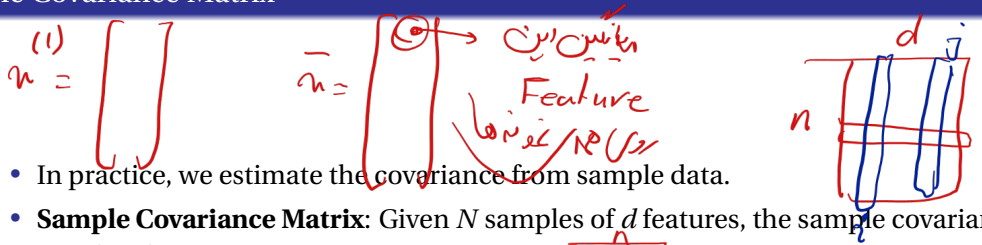


Covariance Matrix Example

- If $\Sigma = \begin{pmatrix} a & b \\ b & d \end{pmatrix}$, $a > d$, and $b < 0$, then:



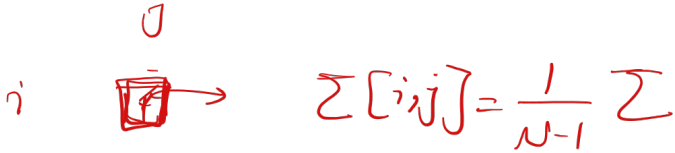
Sample Covariance Matrix



- In practice, we estimate the covariance from sample data.
- **Sample Covariance Matrix**: Given N samples of d features, the sample covariance matrix Σ is:

$$\Sigma_{d\times d} = \frac{1}{N-1}\sum_{i=1}^{N}\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)\left(\mathbf{x}^{(i)}-\bar{\mathbf{x}}\right)^{\top}$$

- Here, $\mathbf{x}^{(i)}$ is the i-th sample vector, and $\bar{\mathbf{x}} \in \mathbb{R}^{d\times 1}$ is the sample mean vector.



Example Calculation of Sample Covariance Matrix

- Suppose we have three samples, each with two features F_1 and F_2:

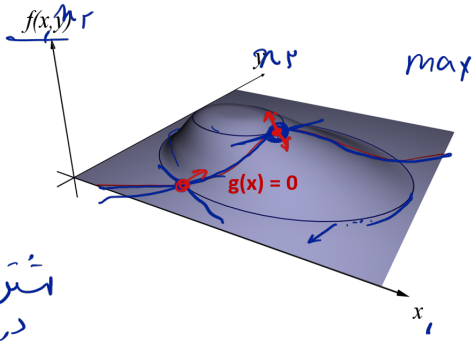
Sample	F_1	F_2
$\mathbf{x}^{(1)}$	3	3
$\mathbf{x}^{(2)}$	4	7
$\mathbf{x}^{(3)}$	5	8
$\bar{\mathbf{x}}$	4	6

$$\Sigma = \frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x}^{(i)} - \bar{\mathbf{x}})(\mathbf{x}^{(i)} - \bar{\mathbf{x}})^{\top}$$

$$\Sigma = \frac{1}{2}\left(\begin{pmatrix}1 & 3\\ 3 & 9\end{pmatrix} + \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} + \begin{pmatrix}1 & 2\\ 2 & 4\end{pmatrix}\right) = \begin{pmatrix}1 & 2.5\\ 2.5 & 7\end{pmatrix}$$

Lagrange Multiplier: Geometrical Interpretation

- We want to maximize $f(x)$ subject to the constraint $g(x) = 0$.
- The optimal point occurs where the gradient of $f(x)$ is proportional to the gradient of $g(x)$; that is, the two gradients are aligned (or point in opposite directions).



Adopted from www.khanacademy.org

Lagrange Multiplier Method

- Combine the objective function and the constraint using the Lagrangian:

$$\mathcal{L}(x, \lambda) = f(x) + \lambda\, g(x)$$

- Solve the system of equations:

$$\nabla \mathcal{L}(x, \lambda) = 0$$

- Where:
  - $\mathcal{L}$: the Lagrangian
  - λ: the Lagrange multiplier
  - $g(x)$: the equality constraint
  - $f(x)$: the objective function

Example Problem

- **Maximize:**

$$f(x_1, x_2) = x_1 + x_2$$

- **Subject to:**

$$x_1^2 + x_2^2 = 1$$

- **Lagrangian:**

$$\mathcal{L}(x_1, x_2, \lambda) = x_1 + x_2 - \lambda\left(x_1^2 + x_2^2 - 1\right)$$

- **Partial derivatives:**

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial x_1} = 1 - 2\lambda x_1 = 0 \\ \dfrac{\partial \mathcal{L}}{\partial x_2} = 1 - 2\lambda x_2 = 0 \\ \dfrac{\partial \mathcal{L}}{\partial \lambda} = -(x_1^2 + x_2^2 - 1) = 0 \end{cases}$$

Example Problem

- Solving the system:

$$\begin{cases} \lambda x_1 = \frac{1}{2} \\ \lambda x_2 = \frac{1}{2} \\ x_1^2 + x_2^2 = 1 \end{cases}$$

- Since $\lambda x_1 = \lambda x_2$, it follows that $x_1 = x_2$.
- Substitute $x_1 = x_2$ into the constraint:

$$2x_1^2 = 1 \implies x_1 = \pm\frac{1}{\sqrt{2}}$$

- Optimal solution:

$$x_1 = x_2 = \frac{1}{\sqrt{2}}, \qquad f_{\max} = x_1 + x_2 = \sqrt{2}$$

Generalization to Multiple Constraints

- The Lagrange multiplier method can be extended to problems with multiple constraints.
- For constraints

$$g_1(x) = 0, \quad g_2(x) = 0, \ldots, g_m(x) = 0,$$

the Lagrangian becomes:

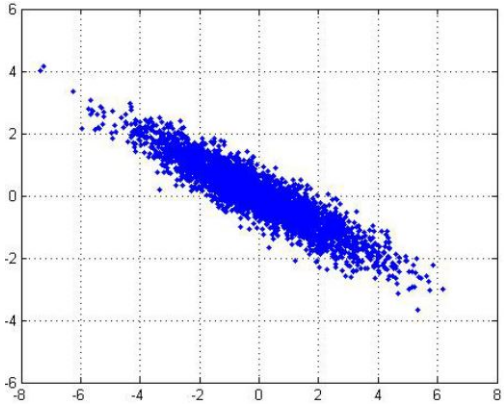
$$\mathcal{L}(x, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x) - \lambda_1 g_1(x) - \lambda_2 g_2(x) - \cdots - \lambda_m g_m(x).$$

Idea

- Given data points in a d-dimensional space, we want to project them into a lower-dimensional space while preserving as much information as possible:
  - Find the best planar approximation of 3D data.
  - Find the best 12D approximation of 104D data.
- In particular, we choose a projection that **minimizes the squared reconstruction error** of the original data.

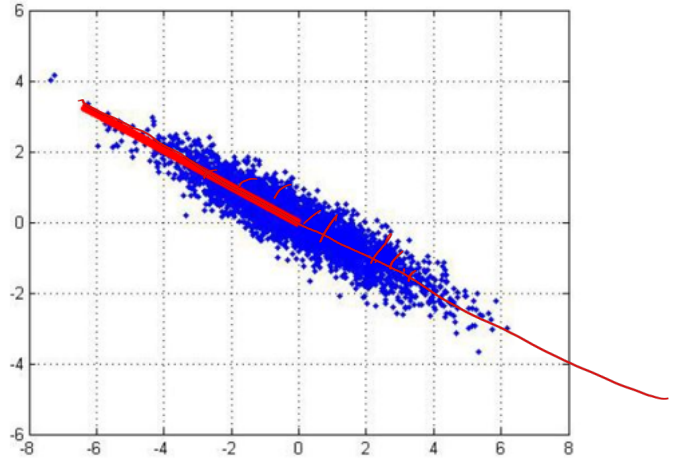
Principal Components Idea

- 2D Gaussian dataset:



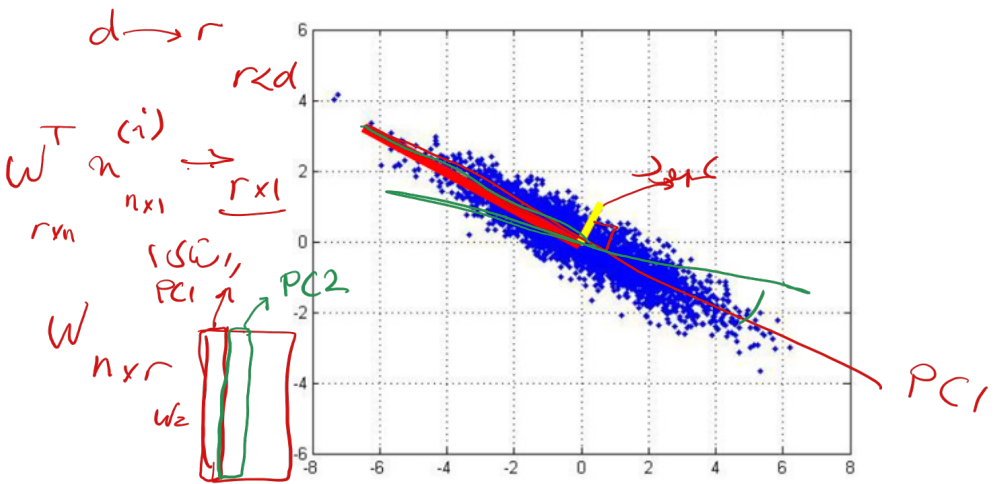
Principal Components Idea

- First PCA axis:



Principal Components Idea

- First and second PCA axes:



Definition

- **Goal**: reducing the dimensionality of the data while preserving important aspects of the dataset.
- Suppose the data matrix is

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}^{(1)\top} \\ \vdots \\ \mathbf{x}^{(N)\top} \end{pmatrix}_{N \times d} = \begin{pmatrix} \overset{F_1}{x_{11}} & \overset{F_2}{x_{12}} & \cdots & \overset{F_d}{x_{1d}} \\ x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nd} \end{pmatrix}$$

- PCA transforms the data as:

$$\mathbf{X}_{N \times d} \xrightarrow{\text{PCA}} \tilde{\mathbf{X}}_{N \times k}, \qquad k \leq d$$

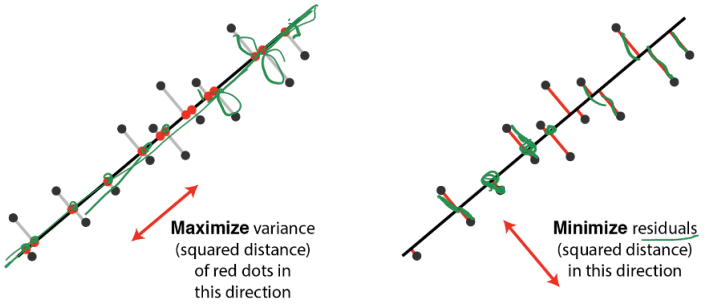
- **Assumption**: the data is mean-centered:

$$\boldsymbol{\mu}_x = \frac{1}{N} \sum_{i=1}^{N} \mathbf{x}^{(i)} = \mathbf{0}_{d \times 1}$$

Interpretations

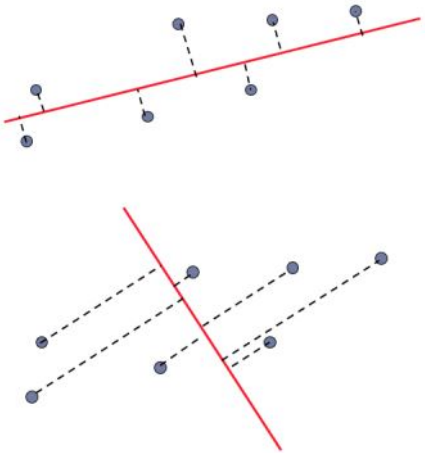
Orthogonal projection of the data onto a **lower-dimensional** linear **subspace** that:

- **Interpretation 1:** Maximizes the variance of the projected data.
- **Interpretation 2:** Minimizes the sum of squared distances to the subspace.



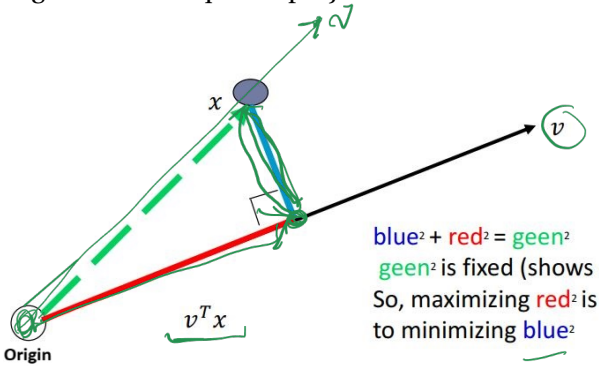
Least Squares Error Interpretation

- Principal components (PCs) are linear least-squares fits to the samples, with each PC orthogonal to all previously computed ones.
  - The first PC is the minimum-distance (least-squares) fit to a vector in the original feature space.
  - The second PC is the minimum-distance fit constrained to lie in the plane perpendicular to the first PC.



Equivalence of the Interpretations

- Minimizing the sum of squared distances to the subspace is **equivalent** to maximizing the sum of squared projections onto that subspace.



Equivalence of the Interpretations

Principal Components (PCs): A set of **orthonormal** vectors

$$\boldsymbol{v} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_k]$$

(where each $\boldsymbol{v}_i \in \mathbb{R}^{d\times 1}$), generated by PCA, that satisfy both interpretations.

Interpretation 1. Maximizes the variance of the projected data

- Projection of data points onto $\boldsymbol{v}_1$:

$$\Pi = \Pi_{\boldsymbol{v}_1}\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\} = \{\boldsymbol{v}_1^\top \mathbf{x}^{(1)}, \ldots, \boldsymbol{v}_1^\top \mathbf{x}^{(N)}\}$$

- Note that for a random variable X:

$$\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

- Since the data are mean-centered ($\mathbb{E}[X] = 0$):

$$\mathrm{Var}(\Pi) = \frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{v}_1^\top \mathbf{x}^{(i)}\right)^2$$

Pre-processing

- **Mean-center the data**
  - Set the mean of each feature to **zero**.
- **Scale the features so that each has variance 1 (optional normalization step)**
  - This may affect the results.
  - It is helpful when the units of measurement differ across features, preventing some features from being unintentionally ignored.

Step 1: Expression for Variance

- The variance of the projected data onto the direction $\mathbf{v}$ is:

$$\text{Var}(\mathbf{Xv}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}^{(i)\top}\mathbf{v}\right)^2$$

- This can be rewritten as:

$$\text{Var}(\mathbf{Xv}) = \frac{1}{n}\|\mathbf{Xv}\|^2 = \frac{1}{n}\mathbf{v}^\top\mathbf{X}^\top\mathbf{Xv} = \mathbf{v}^\top\Sigma\mathbf{v}$$

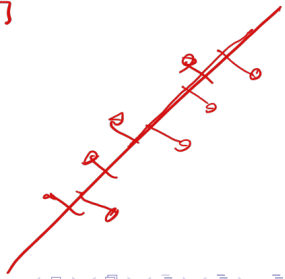
Step 2: Maximization Problem

- We aim to maximize the variance $\mathbf{v}^\top \Sigma \mathbf{v}$ under the constraint $\|\mathbf{v}\| = 1$.
- This leads to the following optimization problem:

$$\max_{\mathbf{v}} \mathbf{v}^\top \Sigma \mathbf{v} \quad \text{subject to} \quad \|\mathbf{v}\| = 1$$

max

Lagrange



Step 3: Use of Lagrange Multipliers

- We introduce a Lagrange multiplier λ and define the Lagrangian:

$$\mathcal{L}(\mathbf{v}, \lambda) = \mathbf{v}^\top \Sigma \mathbf{v} - \lambda (\mathbf{v}^\top \mathbf{v} - 1)$$

- Taking the derivative with respect to $\mathbf{v}$ and setting it to zero:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = 2\Sigma \mathbf{v} - 2\lambda \mathbf{v} = 0$$

- This simplifies to:

$$\Sigma \mathbf{v} = \lambda \mathbf{v}$$

- We find all pairs $(\mathbf{v}_1, \lambda_1), (\mathbf{v}_2, \lambda_2), \ldots, (\mathbf{v}_k, \lambda_k)$ as the k eigenvectors of Σ associated with the largest eigenvalues:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_k$$

Step 4: Interpretation

- The variance $\mathbf{v}^\top \Sigma \mathbf{v}$ is maximized when $\mathbf{v}$ is the eigenvector corresponding to the largest eigenvalue of Σ.
- The eigenvalue λ represents the variance in the direction of the eigenvector $\mathbf{v}$.
- **Conclusion**: Eigenvectors of the covariance matrix maximize the variance of the projected data.

PCA Algorithm

Algorithm 1 Principal Component Analysis (PCA)

1: **Input:** $X \in \mathbb{R}^{N \times d}$ (data matrix with N data points and d dimensions)
2: Compute the mean of each feature: $\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i$
3: Subtract the mean from each data point (center the data): $\tilde{X} \leftarrow X - \bar{x}^T$
4: Compute the covariance matrix: $\Sigma = \frac{1}{N-1} \tilde{X}^T \tilde{X}$
5: Compute the eigenvalues and eigenvectors of Σ: $[\lambda_1, \lambda_2, \ldots, \lambda_d], [v_1, v_2, \ldots, v_d] = \text{eig}(\Sigma)$
6: Select the top k eigenvectors corresponding to the largest eigenvalues: $A \leftarrow [v_1, v_2, \ldots, v_k]$
7: Transform the data into the new subspace: $X' \leftarrow X \cdot A$
8: **Output:** $X' \in \mathbb{R}^{N \times k}$ (transformed data with reduced dimensions)

Numerical Example

- We consider a small dataset with $N = 3$ samples and $d = 2$ features:

$$\mathbf{X} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \\ 3 & 3 \end{bmatrix}.$$

- Each row represents a data point in $\mathbb{R}^2$.
- **Goal**: apply PCA using the sample covariance matrix and compute the principal components and the transformed data.

Step 1: Mean-Center the Data

- Compute the mean of each feature:

$$\bar{\mathbf{x}} = \frac{1}{3}\begin{bmatrix} 2+0+3 \\ 0+2+3 \end{bmatrix} = \begin{bmatrix} \frac{5}{3} \\ \frac{5}{3} \end{bmatrix}.$$

- Subtract the mean from each sample:

$$\tilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}\bar{\mathbf{x}}^\top = \begin{bmatrix} \frac{1}{3} & -\frac{5}{3} \\ -\frac{5}{3} & \frac{1}{3} \\ \frac{4}{3} & \frac{4}{3} \end{bmatrix}.$$

- Now the mean of each column of $\tilde{\mathbf{X}}$ is zero.

Step 2: Sample Covariance Matrix

- With $N = 3$, the sample covariance matrix is

$$\Sigma = \frac{1}{N-1}\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} = \frac{1}{2}\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}.$$

- Compute:

$$\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}} = \begin{bmatrix} \frac{14}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{14}{3} \end{bmatrix} \quad \Rightarrow \quad \Sigma = \begin{bmatrix} \frac{7}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{7}{3} \end{bmatrix}.$$

- Diagonal entries are variances of each feature; off-diagonal entries are covariances.

Step 3: Eigenvalues and Eigenvectors of Σ

- Solve the eigenvalue problem:

$$\Sigma \mathbf{v} = \lambda \mathbf{v}, \qquad \Sigma = \begin{bmatrix} \frac{7}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{7}{3} \end{bmatrix}.$$

- Characteristic equation:

$$\det(\Sigma - \lambda I) = 0 \Rightarrow \lambda_1 = \frac{8}{3}, \quad \lambda_2 = 2.$$

- Corresponding eigenvectors (before normalization) can be chosen as

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

Step 3: Normalize Eigenvectors

- We normalize the eigenvectors to have unit norm:

$$\|\mathbf{v}_1\| = \sqrt{1^2 + 1^2} = \sqrt{2}, \qquad \|\mathbf{v}_2\| = \sqrt{(-1)^2 + 1^2} = \sqrt{2}.$$

- Orthonormal eigenvectors:

$$\mathbf{u}_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \mathbf{u}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$

- $\mathbf{u}_1$ is the first principal component (corresponding to the largest eigenvalue), and $\mathbf{u}_2$ is the second.

Step 4: Projection Matrix

- Collect the principal directions in

$$W = [\mathbf{u}_1 \ \mathbf{u}_2] = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

- This matrix defines the orthogonal projection from the original coordinates to the principal component coordinates.
- If we want to reduce to $k = 1$ dimension, we keep only $\mathbf{u}_1$ (the first column of W).

Step 5: Transform the Data

- The transformed (PCA) coordinates are obtained by

$$Z = \tilde{\mathbf{X}} W.$$

- Using the orthonormal eigenvectors, we obtain approximately

$$Z \approx \begin{bmatrix} -0.94 & -1.41 \\ -0.94 & 1.41 \\ 1.89 & 0.00 \end{bmatrix}.$$

- Interpretation:
  - Column 1: scores along the first principal component $\mathbf{u}_1$.
  - Column 2: scores along the second principal component $\mathbf{u}_2$.

Getting the Eigenvalues: Two Approaches

- Direct eigenvalue decomposition of the covariance matrix:

$$S = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n^\top = \frac{1}{N} X X^\top.$$

- Singular Value Decomposition (SVD).

Algorithms for PCA

How do we find principal components (i.e., eigenvectors)?

- **Power iteration (covered earlier)**
  - Finds each principal component one at a time, in decreasing order of eigenvalues.
- **Singular Value Decomposition (SVD)**
  - Finds all principal components at once.
  - Two options:
    - Option A: run SVD on $X^\top X$
    - Option B: run SVD directly on X *(It is not immediately obvious why Option B should work.)*

Singular Value Decomposition (SVD)

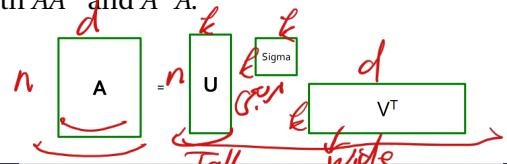
SVD (singular value decomposition) is a factorization of a matrix A into:

$$A = \underbrace{U}_{D\times D}\ \underbrace{\Sigma}_{D\times N}\ \underbrace{V^\top}_{N\times N}$$

$$\frac{1}{N}AA^\top = \frac{1}{N}U\Sigma\underbrace{V^\top V}_{=I_N}\Sigma^\top U^\top = \frac{1}{N}U\Sigma\Sigma^\top U^\top.$$

where:

- The columns of U are the eigenvectors of $AA^\top$.
- The columns of V are the eigenvectors of $A^\top A$.
- Σ is a diagonal (scaling) matrix of singular values, which are the square roots of the eigenvalues of both $AA^\top$ and $A^\top A$.



Generating Principal Components

- Subtract the mean

$$\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}^{(n)}$$

  from each data point to create zero-centered data.
- Create the matrix X with one row vector per zero-centered data point.
- Solve the SVD:

$$X = U \Sigma V^\top.$$

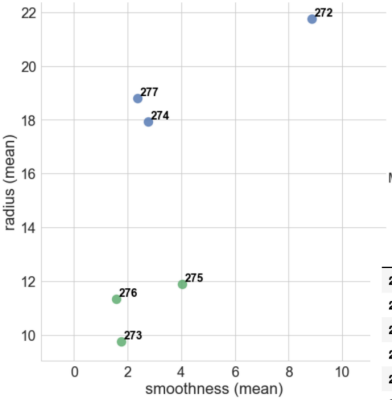
- Output the principal components: the columns of V (equivalently, the rows of $V^\top$):

$$V = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_d].$$

  - The eigenvectors $\mathbf{v}_k$ in V are ordered from largest to smallest eigenvalues.
  - Σ is diagonal, and σ_k^2 gives the eigenvalue corresponding to $\mathbf{v}_k$.

Subset of Patients: Smoothness and Radius

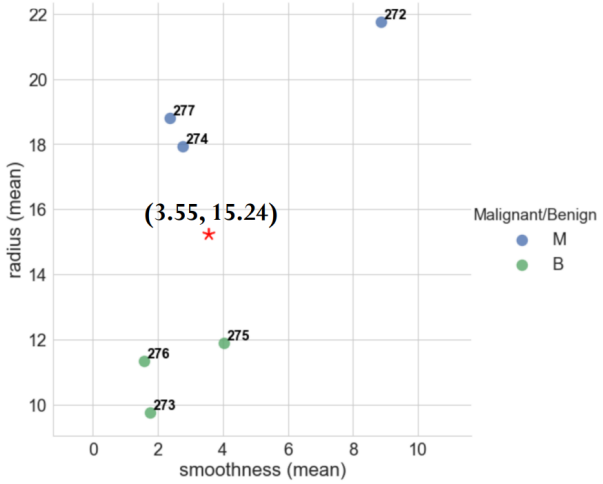
Lets start with a subset of 6 patients and focus on only two of their features: **smoothness** and **radius**.



	smoothness (mean)	radius (mean)
272	8.867	21.750
273	1.750	9.742
274	2.765	17.930
275	4.021	11.890
276	1.565	11.330
277	2.363	18.810

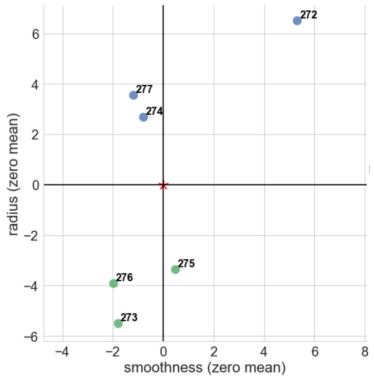
Dataset Center: Mean of Each Feature

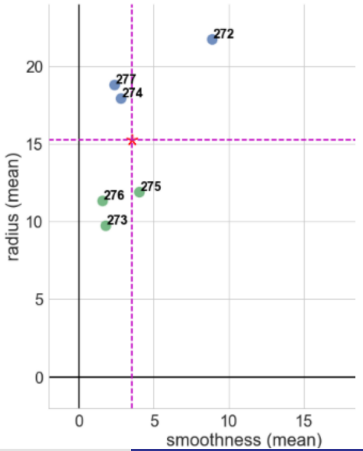
Determine the center of the dataset — the mean value of each feature.



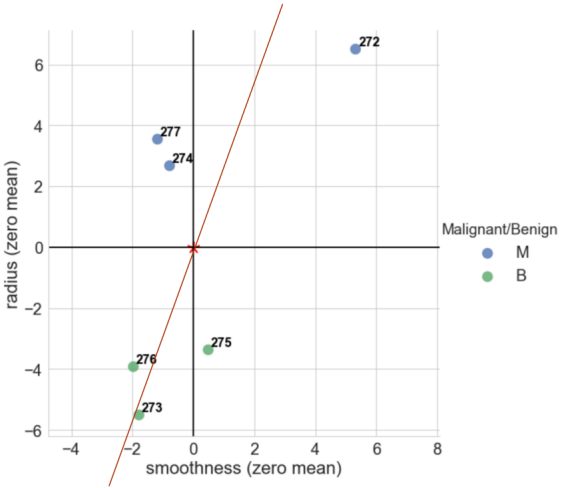
Shifting the Dataset to Zero Mean

We shift the dataset so that its center (the mean value) is moved to the origin $(0,0)$. The transformed dataset therefore has zero mean.



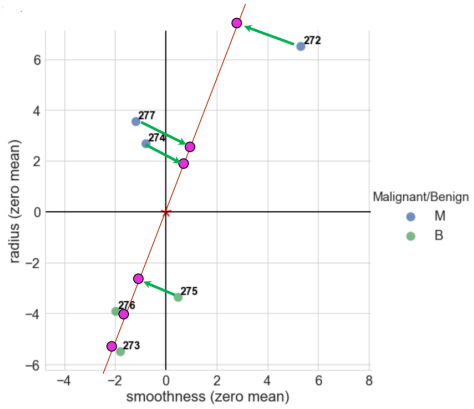


Fitting a Straight Line to the Dataset



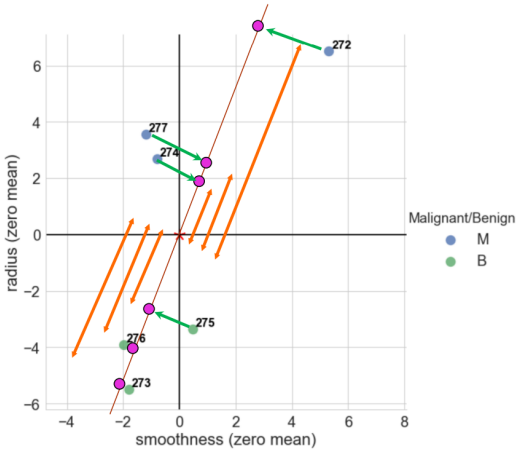
Projection onto a Proposed Line

Lets propose the red line shown below. To quantify how good the fit is, PCA projects the data onto this line. The best-fit line is the one that minimizes the distances from the points to the line (shown in green).



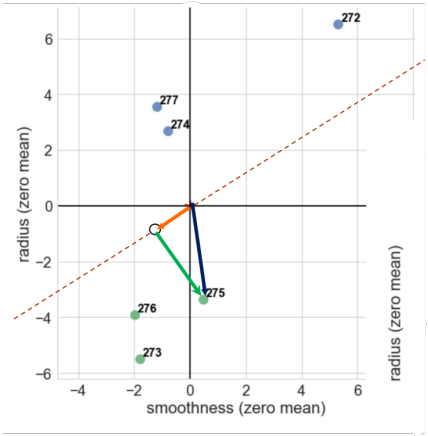
Maximizing the Spread of Projected Points

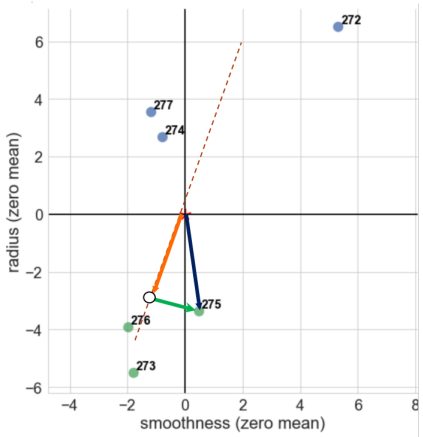
Alternatively, PCA can be viewed as maximizing the distances from the projected points to the origin (indicated in orange).



Why Are They the Same?

Why are these two interpretations equivalent? Consider what happens to the vectors below when we change the fitted curve.

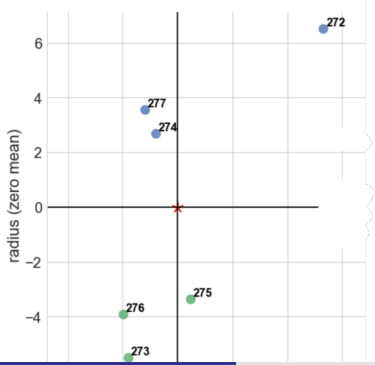




Variance of the Dataset

Lets talk about the variance of the dataset.
The covariance matrix is computed as:

$$\Sigma = \frac{1}{n-1} A^\top A.$$



$A =$

	smoothness (zero mean)	radius (zero mean)
272	5.311833	6.508
273	-1.805167	-5.500
274	-0.790167	2.688
275	0.465833	-3.352
276	-1.990167	-3.912
277	-1.192167	3.568

	smoothness (zero mean)	radius (zero mean)
smoothness (zero mean)	7.539518	8.868854
radius (zero mean)	8.868854	23.819936

Covariance, Eigenvalues, and Maximum Variance

We begin with the centered data matrix A and compute the covariance matrix:

$$\Sigma = \frac{1}{n-1} A^\top A.$$

Diagonalization of the covariance matrix gives:

$$A^\top A = XDX^\top,$$

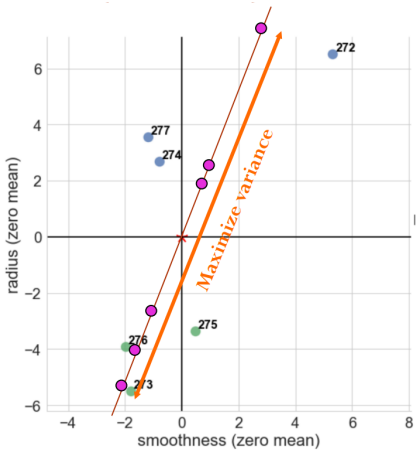
where X contains the eigenvectors of $A^\top A$, D contains the eigenvalues of $A^\top A$.

From the singular value decomposition (SVD):

$$A = U \Sigma V^\top.$$

Maximum variance corresponds to the largest singular value of Σ, and the **direction of maximum variance** is given by the corresponding column of V.

Covariance, Eigenvalues, and Maximum Variance

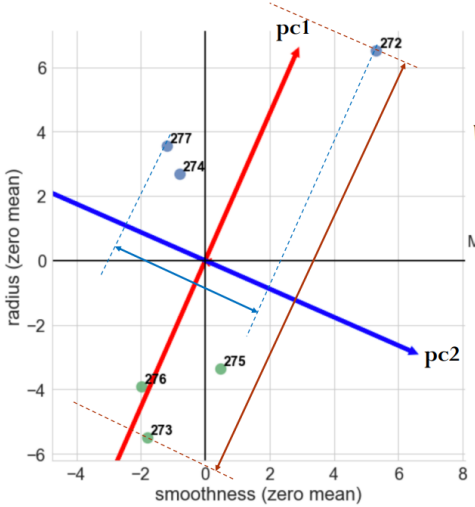


$A =$

	smoothness (zero mean)	radius (zero mean)
272	5.311833	6.508
273	-1.805167	-5.500
274	-0.790167	2.688
275	0.465833	-3.352
276	-1.990167	-3.912
277	-1.192167	3.568

	smoothness (zero mean)	radius (zero mean)
smoothness (zero mean)	7.539518	8.868854
radius (zero mean)	8.868854	23.819936

Principal Components from SVD



pc1 pc2

$$V = \begin{bmatrix} \vdots & \vdots \\ v_1 & v_2 \\ \vdots & \vdots \end{bmatrix}$$

Malignant/Benign
M
B

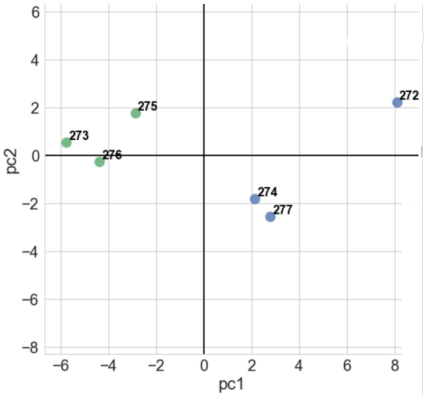
$\sigma_1 = 11.7$

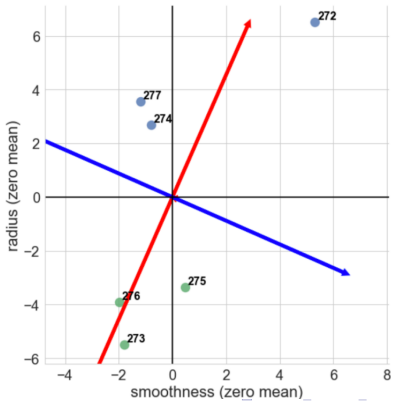
$\sigma_2 = 4.2$

Transformed Dataset in the PCA Coordinate System

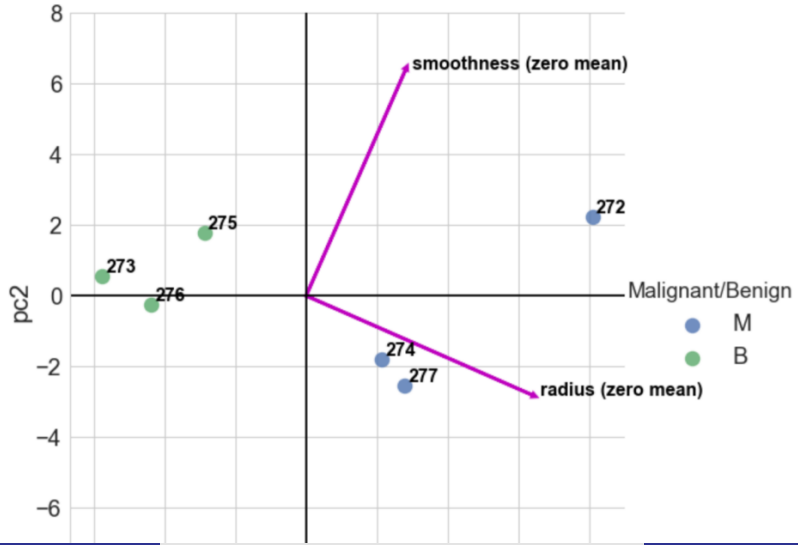
The transformed dataset is obtained by projecting the centered data onto the principal components:

$$A^* = AV = U\Sigma.$$



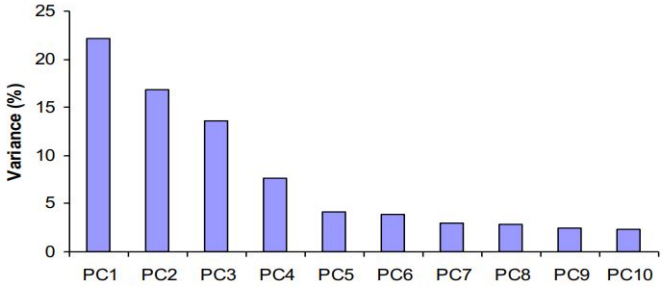


Original Features in the PCA Coordinate System



Number of Principal Components

- For d original dimensions, the sample covariance matrix is $d \times d$ and therefore has up to d eigenvectors. Thus, we can obtain up to d principal components.
- Components with smaller eigenvalues can be ignored.

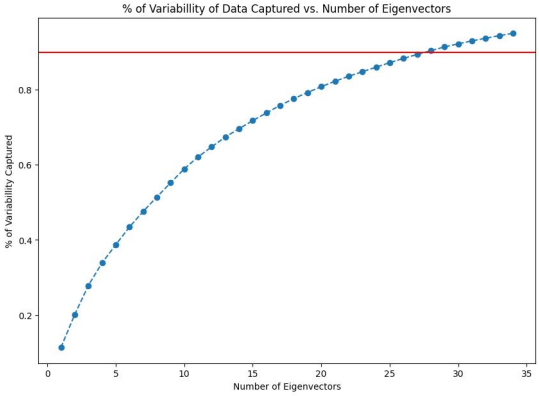


- We lose some information when dropping components, but if their eigenvalues are small, the loss is negligible.

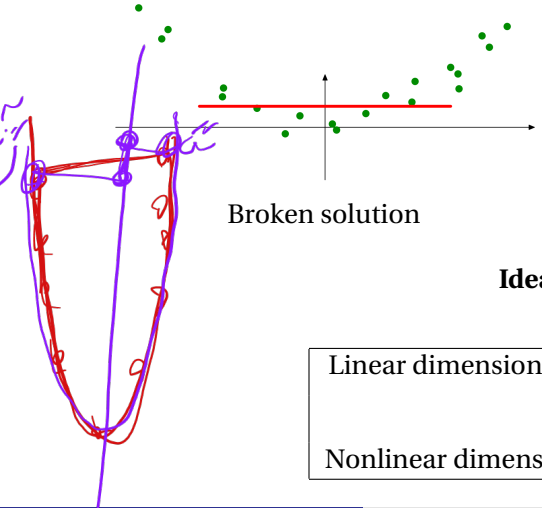
Number of Principal Components

- Select the desired variance ratio, and choose the smallest number of principal components that achieves it.

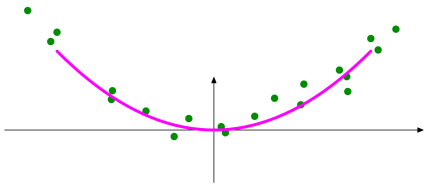
$$\min k \quad \text{s.t.} \quad \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq 0.9$$



Nonlinear PCA



Broken solution



Desired solution

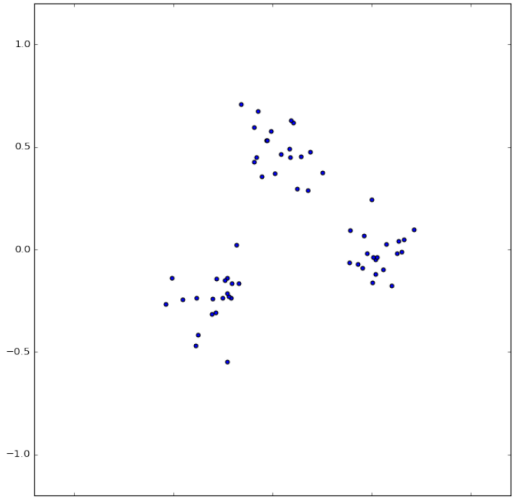
Idea: Use kernels

Linear dimensionality reduction in $\phi(\mathbf{x})$ space

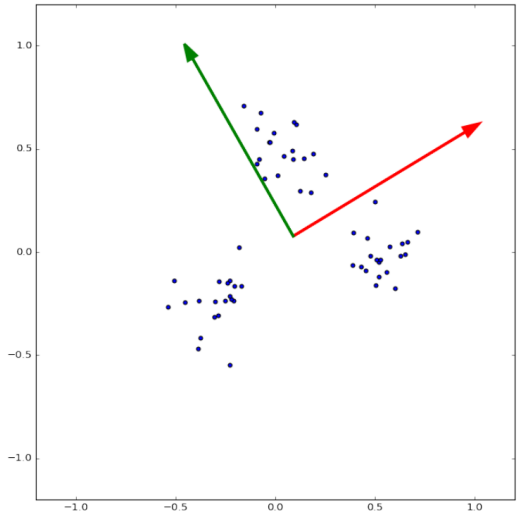
$\Updownarrow$

Nonlinear dimensionality reduction in $\mathbf{x}$ space

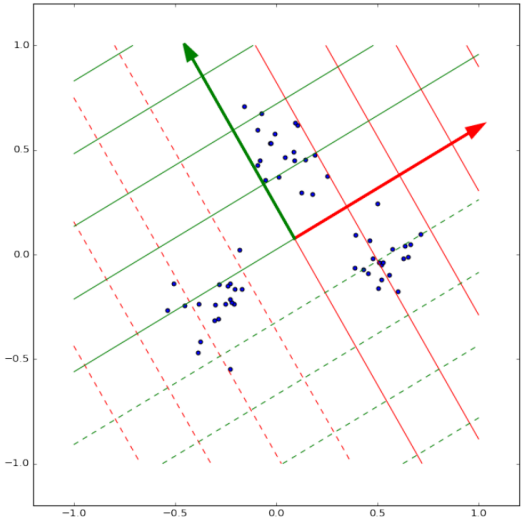
PCA: Beyond Linearity



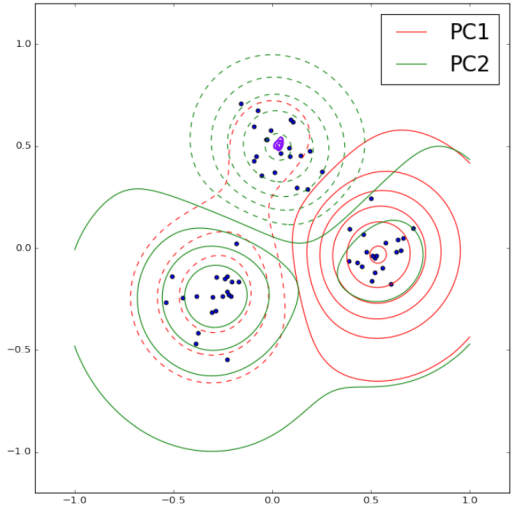
PCA: Beyond Linearity



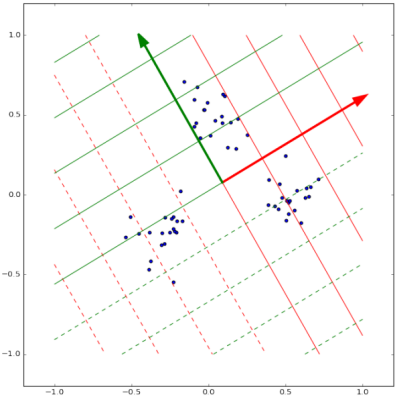
PCA: Beyond Linearity

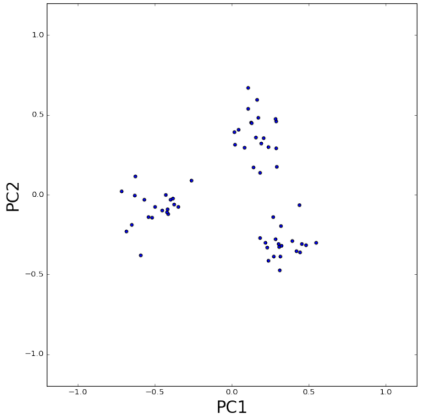


PCA: Beyond Linearity

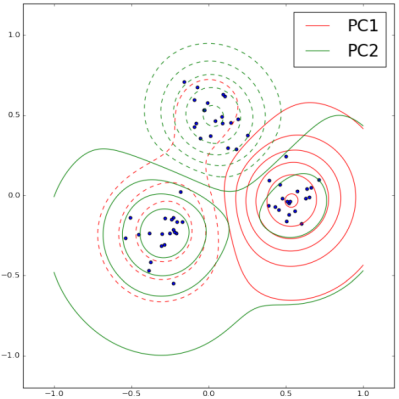


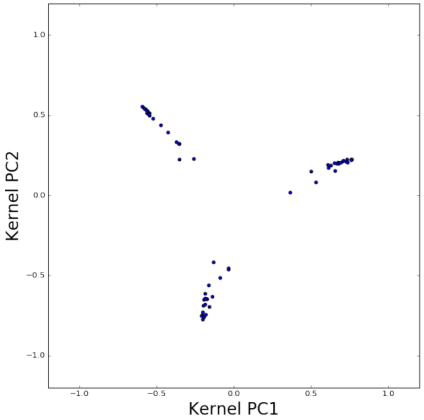
Projection: Linear PCA



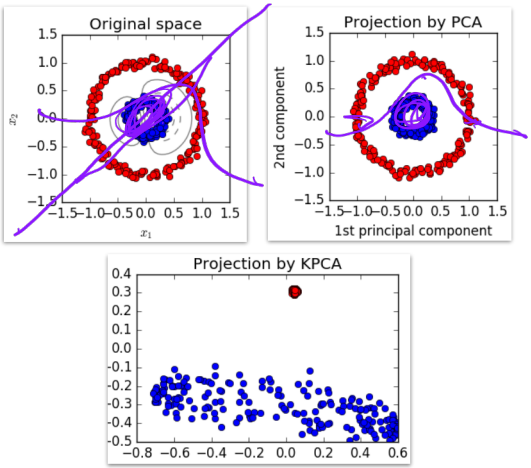


Projection: Kernel PCA





Kernel PCA


Original space
x_1
x_2
Projection by PCA
1st principal component
2nd component
Projection by KPCA

Kernel PCA

Suppose our original data is, for example, $\mathbf{x} \in \mathbb{R}^2$.

We can perform a degree-2 polynomial basis expansion as:

$$\phi(\mathbf{x}) = \left[1,\ \sqrt{2}\,x_1,\ \sqrt{2}\,x_2,\ x_1^2,\ x_2^2,\ \sqrt{2}\,x_1 x_2\right]^\top$$

Recall that we can compute the inner products $\phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$ efficiently using the **kernel trick**:

$$\phi(\mathbf{x}) \cdot \phi(\mathbf{x}') = 1 + 2x_1 x_1' + 2x_2 x_2' + x_1^2 (x_1')^2 + x_2^2 (x_2')^2 + 2x_1 x_2 x_1' x_2'$$

$$= \left(1 + x_1 x_1' + x_2 x_2'\right)^2 = \left(1 + \mathbf{x} \cdot \mathbf{x}'\right)^2 =: \kappa(\mathbf{x}, \mathbf{x}')$$

Kernel PCA

Suppose we use the feature map

$$\phi : \mathbb{R}^D \to \mathbb{R}^M.$$

Let $\phi(X)$ be the $N \times M$ matrix.

We want to find the singular vectors of $\phi(X)$ (i.e., the eigenvectors of $\phi(X)^\top \phi(X)$).

However, in general $M \gg N$ (in fact, M may even be infinite for some kernels).

Instead, we will find the eigenvectors of

$$\phi(X)\,\phi(X)^\top,$$

the kernel matrix.

Kernel PCA

Recall that the kernel matrix is:

$$K = \phi(X)\,\phi(X)^\top = \begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \kappa(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_N) \\ \kappa(\mathbf{x}_2, \mathbf{x}_1) & \kappa(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ \kappa(\mathbf{x}_N, \mathbf{x}_1) & \kappa(\mathbf{x}_N, \mathbf{x}_2) & \cdots & \kappa(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}$$

Let $\mathbf{u} \in \mathbb{R}^N$ be an eigenvector of K (the left singular vector of $\phi(X)$).

The corresponding principal component $\mathbf{v} \in \mathbb{R}^M$ is

$$\mathbf{v} = \sigma^{-1}\,\phi(X)^\top\,\mathbf{u}.$$

Kernel PCA

We will not express $\mathbf{v}$ explicitly. Instead, for a new datapoint $\mathbf{x}_{\text{new}}$ we compute its projection onto the principal component using the kernel function:

$$\phi(\mathbf{x}_{\text{new}})^\top \mathbf{v} = \sigma^{-1} \phi(\mathbf{x}_{\text{new}})^\top \phi(X)^\top \mathbf{u} = \sigma^{-1} \big[\kappa(\mathbf{x}_{\text{new}}, \mathbf{x}_1),\ \kappa(\mathbf{x}_{\text{new}}, \mathbf{x}_2),\ \ldots,\ \kappa(\mathbf{x}_{\text{new}}, \mathbf{x}_N)\big] \mathbf{u}.$$

Thus, to compute projections onto principal components, **we do not need to store the principal components explicitly!**

Kernel PCA

For PCA, we assume that the data matrix X is centered, i.e.,

$$\sum_i \mathbf{x}_i = \mathbf{0}.$$

However, this is not true for the matrix $\phi(X)$.

Instead, we consider the centered feature map:

$$\tilde{\phi}(\mathbf{x}_i) = \phi(\mathbf{x}_i) - \frac{1}{N}\sum_{k=1}^{N} \phi(\mathbf{x}_k).$$

The corresponding matrix $\tilde{K}$ has entries

$$\tilde{K}_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j) - \frac{1}{N}\sum_{\ell=1}^{N} \kappa(\mathbf{x}_i, \mathbf{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N} \kappa(\mathbf{x}_j, \mathbf{x}_\ell) + \frac{1}{N^2}\sum_{k=1}^{N}\sum_{\ell=1}^{N} \kappa(\mathbf{x}_\ell, \mathbf{x}_k).$$

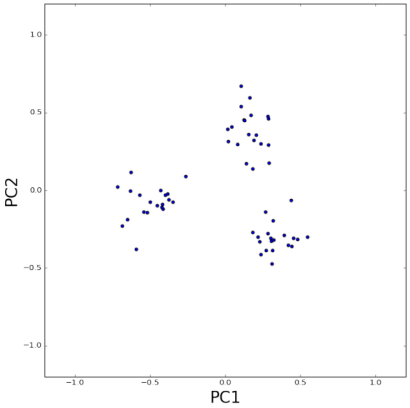
Kernel PCA

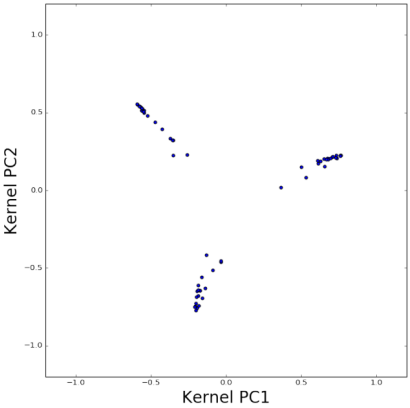
Succinctly, if O is the matrix with every entry $1/N$, i.e. $O = \mathbf{1}\mathbf{1}^\top / N$, then

$$\tilde{K} = K - OK - KO + OKO.$$

To perform kernel PCA, we need to find the eigenvectors of $\tilde{K}$.

Projection: PCA vs Kernel PCA





Kernel PCA Algorithm

Recover basis: Compute

$$K = \Phi(X)^\top \Phi(X)$$

using the kernel matrix. Let V be the matrix of eigenvectors of $\Phi(X)^\top \Phi(X)$ corresponding to the top d eigenvalues. Let Σ be the diagonal matrix of the square roots of the top d eigenvalues.

Encode training data:

$$Y = U^\top \Phi(X) = \Sigma V^\top$$

where Y is a $d \times t$ matrix of encodings of the original data. This is possible because computing Y does not require explicit evaluation of $\Phi(X)$.

Kernel PCA Algorithm

Reconstruct training data:

$$\hat{X} = UY = U\Sigma V^\top = \Phi(X)\, V\Sigma^{-1}\Sigma V^\top = \Phi(X)\, VV^\top.$$

But $\Phi(X)$ is unknown, so reconstruction **cannot be done**.

Encode test example:

$$\mathbf{y} = U^\top \Phi(\mathbf{x}) = \Sigma^{-1} V^\top \Phi(X)^\top \Phi(\mathbf{x}) = \Sigma^{-1} V^\top \Phi(X)^\top \Phi(\mathbf{x}).$$

This works because

$$\Phi(X)^\top \Phi(\mathbf{x}) = K(X, \mathbf{x}).$$

Reconstruct test example:

$$\hat{\mathbf{x}} = U\mathbf{y} = UU^\top \Phi(\mathbf{x}) = \Phi(X)\, V\Sigma^{-2} V^\top \Phi(X)^\top \Phi(\mathbf{x}) = \Phi(X)\, V\Sigma^{-2} V^\top K(X, \mathbf{x}).$$

Since $\Phi(\mathbf{x})$ is unknown, reconstruction **cannot be done**.

Image Compression

- Divide the original 372×492 image into patches.
  - Each patch is an instance containing 12×12 pixels on a grid.
- Treat each patch as a 144-dimensional vector.

Image Compression

- Reducing the representation from 144 dimensions to 60 dimensions.

Image Compression

- Reducing the representation from 144 dimensions to 16 dimensions.

Image Compression

- The 16 most important eigenvectors.

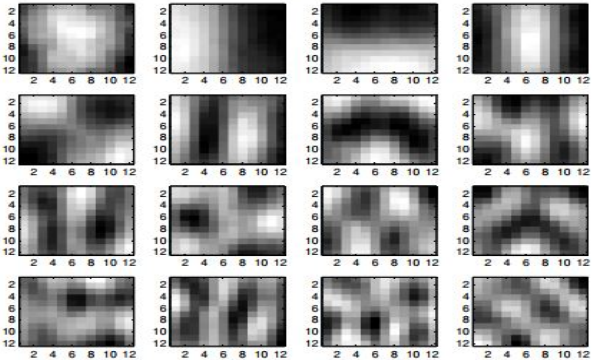
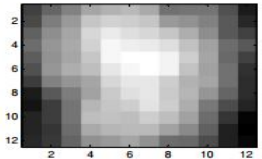


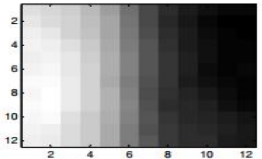
Image Compression

- Reducing the representation from 144 dimensions to 3 dimensions.

Image Compression

- The 3 most important eigenvectors.





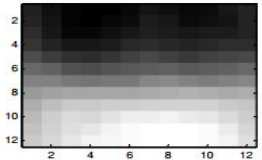
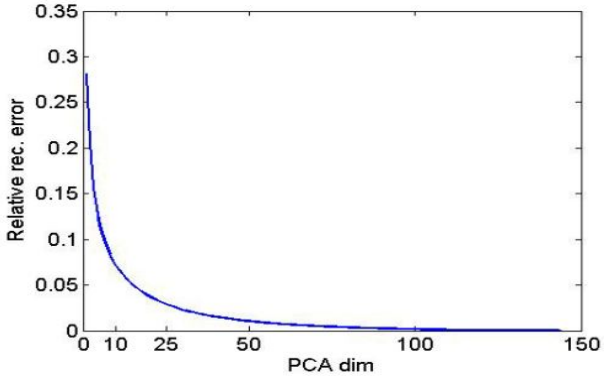


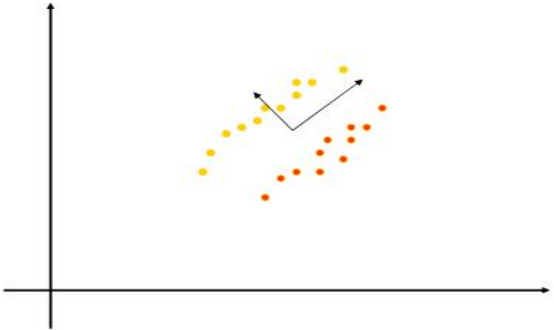
Image Compression

- L^2 error versus PCA dimensions.



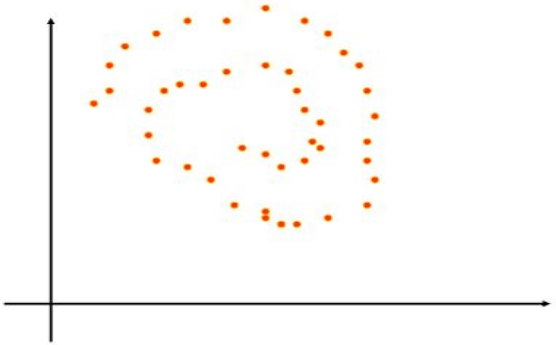
Class Labels

- PCA does not take class labels into account.
- An alternative solution: Linear Discriminant Analysis (LDA)



Non-Linear

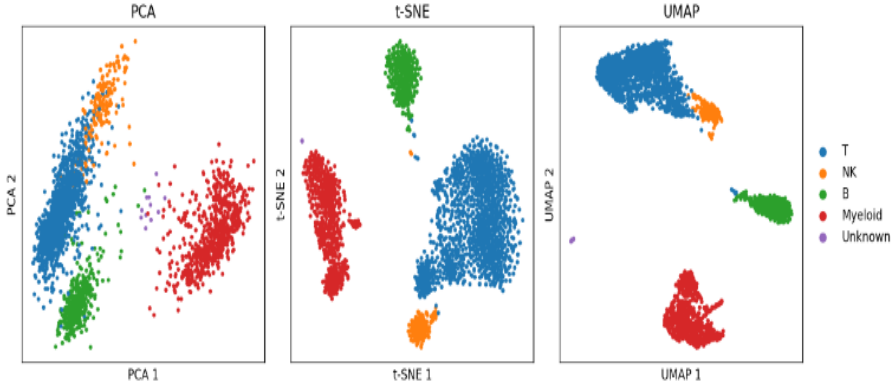
- PCA cannot capture non-linear structure.
- An alternative solution: Kernel PCA



Other Methods

- **t-SNE:**
  - A non-linear method focusing on preserving local structure.
  - Often produces well-separated clusters, making it useful for visualization.
  - Computationally intensive and relatively slow on large datasets.
- **UMAP:**
  - A non-linear method that preserves both local and some global structure.
  - Generally faster than t-SNE and scales better to large datasets.
  - Can capture more complex structures in the data.

PCA vs t-SNE vs UMAP



Adopted from www.mbernste.github.io

Conclusion

- PCA
  - Finds an orthonormal basis for the data.
  - Orders principal components by importance.
  - Discards low-significance principal components.
- Applications
  - Obtain a compact representation.
  - Remove noise.
  - Improve classification (hopefully).
  - Use computational resources more efficiently.
  - Statistical analysis.
- Not magic
  - Does not use class labels.
  - Can capture only linear variation.

Contributions

- **This slide has been prepared thanks to:**
  - Mohammad Mowlavi
  - Mahdi Aghaei

[1] M. Soleymani Baghshah, “Machine learning.” Lecture slides.

[2] B. Póczos, “Advanced introduction to machine learning.” Lecture slides. CMU-10715.

[3] M. Gormley, “Introduction to machine learning.” Lecture slides. 10-701.

[4] M. Gormley, “Introduction to machine learning.” Lecture slides. 10-301/10-601.

[5] F. Seyyedsalehi, “Machine learning and theory of machine learning.” Lecture slides. CE-477/CS-828.

[6] G. Strang, “Linear algebra and its applications,” 2000.