CSC 461: Machine Learning

Fall 2024

Dimensionality Reduction

Prof. Marco Alvarez, Computer Science
University of Rhode Island

High dimensional data

- Prevalent in many areas of machine learning and data science
- Examples:
  - image data, for instance a single 1000 by 1000 RGB image has 3 million dimensions
  - text data, in natural language processing text is often represented in high-dimensional spaces, transformers typically embed tokens into 768 or more dimensions
  - genomic data, gene expression data often has thousands of dimensions.
  - time series from sensor data or financial data
  - audio signals, especially when converted to spectrograms
  - network traffic data
  - …

Dimensionality reduction

- Fundamentally …
  - unsupervised learning algorithms for extracting latent structure (potentially **low dimensional**) from **high-dimensional** data
    - can range from simple feature selection to complex nonlinear transformations
- Examples
  - PCA, Kernel PCA, t-SNE, autoencoders, matrix factorization
- Given $\mathscr{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$ with $\mathbf{x_i} \in \mathbb{R}^d$, find a representation $\mathscr{Z} = \{\mathbf{z_1}, \ldots, \mathbf{z_n}\}$ with $\mathbf{z_i} \in \mathbb{R}^{d'}$, with $d' < d$
  - certain properties should be preserved (e.g., variance, distances, neighborhood structure)

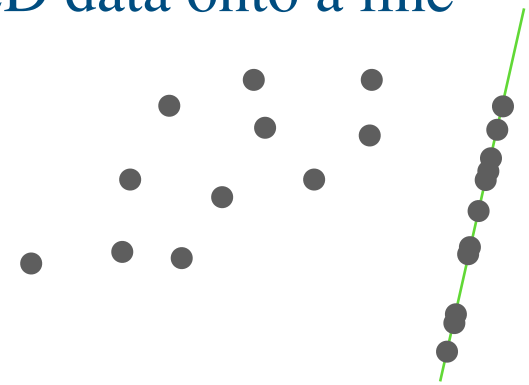
Dimensionality reduction

- Why?
  - visualization (2D or 3D)
  - preprocessing data before machine learning
    - focusing on important features/patterns
    - more efficient training
    - removing noise and redundant information
  - data compression

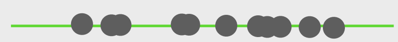
$$\begin{bmatrix} \mathbf{x_1}^T \\ \vdots \\ \mathbf{x_n}^T \end{bmatrix}_{n\times d} \Rightarrow \begin{bmatrix} \mathbf{z_1}^T \\ \vdots \\ \mathbf{z_n}^T \end{bmatrix}_{n\times d'}$$

Project 2D data onto a line

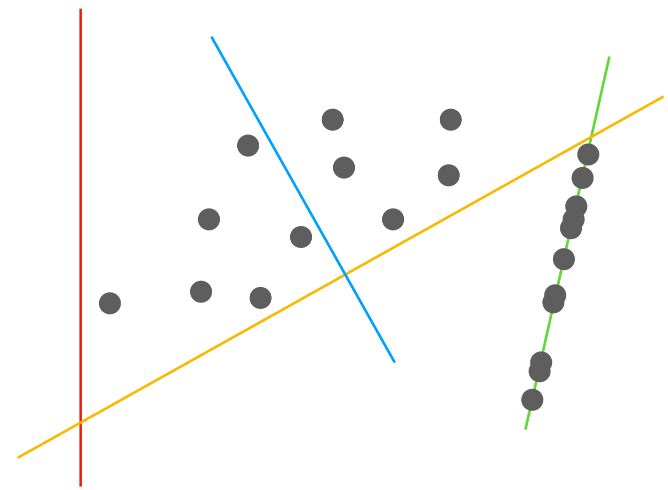
$$\begin{bmatrix} \mathbf{x_1}^T \\ \vdots \\ \mathbf{x_n}^T \end{bmatrix}_{11\times 2}$$



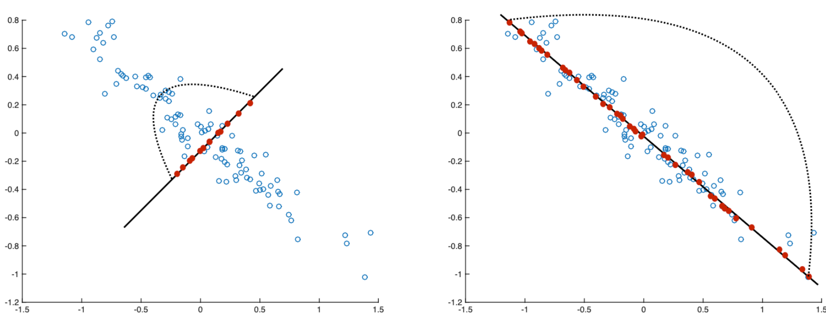
$$\begin{bmatrix} \mathbf{x_1}^T \\ \vdots \\ \mathbf{x_n}^T \end{bmatrix}_{11\times 1}$$



Which line is better? which is worst?

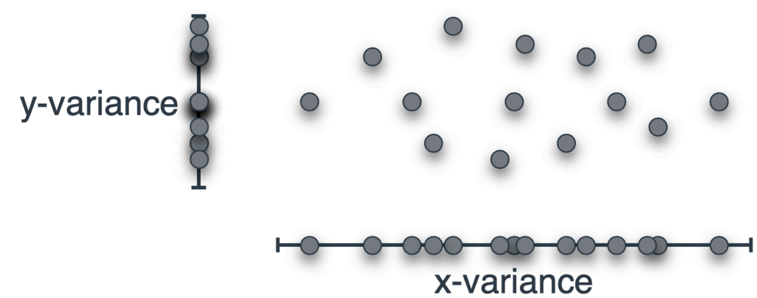


Maximize variance



Idea: minimize sum of square distances to the line (or maximize the sum of squares of the scalar projections)

Variance

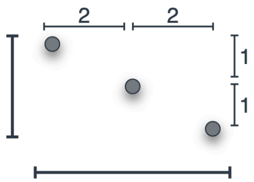


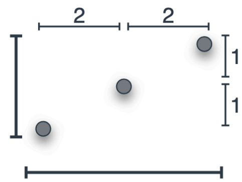
"biased" estimator default in `numpy`

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

Figure credit: https://serrano.academy/unsupervised-learning/

Different data, same variance



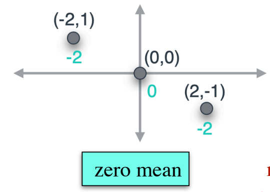


x-variance = $\frac{2^2+0^2+2^2}{3}$ = 8/3

y-variance = $\frac{1^2+0^2+1^2}{3}$ = 2/3

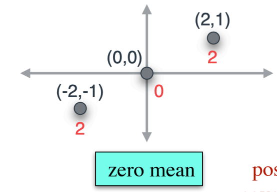
Figure credit: https://serrano.academy/unsupervised-learning/

Covariance



negative covariance

covariance = $\frac{(-2) + 0 + (-2)}{3}$ $\boxed{= -4/3}$



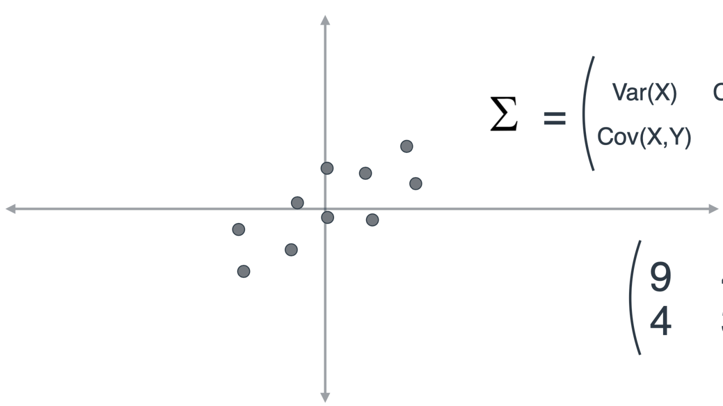
positive covariance

covariance = $\frac{2 + 0 + 2}{3}$ $\boxed{= 4/3}$

$$\mathrm{cov}(\mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

Figure credit: https://serrano.academy/unsupervised-learning/

Covariance matrix



$$\Sigma = \begin{pmatrix} \mathrm{Var(X)} & \mathrm{Cov(X,Y)} \\ \mathrm{Cov(X,Y)} & \mathrm{Var(Y)} \end{pmatrix}$$

$$\begin{pmatrix} 9 & 4 \\ 4 & 3 \end{pmatrix}$$

Every element Σ_{ij} of the covariance matrix is the covariance between column i and column j from the data matrix

Eigenvectors and eigenvalues

- The decomposition of a **square matrix A** into eigenvalues and eigenvectors is known as **eigen decomposition**
  - for **real symmetric matrices** eigenvectors can be chosen real and orthonormal

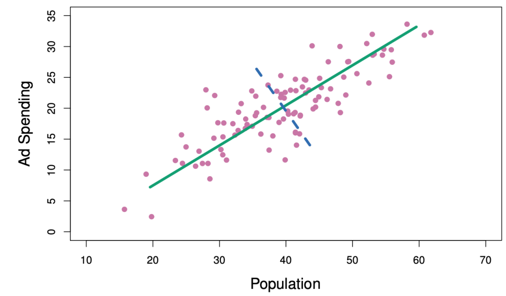
$$A = V \Lambda V^T \qquad A\mathbf{v} = \lambda \mathbf{v}$$

columns of V are the eigenvectors of A and Λ is a diagonal matrix whose entries are the eigenvalues of A

Principal Component Analysis (PCA)

Goal of PCA

- Find projections of the data onto directions that maximize variance
  - these directions are **orthogonal** to each other



https://www.dataschool.io/15-hours-of-expert-machine-learning-videos/

PCA approach

- Input: data matrix $X_{d \times n}$
  - **center** the data (subtract the mean)
  - calculate the **covariance** matrix $\frac{1}{n}XX^T$
  - compute **eigendecomposition** $V\Lambda V^T$ of the covariance matrix
  - **sort** the eigenvectors by eigenvalues in decreasing order
- Output
  - sorted orthonormal eigenvectors V and eigenvalues Λ

eigenvectors can then be used for projecting the data into lower dimensions (XV)

Remarks

- The **larger** the eigenvalue, the **more important** the corresponding eigenvector
  - that's why we **sort** eigenvalues (and corresponding eigenvectors) in **decreasing order**
- All eigenvalues of a positive semidefinite matrix are **non-negative**
  - **covariance matrix** is always symmetric and p.s.d.
- For dimensionality reduction, we can ignore eigenvectors associated with smaller eigenvalues

Explained variance

- Each eigenvalue corresponds to the amount of variance explained by its associated eigenvector
  - **explained variance** is often presented as a **percentage**, i.e., eigenvalues divided by the total sum of eigenvalues
- The sum of percentages of the top-k principal components is usually referred to as the **"cumulative explained variance"**
  - often used to select how many components to keep for a reduced dataset

Explained variance

PC	Eigenvalue	Variance (%)	Cumulative Variance
1st	23.31800	59.072%	59.072%
2nd	7.01200	17.764%	76.835%
3rd	4.61800	11.699%	88.534%
4th	1.98100	5.018%	93.553%
5th	1.00100	2.536%	96.089%
6th	0.82100	2.080%	98.168%
7th	0.64100	1.624%	99.792%
8th	0.03100	0.079%	99.871%
9th	0.02900	0.073%	99.944%
10th	0.02200	0.056%	100.000%

PCA Notebooks

https://colab.research.google.com/drive/1MzPdVsJi8gUxhwiXFcKA8RJsw8DY1OhE

https://colab.research.google.com/drive/1r7JPdmmWS11yl2WVOMi0GMhlDM9s37GI