Lecture 1: Matrix Decompositions

Jinwoo Shin

Kim Jaechul Graduate School of AI, KAIST

Roadmap

(1) Determinant and Trace

(2) Eigenvalues and Eigenvectors

(3) Cholesky Decomposition

(4) Eigendecomposition and Diagonalization

(5) Singular Value Decomposition

Summary

- How to summarize matrices: determinants and eigenvalues
- How matrices can be decomposed: Cholesky decomposition, diagonalization, singular value decomposition

Roadmap

(1) Determinant and Trace

(2) Eigenvalues and Eigenvectors

(3) Cholesky Decomposition

(4) Eigendecomposition and Diagonalization

(5) Singular Value Decomposition

Determinant: Motivation (1)

KAIST AI
Graduate School of AI

- For $\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$, $\boldsymbol{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$.
- $\boldsymbol{A}$ is invertible iff $a_{11}a_{22} - a_{12}a_{21} \neq 0$
- Let's define $\det(\boldsymbol{A}) = a_{11}a_{22} - a_{12}a_{21}$.
- Notation: $\det(\boldsymbol{A})$ or $|\text{whole matrix}|$
- What about 3×3 matrix? By doing some algebra (e.g., Gaussian elimination),

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23}$$
$$- a_{31}a_{22}a_{13} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33}$$

L4(1)

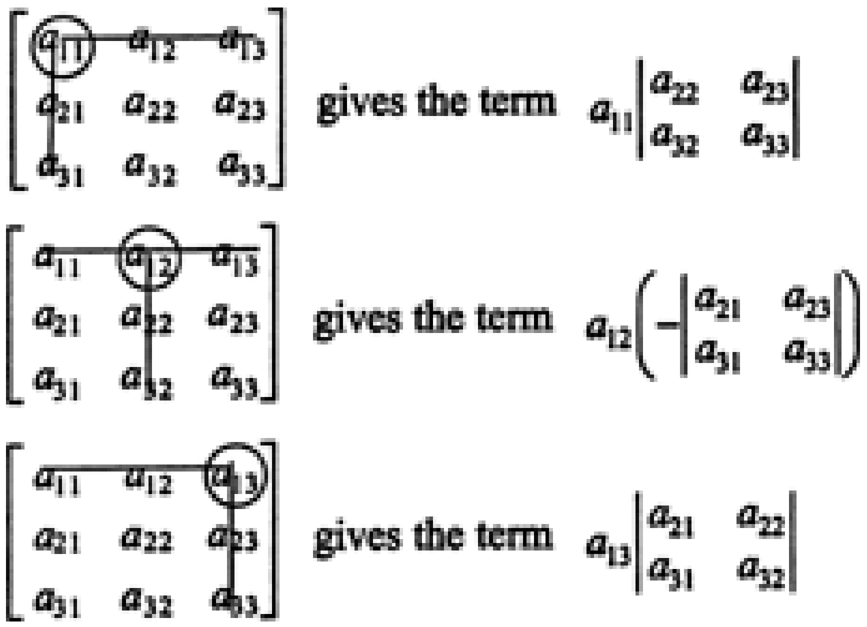
Determinant: Motivation (2)

KAIST AI
Graduate School of AI

- Try to find some pattern ...

$$a_{11}a_{22}a_{33} + a_{21}a_{32}a_{13} + a_{31}a_{12}a_{23}$$
$$- a_{31}a_{22}a_{13} - a_{11}a_{32}a_{23} - a_{21}a_{12}a_{33} =$$
$$a_{11}(-1)^{1+1}\det(\boldsymbol{A}_{1,1}) + a_{12}(-1)^{1+2}\det(\boldsymbol{A}_{1,2})$$
$$+ a_{13}(-1)^{1+3}\det(\boldsymbol{A}_{1,3})$$

- $\boldsymbol{A}_{k,j}$ is the submatrix of $\boldsymbol{A}$ that we obtain when deleting row k and column j.


gives the term $a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}$
gives the term $a_{12}\left(-\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}\right)$
gives the term $a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$

source: www.cliffsnotes.com

- This is called Laplace expansion.
- Now, we can generalize this and provide the formal definition of determinant.

L4(1)

Determinant: Formal Definition

Determinant

For a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, for all $j = 1, \ldots, n$,

1. Expansion along column j: $\det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k+j} a_{kj} \det(\boldsymbol{A}_{k,j})$
2. Expansion along row j: $\det(\boldsymbol{A}) = \sum_{k=1}^{n} (-1)^{k+j} a_{jk} \det(\boldsymbol{A}_{j,k})$

- All expansion are equal, so no problem with the definition.
- Theorem. $\det(\boldsymbol{A}) \neq 0 \iff \text{rk}(\boldsymbol{A}) = n \iff \boldsymbol{A}$ is invertible.

Determinant: Properties

(1) $\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B})$

(2) $\det(\boldsymbol{A}) = \det(\boldsymbol{A}^\top)$

(3) For a regular $\boldsymbol{A}$, $\det(\boldsymbol{A}^{-1}) = 1/\det(\boldsymbol{A})$

(4) For two similar matrices $\boldsymbol{A}, \boldsymbol{A}'$ (i.e., $\boldsymbol{A}' = \boldsymbol{S}^{-1}\boldsymbol{AS}$ for some $\boldsymbol{S}$), $\det(\boldsymbol{A}) = \det(\boldsymbol{A}')$

(5) For a triangular matrix[1] $\boldsymbol{T}$, $\det(\boldsymbol{T}) = \prod_{i=1}^{n} T_{ii}$

(6) Adding a multiple of a column/row to another one does not change $\det(\boldsymbol{A})$

(7) Multiplication of a column/row with λ scales $\det(\boldsymbol{A})$: $\det(\lambda\boldsymbol{A}) = \lambda^n\boldsymbol{A}$

(8) Swapping two rows/columns changes the sign of $\det(\boldsymbol{A})$

- Using (5)-(8), Gaussian elimination (reaching a triangular matrix) enables to compute the determinant.

[1] This includes diagonal matrices.

L4(1)

Trace

- Definition. The trace of a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is defined as

$$\operatorname{tr}(\boldsymbol{A}) := \sum_{i=1}^{n} a_{ii}$$

- $\operatorname{tr}(\boldsymbol{A} + \boldsymbol{B}) = \operatorname{tr}(\boldsymbol{A}) + \operatorname{tr}(\boldsymbol{B})$
- $\operatorname{tr}(\alpha \boldsymbol{A}) = \alpha \operatorname{tr}(\boldsymbol{A})$
- $\operatorname{tr}(\boldsymbol{I}_n) = n$

L4(1)

Roadmap

Eigenvalue and Eigenvector

- Definition. Consider a square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$. Then, $\lambda \in \mathbb{R}$ is an eigenvalue of $\boldsymbol{A}$ and $\boldsymbol{x} \in \mathbb{R}^n \setminus \{0\}$ is the corresponding eigenvector of $\boldsymbol{A}$ if

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$

- Equivalent statements
  - λ is an eigenvalue.
  - $(\boldsymbol{A} - \lambda \boldsymbol{I}_n)\boldsymbol{x} = 0$ can be solved non-trivially, i.e., $\boldsymbol{x} \neq 0$.
  - $\det(\boldsymbol{A} - \lambda \boldsymbol{I}_n) = 0$

L4(2)

Example

- For $\boldsymbol{A} = \left(\begin{smallmatrix} 4 & 2 \\ 1 & 3 \end{smallmatrix}\right)$, $p_{\boldsymbol{A}}(\lambda) = \begin{vmatrix} 4-\lambda & 2 \\ 1 & 3-\lambda \end{vmatrix} = (4-\lambda)(3-\lambda) - 2 \cdot 1 = \lambda^2 - 7\lambda + 10$
- Eigenvalues $\lambda = 2$ or $\lambda = 5$.
- Eigenvector E_5 for $\lambda = 5$

$$\begin{pmatrix} 4-\lambda & 2 \\ 1 & 3-\lambda \end{pmatrix} \boldsymbol{x} = 0 \implies \begin{pmatrix} -1 & 2 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 0 \implies E_5 = \text{span}[\begin{pmatrix} 2 \\ 1 \end{pmatrix}]$$

- Eigenvector E_2 for $\lambda = 2$. Similarly, we get $E_2 = \text{span}[\begin{pmatrix} 1 \\ -1 \end{pmatrix}]$
- Message. Eigenvectors are not unique.

Properties

- For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, n distinct eigenvalues $\implies$ eigenvectors are linearly independent, which form a basis of $\mathbb{R}^n$.
  - Converse is not true.
  - Example of n linearly independent eigenvectors for less than n eigenvalues???
- Determinant. For (possibly repeated) eigenvalues λ_i of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$,

$$\boxed{\det(\boldsymbol{A}) = \prod_{i=1}^n \lambda_i}$$

- Trace. For (possibly repeated) eigenvalues λ_i of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$,

$$\boxed{\operatorname{tr}(\boldsymbol{A}) = \sum_{i=1}^n \lambda_i}$$

Roadmap

Cholesky Decomposition

- A real number: decomposition of two identical numbers, e.g., $9 = 3 \times 3$
- Theorem. For a symmetric, positive definite matrix $\boldsymbol{A}$, $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}$, where
  - $\boldsymbol{L}$ is a lower-triangular matrix with positive diagonals
  - Such a $\boldsymbol{L}$ is unique, called Cholesky factor of $\boldsymbol{A}$.
- Applications
  (a) factorization of covariance matrix of a multivariate Gaussian variable
  (b) linear transformation of random variables
  (c) fast determinant computation: $\det(\boldsymbol{A}) = \det(\boldsymbol{L})\det(\boldsymbol{L}^{\mathsf{T}}) = \det(\boldsymbol{L})^2$, where $\det(\boldsymbol{L}) = \prod_i l_{ii}$. Thus, $\det(\boldsymbol{A}) = \prod_i l_{ii}^2$.

Roadmap

Diagonal Matrix and Diagonalization

- Diagonal matrix. zero on all off-diagonal elements, $\boldsymbol{D} = \begin{pmatrix} d_1 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & d_n \end{pmatrix}$

$$\boldsymbol{D}^k = \begin{pmatrix} d_1^k & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & d_n^k \end{pmatrix}, \quad \boldsymbol{D}^{-1} = \begin{pmatrix} 1/d_1 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 1/d_n \end{pmatrix}, \quad \det(\boldsymbol{D}) = d_1 d_2 \cdots d_n$$

- Definition. $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is diagonalizable if it is similar to a diagonal matrix $\boldsymbol{D}$, i.e., $\exists$ an invertible $\boldsymbol{P} \in \mathbb{R}^{n \times n}$, such that $\boldsymbol{D} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P}$.

- Definition. $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is orthogonally diagonalizable if it is similar to a diagonal matrix $\boldsymbol{D}$, i.e., $\exists$ an orthogonal $\boldsymbol{P} \in \mathbb{R}^{n \times n}$, such that $\boldsymbol{D} = \boldsymbol{P}^{-1}\boldsymbol{A}\boldsymbol{P} = \boldsymbol{P}^{\mathsf{T}}\boldsymbol{A}\boldsymbol{P}$.

Power of Diagonalization

- $\boldsymbol{A}^k = \boldsymbol{P}\boldsymbol{D}^k\boldsymbol{P}^{-1}$
- $\det(\boldsymbol{A}) = \det(\boldsymbol{P})\det(\boldsymbol{D})\det(\boldsymbol{P}^{-1}) = \det(\boldsymbol{D}) = \prod_i d_{ii}$
- Many other things ...
- Question. Under what condition is $\boldsymbol{A}$ diagonalizable (or orthogonally diagonalizable) and how can we find $\boldsymbol{P}$ (thus $\boldsymbol{D}$)?

Orthogonally Diagonaliable and Symmetric Matrix

Theorem. $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is orthogonally diagonalizable $\iff$ $\boldsymbol{A}$ is symmetric.

- Question. How to find $\boldsymbol{P}$ (thus $\boldsymbol{D}$)?
- Spectral Theorem. If $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is symmetric,
  (a) the eigenvalues are all real
  (b) the eigenvectors to different eigenvalues are perpendicular.
  (c) there exists an orthogonal eigenbasis
- The above implies the columns of $\boldsymbol{P}$ are the n eigenvectors of $\boldsymbol{A}$ because $\boldsymbol{AP} = \boldsymbol{PD}$ and $\boldsymbol{P}^{\mathsf{T}} = \boldsymbol{P}^{-1}$ ($\boldsymbol{P}$ is an orthogonal matrix).

Roadmap

Storyline

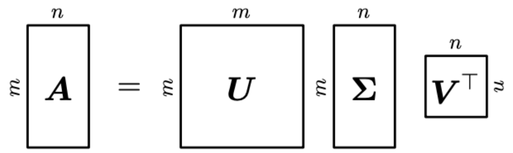
- Eigendecomposition (also called EVD: EigenValue Decomposition): (Orthogonal) Diagonalization for symmetric matrices $\boldsymbol{A} \in \mathbb{R}^{n \times n}$.
- Extensions: Singular Value Decomposition (SVD)
  1. First extension: diagonalization for non-symmetric, but still square matrices $\boldsymbol{A} \in \mathbb{R}^{n \times n}$
  2. Second extension: diagonalization for non-symmeric, and non-square matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
- Background. For $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, a matrix $\boldsymbol{S} := \boldsymbol{A}^\mathsf{T}\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is always symmetric, positive semidefinite.
  - Symmetric, because $\boldsymbol{S}^\mathsf{T} = (\boldsymbol{A}^\mathsf{T}\boldsymbol{A})^\mathsf{T} = \boldsymbol{A}^\mathsf{T}\boldsymbol{A} = \boldsymbol{S}$.
  - Positive semidefinite, because $\boldsymbol{x}^\mathsf{T}\boldsymbol{S}\boldsymbol{x} = \boldsymbol{x}^\mathsf{T}\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{A}\boldsymbol{x})^\mathsf{T}(\boldsymbol{A}\boldsymbol{x}) \geq 0$.

L4(5)

Singular Value Decomposition

- Theorem. $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with rank $r \in [0, \min(m,n)]$. The SVD of $\boldsymbol{A}$ is a decomposition of the form

$$\boldsymbol{A} = \boldsymbol{U}\Sigma\boldsymbol{V}^\top,$$



with an orthogonal matrix $\boldsymbol{U} = \begin{pmatrix}\boldsymbol{u}_1 & \cdots & \boldsymbol{u}_m\end{pmatrix} \in \mathbb{R}^{m \times m}$ and an orthogonal matrix $\boldsymbol{V} = \begin{pmatrix}\boldsymbol{v}_1 & \cdots & \boldsymbol{v}_n\end{pmatrix} \in \mathbb{R}^{n \times n}$. Moreoever, Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geq 0$ and $\Sigma_{ij} = 0, \ i \neq j$, which is uniquely determined for $\boldsymbol{A}$.

- Note
  - The diagonal entries $\sigma_i, \ i = 1, \ldots, r$ are called singular values.
  - $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are called left and right singular vectors, respectively.

L4(5)

SVD: How It Works (for $\boldsymbol{A} \in \mathbb{R}^{n \times n}$)

KAIST AI
Graduate School of AI

- $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ with rank $r \le n$. Then, $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$ is symmetric.
- Orthogonal diagonalization of $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$:

$$\boldsymbol{A}^\mathsf{T}\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\mathsf{T}.$$

- $\boldsymbol{D} = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$ and an orthogonal matrix $\boldsymbol{V} = \begin{pmatrix} \boldsymbol{v}_1 \cdots \boldsymbol{v}_n \end{pmatrix}$, where $\lambda_1 \ge \cdots \ge \lambda_r \ge \lambda_{r+1} = \cdots \lambda_n = 0$ are the eigenvalues of $\boldsymbol{A}^\mathsf{T}\boldsymbol{A}$ and $\{\boldsymbol{v}_i\}$ are orthonormal.
- All λ_i are positive
  $\forall \boldsymbol{x} \in \mathbb{R}^n, \|\boldsymbol{A}\boldsymbol{x}\|^2 = \boldsymbol{A}\boldsymbol{x}^\mathsf{T}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^\mathsf{T}\boldsymbol{A}^\mathsf{T}\boldsymbol{A}\boldsymbol{x} = \lambda_i \|\boldsymbol{x}\|^2$

- $\text{rk}(\boldsymbol{A}) \;=\text{rk}(\boldsymbol{A}^\mathsf{T}\boldsymbol{A}) \;= \text{rk}(D) \;=$r
- Choose $\boldsymbol{U}' = \begin{pmatrix} \boldsymbol{u}_1 & \cdots \boldsymbol{u}_r \end{pmatrix}$, where

$$\boldsymbol{u}_i = \frac{\boldsymbol{A}\boldsymbol{v}_i}{\sqrt{\lambda_i}}, \quad 1 \le i \le r.$$

- We can construct $\{\boldsymbol{u}_i\}$, $i = r+1, \cdots, n$, so that $\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_1 & \cdots \boldsymbol{u}_n \end{pmatrix}$ is an orthonormal basis of $\mathbb{R}^n$.
- Define $\Sigma = \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_n} \end{pmatrix}$
- Then, we can check that $\boldsymbol{U}\Sigma = \boldsymbol{A}\boldsymbol{V}$.
- Similar arguments for a general $\boldsymbol{A} \in \mathbb{R}^{m\times n}$

EVD ($\boldsymbol{A} = \boldsymbol{P}\boldsymbol{D}\boldsymbol{P}^{-1}$) vs. SVD ($\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$)

- SVD: always exists, EVD: square matrix and exists if we can find a basis of eigenvectors (such as symmetric matrices)
- $\boldsymbol{P}$ in EVD is not necessarily orthogonal (only true for symmetric $\boldsymbol{A}$), but $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal (so representing rotations)
- Both EVD and SVD: (i) basis change in the domain, (ii) independent scaling of each new basis vector and mapping from domain to codomain, (iii) basis change in the codomain. The difference: for SVD, different vector spaces of domain and codomain.
- SVD and EVD are closely related through their projections
  - The left-singular (resp. right-singular) vectors of $\boldsymbol{A}$ are eigenvectors of $\boldsymbol{A}\boldsymbol{A}^\top$ (resp. $\boldsymbol{A}^\top\boldsymbol{A}$)
  - The singular values of $\boldsymbol{A}$ are the square roots of eigenvalues of $\boldsymbol{A}\boldsymbol{A}^\top$ and $\boldsymbol{A}^\top\boldsymbol{A}$
  - When $\boldsymbol{A}$ is symmetric, EVD $=$ SVD (from spectral theorem)

L4(5)

Questions?

L4(5)

Lecture 2: Convex Optimization

Jinwoo Shin

Kim Jaechul Graduate School of AI, KAIST

Roadmap

Summary

- Training machine learning models $=$ finding a good set of parameters
- A good set of parameters $=$ Solution (or close to solution) to some optimization problem
- Directions: Unconstrained optimization, Constrained optimization, Convex optimization
- High-school math: A necessary condition for the optimal point: $f'(x) = 0$ (stationary point)
    - Gradient will play an important role

Roadmap

Unconstrained Optimization and Gradient Algorithms

KAIST AI
Graduate School of AI

- Goal

$$\min f(\boldsymbol{x}), \quad f(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}, \quad f \in C^1$$

- Graident-type algorithms

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k \boldsymbol{d}_k, \quad k = 0, 1, 2, \dots$$

- Lemma. Any direction $\boldsymbol{d} \in \mathbb{R}^{n \times 1}$ that satisfies $\nabla f(\boldsymbol{x}) \cdot \boldsymbol{d} < 0$ is a descent direction of f at $\boldsymbol{x}$. That is, if we let $\boldsymbol{x}_\alpha = \boldsymbol{x} + \alpha \boldsymbol{d}$, $\exists \bar{\alpha} > 0$, such that for all $\alpha \in (0, \bar{\alpha}]$, $f(\boldsymbol{x}_\alpha) < f(\boldsymbol{x})$.
- Steepest gradient descent[1]. $\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k)^\mathsf{T}$.
- Finding a local optimum $f(\boldsymbol{x}_\star)$, if the step-size γ_k is suitably chosen.

[1] In some cases, just gradient descent often means this steepest gradient descent.

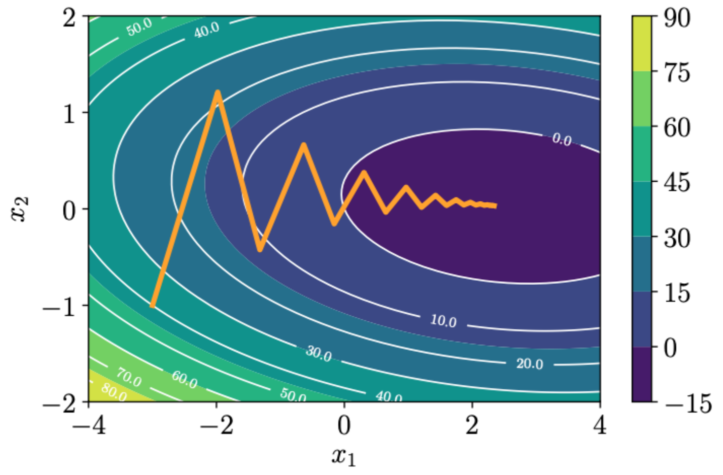
Example

- A quadratic function $f : \mathbb{R}^2 \mapsto \mathbb{R}$.

$$f\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) = \frac{1}{2}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} 2 & 1 \\ 1 & 20 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} 5 \\ 3 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

whose gradient is $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{\mathsf{T}} \begin{pmatrix} 2 & 1 \\ 1 & 20 \end{pmatrix} - \begin{pmatrix} 5 \\ 3 \end{pmatrix}^{\mathsf{T}}$

- $\boldsymbol{x}_0 = (-3 - 1)^{\mathsf{T}}$
- constant step size $\alpha = 0.085$
- Zigzag pattern



Taxonomy

- Goal: min $L(\boldsymbol{\theta})$ for n training data
- Based on the amount of training data used for each iteration
  - Batch gradient descent (the entire n)
  - Mini-batch gradient descent($k < n$ data)
  - Stochastic gradient descent ($k < n$ data with unbiased gradient estimation)
- Based on the adaptive method of update
  - Momentum, NAG, Adagrad, RMSprop, Adam, etc
- `https://ruder.io/optimizing-gradient-descent/`

Stochastic Gradient Descent (SGD)

- Assume $L(\boldsymbol{\theta}) = \sum_{i=1}^{n} L_n(\boldsymbol{\theta})$ (which happens in many cases in machine learning)
- Gradient update

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \gamma_k \nabla L(\boldsymbol{\theta}_k)^{\mathsf{T}} = \boldsymbol{\theta}_k - \gamma_k \sum_{n=1}^{N} \nabla L_n(\boldsymbol{\theta}_k)^{\mathsf{T}}$$

  - Batch gradient: $\sum_{n=1}^{N} \nabla L_n(\boldsymbol{\theta}_k)^{\mathsf{T}}$
  - Mini-batch gradient: $\sum_{n\in\mathcal{K}} \nabla L_n(\boldsymbol{\theta}_k)^{\mathsf{T}}$ for a suitable choice of $\mathcal{K}$, $|\mathcal{K}| < n$
  - Stochastic gradient: Randomly choose the subset $\mathcal{K}$ of mini-batch gradient such that

$$\sum_{n=1}^{N} \nabla L_n(\boldsymbol{\theta}_k)^{\mathsf{T}} = E\left[\sum_{n\in\mathcal{K}} \nabla L_n(\boldsymbol{\theta}_k)^{\mathsf{T}}\right]$$

i.e., noisy approximation to the real gradient.

L7(1)

Adaptivity for Better Convergence: Momemtum

KAIST AI
Graduate School of AI

- Step size.
  - Too small: slow update, Too big: overshoot, zig-zag, often fail to converge
- Adaptive update: smooth out the erratic behavior and dampens oscillations
- Gradient descent with momentum

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_i \nabla f(\boldsymbol{x}_k)^{\mathsf{T}} + \alpha \Delta \boldsymbol{x}_k, \quad \alpha \in [0,1]$$
$$\Delta \boldsymbol{x}_k = \boldsymbol{x}_k - \boldsymbol{x}_{k-1}$$

  - Memory term: $\alpha \Delta \boldsymbol{x}_k$, where α is the degree of how much we remember the past
  - Next update = a linear combination of current and previous updates

Roadmap

(1) Optimization Using Gradient Descent

(2) Constrained Optimization and Lagrange Multipliers

(3) Convex Sets and Functions

(4) Convex Optimization

Standard Constrained Optimization Problem

- An optimization problem in standard form:

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) \\ &\text{subject to } g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \ldots, m \quad \textit{(Inequality constraints)} \\ &\qquad\qquad\quad h_j(\boldsymbol{x}) = 0, \quad j = 1, 2, \ldots, p \quad \textit{(Equality constraints)} \end{aligned}$$

- Variables: $\boldsymbol{x} \in \mathbb{R}^n$. Assume nonempty feasible set

- Optimal value: p^*. Optimizer: $\boldsymbol{x}^*$

Problem Solving via Langrange Multipliers

- Duality Mentality
  - Bound or solve an optimization problem via a different optimization problem!
  - We'll develop the basic Lagrange duality theory for a general optimization problem, then specialize for convex optimization
- Idea: augment the objective with a weighted sum of constraints
  - Lagrangian:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) + \sum_{i=1}^{p} \nu_i h_i(\boldsymbol{x})$$

  - Lagrange multipliers (dual variables): $\boldsymbol{\lambda} = (\lambda_i : i = 1, \cdots, m) \succeq 0$, $\boldsymbol{\nu} = (\nu_1, \cdots, \nu_p)$
  - Lagrange dual function:

$$\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$$

Lower Bound on the Optimal Value

- The dual function $\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu})$ is a lower bound on the optimal value p^*.
- Theorem. $\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \le p^*, \quad \forall \boldsymbol{\lambda} \succeq 0, \ \boldsymbol{\nu}$
- Proof. Consider a feasible $\tilde{\boldsymbol{x}}$. Then,

$$\mathcal{L}(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{m} \lambda_i g_i(\tilde{\boldsymbol{x}}) + \sum_{i=1}^{p} \nu_i h_i(\tilde{\boldsymbol{x}}) \le f(\tilde{\boldsymbol{x}})$$

since $g_i(\tilde{\boldsymbol{x}}) \le 0$, $\lambda_i \ge 0$ and $h_i(\tilde{\boldsymbol{x}}) = 0$.
Hence, $\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \le \mathcal{L}(\tilde{\boldsymbol{x}}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \le f(\tilde{\boldsymbol{x}})$ for all feasible $\tilde{\boldsymbol{x}}$. Therefore, $\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu}) \le p^*$.

Lagrangian Dual Problem

- Lower bound from Lagrange dual function depends on $(\boldsymbol{\lambda}, \boldsymbol{\nu})$.
- Question. What's the best lower bound?

  Langrangian dual problem $\quad$ maximize $\quad \mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu})$

  subject to $\quad \boldsymbol{\lambda} \succeq 0$

- Dual variables: $(\boldsymbol{\lambda}, \boldsymbol{\nu})$
- Always a convex optimization, because $\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu})$ is always concave over $\boldsymbol{\lambda}, \boldsymbol{\nu}$.
  - Infimum over $\boldsymbol{x}$ of a family of affine functions in $(\boldsymbol{\lambda}, \boldsymbol{\nu})$ (we will see this later)
- Denote the optimal value of Lagrange dual problem by d^*.

L7(2)

Weak Duality

- What's the relationship between d^* and p^*?

Weak Duality

$d^* \leq p^*$

- Weak duality always hold (even if the primal problem is not convex):
- Optimal duality gap: $p^* - d^*$
- Efficient generation of the lower bounds through the dual problem

Roadmap

Convex Optimization

KAIST AI
Graduate School of AI

- Convex optimization problem

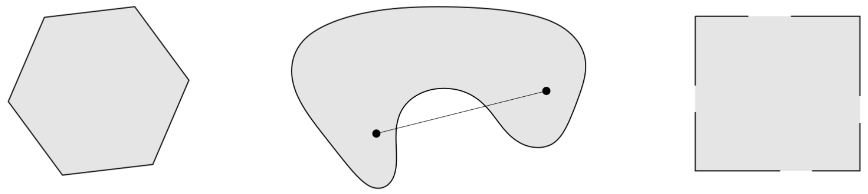
$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{X}, \end{aligned}$$

  where $f(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ is a convex function, and $\mathcal{X}$ is a convex set.

- The watershed between easily solvable problem and intractable ones is not 'linearity', but 'convexity'
- Let's overview the background of convex functions, convex sets, and their basic properties.

Convex Set

- Set $\mathcal{C}$ is a convex set if the line segment between any two points in $\mathcal{C}$ lies in $\mathcal{C}$, i.e., if for any $x_1, x_2 \in \mathcal{C}$ and any $\theta \in [0,1]$, we have $\theta x_1 + (1-\theta)x_2 \in \mathcal{C}$
- Examples of convex and non-convex sets



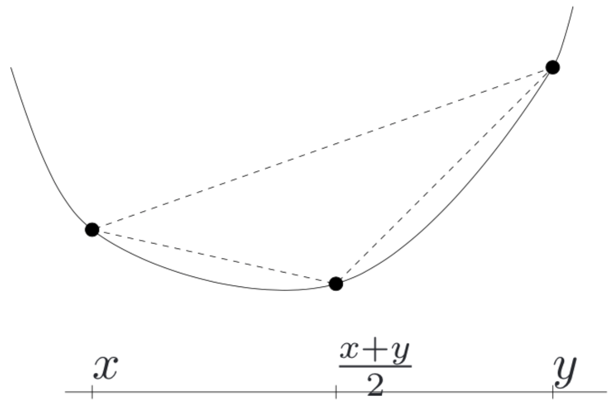
Convex Functions

- $f : \mathbb{R}^n \mapsto \mathbb{R}$ is a convex function if dom f is a convex set and for all $x, y \in$ dom f and $\theta \in [0,1]$, we have

  $f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y)$

- f is strictly convex if the strict inequality in the above holds for all $x \neq y$ and $0 < \theta < 1$.

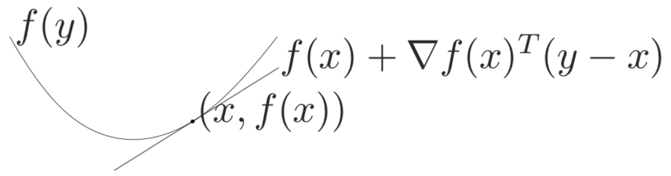
- f is concave if $-f$ is convex
- Affine functions are convex and concave
- Jensen's inequality. For a rv X, $f(\mathbb{E}[X]) \le \mathbb{E}[f(X)]$.



Conditions of Convex Functions (1)

- First-order condition. For differentiable functions, f is convex iff

  $f(y) - f(x) \geq \nabla f(x)^{\mathsf{T}}(y - x), \quad \forall x, y \in \text{dom } f$, and dom f is convex



- Example. $f(y) = y^2$.
- $f(y) \geq \tilde{f}_x(y)$ where $\tilde{f}_x(y)$ is the first order Taylor expansion of $f(y)$ at x.
- Local information (first order Taylor approximation) about a convex function provides global information (global underestimator).
- If $\nabla f(x) = 0$, then $f(y) \geq f(x)$, $\forall y$. Thus, x is a global minimizer of f

Conditions of Convex Functions (2)

- Second-order condition. For twice differentiable functions, f is convex iff

  $\nabla^2 f(x) \succeq 0$

  for all $x \in \operatorname{dom} f$ (upward slope) and $\operatorname{dom} f$ is convex
- Example: $f(x) = x^2$.
- Meaning: The graph of the function have positive (upward) curvature at x.

Examples of Convex or Concave Functions

- e^{ax} is convex on $\mathbb{R}$, for any $a \in \mathbb{R}$
- x^a is convex on $\mathbb{R}_{++}$ when $a \geq 1$ or $a \leq 0$, and concave for $0 \leq a \leq 1$
- $|x|^p$ is convex on $\mathbb{R}$ for $p \geq 1$
- $\log x$ is concave on $\mathbb{R}_{++}$
- $x \log x$ is strictly convex on $\mathbb{R}_{++}$
- Every norm on $\mathbb{R}^n$ is convex
- $f(x) = \max\{x_1, \ldots, x_n\}$ is convex on $\mathbb{R}^n$
- $f(x) = \log \sum_{i=1}^{n} e^{x_i}$ is convex on $\mathbb{R}^n$
- $f(x) = (\prod_{i=1}^{n} x_i)^{\frac{1}{n}}$ is concave on $\mathbb{R}^n_{++}$

L7(3)

Convexity-Preserving Operations

- $f = \sum_{i=1}^{n} w_i f_i$ convex if f_i are all convex and $w_i \geq 0$
- $g(x) = f(Ax + b)$ is convex iff $f(x)$ is convex
- $f(x) = \max\{f_1(x), f_2(x)\}$ convex if f_i convex, e.g., sum of r largest components is convex
- $f(x) = h(g(x))$, where $h : \mathbb{R}^k \mapsto \mathbb{R}$ and $g : \mathbb{R}^n \mapsto \mathbb{R}^k$.

  If $k = 1$: $f''(x) = h''(g(x))g'(x)^2 + h'(g(x))g''(x)$. So, f is convex if h is convex and nondecreasing and g is convex, or if h is convex and nonincreasing and g is concave ...

Point-wise Supremum

KAIST AI
Graduate School of AI

- If $f(x, y)$ is convex in x for each $y \in \mathcal{A}$, then
$$g(x) = \sup_{y \in \mathcal{A}} f(x, y)$$
is convex. Similarly, if $f(x, y)$ is concave in x for each $y \in \mathcal{A}$, then
$$g(x) = \inf_{y \in \mathcal{A}} f(x, y)$$
is concave.
- Example. distance to farthest point in a set $\mathcal{C}$: $f(x) = \sup_{y \in \mathcal{C}} ||x - y||$ is convex.
- Example. Lagrange dual function
$$\mathcal{D}(\boldsymbol{\lambda}, \boldsymbol{\nu}) = \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu})$$
is concave.

Roadmap

Standard Convex Optimization

- A standard convex optimization problem with variables $\boldsymbol{x}$:

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq 0, \quad i = 1, 2, \dots, m \\ & {a_i}^{\mathsf{T}} \boldsymbol{x} = b_i, \quad i = 1, 2, \dots, p \end{array}$$

  where $f, g_1, \dots, g_m$ are convex functions.

- Minimize convex objective function (or maximize concave objective function)
- Upper bound inequality constraints on convex functions ($\Rightarrow$ Constraint set is convex)
- Equality constraints must be affine (Only affine functions leads to a convex set for the equality constraints)

Strong Duality

- Strong duality (zero optimal duality gap):

  $d^* = p^*$

- If strong duality holds, solving dual is ‘equivalent’ to solving primal. But strong duality does not always hold
- Convexity and constraint qualifications $\Longrightarrow$ Strong duality
- Another reason why convex optimization is ‘easy’

L7(4)

KKT Condition

- Since $\boldsymbol{x}^*$ minimizes $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}^*, \boldsymbol{\nu}^*)$ over $\boldsymbol{x}$,

$$\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{m} \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\boldsymbol{x}^*) = 0$$

Karush-Kuhn-Tucker optimality condition

$$g_i(\boldsymbol{x}^*) \leq 0, \quad h_i(\boldsymbol{x}^*) = 0, \quad \lambda_i^* \succeq 0$$

$$\lambda_i^* g_i(\boldsymbol{x}^*) = 0$$

$$\nabla f(\boldsymbol{x}^*) + \textstyle\sum_{i=1}^{m} \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(\boldsymbol{x}^*) = 0$$

- Any optimization with strong duality, KKT condition is necessary for primal-dual optimality
- Convex optimization (with Slater's condition), KKT is also sufficient for primal-dual optimality.

L7(4)

Linear Programming

- Primal problem

$$\begin{array}{ll} \min_{\boldsymbol{x}\in\mathbb{R}^d} & \boldsymbol{c}^\top \boldsymbol{x} \\ \text{subject to} & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}, \end{array}$$

where $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $\boldsymbol{b} \in \mathbb{R}^m$.

- Dual problem

$$\begin{array}{ll} \max_{\boldsymbol{\lambda}\in\mathbb{R}^m} & -\boldsymbol{b}^\top \boldsymbol{\lambda} \\ \text{subject to} & \boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = 0, \ \ \boldsymbol{\lambda} \succeq 0, \end{array}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$.

- The Lagrangian: $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda})^\top \boldsymbol{x} - \boldsymbol{\lambda}^\top \boldsymbol{b}$, whose derivative w.r.t. $\boldsymbol{x}$ becomes zero, when $\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda} = 0$.
- The dual function: $\mathcal{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^\top \boldsymbol{b}$

Quadratic Programming

KAIST AI
Graduate School of AI

- Primal problem

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathbb{R}^d} \quad & \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + c^\top \boldsymbol{x} \\ \text{subject to} \quad & \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}, \end{aligned}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times d}$, $\boldsymbol{b} \in \mathbb{R}^m$, $\boldsymbol{c} \in \mathbb{R}^d$, the square matrix $\boldsymbol{Q}$ is symmetric, positive definite.

- Dual problem

$$\begin{aligned} \max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \quad & \left(-\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda})^\top \boldsymbol{A}\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^\top \boldsymbol{\lambda}) - \boldsymbol{\lambda}^\top \boldsymbol{b} \right) \\ \text{subject to} \quad & \boldsymbol{\lambda} \succeq 0, \end{aligned}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$.

L7(4)

Roadmap

Questions?

L7(5)

Lecture 3: Principal Component Analysis

Jinwoo Shin

Kim Jaechul Graduate School of AI, KAIST

Roadmap

(1) Problem Setting

(2) Maximum Variance Perspective

(3) Eigenvector Computation and Low-Rank Approximations

(4) PCA in High Dimensions

Roadmap

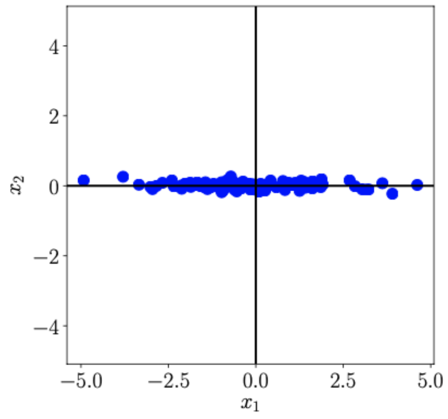
(1) Problem Setting

(2) Maximum Variance Perspective

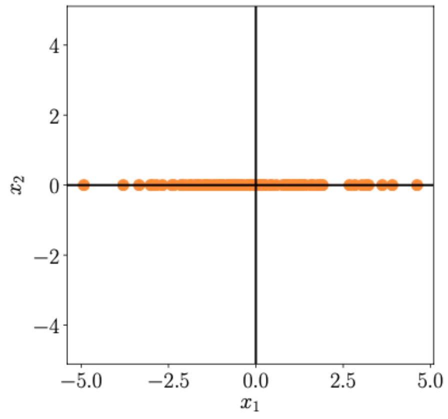
(3) Eigenvector Computation and Low-Rank Approximations

(4) PCA in High Dimensions

Dimensionality Reduction



(a) Dataset with x_1 and x_2 coordinates.



(b) Compressed dataset where only the x_1 coordinate is relevant.

- High-dimensional data
  - hard to analyze and visualize
  - Often, overcomplete and many dimensionas are redundant
- Compact data representation is always preferred just like compression.
- PCA (Principal Component Analysis) is a representative method.

L10(1)

Example: Housing Data

- 5 dimensions
  1. Size
  2. Number of rooms
  3. Number of bathrooms
  4. Schools around
  5. Crime rate
- 2 dimensions
  - Size feature
  - Location feature

PCA Algorithm

KAIST AI
Graduate School of AI

S1. Centering. Centering the data by subtracting mean

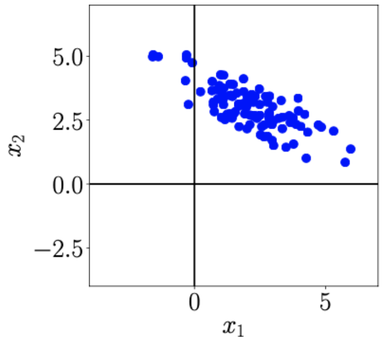
S2. Standardization. Divide the data points by the standard deviation for every dimension (original feature) $d = 1, \dots, D$

S3. Eigenvalue/vector. Compute the M-largest eigenvalues and the eigenvectors of the data covariance matrix (M is the dimension that needs to be reduced)

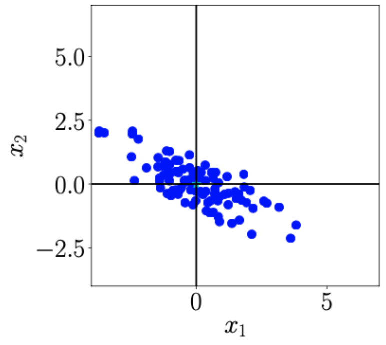
S4. Projection. Project all data points onto the space defined by the eigenvectors (i.e., principal subspace).

S5. Undo standardization and centering.

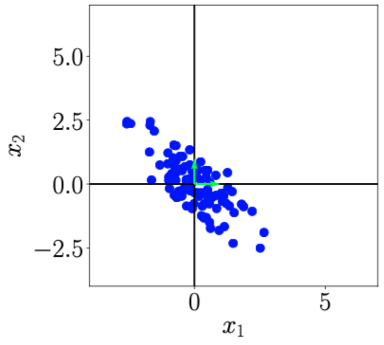
PCA Illustration



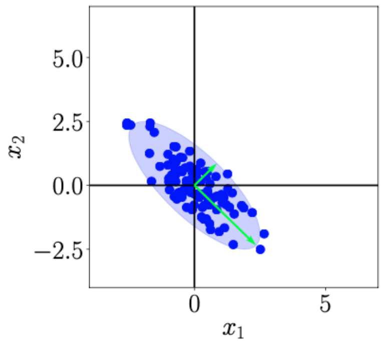
(a) Original dataset.



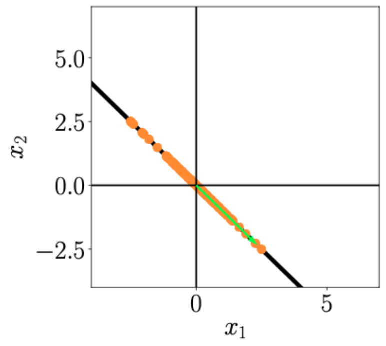
(b) Step 1: Centering by subtracting the mean from each data point.



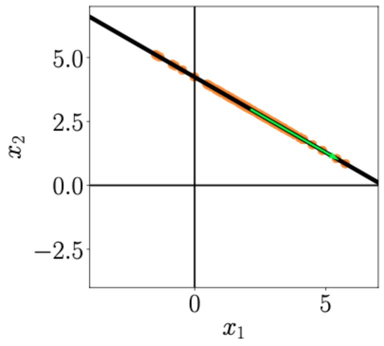
(c) Step 2: Dividing by the standard deviation to make the data unit free. Data has variance 1 along each axis.



(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).



(e) Step 4: Project data onto the principal subspace.



(f) Undo the standardization and move projected data back into the original data space from (a).

L10(1)

Data Matrix and Data Covariance Matrix

- N: number of samples, D: number of measurements (or original features)
- iid dataset $\mathcal{X} = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_N\}$ whose mean is 0 (well-centered), where each $\boldsymbol{x}_i \in \mathbb{R}^D$, and its corresponding data matrix

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_N \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,N} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,N} \\ & \vdots & & \\ x_{D,1} & x_{D,2} & \cdots & x_{D,N} \end{pmatrix} \in \mathbb{R}^{D \times N}$$

- (data) covariance matrix

$$\boxed{\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\mathsf{T} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\mathsf{T} \in \mathbb{R}^{D \times D}}$$

L10(1)

Code: Low Dimensional Representation

KAIST AI
Graduate School of AI

- Low-dimensional compressed representation, also called code:

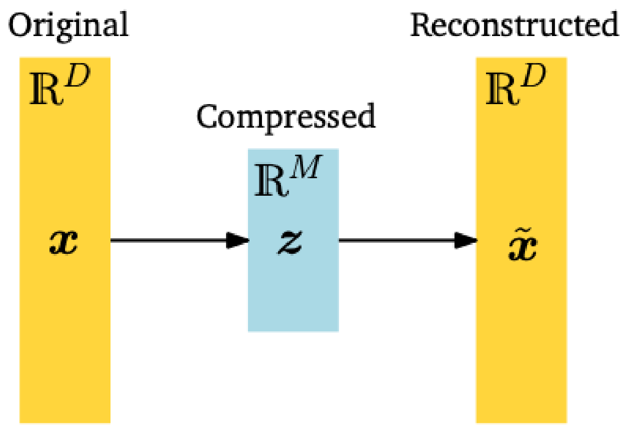
$$\boldsymbol{z}_n = \boldsymbol{B}^\top \boldsymbol{x}_n \in \mathbb{R}^M,$$

  where the projection[1] matrix is $\boldsymbol{B} := (\boldsymbol{b}_1, \dots, \boldsymbol{b}_M) \in \mathbb{R}^{D \times M}$,
- Assume that the columns of $\boldsymbol{B}$ are orthonormal, i.e., $\boldsymbol{b}_i^\top \boldsymbol{b}_j = 0$ if $i \neq j$, and $\boldsymbol{b}_i^\top \boldsymbol{b}_i = 1$ if $i = j$.
- Seek an M-dimensional subspace $U \subset \mathbb{R}^D$, $\dim(U) = M < D$ onto which we project data
- $\tilde{\boldsymbol{x}}_n \in \mathbb{R}^D$: projected data, $\boldsymbol{z}_n$: their coordinates w.r.t. the basis vectors of $\boldsymbol{B}$.

[1] In L3(8), the coordinate in the projected space becomes $\boldsymbol{\lambda} = (\boldsymbol{B}^\top \boldsymbol{B})^{-1} \boldsymbol{B}^\top \boldsymbol{x}$, which is simply $\boldsymbol{B}^\top \boldsymbol{x}$ for orthonormal bases $\boldsymbol{B}$.

L10(1)

PCA: Encoder and Decoder Viewpoint



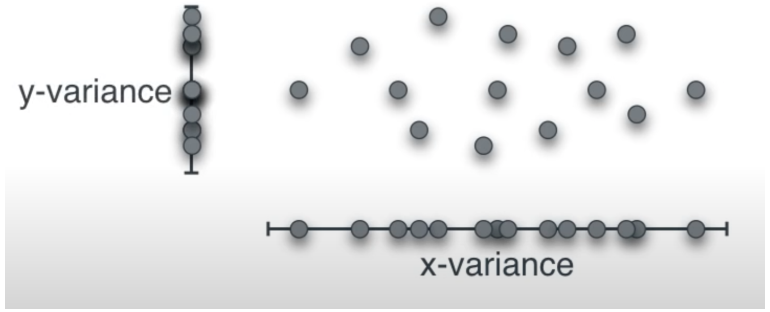
- Find a suitable matrix B such that $\boldsymbol{z} = \boldsymbol{B}^\mathsf{T}\boldsymbol{x}$ and $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z}$
- $\boldsymbol{B}^\mathsf{T}$: encoder, $\boldsymbol{B}$: decoder
- Example. MNIST dataset
  - handwritten digits, $N = 60,000$ data samples, $D = 28 \times 28 = 784$ pixels

L10(1)

Roadmap

Idea

- Information content in the data
  - space filling
  - information in the data by looking at how much data is spread out
- PCA
  - a dimensinoality reduction algorithm that maximizes the variance in the low-dimensional data representation.



source: Youtube channel by Luis Serrano

Matrix Again: B, $\boldsymbol{z}_n$, and $\boldsymbol{x}_n$

- $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \dots & \boldsymbol{b}_M \end{pmatrix}$, where $\boldsymbol{b}_i \in \mathbb{R}^D$ and $\boldsymbol{B} \in \mathbb{R}^{D \times M}$
- $\boldsymbol{B}^\mathsf{T} = \begin{pmatrix} \boldsymbol{b}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{b}_M^\mathsf{T} \end{pmatrix} \in \mathbb{R}^{M \times D}$, $\boldsymbol{b}_i^\mathsf{T} \in \mathbb{R}^{1 \times D}$, $\boldsymbol{x}_i \in \mathbb{R}^{D \times 1}$
- $\boldsymbol{z}_n = \begin{pmatrix} z_{1n} \\ \vdots \\ z_{Mn} \end{pmatrix} = \boldsymbol{B}^\mathsf{T} \boldsymbol{x}_n = \begin{pmatrix} \boldsymbol{b}_1^\mathsf{T} \\ \vdots \\ \boldsymbol{b}_M^\mathsf{T} \end{pmatrix} \boldsymbol{x}_n = \begin{pmatrix} \boldsymbol{b}_1^\mathsf{T} \boldsymbol{x}_n \\ \vdots \\ \boldsymbol{b}_M^\mathsf{T} \boldsymbol{x}_n \end{pmatrix}$
- z_{in}: new coordinate (for $\boldsymbol{x}_n$) in the projected space by the basis $\boldsymbol{b}_i$

What We Will Do Is ...

KAIST AI
Graduate School of AI

- Goal: Find the orthonormal bases $\boldsymbol{B} = \begin{pmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \dots & \boldsymbol{b}_M \end{pmatrix}$ that maximizes the variance.
- Result: For the M-largest eigenvalues $\lambda_1, \dots, \lambda_M$ of the data covariance matrix $\boldsymbol{S}$, their corresponding M eigenvectors become $\boldsymbol{b}_1, \dots, \boldsymbol{b}_M$
- Question. Why data covariance matrix? Why eigenvectors ordered by their eigenvalues?
- Strategy: Induction

  Step 1. We seek a single vector $\boldsymbol{b}_1$ that maximizes the variance of the projected data, assuming that we project the data onto an 1D line. We show that $\boldsymbol{b}_1$ is the eigenvector of the largest eigenvalue.

  Step k. Suppose that we found $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{k-1}$ for the variance maximization. Then, we seek $\boldsymbol{b}_k$ that maximizes the variance of the projected data onto k-D plain with the constraint that $\boldsymbol{b}_k$ is orthogonal to $\boldsymbol{b}_1, \dots, \boldsymbol{b}_{k-1}$. We prove that $\boldsymbol{b}_k$ is the eigenvector of the k-th largest eigenvalue.

L10(2)

Step 1: Finding $\boldsymbol{b}_1$ (1)

KAIST AI
Graduate School of AI

- Variance (over N sample data) of the first coordinate z_1 of $\boldsymbol{z} \in \mathbb{R}^M$, so that

$$V_1 := \mathrm{var}[z_1] = \frac{1}{N}\sum_{n=1}^{N} z_{1n}^2, \quad z_{1n} = \boldsymbol{b}_1^\top \boldsymbol{x}_n$$

where z_{1n} (z_{in}) is the first (i-th) coordinate of the low-dimensional representation $\boldsymbol{z}_n$ of $\boldsymbol{x}_n$

$$V_1 = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{b}_1^\top \boldsymbol{x}_n)^2 = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{b}_1^\top \boldsymbol{x}_n \boldsymbol{x}_n^\top \boldsymbol{b}_1 = \boldsymbol{b}_1^\top \left(\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n \boldsymbol{x}_n^\top\right)\boldsymbol{b}_1 = \boldsymbol{b}_1^\top \boldsymbol{S}\boldsymbol{b}_1$$

- Find $\boldsymbol{b}_1$ that maximizes V_1.

$$\max_{\boldsymbol{b}_1} \boldsymbol{b}_1^\top \boldsymbol{S}\boldsymbol{b}_1, \quad \text{subject to} \quad \|\boldsymbol{b}_1\|^2 = 1$$

Step 1: Finding $\boldsymbol{b}_1$ (2)

- Optimization problem

$$\max_{\boldsymbol{b}_1} \boldsymbol{b}_1^{\mathsf{T}} \boldsymbol{S} \boldsymbol{b}_1, \quad \text{subject to} \quad \|\boldsymbol{b}_1\|^2 = 1$$

- Using the Lagrange multiplier method, we get: L7(2), L7(4)

$$\boldsymbol{S}\boldsymbol{b}_1 = \lambda_1 \boldsymbol{b}_1, \quad \boldsymbol{b}_1^{\mathsf{T}} \boldsymbol{b}_1 = 1 \implies \lambda_1\text{: eigenvalue, } \boldsymbol{b}_1\text{: eigenvector of } \boldsymbol{S}$$

- Then, $V_1 = \boldsymbol{b}_1^{\mathsf{T}} \boldsymbol{S} \boldsymbol{b}_1 = \lambda_1 \boldsymbol{b}_1^{\mathsf{T}} \boldsymbol{b}_1 = \lambda_1$ (the variance V_1 is the eigenvalue of S)
- To maximize the variance, we take the largest eigenvalue, and the corresponding eigenvector is called the (first) principal component.

Step k: Finding $\boldsymbol{b}_k$ (1)

KAIST AI
Graduate School of AI

- Finding k-th principal component: Solving the following optimization problem

$$\boxed{\max_{\boldsymbol{b}} \boldsymbol{b}^\top \boldsymbol{S}\boldsymbol{b}, \quad \text{subject to} \quad \boldsymbol{b}^\top \boldsymbol{b} = 1 \text{ and } \boldsymbol{b}^\top \boldsymbol{b}_i = 0, \ i = 1, \dots, k-1}$$

- Claim. The solution of the above is the eigenvector of $\boldsymbol{S}$ corresponding to its k-th largest eigenvalue.
- Proof. By induction hypothesis, $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k$ are the orthonormal eigenvectors of $\boldsymbol{S}$. Denote the i-th largest eigenvalue of $\boldsymbol{S}$ by λ_i, where note that $\boldsymbol{S}\boldsymbol{b}_i = \lambda_i \boldsymbol{b}_i$.
  The lagrangian of the objective function is:

$$\mathcal{L}(\boldsymbol{b}) = \boldsymbol{b}^\top \boldsymbol{S}\boldsymbol{b} - \lambda(\boldsymbol{b}^\top \boldsymbol{b} - 1) + \sum_{i=1}^{k} \eta_i \boldsymbol{b}^\top \boldsymbol{b}_i$$

L10(2)

Step k: Finding $\boldsymbol{b}_k$ (2)

- Letting the solution be denoted by $\boldsymbol{b}_{k+1}$, the first-order necessary condition for optimality is:

$$\nabla \mathcal{L}(\boldsymbol{b}_{k+1}) = 2\boldsymbol{S}\boldsymbol{b}_{k+1} - 2\lambda \boldsymbol{b}_{k+1} + \sum_{i=1}^{k} \eta_i \boldsymbol{b}_i = 0 \qquad (*)$$

- Now, for any $j \in \{1, \dots, k\}$,

$$0 = \boldsymbol{b}_j^\mathsf{T} \nabla \mathcal{L}(\boldsymbol{b}_{k+1}) = 2\boldsymbol{b}_j^\mathsf{T} \boldsymbol{S}\boldsymbol{b}_{k+1} - 2\lambda \boldsymbol{b}_j^\mathsf{T} \boldsymbol{b}_{k+1} + \sum_{i=1}^{k} \eta_i \boldsymbol{b}_j^\mathsf{T} \boldsymbol{b}_i = 2(\boldsymbol{S}\boldsymbol{b}_j)^\mathsf{T} \boldsymbol{b}_{k+1} + \eta_j$$
$$= 2(\lambda \boldsymbol{b}_j)^\mathsf{T} \boldsymbol{b}_{k+1} + \eta_j = 2\lambda \boldsymbol{b}_j^\mathsf{T} \boldsymbol{b}_{k+1} + \eta_j = \eta_j$$

- From $\eta_j = 0$ and (*), $\boldsymbol{S}\boldsymbol{b}_{k+1} = \lambda \boldsymbol{b}_{k+1}$. $\Longrightarrow$ λ is an eigenvalue and its corresponding eigenvector is $\boldsymbol{b}_{k+1}$.
- Note that the objective function is λ, because $\boldsymbol{b}^\mathsf{T} \boldsymbol{S}\boldsymbol{b} = \lambda \boldsymbol{b}^\mathsf{T} \boldsymbol{b}$.

L10(2)

Step k: Finding $\boldsymbol{b}_k$ (3)

KAIST AI
Graduate School of AI

- Question. How can we choose the largest λ with the constraint that $\boldsymbol{b}_{k+1} \perp (\boldsymbol{b}_1, \dots \boldsymbol{b}_k)$?
- Clearly, if $\boldsymbol{b}_{k+1}$ is equal to any of these eigenvectors (up to sign), the constraint will be violated, so, to maximize λ, $\boldsymbol{b}_{k+1}$ should be a unit eigenvector of $\boldsymbol{S}$ corresponding to $(k+1)$-th largest eigenvalue.
- By spectral theorem, we can choose this vector in such a way that it is orthogonal to $\boldsymbol{b}_1, \dots, \boldsymbol{b}_k$. L4(4)

Roadmap

Eigenvector Computation

KAIST AI
Graduate School of AI

- Approach 1: EVD — L4(4)
  - Perform an eigendecomposition and compute the eigenvalues and eigenvectors of the symmetric matrix $\boldsymbol{S}$ directly.
- Approach 2: SVD — L4(5)
  - SVD of the data matrix $\boldsymbol{X}$: $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top$ $([D \times N] = [D \times D] \cdot [D \times N] \cdot [N \times N])$
  - $\boldsymbol{U}$ and $\boldsymbol{V}^\top$: orthogonal matrices, $\boldsymbol{\Sigma}$: only nonzero entries are the singular values $\sigma_{ii} \geq 0$.

$$\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top = \frac{1}{N}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top \overset{(\boldsymbol{V}^\top=\boldsymbol{V}^{-1})}{=} \frac{1}{N}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^\top\boldsymbol{U}^\top$$

  - The columns of $\boldsymbol{U}$ are the eigenvectors of $\boldsymbol{X}\boldsymbol{X}^\top$ (thus $\boldsymbol{S}$)
  - The eigenvalues λ_d of $\boldsymbol{S}$ are related to the singular values of $\boldsymbol{X}$: $\lambda_d = \dfrac{\sigma_d^2}{N}$

L10(4)

PCA as Low-Rank Matrix Approximations

- In SVD, $\boldsymbol{U}$ corresponds to the projection matrix $\boldsymbol{B}$, so that we maximize the variance of the projected data or minimize the average squared reconstruction error.
- Consider the best rank-M approximation

$$\tilde{\boldsymbol{X}}_M := \arg \min_{\mathrm{rk}(\boldsymbol{A})=M} \|\boldsymbol{X} - \boldsymbol{A}\|_2$$

- From Eckart-Young Theorem, by truncating the SVD at the top-M singular value, we obtain the reconstructed data matrix $\tilde{\boldsymbol{X}}_M$ as: L4(5), L4(6)

$$\tilde{\boldsymbol{X}}_M = \overbrace{\boldsymbol{U}_M}^{D\times M} \overbrace{\boldsymbol{\Sigma}_M}^{M\times M} \overbrace{\boldsymbol{V}_M}^{M\times N}{}^{\mathsf{T}} \iff \tilde{\boldsymbol{X}}_M = \sum_{i=1}^{M} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i{}^{\mathsf{T}},$$

where σ_i is the i-th singular value.

L10(4)

Roadmap

PCA in High Dimensions

KAIST AI
Graduate School of AI

- In some practical cases, $\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{D\times D}$, where D is pretty high.
  - Example. 100×100 pixel image: $D = 10,000$.
- What if $N << D$?
  - With no duplicate data, $\text{rk}(\boldsymbol{S}) = N$, and $D - N + 1$ eigenvalues are 0! $\implies$ no need to maintain $D \times D$ data covariance matrix.
- In PCA, $\boldsymbol{S}\boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m$, $m = 1, \ldots, M$.

$$\boldsymbol{S}\boldsymbol{b}_m = \frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{b}_m = \lambda_m\boldsymbol{b}_m \quad \implies \quad \frac{1}{N}\underbrace{\boldsymbol{X}^\top\boldsymbol{X}}_{N\times N}\underbrace{\boldsymbol{X}^\top\boldsymbol{b}_m}_{:=\boldsymbol{c}_m} = \lambda_m\boldsymbol{X}^\top\boldsymbol{b}_m \iff \frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{c}_m = \lambda_m\boldsymbol{c}_m$$

- λ_m is an eigenvalue of $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}$ with its associated eigenvector $\boldsymbol{c}_m = \boldsymbol{X}^\top\boldsymbol{b}_m$
- $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X} \in \mathbb{R}^{N\times N}$, so much easier to compute the eigenstuff
- To recover the eigenvector of $\boldsymbol{S}$, by left-multiplying X, we get $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{c}_m = \lambda_m\boldsymbol{X}\boldsymbol{c}_m$

Questions?