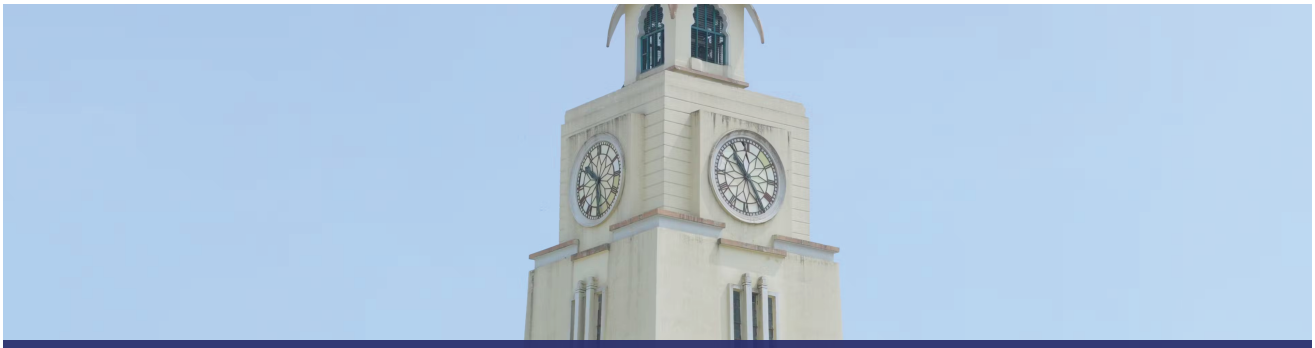


BITS Pilani
Pilani | Dubai | Goa | Hyderabad | Mumbai

WORK INTEGRATED LEARNING PROGRAMMES

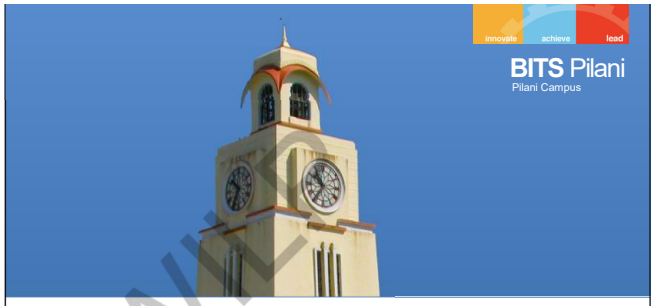
MFML

PPT Booklet

BITS Pilani
Pilani Campus

Mathematical Foundations

MFDS Team



BITS Pilani
Pilani Campus

* ZC416, Lecture 0

Agenda

- Matrices and their types
- REF and RREF
- Rank, its computation and properties
- Determinant, its computation and properties
- Consistency and inconsistency of linear systems
- Nature of solutions of linear systems

Matrices

- A **matrix** is a rectangular array of numbers or functions which we will enclose in brackets. For example,

$$\begin{bmatrix} 0.3 & 1 & -5 \\ 0 & -0.2 & 16 \end{bmatrix}, \quad \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad \begin{bmatrix} e^{-x} & 2x^2 \\ e^{6x} & 4x \end{bmatrix}, \quad \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}, \quad \begin{bmatrix} 4 \\ \frac{1}{2} \end{bmatrix} \qquad (1)$$

- The numbers (or functions) are called **entries** or, less commonly, *elements* of the matrix.
- The first matrix in (1) has two **rows**, which are the horizontal lines of entries.

Matrix – Notations

- We shall denote matrices by capital boldface letters **A**, **B**, **C**, … , or by writing the general entry in brackets; thus $\mathbf{A} = [a_{jk}]$, and so on.
- By an $m \times n$ **matrix** (read *m by n matrix*) we mean a matrix with m rows and n columns—rows always come first! $m \times n$ is called the **size** of the matrix. Thus an $m \times n$ matrix is of the form

$$\mathbf{A} = [a_{jk}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}. \qquad (2)$$

Vectors

- A **vector** is a matrix with only one row or column. Its entries are called the **components** of the vector.
- We shall denote vectors by ***lowercase*** boldface letters **a**, **b**, … or by its general component in brackets, $\mathbf{a} = [a_j]$, and so on. Our special vectors in (1) suggest that a (general) **row vector** is of the form

$$\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}. \quad \text{For instance,} \quad \mathbf{a} = \begin{bmatrix} -2 & 5 & 0.8 & 0 & 1 \end{bmatrix}.$$

A **column vector**

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}. \quad \text{For instance,} \quad \mathbf{b} = \begin{bmatrix} 4 \\ 0 \\ -7 \end{bmatrix}.$$

Equality of Matrices

- Two matrices $\mathbf{A} = [a_{jk}]$ and $\mathbf{B} = [b_{jk}]$ are **equal**, written $\mathbf{A} = \mathbf{B}$, if and only if (1) they have the same size and (2) the corresponding entries are equal, that is, $a_{11} = b_{11}$, $a_{12} = b_{12}$, and so on.
- Matrices that are not equal are called **different**. Thus, matrices of different sizes are always different.

Algebra of Matrices

1. Addition of Matrices

- The **sum** of two matrices $\mathbf{A} = [a_{jk}]$ and $\mathbf{B} = [b_{jk}]$ ***of the same size*** is written $\mathbf{A} + \mathbf{B}$ and has the entries $a_{jk} + b_{jk}$ obtained by adding the corresponding entries of **A** and **B**. Matrices of different sizes cannot be added.

2. Scalar Multiplication (Multiplication by a Number)

- The **product** of any $m \times n$ matrix $\mathbf{A} = [a_{jk}]$ and any **scalar** c (number c) is written $c\mathbf{A}$ and is the $m \times n$ matrix $c\mathbf{A} = [ca_{jk}]$ obtained by multiplying each entry of **A** by c.

(a) $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$
(b) $(\mathbf{A} + \mathbf{B}) + \mathbf{C} = \mathbf{A} + (\mathbf{B} + \mathbf{C})$ (written $\mathbf{A} + \mathbf{B} + \mathbf{C}$)
(c) $\mathbf{A} + \mathbf{0} = \mathbf{A}$
(d) $\mathbf{A} + (-\mathbf{A}) = \mathbf{0}$.

(a) $c(\mathbf{A} + \mathbf{B}) = c\mathbf{A} + c\mathbf{B}$
(b) $(c + k)\mathbf{A} = c\mathbf{A} + k\mathbf{A}$
(c) $c(k\mathbf{A}) = (ck)\mathbf{A}$ (written $ck\mathbf{A}$)
(d) $1\mathbf{A} = \mathbf{A}$.

- Here **0** denotes the **zero matrix** (of size $m \times n$), that is, the $m \times n$ matrix with all entries zero.

Matrix Multiplication

Multiplication of a Matrix by a Matrix

•The **product** $\mathbf{C} = \mathbf{AB}$ (in this order) of an $m \times n$ matrix $\mathbf{A} = [a_{jk}]$ times an $r \times p$ matrix $\mathbf{B} = [b_{jk}]$ is defined if and only if $r = n$ and is then the $m \times p$ matrix $\mathbf{C} = [c_{jk}]$ with entries

$$(3) \qquad c_{jk} = \sum_{l=1}^{n} a_{jl} b_{lk} = a_{j1} b_{1k} + a_{j2} b_{2k} + \cdots + a_{jn} b_{nk} \qquad \begin{matrix} j = 1, \cdots, m \\ k = 1, \cdots, p. \end{matrix}$$

- The condition $r = n$ means that the second factor, **B**, must have as many rows as the first factor has columns, namely n. A diagram of sizes that shows when matrix multiplication is possible is as follows:

$$\begin{matrix} \mathbf{A} & \mathbf{B} & = & \mathbf{C} \\ [m \times n] & [n \times p] & = & [m \times p]. \end{matrix}$$

Matrix Multiplication

EXAMPLE 1

$$\mathbf{AB} = \begin{bmatrix} 3 & 5 & -1 \\ 4 & 0 & 2 \\ -6 & -3 & 2 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 & 1 \\ 5 & 0 & 7 & 8 \\ 9 & -4 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 22 & -2 & 43 & 42 \\ 26 & -16 & 14 & 6 \\ -9 & 4 & -37 & -28 \end{bmatrix}$$

•Here $c_{11} = 3 \cdot 2 + 5 \cdot 5 + (-1) \cdot 9 = 22$, and so on. The entry in the box is $c_{23} = 4 \cdot 3 + 0 \cdot 7 + 2 \cdot 1 = 14$.

•The product **BA** is not defined.

Matrix Multiplication

Matrix Multiplication Is *Not Commutative*, AB ≠ BA in General

- This is illustrated by Example 1, where one of the two products is not even defined. But it also holds for square matrices. For instance,

$$\begin{bmatrix} 1 & 1 \\ 100 & 100 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

but $\begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 100 & 100 \end{bmatrix} = \begin{bmatrix} 99 & 99 \\ -99 & -99 \end{bmatrix}$.

- It is interesting that this also shows that **AB = 0** does ***not*** necessarily imply **BA = 0** or **A = 0** or **B = 0**.

Transposition of Matrices & Vectors

- The transpose of an $m \times n$ matrix $\mathbf{A} = [a_{jk}]$ is the $n \times m$ matrix $\mathbf{A}^T$ (read *A transpose*) that has the first *row* of **A** as its first *column*, the second *row* of **A** as its second *column*, and so on. Thus the transpose of **A** in (2) is $\mathbf{A}^T = [a_{kj}]$, written out

$$\mathbf{A}^T = [a_{kj}] = \begin{bmatrix} a_{11} & a21 & \cdots & a_{m1} \\ a12 & a_{22} & \cdots & a_{m2} \\ \cdot & \cdot & \cdots & \cdot \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{bmatrix}.$$

- As a special case, transposition converts row vectors to column vectors and conversely.

Transposition of Matrices

- Rules for transposition are

$$\begin{aligned} &(a) \quad (\mathbf{A}^{\mathsf{T}})^{\mathsf{T}} = \mathbf{A} \\ &(b) \quad (\mathbf{A}+\mathbf{B})^{\mathsf{T}} = \mathbf{A}^{\mathsf{T}} + \mathbf{B}^{\mathsf{T}} \\ &(c) \quad (c\mathbf{A})^{\mathsf{T}} = c\mathbf{A}^{\mathsf{T}} \\ &(d) \quad (\mathbf{AB})^{\mathsf{T}} = \mathbf{B}^{\mathsf{T}}\mathbf{A}^{\mathsf{T}}. \end{aligned} \qquad (5)$$

CAUTION! Note that in (5d) the transposed matrices are ***in reversed order.***

Special Matrices

- Symmetric: $a_{ij} = a_{ji}$ Eg: $\begin{bmatrix} 1 & 1 & -1 \\ 1 & 2 & 0 \\ -1 & 0 & 5 \end{bmatrix}$
- Skew Symmetric : $a_{ij} = -a_{ji}$ Eg: $\begin{bmatrix} 0 & 1 & -2 \\ -1 & 0 & 3 \\ 2 & -3 & 0 \end{bmatrix}$

Upper triangular matrix: U

$\begin{bmatrix} 1 & 1/2 & 3 & 0 \\ 0 & 5 & 0 & 1 \\ 0 & 0 & 4 & -2 \\ 0 & 0 & 0 & 3 \end{bmatrix}$

- Triangular: Upper Triangular → $a_{ij} = 0$ for all $i > j$
  Lower Triangular → $a_{ij} = 0$ for all $i < j$

Lower triangular matrix: L

$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 3 & 0 \\ 1 & -2 & 0 \end{bmatrix}$

- Diagonal Matrix: $a_{ij} = 0$ for all $i \neq j$ Eg: $\begin{bmatrix} \square & 0 & \cdots & 0 \\ 0 & \square & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \square \end{bmatrix}$
- Sparse Matrix: Many zeroes and few non-zero entities

$\begin{bmatrix} 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 5 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 6 & 0 & 0 \end{bmatrix}$

Positive Definite matrix

- Let **A** be a real symmetric matrix. Then **A** is positive definite if for any x ≠ 0,

$$x^T Ax > 0$$

- Example:
- $A = \begin{bmatrix} 2 & 6 \\ 6 & 20 \end{bmatrix}$ $x^T Ax = [x_1 \quad x_2]\begin{bmatrix} 2 & 6 \\ 6 & 20 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$= 2x_1^2 + 12x_1x_2 + 20x_2^2 = 2(x_1 + 3x_2)^2 + 2x_2^2 > 0$$

- **A** is positive semi-definite **if** $x^T Ax \geq 0$
- $A = \begin{bmatrix} 2 & 6 \\ 6 & 18 \end{bmatrix}$ → $x^T Ax = 2(x_1 + 3x_2)^2 = 0$ when x_1=3 and $x_2 = -1$

Elementary Row Operations

Given a matrix A, the following operations are called Elementary Row Operations

- *Interchange of two rows*
- *Addition of a constant multiple of one row to another row*
- *Multiplication of a row by a* ***non-zero*** *constant c*

CAUTION! These operations are for rows, ***not for columns***!

Row Echelon Form (REF) of a matrix

- Any rows of all zeros are below any other non zero rows.
- Each leading entry of a row is in a column to the right of the leading entry of the row above it.
- All entries in a column below a leading entry are zeros
- Example

$$\begin{bmatrix} 3 & 2 & 0 & 7 & 9 \\ 0 & 4 & 5 & 10 & 0 \\ 0 & 0 & 0 & -4 & 1 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Reduced Rpw Echelon Form (RREF)

- We say that a matrix is in Reduced Row Echelon Form if it is in Echelon form and additionally,
  1. The leading entry in each row is 1.
  2. Each leading 1 is the only non zero entry in its column
- Example

$$\begin{bmatrix} 1 & 0 & 3 & 0 & 9 \\ 0 & 1 & 4 & 0 & -6 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Uniqueness of Row Reduced Echelon Form

- We can transform any matrix into a matrix in reduced row echelon form by using elementary row operations.
- No matter what sequence of row operations we use each matrix is row equivalent to one and only one reducedrow echelon matrix

REF of a Matrix

$\begin{bmatrix} 0 & 1 & 2 & 0 & -1 \\ -2 & 2 & 0 & 3 & 2 \\ 2 & 8 & 20 & 0 & 6 \end{bmatrix}$ Swap rows 1 and 2 → $\begin{bmatrix} -2 & 2 & 0 & 3 & 2 \\ 0 & 1 & 2 & 0 & -1 \\ 2 & 8 & 20 & 0 & 6 \end{bmatrix}$

Replace R3 by R3+1.R1 ↓

$\begin{bmatrix} -2 & 2 & 0 & 3 & 2 \\ 0 & 1 & 2 & 0 & -1 \\ 0 & 10 & 20 & 3 & 8 \end{bmatrix}$

Replace R3 by R3+(-10).R2 ←

$\begin{bmatrix} -2 & 2 & 0 & 3 & 2 \\ 0 & 1 & 2 & 0 & -1 \\ 0 & 0 & 0 & 3 & 18 \end{bmatrix}$

↓ Row Echelon Form

Rank of a matrix

- The number of nonzero rows, *r*, in the reduced row (or row echelon form) coefficient matrix **R** is called the **rank of R** and also the **rank of A**.
- The rank is **invariant** under elementary row operations:

Determination of Rank

$$\mathbf{A} = \begin{bmatrix} 3 & 0 & 2 & 2 \\ -6 & 42 & 24 & 54 \\ 21 & -21 & 0 & -15 \end{bmatrix} \quad \text{(given)}$$

$$= \begin{bmatrix} 3 & 0 & 2 & 2 \\ 0 & 42 & 28 & 58 \\ 0 & -21 & -14 & -29 \end{bmatrix} \quad \begin{matrix} \text{Row 2 + 2 Row 1} \\ \text{Row 3 } - \text{ 7 Row 1} \end{matrix}$$

$$= \begin{bmatrix} 3 & 0 & 2 & 2 \\ 0 & 42 & 28 & 58 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{Row 3 + } \frac{1}{2} \text{ Row 2.}$$

- The last matrix is in row-echelon form and has two nonzero rows. Hence rank **A** = 2.

Minor

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Each element in **A** has a minor.

Delete first row and column from **A**. The determinant of the remaining 2x2 submatrix is the minor of a_{11} which is $a_{22}a_{33} - a_{23}a_{32}$

$$m_{11} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}$$

Minor

$$m_{12} = \begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}$$

And the minor for a_{13} is:

$$m_{13} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

Cofactor

The cofactor C_{ij} of an element a_{ij} is defined as:

$$C_{ij} = (-1)^{i+j} m_{ij}$$

When the sum of a row number *i* and column *j* is even, $c_{ij} = m_{ij}$ and when *i*+*j* is odd, $c_{ij} = -m_{ij}$

$$c_{11}(i=1, j=1) = (-1)^{1+1} m_{11} = +m_{11}$$
$$c_{12}(i=1, j=2) = (-1)^{1+2} m_{12} = -m_{12}$$
$$c_{13}(i=1, j=3) = (-1)^{1+3} m_{13} = +m_{13}$$

Determinant

The determinant of an n x n matrix **A** can now be defined as

$$|A| = \det A = a_{11}c_{11} + a_{12}c_{12} + \ldots + a_{1n}c_{1n}$$

The determinant of **A** is therefore the sum of the products of the elements of the first row of **A** and their corresponding cofactors.

(It is possible to define |**A**| in terms of any other row or column but for simplicity, the first row only is used)

Determinant -Example

Therefore the 2 x 2 matrix :

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Has cofactors :

$$c_{11} = m_{11} = |a_{22}| = a_{22}$$

And:

$$c_{12} = -m_{12} = -|a_{21}| = -a_{21}$$

And the determinant of **A** is:

$$|A| = a_{11}c_{11} + a_{12}c_{12} = a_{11}a_{22} - a_{12}a_{21}$$

Example 1:

$$A = \begin{bmatrix} 3 & 1 \\ 1 & 2 \end{bmatrix}$$

$$|A| = (3)(2) - (1)(1) = 5$$

Determinant -Example

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The cofactors of the first row are:

$$c_{11} = \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} = a_{22}a_{33} - a_{23}a_{32}$$

$$c_{12} = -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} = -(a_{21}a_{33} - a_{23}a_{31})$$

$$c_{13} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} = a_{21}a_{32} - a_{22}a_{31}$$

Determinants – Example

$$|A| = a_{11}c_{11} + a_{12}c_{12} = a_{11}a_{22} - a_{12}a_{21}$$

Which by substituting for the cofactors in this case is:

$$|A| = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

Example 2: $A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 3 \\ -1 & 0 & 1 \end{bmatrix}$

$$|A| = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{23}a_{31}) + a_{13}(a_{21}a_{32} - a_{22}a_{31})$$

$$|A| = (1)(2-0) - (0)(0+3) + (1)(0+2) = 4$$

Adjoint

The adjoint matrix of **A**, denoted by adj **A**, is the transpose of its cofactor matrix

$$adjA = C^T$$

It can be shown that:

$$\mathbf{A}(\text{adj }\mathbf{A}) = (\text{adj}\mathbf{A})\,\mathbf{A} = |\mathbf{A}|\,\mathbf{I}$$

Example: $A = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}$

$$|A| = (1)(4) - (2)(-3) = 10$$

$$adjA = C^T = \begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix}$$

Adjoint

$$A(adjA) = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} = 10I$$

$$(adjA)A = \begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} = 10I$$

Properties of Determinants

1. det(AB) =det(A) * det (B)
2. det(A) nonzero implies there exists a matrix B such that AB=BA=I
3. Two Rows Equal➔ det = 0(Singular)
4. R_i and R_j swapped ➔ det gets a minus sign ($i \neq j$)
5. det(A) = det(A^T)
6. $R_i \leftarrow cR_j$ ➔det A ←c detA

Inverse

- $A^{-1} = \frac{\text{adj}(A)}{\det(A)}$ where $\det(A) \neq 0$

Reiterate $\det(A) \neq 0$ → A is Non singular

$$A = \begin{bmatrix} 3 & 1 \\ 2 & 1 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 1 & -1 \\ -2 & 3 \end{bmatrix}$$

Inverse – 2x2 Example

Example

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}$$

$$A^{-1} = \frac{1}{10}\begin{bmatrix} 4 & -2 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 0.4 & -0.2 \\ 0.3 & 0.1 \end{bmatrix}$$

To check $\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$

$$AA^{-1} = \begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix}\begin{bmatrix} 0.4 & -0.2 \\ 0.3 & 0.1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

$$A^{-1}A = \begin{bmatrix} 0.4 & -0.2 \\ 0.3 & 0.1 \end{bmatrix}\begin{bmatrix} 1 & 2 \\ -3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Inverse – 3x3 Example

Example 2

$$A = \begin{bmatrix} 3 & -1 & 1 \\ 2 & 1 & 0 \\ 1 & 2 & -1 \end{bmatrix}$$

The determinant of **A** is

$$|\mathbf{A}| = (3)(-1-0)-(-1)(-2-0)+(1)(4-1) = -2$$

The elements of the cofactor matrix are

$$c_{11} = +(-1), \quad c_{12} = -(-2), \quad c_{13} = +(3),$$
$$c_{21} = -(-1), \quad c_{22} = +(-4), \quad c_{23} = -(7),$$
$$c_{31} = +(-1), \quad c_{32} = -(-2), \quad c_{33} = +(5),$$

Inverse – 3x3 Example

The cofactor matrix is therefore

$$C = \begin{bmatrix} -1 & 2 & 3 \\ 1 & -4 & -7 \\ -1 & 2 & 5 \end{bmatrix}$$

so

$$adjA = C^T = \begin{bmatrix} -1 & 1 & -1 \\ 2 & -4 & 2 \\ 3 & -7 & 5 \end{bmatrix}$$

and

$$A^{-1} = \frac{adjA}{|A|} = \frac{1}{-2}\begin{bmatrix} -1 & 1 & -1 \\ 2 & -4 & 2 \\ 3 & -7 & 5 \end{bmatrix} = \begin{bmatrix} 0.5 & -0.5 & 0.5 \\ -1.0 & 2.0 & -1.0 \\ -1.5 & 3.5 & -2.5 \end{bmatrix}$$

Inverse

The result can be checked using

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$$

The determinant of a matrix must not be zero for the inverse to exist as there will not be a solution

Nonsingular matrices have non-zero determinants

Singular matrices have zero determinants

Inverse -Simple 2 x 2 case

So that for a 2 x 2 matrix the inverse can be constructed in a simple fashion as

$$A^{-1} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \frac{d}{|A|} & \frac{-b}{|A|} \\ \frac{-c}{|A|} & \frac{a}{|A|} \end{bmatrix} = \frac{1}{|A|}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

- Exchange elements of main diagonal
- Change sign in elements off main diagonal
- Divide resulting matrix by the determinant

Inverse -Simple 2 x 2 case

Example

$$A = \begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix}$$

$$A^{-1} = -\frac{1}{10}\begin{bmatrix} 1 & -3 \\ -4 & 2 \end{bmatrix} = \begin{bmatrix} -0.1 & 0.3 \\ 0.4 & -0.2 \end{bmatrix}$$

Check inverse

$\mathbf{A}^{-1}\mathbf{A}=\mathbf{I}$

$$-\frac{1}{10}\begin{bmatrix} 1 & -3 \\ -4 & 2 \end{bmatrix}\begin{bmatrix} 2 & 3 \\ 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I$$

Linear System

- A **linear system of *m* equations in *n* unknowns** $x_1, \dots, x_n$ is a set of equations of the form

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n &= b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots & \\ a_{m1}x_1 + \cdots + a_{mn}x_n &= b_m. \end{aligned} \qquad (1)$$

- The system is called *linear* because each variable x_j appears in the first power only, just as in the equation of a straight line.

Linear System

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n &= b_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots & \\ a_{m1}x_1 + \cdots + a_{mn}x_n &= b_m. \end{aligned} \qquad (1)$$

- $a_{11}, \dots, a_{mn}$ are given numbers, called the **coefficients** of the system.
- $b_1, \dots, b_m$ on the right are also given numbers.
- If all the bj are zero, then **(1)** is called a **homogeneous system**.
- If at least one bj is not zero, then **(1)** is called a **non-homogeneous system**.

Coefficient Matrix

Matrix Form of the Linear System (1).

- From the definition of matrix multiplication we see that the *m* equations of (1) may be written as a single vector equation

$$\mathbf{Ax} = \mathbf{b} \qquad (2)$$

where the **coefficient matrix A** = $[a_{jk}]$ is the $m \times n$ matrix are column vectors.

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad \text{and} \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

Augmented Matrix

Matrix Form of the Linear System (1). (continued)

- We assume that the coefficients a_{jk} are not all zero, so that **A** is not a zero matrix. Note that **x** has *n* components, whereas **b** has *m* components. The matrix

$$\tilde{\mathbf{A}} = \left[\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \cdot & \cdots & \cdot & \cdot \\ \cdot & \cdots & \cdot & \cdot \\ a_{m1} & \cdots & a_{mn} & b_m \end{array}\right]$$

is called the **augmented matrix** of the system (1).

- The dashed vertical line could be omitted, as we shall do later. It is merely a reminder that the last column of $\tilde{\mathbf{A}}$ did not come from matrix **A** but came from vector **b**. Thus, we *augmented* the matrix **A**.

Solution to System of Linear Equations

- A linear system (1) is called **overdetermined** if it has more equations than unknowns, **determined** if $m = n$, and **underdetermined** if it has fewer equations than unknowns.
- A linear system is **consistent** if rank(**A**) = rank($\tilde{\mathbf{A}}$)
- A **consistent** system has at least one solution (thus, one solution or infinitely many solutions), but **inconsistent** has no solutions at all, as $x_1 + x_2 = 1$, $x_1 + x_2 = 0$.

Solution

The method for determining whether **Ax** = **b** has solutions and what they are:

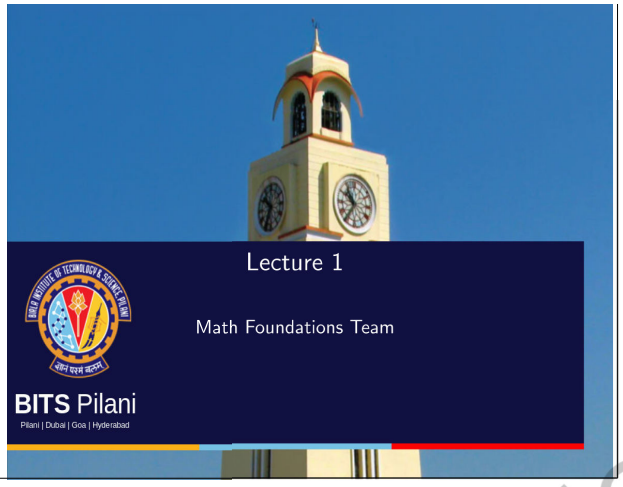
(a) No solution. If r is less than m (meaning that **R** actually has at least one row of all 0s) *and* at least one of the numbers f_{r+1}, f_{r+2}, … , f_m is not zero, then the system **Rx** = **f** is inconsistent: No solution is possible. Therefore the system **Ax** = **b** is inconsistent as well.

Solution

If the system is consistent (either $r = m$, or $r < m$ *and* all the numbers f_{r+1}, f_{r+2}, … , f_m are zero), then there are solutions.

(b) Unique solution. If the system is consistent and $r = n$, there is exactly one solution, which can be found by back substitution.

(c) Infinitely many solutions. To obtain any of these solutions, choose values of x_{r+1}, … , x_n arbitrarily. Then solve the rth equation for x_r (in terms of those arbitrary values), then the $(r-1)$st equation for x_{r-1}, and so on up the line.



What is linear algebra?

- Linear algebra is the study of vectors and rules to manipulate vectors.
- Vectors are not only the familiar geometric vectors from high school (points in 2D/3D space) but any special objects which can be added together and multiplied by scalar values to produce another object of the same kind. For example, polynomials can also be treated as vectors.
- We shall deal with vectors in the space $\mathbb{R}^n$

Systems of linear equations

- Systems of linear equations form a central part of linear algebra.
- Many problems can be formulated as systems of linear equations.
- Tools of linear algebra can be used to solve such problems.
- We would use analytical techniques and numerical methods to solve the systems of linear equations.

Motivating problem

Consider the following problem.
A company produces products $N_1, N_2, \ldots N_n$ for which resources $R_1, R_2, \ldots R_m$ are required. To produce a unit of product N_i , a_{ij} units of resource R_j are needed, where $1 \leq i \leq n, 1 \leq j \leq m$. Find an optimal production plan where x_i units of product N_i are produced if a total b_j units of resource R_j are available, and no resources are left over.

Formulation

If we produce $x_1, x_2, \ldots x_n$ units of the products $N_1, N_2 \ldots N_n$ we need a total of $a_{1i}x_1 + a_{2i}x_2 + \cdots + a_{ni}x_n$ units of resource R_j. Thus we set up the equation:

$$a_{1i}x_1 + a_{2i}x_2 + \ldots + a_{ni}x_n = b_i$$

We can similarly set up the following set of linear equations in n unknowns, $x_1, x_2 \ldots x_n$.

$$\begin{aligned} a_{11}x_1 + a_{21}x_2 + \cdots + a_{n1}x_n &= b_1 \\ &\vdots \\ a_{1m}x_1 + a_{2m}x_2 + \ldots + a_{nm}x_n &= b_m \end{aligned}$$

A linear system has zero, one or infinitely many solutions

Example

Consider the folllowing system of linear equations

$$\begin{aligned} x_1 + x_2 + x_3 &= 3 \\ x_1 - x_2 + 2x_3 &= 2 \\ 2x_1 + 3x_3 &= 1 \end{aligned}$$

Adding the first and second equations gives $2x_1 + 3x_3 = 5$ which contradicts the third equation. Thus there is no set of values for the variables x_1, x_2, x_3 such that the equations above are simultaneously satisfied.

Modified example

Consider a slightly modified example

$$\begin{aligned} x_1 + x_2 + x_3 &= 3 \\ x_1 - x_2 + 2x_3 &= 2 \\ x_2 + x_3 &= 2 \end{aligned}$$

In this case we can see from the first and third equations that $x_1 = 1$. Substituting this value of x_1 into equation (2), we get $-x_2 + 2x_3 = 1$. Adding this equation to equation (3), we get $3x_3 = 3$ which means $x_3 = 1$. Substituting $x_3 = 1$ into equation (3) shows $x_2 = 1$, so the overall solution is $x_1 = x_2 = x_3 = 1$. This is the unique solution to the problem

Infinite solutions

Now consider another modification to the original set of equations

$$\begin{aligned} x_1 + x_2 + x_3 &= 3 \\ x_1 - x_2 + 2x_3 &= 2 \\ 2x_1 + 3x_3 &= 5 \end{aligned}$$

Adding the first and second equations gives $2x_1 + 3x_3 = 5$ which is the same as the third equation. Thus the solution to the three equations is any tuple x_1, x_2, x_3 which satisfies $2x_1 + 3x_3 = 5$, and there are infinite solutions. We now express these solutions in a way whose motivation will become clear later: adding equations (1) and (2) above we get $2x_1 = 5 - 3x_3$.

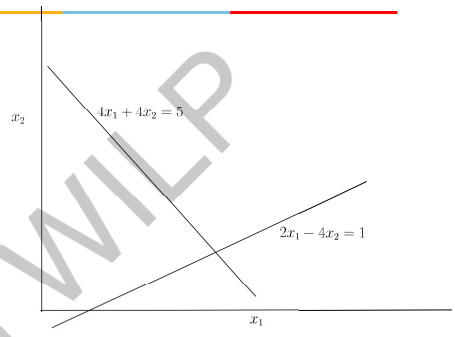
Infinite solutions

- Subtracting equation (2) from (1) we get $2x_2 - x_3 = 1$, so we can write

$$\begin{aligned} x_1 &= \frac{5}{2} - \frac{3}{2}x_3 \\ x_2 &= \frac{1}{2} + \frac{x_3}{2} \end{aligned}$$

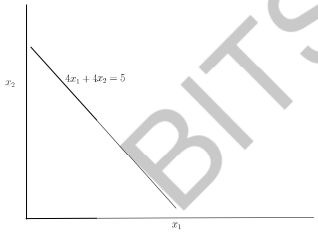
- For the previous problem we can express the set of infinite solutions in terms of the free variable x_3.
- Once x_3 is fixed, the other two variables have to take on specific values - they are known as pivot variables.
- We will show later how to identify pivot and free variables using Gaussian Elimination

Geometrical Interpretation



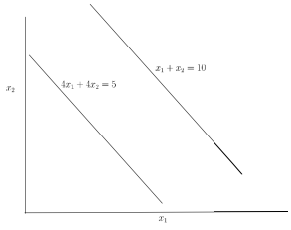
Geometrical interpretation

In the second case both constraints are the same, so there are an infinite number of solutions:



Geometrical interpretation

In the third case the constraints are mutually incompatible, so there is no assignment to x_1, x_2 which satisfies both constraints. The graph of both constraints shows a pair of parallel lines:



Higher dimensions

- In 3D each constraint is a plane.
- The intersection of two planes is a line.
- The intersection of the third plane with the first two planes will be a point on the line in case of a unique solution, or it may lead to pairs of parallel lines (constraint 1 intersection constraint 2 gives one line, constraint 1 intersection constraint 3 gives parallel line, constraint 2 intersection constraint gives parallel line) which means there is no solution.
- All three constraints or planes may intersect in the same line which means infinite solutions.

Example of system of equations

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 42 \\ 8 \end{bmatrix}$$

- This system has two equations and four unknowns, so it is underconstrained. We expect an infinity of solutions.
- Is there a special way in which to express the solutions to this system?
- Let us examine the structure of the given problem matrix.

Structure of the solution

- Looking at the previous slide we can see that a linear combination of columns of the matrix will give the right hand side.
- The ith column vector in the matrix appears in the linear combination, scaled by the corresponding x_i as below.

$$x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + x_3 \begin{bmatrix} 8 \\ 2 \end{bmatrix} + x_4 \begin{bmatrix} -4 \\ 12 \end{bmatrix} = \begin{bmatrix} 42 \\ 8 \end{bmatrix}$$

Particular solution to the example

- A closer look at the linear combination to give the right hand side shows that we can take $x_1 = 42$, $x_2 = 8$, $x_3 = 0$, $x_4 = 0$ since the first two columns are $(1, 0)^T$ and $(0, 1)^T$ respectively.
- Therefore a solution is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 42 \\ 8 \\ 0 \\ 0 \end{bmatrix}$$

- This solution is called the particular solution

Any other solutions possible?

- We can generate other solutions than the particular solution, by adding the vector $\mathbf{0}$ to the particular solution
- But isn't this the same as the particular solution as any vector $+ \mathbf{0}$ is that vector itself?
- The trick is to express $\mathbf{0}$ in terms of the linear combination of some vectors.
- Describing $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}, \mathbf{c_4}$ as the four column vectors associated with the given matrix in the example we can see that $8\mathbf{c_1} + 2\mathbf{c_2}$ - $1\mathbf{c_3} + 0\mathbf{c_4} = \mathbf{0}$.

Any other solutions possible?

- Writing the linear combination in terms of a matrix-vector product we have

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

- Any vector $\lambda(8, 2, -1, 0)^T, \lambda \in R$ will also produce the $\mathbf{0}$ vector

Any other solutions possible?

- We can add the vector $(8, 2, -1, 0)^T$ to the original particular solution $(42, 8, 0, 0)^T$ to get another solution since

$$A\left(\begin{bmatrix} 42 \\ 8 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 42 \\ 8 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 42 \\ 8 \end{bmatrix}$$

Any other solutions possible?

- Following the same line of reasoning as before, we can create the $\mathbf{0}$ vector by expressing the fourth column of the matrix $\boldsymbol{A}$ in terms of the first two columns - note that the first two columns appear capable of generating any two-dimensional vector!
- We can see that $-4\mathbf{c_1} + 12\mathbf{c_2} + 0\mathbf{c_3}$ - $1\mathbf{c_4} = \mathbf{0}$.
- Thus we have

$$\begin{bmatrix} 1 & 0 & 8 & -4 \\ 0 & 1 & 2 & 12 \end{bmatrix} \left(\lambda_2 \begin{bmatrix} -4 \\ 12 \\ 0 \\ -1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Putting things together

- We obtain the following general solution as the sum of the particular solution and a linear combination of solutions to the equation $\mathbf{Ax} = \mathbf{0}$ as follows:

$$\{\mathbf{x} \in \mathbb{R}^4 : \mathbf{x} = \begin{bmatrix} 42 \\ 8 \\ 0 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} 8 \\ 2 \\ -1 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} -4 \\ 12 \\ 0 \\ -1 \end{bmatrix}\}$$

- The general approach consisted of finding a particular solution to $\mathbf{Ax} = \mathbf{b}$, finding all solutions to $\mathbf{Ax} = \mathbf{0}$ and combining the particular and general solutions.
- Neither the particular nor general solutions are unique $\rightarrow$ why?

Algorithmic way of solving equations

- The system of equations in our example was easy to solve because of the special structure of the matrix - we could guess the solution without much difficulty.
- Can we develop an algorithmic way of solving a general system of equations?
- The answer is yes $\rightarrow$ we call the procedure Gaussian elimination

Elementary transformations

- The key idea is to take a complex looking matrix and transform it using elementary row operations to a simple looking matrix like the one we just handled, for which solutions could be obtained essentially by inspection.
- To make this work we need to preserve solutions of the original system of equations, i.e ensure that elementary transformations of the original matrix do not change its solutions.
- Do such elementary transformations exist?

What are the elementary operations?

- Exchange of rows
- Multiplying a row by a constant $\lambda \in R \backslash \{0\}$
- Adding a a row to another row
- Question $\rightarrow$ why must any multiplier to a row be non-zero?

Example to illustrate elementary operations

Consider the folliowing system where we seek all solutions for some $a \in R$.

$$\begin{aligned} -2x_1 + 4x_2 - 2x_3 - x_4 + 4x_5 &= -3 \\ 4x_1 - 8x_2 + 3x_3 - 3x_4 + x_5 &= 2 \\ x_1 - 2x_2 + x_3 - x_4 + x_5 &= 0 \\ x_1 - 2x_2 - 3x_4 + 4x_5 &= a \end{aligned}$$

Compact representation

Let us take the preceding equations and express them compactly in matrix form:

$$\left[\begin{array}{ccccc|c} -2 & 4 & -2 & -1 & 4 & -3 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ 1 & -2 & 1 & -1 & 1 & 0 \\ 1 & -2 & 0 & -3 & 4 & a \end{array}\right]$$

This matrix is called the augmented matrix. It is on this matrix that we will perform the elementary row operations.

Swap rows

Now swap rows 1 and 3 in the augmented matrix to get

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ -2 & 4 & -2 & -1 & 4 & -3 \\ 1 & -2 & 0 & -3 & 4 & a \end{array}\right]$$

Does this change the system of equations? No, because we are swapping **both left and right hand sides of the equality sign, so we are still dealing with the same set of equations.**

Subtract rows

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 4 & -8 & 3 & -3 & 1 & 2 \\ -2 & 4 & -2 & -1 & 4 & -3 \\ 1 & -2 & 0 & -3 & 4 & a \end{array}\right] \begin{array}{c} \\ -4R_1 \\ +2R_1 \\ +-1R_1 \end{array}$$

The notation above is used to convey that we would like to add $-4\times$ first row to the second row, $2\times$ the first row to the third row, and $-1\times$ the first row to the fourth row to get a new augmented matrix.

New Augmented Matrix

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & -1 & -2 & 3 & a \end{array}\right] \quad -R_2 - R_3$$

Note that the augmented matrix shown is obtained by performing the operations shown on the previous slide. To get the next augmented matrix we subtract the second and third rows of this augmented matrix from the last row.

New Augmented Matrix

$$\left[\begin{array}{ccccc|c} 1 & -2 & 1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 & -3 & 2 \\ 0 & 0 & 0 & -3 & 6 & -3 \\ 0 & 0 & 0 & 0 & 0 & a+1 \end{array}\right] \begin{array}{c} \\ \\ -1 \\ 1/3 \end{array}$$

Now multiply the second row by -1 and the third row by $\frac{1}{3}$ to get the augmented matrix in its final form, known as the row-echelon form.

Finding the particular solution

- The row echelon form makes finding a particular solution easy
- Remember that the idea is that a linear combination of the pivot columns must give the right hand side.
- In the example above this means that

$$\lambda_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \lambda_3 \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \\ 1 \\ 0 \end{bmatrix}$$

- This looks like any regular linear combination for which we need to find the coefficients $\lambda_1, \lambda_2, \lambda_3$, so how is this really different from the original problem $\mathbf{Ax} = \mathbf{b}$?

Finding a particular solution

- The linear combination from the previous slide is easily solved.
- Start with finding the value of λ_3. We can see that the third equation establishes $\lambda_3 = 1$.
- The second equation involves only λ_2 and λ_3. Plugging the just discovered value of λ_3 into the second equation, we can find $\lambda_2 = -1$.
- Now we can plug the values of λ_2, λ_3 into the first equation to get $\lambda_1 = 2$

RREF Example

- Consider the following matrix in reduced row-echelon form.

$$A = \begin{bmatrix} 1 & 3 & 0 & 0 & 3 \\ 0 & 0 & 1 & 0 & 9 \\ 0 & 0 & 0 & 1 & -4 \end{bmatrix}$$

- To find solutions for $\mathbf{Ax} = \mathbf{0}$ we need to look at non-pivot columns and note that the pivot columns are "strong enough" to generate the non-pivot columns.
- Our strategy to find solutions to $\mathbf{Ax} = \mathbf{0}$ is to find linear combinations of the pivot columns to the left of a non-pivot column to cancel out the non-pivot column, while setting all other coefficients to zero.

Example continued

- Thus we note that the second column is a non-pivot column which can be expressed as a multiple of the first column such that 3 times the first column + -1 * second column is equal to zero. This gives us our first solution.

$$\begin{bmatrix} 3 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example continued

- Similarly we note that 3 $\times$ the first column + 9 $\times$ the third column + -4 $\times$ the fourth column + $-1\times$ the fifth column is equal to zero. This gives us our second solution:

$$\begin{bmatrix} 3 \\ 0 \\ 9 \\ -4 \\ -1 \end{bmatrix}$$

General solution

- If $\mathbf{x_1}$ and $\mathbf{x_2}$ are solutions to $\mathbf{Ax} = \mathbf{0}$, then any linear combination $\lambda_1\mathbf{x_1} + \lambda_2\mathbf{x_2}$, $\lambda_1, \lambda_2 \in R$ is also a solution
- Thus the general solution to the problem is

$$x \in \mathbb{R}^5 : x = \lambda_1 \begin{bmatrix} 3 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 3 \\ 0 \\ 9 \\ -4 \\ -1 \end{bmatrix}, \lambda_1, \lambda_2 \in R$$

Gaussian elimination and Gauss Jordan

- Repeated use of elementary row operations done to convert a matrix to upper triangular form is called Gaussian elimination.
- Consider the system of equations $\mathbf{Ax} = \mathbf{b}$ where A is a $n \times n$ matrix.
- If A is invertible, it means that A^{-1} exists such that $AA^{-1} = A^{-1}A = I_n$
- In such a case the row-reduced echelon form of A is I_n, i.e every column is a pivot column where the pivot is 1.
- The process of converting A to I_n that we have discussed above is called the Gauss-Jordan method.

Gaussian elimination and Gauss Jordan

- In Gaussian Elimination we use multiples of the first row to eliminate the entries in the first column below the first row.
- Then we use multiples of the second row to eliminate entries in the second column below the second row and so on until we get an upper-triangular matrix.
- This Gauss Jordan process is shown diagramatically in the next slide.
- Then we take multiples of the last row to eliminate non-zero entries in the last column above the last entry, followed by multiples of the last but one row to eliminate non-zero entries in the last but one column and so on. This gives us a diagonal matrix.

Gauss Jordan diagram

$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \rightarrow \begin{bmatrix} a & b & c \\ 0 & e' & f' \\ 0 & h' & i' \end{bmatrix} \rightarrow \begin{bmatrix} a & b & c \\ 0 & e' & f' \\ 0 & 0 & i'' \end{bmatrix} \rightarrow \begin{bmatrix} a & 0 & c' \\ 0 & e' & f' \\ 0 & 0 & i'' \end{bmatrix}$$

$$\downarrow$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \begin{bmatrix} a & 0 & 0 \\ 0 & e' & 0 \\ 0 & 0 & i'' \end{bmatrix}$$

Calculating inverse

- Can the Gauss Jordan procedure calculate the inverse of a matrix?
- For example, let A be a $n \times n$ matrix whose inverse A^{-1} exists. We would like to compute its inverse using Gauss Jordan procedure. Is this possible?
- Yes we can compute the inverse in the following way: we simply set up n linear systems of the form $\mathbf{Ax} = \mathbf{e_i}$, $1 \leq i \leq n$ where $\mathbf{e_i}$ is the ith canonical basis vector and find their solutions $\mathbf{x}$. Each solution vector constitutes a column in A^{-1}. Why is this true?

Calculating Inverse

- Consider the linear system $\mathbf{Ax} = \mathbf{e_i}$.
- Gauss Jordan procedure will convert this sytem to the equivalent system $\mathbf{I_n x} = \mathbf{c_i}$ whose solution is $\mathbf{x} = \mathbf{c_i}$.
- On the other hand, the solution to $\mathbf{Ax} = \mathbf{e_i}$ is $\mathbf{x} = \mathbf{A^{-1}e_i}$.
- Since the two systems are equivalent they have the same solution, so $\mathbf{x} = \mathbf{c_i} = \mathbf{A^{-1}e_i}$ which means $\mathbf{c_i}$ is the ith column of $\mathbf{A^{-1}}$.
- Thus when we create the augmented matrix $[Ae_i]$, Gauss Jordan procedure will convert it into $[I_n c_i]$.
- We can solve n linear systems at once by letting the augmented matrix be $[\ A \ | \ I_n\ ]$ which will become $[I_n A^{-1}]$.

Lecture 2

Math Foundations Team

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Vector Spaces

- We would like to shift focus from systems of linear equations and their solutions to vector spaces.
- Vector spaces are structured spaces in which vectors live.
- What do we mean by "structure" here?

Groups

- We already have some notion about the structure of a vector space, i.e adding two vectors returns a vector, multiplying a vector by a scalar also returns a vector.
- To formalize these notions, we need the concept of a *Group*
- *A Group is a set G and an operation $\otimes : G \times G \rightarrow G$ defined on G. Then $(G, \otimes)$ is called a group if the following properties hold:*
  - *Closure of G under $\otimes$: $\forall x, y \in G, x \otimes y \in G$*
  - *Associativity: $\forall x, y, z \in G, (x \otimes y) \otimes z = x \otimes (y \otimes z)$*
  - *Neutral or Identity element: $\exists\, e \in G, \forall x \in G\ x \otimes e = x$*
  - *Inverse element: $\forall x \in G, \exists\, y \in G, x \otimes y = y \otimes x = e$*
- *If in addition to all the above properties we have $\forall x, y \in G, x \otimes y = y \otimes x$ then $(G, \otimes)$ is an Abelian group*

Some examples

- $(Z, +)$ is an Abelian group. We can see that all the aforementioned properties hold.
- What about $(N_0, +)$? This is not a group since there is no inverse element for an arbitrary element in it.
- $(Z, .)$ where Z is the set of integers, and . is product. It has the identity element but there exist elements in it that don't have an inverse.

Vector Spaces

- We will now consider sets that are just like groups in terms of its properties with respect to an inner operation $+$, and also have an outer operation called . which denotes the multiplication of a vector $x \in G$ by a scalar $\lambda \in R$.
- The inner operation can be viewed as a form of addition while the outer operation can be viewed as a form of scaling.
- A real-valued vector space is a set $V = (\mathcal{V}, +, .)$ with operations $+, .$ such that $+ : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$ and $. : R \times \mathcal{V} \rightarrow \mathcal{V}$ where $(\mathcal{V}, +)$ is an Abelian group, and the following properties hold:
  - Distributivity
  - associativity with respect to the outer operation.
  - there exists a neutral or identity element with respect to the outer operation.

Vector Spaces

Let us look at the properties of a Vector Space more carefully:

- Distributivity: $\forall \lambda \in R, \mathbf{x}, \mathbf{y} \in \mathcal{V}, \lambda.(\mathbf{x} + \mathbf{y}) = \lambda.\mathbf{x} + \lambda.\mathbf{y}$ and $\forall \lambda, \psi \in R, (\lambda + \psi).\mathbf{x} = \lambda.\mathbf{x} + \psi.\mathbf{x}$
- Associativity with respect to the outer operation: $\forall \lambda, \psi \in R, \mathbf{x} \in \mathcal{V}, \lambda.(\psi.\mathbf{x}) = (\lambda\psi).\mathbf{x}$
- Neutral element with respect to the outer operation: $\forall \mathbf{x} \in \mathcal{V}$, $1.\mathbf{x} = \mathbf{x}$
- The elements $\mathbf{x} \in \mathcal{V}$ are called vectors.
- The neutral element with respect to $(\mathcal{V}, +)$ is the zero vector $[0, 0, \ldots 0]^T$ and the inner operation is called vector addition.

Examples of vector spaces

Consider some examples of vector spaces, for example $\mathcal{V} = \mathbb{R}^n$

- We can define addition:
  $\mathbf{x} + \mathbf{y} = (x_1, x_2, \ldots x_n) + (y_1, y_2, \ldots y_n) = (x_1 + y_1, x_2 + y_2, \ldots x_n + y_n)$
- Multiplication by scalars:
  $\lambda\mathbf{x} = \lambda(x_1, x_2, \ldots x_n) = (\lambda x_1, \lambda x_2, \ldots \lambda x_n)$, $\forall \lambda \in R, \mathbf{x} \in \mathbb{R}^{\mathbf{n}}$.

Another example

What we call a vector need not be the standard column vector that we are accustomed to treating as a vector. We can think of $m \times n$ matrices as vectors and create a vector space out of them. Thus $\mathcal{V} = \mathbb{R}^{m \times n}$ with addition and multiplication defined as below:

- Addition $\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} \ldots a_{1n} + b_{1n} \\ \vdots \\ a_{n1} + b_{n1} \ldots a_{nn} + b_{nn} \end{bmatrix}$ is defined element-wise for two matrices $A, B \in \mathcal{V}$.
- Multiplication by scalars: $\lambda\mathbf{A} = \begin{bmatrix} \lambda a_{11} \ldots \lambda a_{1n} \\ \vdots \\ \lambda a_{n1} \ldots \lambda a_{nn} \end{bmatrix}$

Vector subspaces

- Let $V = (\mathcal{V}, +, .)$ be a vector space and let $\mathcal{U} \subseteq \mathcal{V}, \mathcal{U} \neq \Phi$. then $U = (\mathcal{U}, +, .)$ is called a vector subspace of V if U is a vector space with the vector space operations $+$ and $.$ restricted to $\mathcal{U} \times \mathcal{U}$ and $R \times \mathcal{U}$.
- We use the notation $U \subseteq V$ to denote that U is a vector subspace of V.
- If $U \subseteq V$ and V is a vector space, then U naturally inherits many properties directly from V because they hold for all $\mathbf{x} \in V$, and in particular for all $\mathbf{x} \in U \subseteq V$. These properties include the Abelian group property, associativity, distributivity and the neutral element.

Establishing vector subspaces

How do we show if $(\mathcal{U}, +, .)$ is subspace of V?
We need to show that

- $\mathcal{U} \neq \Phi$
- Closure of U with respect to the outer and inner operations, i.e $\forall \lambda \in R, \forall \mathbf{x}, \in \mathcal{U}, \lambda \mathbf{x} \in \mathcal{U}$ and $\forall \mathbf{x}, \mathbf{y} \in \mathcal{U}, \mathbf{x} + \mathbf{y} \in \mathcal{U}$.

Examples

- Let $\mathcal{V} = \mathbb{R}^2$ and $\mathcal{U}$ be the y-axis. . Is $U = (\mathcal{U}, +, .)$ a subspace of $V = (\mathcal{V}, +, .)$? Answer: Yes, because the addition of any two vectors on the y-axis remains on the y-axis so closure with respect to the addition operation is satisfied. Also any vector on the y-axis when scaled by any real number (including 0) will yield a vector on y-axis.
- What about when we shift the y-axis one unit to the right, i.e $\mathcal{U} = \{x = 1\}$? Answer: We no longer have a vector subspace since scaling with respect to the outer operation does not have the closure property.

More examples

- What about the subset of $\mathbb{R}^2$ that represents a square around the origin , i.e $-1 \leq x \leq 1, -1 \leq y \leq 1$? This is again not a subspace.
- This subspace is of particular interest to us called the nullspace of a matrix, i.e the set of solutions to a linear system of equations $A\mathbf{x} = 0$. Why is this a vector subspace?

Linear combination and linear independence

- We know that we can stay in a vector space by adding vectors belonging to the space, and scaling them?
- We are now interested in a different question $\rightarrow$ can we come up with a set of vectors such that every vector in the vector space can be represented as a sum of these vectors with scaling as necessary?
- The answer is yes. A set of vectors capable of representing all vectors in a vector space is called a basis.
- To explore the question of finding a basis, we need to learn the concepts of linear combination and linear independence.

Linear combination

- Consider a vector space V and a set of vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k \in V$. Then every $\mathbf{v} \in V$ that is of the form $v = \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 \ldots + \lambda_k \mathbf{x}_k$ is a linear combination of the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k$.
- Note that the $\mathbf{0}$ can be written trivially as a linear combination of the given k vectors $\mathbf{x}_i$s. We are however interested in non-trivial linear combinations of vectors to get the $\mathbf{0}$-vector.

Linear (in)dependence

- Consider a vector space V and k vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k \in V$. Then if there is a non-trivial linear combination of the given vectors such that $0 = \sum_{i=1}^{i=k} \lambda_i \mathbf{x}_i$ where at least one of the λ_is is non-zero, then the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k$ are linearly dependent.
- Put another way, if the only way we can combine vectors to get the $\mathbf{0}$-vector is by letting all the λ_i be zero, then the vectors are linearly independent.

Linear (in)dependence

- What does the concept of linear independence capture? If the vectors are linearly dependent we can write one of the vectors in terms of the others, so that vector is redundant. On the other hand, when the vectors are linearly independent, each vector brings something to the table which the other vectors collectively cannot replace.

Linear independence

- k vectors in a vector space are either linearly independent or linearly dependent. There is no third option.
- If at least one of the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k$ is the 0-vector, then the vectors are linearly dependent. Why? Apply the definition to see that this is the case. We can choose a non-zero λ for the 0-vector and zero λs for all the other vectors to get the linear combination to be zero.
- If all the vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_k$ are non-zero, then the vectors are linearly dependent if and only if one of them is a linear combination of the others.

Using Gaussian elimination to check for linear independence

- A practical way of checking whether a bunch of vectors are linearly independent is to fill out the columns of a matrix with the given vectors and then perform Gaussian elimination to get the row-echelon form
- The pivot columns indicate all the vectors that are linearly independent, and they are all on the left.
- The non-pivot columns can be expressed as a linear combination of the pivot columns

Example

- Let us start with the matrix

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 4 \end{bmatrix}$$

- Gaussian elimination of this matrix will give rise to the following row-echelon form

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -2 \end{bmatrix}$$

- The pivot columns are the first and the third column and we see that the second column which is a non-pivot column can be expressed as a linear combination of the pivot columns to its left, which is just twice the first column.

Another example

Consider the vectors $\mathbf{x}_1 = \begin{bmatrix} 1 \\ 2 \\ -3 \\ 4 \end{bmatrix}$, $\mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 2 \end{bmatrix}$, $\mathbf{x}_3 = \begin{bmatrix} -1 \\ -2 \\ 1 \\ 1 \end{bmatrix}$

To check for linear independence we set up the equation with λs as follows: $\lambda_1\mathbf{x}_1 + \lambda_2\mathbf{x}_2 + \lambda_3\mathbf{x}_3 = 0$.

As before we put the given vectors into a matrix to get:

$$\begin{bmatrix} 1 & 1 & -1 \\ 2 & 1 & -2 \\ -3 & 0 & 1 \\ 4 & 2 & 1 \end{bmatrix}$$

.

Another example continued

We now perform Gaussian elimination on this matrix to identify pivot and non-pivot columns. If all columns are pivot columns then the given three vectors are linearly independent. After Gaussian elimination we get

$$\begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

.

Note that all columns are pivot columns, so we conclude that the original three vectors are linearly independent. The only way to combine those vectors and get the 0-vector is to take 0s as the λs.

More on linear independence

- Let us say that we have a bunch of k independent vectors $\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_k$
- We now have m vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_m$, each of which is some linear combination of the vectors $\mathbf{b}_i$.
- The goal is to find under what conditions the vectors $\mathbf{x}_i$ are linearly independent.
- Can we express the criterion of linear independence of the $\mathbf{x}_i$s in terms of the way in which we combine the $\mathbf{b}_i$s to get the $\mathbf{x}_i$s?

More on linear independence

We have the following equations:

$$\begin{aligned} \mathbf{x}_1 &= \sum_{i=1}^{i=k} \lambda_{i1}\mathbf{b}_i; \\ \mathbf{x}_2 &= \sum_{i=1}^{i=k} \lambda_{i2}\mathbf{b}_i; \\ &\vdots \\ \mathbf{x}_m &= \sum_{i=1}^{i=k} \lambda_{im}\mathbf{b}_i; \end{aligned}$$

More on linear independence

- We can define a matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_k]$ whose columns are the original linearly independent vectors $\mathbf{b}_1, \mathbf{b}_2 \ldots \mathbf{b}_k$ and then we see that $\mathbf{x}_j = \mathbf{B}\boldsymbol{\lambda}_j$ where $\boldsymbol{\lambda}_j$ is a vector of coefficients.
- Now to test whether $\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots \boldsymbol{x}_m$ are linearly independent, we set up the equation $\sum_{j=0}^{j=m} \psi_j \boldsymbol{x}_j = 0$ and check whether the ψ_js have to be zero. This means

$$\sum_{j=1}^{j=m} \psi_j \boldsymbol{x}_j = \sum_{j=1}^{j=m} \psi_j \boldsymbol{B}\boldsymbol{\lambda}_j = \boldsymbol{B}\sum_{j=1}^{j=m} \psi_j \boldsymbol{\lambda}_j$$

More on linear independence

Now $\sum_{j=1}^{j=m} \psi_j \boldsymbol{\lambda}_j = \boldsymbol{v}$, a vector so $\boldsymbol{Bv}$ is a linear combination of the column vectors of $\boldsymbol{B}$, and the only way a linear combination of the columns of $\boldsymbol{B}$ can be equal to zero if all the combining coefficients are zero. This means that $\boldsymbol{v} = \boldsymbol{0} = \sum_{j=1}^{j=m} \psi_j \boldsymbol{\lambda}_j$.

The fact that $\sum_{j=0}^{j=m} \psi_j \boldsymbol{x}_j = 0$ implies that $\sum_{j=1}^{j=m} \psi_j \boldsymbol{\lambda}_j = 0$ means that $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_m$ are linearly independent if and only if $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots \boldsymbol{\lambda}_m$ are linearly independent.

Example

Consider the following set of equations:

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{b}_1 - 2\boldsymbol{b}_2 + \boldsymbol{b}_3 - \boldsymbol{b}_4 \\ \boldsymbol{x}_2 &= -4\boldsymbol{b}_1 - 2\boldsymbol{b}_2 + 4\boldsymbol{b}_4 \\ \boldsymbol{x}_3 &= 2\boldsymbol{b}_1 + 3\boldsymbol{b}_2 - \boldsymbol{b}_3 - 3\boldsymbol{b}_4 \\ \boldsymbol{x}_4 &= 17\boldsymbol{b}_1 - 10\boldsymbol{b}_2 + 11\boldsymbol{b}_3 + \boldsymbol{b}_4 \end{aligned}$$

Example

We would like to find if the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ are linearly independent. Following the theory established in the last slides we check if the vectors corresponding to the $\boldsymbol{\lambda}_j$s are linearly independent.

We therefore need to check whether the vectors

$$\begin{bmatrix} 1 \\ -2 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -4 \\ -2 \\ 0 \\ 4 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ -1 \\ -3 \end{bmatrix}, \begin{bmatrix} 17 \\ -10 \\ 11 \\ 1 \end{bmatrix} \quad (1)$$

are linearly independent.

Example

We form a matrix consisting of all these column vectors:

$$A = \begin{bmatrix} 1 & -4 & 2 & 17 \\ -2 & -2 & 3 & -10 \\ 1 & 0 & -1 & 11 \\ -1 & 4 & -3 & 1 \end{bmatrix}$$

and perform Gaussian elimination to get the reduced row-echelon form.

Example

We get the following matrix:

$$B = \begin{bmatrix} 1 & 0 & 0 & -7 \\ 0 & 1 & 0 & -15 \\ 0 & 0 & 1 & -18 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Clearly there is a non-trivial way of combining the columns of the above matrix to get the **0**-vector , i.e take 7 $\times$ the first column, add it to 15 $\times$ the second column, add the result to 18 $\times$ the third column and add the result to the fourth column to get **0**. Thus the original set of vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ are not linearly independent.

Insight from the example

- We had a set of k vectors $\boldsymbol{b}_i$ which we combined linearly to get m vectors $\boldsymbol{x}_i$.
- Determining whether the m vectors $\boldsymbol{x}_i$ were linearly independent boiled down to checking if the column vectors of a $k \times m$ matrix were linearly independent.
- We performed Gaussian elimination on this matrix to get the reduced row-echelon form.

Insight from the example

- Here is an interesting insight: if $m > k$, we have more columns than rows and there can only be as many pivots as there are non-zero rows, so the number of pivots and therefore the number of pivot columns is less than k.
- We are therefore **guaranteed** to have non-pivot columns in this case, so that we can express the non-pivot columns in terms of the pivot columns to their left and get a non-trivial linear combination of the columns to get the **0** vector. This would make the m vectors $\boldsymbol{x}_i$ **linearly dependent**.

Basis of vector space

- Do there exist a set of vectors in a vector space which "span" the entire space?
- What we mean here is that any vector $v \in V$ can be generated as a linear combination of the vectors in question.
- Is such a set of vectors unique to the vector space?
- Are all sets capable of generating all vectors in a vector space of the same size?

Generating set and basis

- Consider a vector space $V = (\mathcal{V}, +, .)$ and a set of vectors $\mathcal{A} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_k\} \subseteq V$.
- If every vector $v \in V$ can be expressed as a linear combination of the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_k$, then $\mathcal{A}$ is called a **generating set** of $\mathcal{V}$.
- The set of all linear combinations of the vectors in $\mathcal{A}$ is known as the **span** of $\mathcal{A}$.
- If $\mathcal{A}$ spans the vector space V we write $V = span[\mathcal{A}]$.

Basis as smallest generating set

- Consider a vector space $V = (\mathcal{V}, +, .)$ and $\mathcal{A} \subseteq \mathcal{V}$.
- A generating set $\mathcal{A}$ of V is called minimal if there is no smaller set $\tilde{\mathcal{A}}$ such that $\tilde{\mathcal{A}} \subset \mathcal{A} \subseteq \mathcal{V}$ that spans V.
- Every linearly independent generating set of V is minimal and is called a basis of V.

Characterizations of a basis

The following statements are equivalent:

- $\mathcal{B}$ is a basis of V.
- $\mathcal{B}$ is a minimal generating set.
- $\mathcal{B}$ is a maximal linearly independent set of vectors in V such that adding a vector to it will make it a linearly dependent set.
- Every vector $\boldsymbol{x} \in V$ is a linear combination of vectors from $\mathcal{B}$, and every linear combination is unique. This means that if $\boldsymbol{x} = \sum_{i=1}^{i=k} \lambda_i \boldsymbol{b}_i = \sum_{i=1}^{i=k} \psi_i \boldsymbol{b}_i$, then $\lambda_i = \psi_i, 1 \leq i \leq k$.

Basis is not unique

- Consider the space $\mathbb{R}^3$. The canonical basis is $\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}$
- Another basis for $\mathbb{R}^3$ is $\begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}$
- Yet another basis for $\mathbb{R}^3$ is $\begin{bmatrix}0.5\\0.8\\0.4\end{bmatrix}, \begin{bmatrix}1.8\\0.3\\0.3\end{bmatrix}, \begin{bmatrix}-2.2\\-3.3\\1.5\end{bmatrix}$

Not a basis

- Is any linear independent set a basis?
- No, we can have a linearly independent set that has too few vectors to become a basis
- As an example, consider the following set of vectors in $\mathbb{R}^4$
  $\begin{bmatrix}1\\2\\3\\4\end{bmatrix}, \begin{bmatrix}2\\-1\\0\\2\end{bmatrix}, \begin{bmatrix}1\\1\\0\\4\end{bmatrix}$
- Why is the above not a basis?

Some properties

- Vector spaces can be finite or infinite-dimensional.
- We consider only finite-dimensional vector spaces.
- In a finite-deimensional vector space V, the number of vectors in the basis is known as $dim(V)$.
- If $U \subseteq V$, then $dim(U) \leq dim(V)$.
- $dim(U) = dim(V)$ if and only if $U = V$.

Dimension of a vector space

- Is the size of the basis merely the number of components in the vector?
- So far, that has been the case.
- But consider the vector space $span(\begin{bmatrix}0\\1\end{bmatrix})$ - the vector has two components, but the basis is just one vector. All vectors in this vector space are multiples of the vector $\begin{bmatrix}0\\1\end{bmatrix}$.

Finding a basis

There are three steps to finding a basis for a vector space. A basis of a subspace $U = span(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m) \subseteq \mathbb{R}^n$ can be found by executing the following steps:

- Write the spanning vectors as columns of a matrix $\boldsymbol{A}$.
- Determine the row-echelon form of $\boldsymbol{A}$.
- The spanning vectors associated with the pivot columns are a basis of U.

Example

- Consider the vector subspace $U \subseteq \mathbb{R}^5$ which is spanned by the following vectors:

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \\ -1 \\ -1 \end{bmatrix}, \boldsymbol{x}_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \\ 2 \\ -2 \end{bmatrix}, \boldsymbol{x}_3 = \begin{bmatrix} 3 \\ -4 \\ 3 \\ 5 \\ -3 \end{bmatrix}, \boldsymbol{x}_4 \begin{bmatrix} -1 \\ 8 \\ -5 \\ -6 \\ 1 \end{bmatrix}$$

- We would like to check if the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ constitute a basis for the subspace U.
- We therefore need to check whether the given vectors are linearly independent or not.

Analytic geometry

- We have studied vector spaces in the previous lecture.
- Now we would like to provide some geometric interpretation to these concepts.
- We shall take a close look at geometric vectors and the concepts of lengths of vectors and angles between vectors.
- But first we need to add the concept of an inner product to our vector space.

Norms

- A norm on a vector space is a function $\|.\| : V \to \mathbb{R}$, $\boldsymbol{x} \to \|\boldsymbol{x}\|$ which assigns to each vector $\boldsymbol{x}$ a length $\|\boldsymbol{x}\|$ such that for all $\lambda \in \mathbb{R}$ and $\boldsymbol{x}, \boldsymbol{y} \in V$ the following properties hold:
  - Absolutely homogeneous: $\|\lambda \boldsymbol{x}\| = |\lambda| \|\boldsymbol{x}\|$
  - Triangle inequality: $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$
  - Positive definite: $\|\boldsymbol{x}\| \geq 0$ and $\|\boldsymbol{x}\| = 0 \implies \boldsymbol{x} = 0$
- Manhattan norm : $\|\boldsymbol{x}\| = \sum_{i=1}^{i=n} |x_i|$
- Euclidean norm : $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{i=1}^{i=n} x_i^2}$.

Inner products

- Dot product in $\mathbb{R}^n$ is given by $\boldsymbol{x}^T \boldsymbol{y} = \sum_{i=1}^{i=n} x_i y_i$
- A bilinear mapping Ω is a mapping with two arguments and is linear in both arguments: Let V be a vector space such that $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$, and let $\lambda, \psi \in \mathbb{R}$. Then we have $\Omega(\lambda \boldsymbol{x} + \psi \boldsymbol{y}, \boldsymbol{z}) = \lambda \Omega(\boldsymbol{x}, \boldsymbol{z}) + \psi \Omega(\boldsymbol{y}, \boldsymbol{z})$, and $\Omega(\boldsymbol{x}, \lambda \boldsymbol{y} + \psi \boldsymbol{z}) = \lambda \Omega(\boldsymbol{x}, \boldsymbol{y}) + \psi \Omega(\boldsymbol{x}, \boldsymbol{z})$.
- Let V be a vector space and $\Omega : V \times V \to \mathbb{R}$ be a bilinear mapping that takes two vectors as arguments and returns a real number. Then Ω is called symmetric if $\Omega(\boldsymbol{x}, \boldsymbol{y}) = \Omega(\boldsymbol{y}, \boldsymbol{x})$. Also Ω is called positive-definite if $\forall x \in V \setminus \{0\}$, $\Omega(\boldsymbol{x}, \boldsymbol{x}) > 0$ and $\Omega(\boldsymbol{0}, \boldsymbol{0}) = 0$.

Inner products

- A positive-definite, symmetric bilinear mapping $\Omega : V \times V \to \mathbb{R}$ is called an inner product. To denote an inner product on V we generally write $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$.
- The pair $(V, \langle ., . \rangle)$ is called an inner product space.
- Next we introduce the concept of symmetric, positive-definite matrices and show we can express an inner product using such matrices.
- We recall that in a vector space V any vector $\boldsymbol{x}$ can be written as linear combination of the basis vectors. We use this to express an inner product in terms of a matrix.

Symmetric, positive-definite matrices

Theorem: For a real-valued, finite-dimensional vector space V and an ordered basis B of V, it holds that $\langle ., . \rangle : V \times V \to \mathbb{R}$ is an inner product if and only if there exists a symmetric, positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ with $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{y}}$.
Proof: One direction $\to$: $\langle ., . \rangle$ is an inner product $\implies$ $\boldsymbol{A}$ is symmetric, positive-definite such that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{y}}$.
Other direction $\leftarrow$: $\boldsymbol{A}$ is symmetric, positive definite such that the operation $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ is defined as $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{y}}$ $\implies$ the operation defined is an inner product.

Symmetric, positive-definite matrices

- We prove the $\to$ direction.
- Let $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ be the inner product between the vectors $\boldsymbol{x}, \boldsymbol{y}$ in V. We can write $\boldsymbol{x}$ in terms of say n basis vectors as $\boldsymbol{x} = \sum_{i=1}^{i=n} \psi_i \boldsymbol{b}_i$. Similarly $\boldsymbol{y} = \sum_{i=1}^{i=n} \lambda_i \boldsymbol{b}_i$.
- Since the inner product is bilinear we can write $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \sum_{i=1}^{i=n} \psi_i \boldsymbol{b}_i, \sum_{i=1}^{i=n} \lambda_i \boldsymbol{b}_i \rangle = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} \psi_i \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle \lambda_j = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{y}}$ where $A_{ij} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle$.
- Here $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$ are vectors which represent the coordinates of the original vectors $\boldsymbol{x}, \boldsymbol{y}$ with respect to the basis vectors.

Symmetric, positive-definite matrices

- This means that the inner product is entirely determined through the matrix $\boldsymbol{A}$. The symmetry of the inner product means that $A_{ij} = \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = A_{ji} = \langle \boldsymbol{b}_j, \boldsymbol{b}_i \rangle$. Thus $\boldsymbol{A}$ is symmetric.
- The positive-definiteness of the inner product means that $\forall \boldsymbol{x} \in V \backslash \{0\}, \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0$.

Symmetric, positive-definite matrices

- Now let us consider an operation op such that $\boldsymbol{x} op \boldsymbol{y} = \hat{\boldsymbol{x}}^T A \hat{\boldsymbol{y}}$ where $\boldsymbol{A}$ is a symmetric, positive definite matrix.
- We shall show that "op" is an inner product by showing that it has all the properties of an inner product:
  - "op" has symmetry because $\boldsymbol{x} op \boldsymbol{y} = \hat{\boldsymbol{x}}^T A \hat{\boldsymbol{y}}$ and $\boldsymbol{y} op \boldsymbol{x} = \hat{\boldsymbol{y}}^T A \hat{\boldsymbol{x}} = \hat{\boldsymbol{y}}^T (A \hat{\boldsymbol{x}})$. By a property of the dot product we can write $\hat{\boldsymbol{y}}^T (A \hat{\boldsymbol{x}}) = (A \hat{\boldsymbol{x}})^T \hat{\boldsymbol{y}} = \hat{\boldsymbol{x}}^T \boldsymbol{A}^T \hat{\boldsymbol{y}} = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{y}}$ where the last equality in the chain is possible since $\boldsymbol{A}$ is symmetric.
  - "op" also has bilinearity since we see that for $r \in R, (r\boldsymbol{x}) op \boldsymbol{y} = (r \hat{\boldsymbol{x}})^T \boldsymbol{A} \hat{\boldsymbol{y}} = r \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{y}} = r \boldsymbol{x} op \boldsymbol{y}$.
  - $(\boldsymbol{x} + \boldsymbol{y}) op \boldsymbol{z} = (\hat{\boldsymbol{x}} + \hat{\boldsymbol{y}})^T \boldsymbol{A} \hat{\boldsymbol{z}} = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{z}} + \hat{\boldsymbol{y}}^T \boldsymbol{A} \hat{\boldsymbol{z}} = \boldsymbol{x} op \boldsymbol{z} + \boldsymbol{y} op \boldsymbol{z}$.
  - Finally if $\boldsymbol{x}$ is a non-zero vector then $\hat{\boldsymbol{x}}$ is also a non-zero vector, $\boldsymbol{x} op \boldsymbol{x} = \hat{\boldsymbol{x}}^T \boldsymbol{A} \hat{\boldsymbol{x}} > 0$ since we are given that $\boldsymbol{A}$ is positive-definite.

Symmetric, positive definite matrices

- Can a symmetric, positive-definite matrix have less than full rank? We have $\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0$ for all non-zero $\boldsymbol{x}$. Thus $\boldsymbol{x} = \boldsymbol{0}$ is the only vector allowed in the nullspace. The nullspace is 0-dimensional so $\boldsymbol{A}$ has full rank.
- What can be said about the diagonal elements of a positive-definite matrix? From $(\boldsymbol{e}_i)^T \boldsymbol{A} \boldsymbol{e}_i > 0$ where $\boldsymbol{e}_i$ is the ith canonical basis vector, we see that $A_{ii} > 0$. Thus the diagonal entries are all strictly positive.

Lengths and distances

- Inner products and norms are closely related in the sense that any inner product induces a norm: $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$
- Not every norm is induced by an inner product, for example the Manhattan norm.
- For an inner product vector space $(V, \langle ., . \rangle)$, the induced norm $\|.\|$ satisfies the Cauchy-Schwarz inequality: $\langle \boldsymbol{x}, \boldsymbol{y} \rangle \leq \|\boldsymbol{x}\| \|\boldsymbol{y}\|$. Why is this true?

Cauchy-Schwarz inequality

- Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two vectors and let us consider the length of the vector $\boldsymbol{u} - \alpha \boldsymbol{v}$ where α is a constant.
- The length of the vector $\boldsymbol{u} - \alpha \boldsymbol{v}$ is greater than or equal to zero. The length of the vector $\boldsymbol{u} - \alpha \boldsymbol{v}$ is $\|\boldsymbol{u} - \alpha \boldsymbol{v}\| = \langle \boldsymbol{u} - \alpha \boldsymbol{v}, \boldsymbol{u} - \alpha \boldsymbol{v} \rangle = (\boldsymbol{u} - \alpha \boldsymbol{v})^T (\boldsymbol{u} - \alpha \boldsymbol{v})$.
- We can expand the dot product $(\boldsymbol{u} - \alpha \boldsymbol{v})^T (\boldsymbol{u} - \alpha \boldsymbol{v}) = \boldsymbol{u}^T \boldsymbol{u} - \alpha \boldsymbol{u}^T \boldsymbol{v} - \alpha \boldsymbol{v}^T \boldsymbol{u} + \alpha^2 \boldsymbol{v}^T \boldsymbol{v} \geq 0$
- Now set $\alpha = \frac{\boldsymbol{u}^T \boldsymbol{v}}{\boldsymbol{v}^T \boldsymbol{v}}$ to get $\boldsymbol{u}^T \boldsymbol{u} - \frac{(\boldsymbol{u}^T \boldsymbol{v})^2}{\boldsymbol{v}^T \boldsymbol{v}} \geq 0$ which leads us to $(\boldsymbol{u}^T \boldsymbol{u})(\boldsymbol{v}^T \boldsymbol{v}) \geq (\boldsymbol{u}^T \boldsymbol{v})^2$ which is Cauchy-Schwarz inequality.
- Note that although this proof was developed using $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u}^T \boldsymbol{v}$, it works for any definition of the inner product.

Metric space

- Consider an inner product space $(V, \langle ., . \rangle)$. Define $d(\boldsymbol{x}, \boldsymbol{y})$ the distance between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ to be $d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y} \rangle}$.
- If we use the dot product as the inner product, then the distance is called the Euclidean distance.
- The mapping $d : V \times V \to \mathbb{R}$ is called a metric.

Properties of a metric space

A metric d has the following properties:

- d is positive-definite which means $d(\boldsymbol{x}, \boldsymbol{y}) \geq 0 \ \ \forall \boldsymbol{x}, \boldsymbol{y} \in V$. $d(\boldsymbol{x}, \boldsymbol{y}) = 0 \implies \boldsymbol{x} = \boldsymbol{y}$.
- d is symmetric which means $d(\boldsymbol{x}, \boldsymbol{y}) = d(\boldsymbol{y}, \boldsymbol{x}) \ \ \forall \boldsymbol{x}, \boldsymbol{y} \in V$.
- d obeys the triangle inequality as follows: $d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}) \ \ \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in V$

Inner products and metrics seem to be very similar in terms of their properties - however there is one important difference. When $\boldsymbol{x}$ and $\boldsymbol{y}$ are close to each other the inner product is large but the distance metric is small. On the other hand when $\boldsymbol{x}$ and $\boldsymbol{y}$ are far apart, then the inner product is small but the distance metric is large.

Angles and orthogonality

- In addition to being able to capture the lengths of vectors and the distance between vectors, inner products can also capture the **angle** ω between two vectors and can thus capture the geometry of a vector space.
- The key to using the inner product to characterize the angle between two vectors is the Cauchy-Schwarz inequality.
- Assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are not the $\boldsymbol{0}$ vector. Then the Cauchy-Schwarz inequality tells us that

$$-1 \leq \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} \leq 1 \qquad (1)$$

Angles and orthogonality

- Since the Cauchy-Schwarz ratio lies between -1 and 1 we can set it equal to the cosine of a unique angle $\omega \in [0, \pi]$ such that

$$\cos(\omega) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|} \qquad (2)$$

- The angle ω is the angle between two vectors. What does it capture?
- The notion of angle captures similarity of orientation between two vectors. When the dot product is close to zero, the vectors are more or less pointing in orthogonal directions and $\omega \approx \pi/2$.

Angles and orthogonality

- Food for thought: Suppose we choose vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ uniformly at random in high dimensions. What happens to the dot product between the vectors and hence the angle between them?
- To choose a vector uniformly at random over a sphere let every component in the vector be an independent Gaussian random variable of mean 0 and unit variance.
- Write a small program to see what happens ...

Angles and orthogonality

- A key feature of the inner product is that we can use it to characterize vectors that are orthogonal.
- Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal if and only if the inner product between them is 0. For an orthogonal pair of vectors $\boldsymbol{x}$, $\boldsymbol{y}$ we can write $\boldsymbol{x} \perp \boldsymbol{y}$.
- By the above definition the $\boldsymbol{0}$-vector is orthogonal to all vectors.
- Vectors which are orthogonal with respect to one inner product need not be orthogonal with respect to another inner product.

Example - angles and orthogonality

- Consider the vectors $\boldsymbol{x} = [1, 1]^T$ and $\boldsymbol{y} = [-1, 1]^T$
- With respect to the inner product defined as a dot product we see that $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^T \boldsymbol{y} = 1 * -1 + 1 * 1 = 0$.
- With respect to the inner product $\boldsymbol{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{y}$, the angle between the two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ becomes

$$\cos(\omega) = \frac{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}{\|\boldsymbol{x}\| \|\boldsymbol{y}\|}$$

Example - angles and orthogonality

- Continuing with our example we have

$$\begin{aligned} \cos(\omega) &= \frac{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{y}}{\sqrt{\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}} \sqrt{\boldsymbol{y}^T \boldsymbol{A} \boldsymbol{y}}} \\ &= \frac{2x_1 y_1 + x_2 y_2}{\sqrt{(2x_1^2 + x_2^2)(2y_1^2 + y_2^2)}} \\ &= \frac{-1}{3} \end{aligned}$$

where $\boldsymbol{A} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$.

- Thus, with respect to the new definition of inner product the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are no longer orthogonal.

Orthonormal matrix

- A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix if and only if its columns are orthonormal:

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I} = \boldsymbol{A}\boldsymbol{A}^T$$
$$\boldsymbol{A}^T = \boldsymbol{A}^{-1}$$

- If the columns of a matrix are orthonormal, why are its rows orthonormal too? This follows from the fact that the left-inverse of a square matrix is the same as the right-inverse. Let $\boldsymbol{A}$ be a square matrix with $\boldsymbol{B}$ and $\boldsymbol{C}$ the left and right inverses of $\boldsymbol{A}$: $\boldsymbol{B}\boldsymbol{A} = I = \boldsymbol{A}\boldsymbol{C} \implies \boldsymbol{B} = \boldsymbol{C}$. Why is this true?

Orthonormal matrix

- Transformations by an orthonormal matrix preserve lengths. This can be seen as follows, using the dot product as the definition of the inner product:
$\|\boldsymbol{A}\boldsymbol{x}\|^2 = (\boldsymbol{A}\boldsymbol{x})^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}^T\boldsymbol{x}$.
- An example of an orthonormal matrix is the 2D-rotation matrix which can be expressed as $\begin{bmatrix} \cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}$ where θ is the angle of rotation.

Orthonormal matrix

- Also the angle between two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ does not change after transformation by an orthonormal matrix. This can be seen as follows:

$$\begin{aligned} \cos(\omega) &= \frac{(\boldsymbol{A}\boldsymbol{x})^T\boldsymbol{A}\boldsymbol{y}}{\|\boldsymbol{A}\boldsymbol{x}\|\,\|\boldsymbol{A}\boldsymbol{y}\|} \\ &= \frac{\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{y}}{\|\boldsymbol{x}\|\,\|\boldsymbol{y}\|} \\ &= \frac{\boldsymbol{x}^T\boldsymbol{y}}{\|\boldsymbol{x}\|\,\|\boldsymbol{y}\|} \end{aligned}$$

Orthonormal basis

- We already looked at the concept of a basis of a vector space, and found that for the vector space $\mathbb{R}^n$ we need n basis vectors.
- Our basis vectors needed only to be linearly independent - we can ensure linear independence by ensuring that our basis vectors point in different directions, so that a linear combination of $n-1$ basis vectors cannot cancel out the nth basis vector.
- Now we will look at a special case of a basis where the vectors are all mutually orthogonal in the sense of the inner product, and each vector is of unit length. We call such a basis an orthonormal basis.

Orthonormal basis

- Question: Can you immediately think of an orthonormal basis for $\mathbb{R}^n$? Is an orthonormal basis for a vector space unique?
- Formal definition of an orthonormal basis: Consider an n-dimensional vector space V and n basis vectors $\{\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_n\}$. If it is true that $\forall i, j = 1, \ldots n, i \neq j \ \langle \boldsymbol{b}_i, \boldsymbol{b}_j \rangle = 0$ and $\langle \boldsymbol{b}_i, \boldsymbol{b}_i \rangle = 1$, then the basis is called an orthonormal basis.
- If the basis vectors are only mutually orthogonal but not of length unity, then we have an orthogonal basis.

Gram-Schmidt process

- Given a set of basis vectors for a vector space, can we convert the given basis into an orthogonal basis? Yes, we shall use Gaussian elimination to construct such a basis.
- Let us start with an example: Consider $\mathbb{R}^2$ and two basis vectors $\boldsymbol{v}_1 = (3,1)^T$ and $\boldsymbol{v}_2 = (2,2)^T$. Put these vectors into columns of a matrix $\boldsymbol{A}$ such that $\boldsymbol{A} = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$.
- The next step is to perform Gaussian elimination on the following augmented matrix: $[\boldsymbol{A}^T\boldsymbol{A}|\boldsymbol{A}^T] = \left[\begin{array}{cc|cc} 10 & 8 & 3 & 1 \\ 8 & 8 & 2 & 2 \end{array}\right]$
- On performing Gaussian elimination of this augmented matrix we end up with $\left[\begin{array}{cc|cc} 1 & 0.8 & 0.3 & 0.1 \\ 0 & 1 & -0.25 & 0.75 \end{array}\right]$

Gram-Schmidt process

- Note that after the completion of Gaussian elimination the two rows on the right hand side are orthogonal. They form a basis for $\mathbb{R}^2$. We can normalize the vectors to get an orthonormal basis. Let us justify this technique.
- First we see that when the $m \times n$ matrix $\boldsymbol{A}$ has full column rank, then the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is positive definite. To see this note that any solution $\boldsymbol{x}$ to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ is also a solution to $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ and vice-versa. Why is this the case?
- When $\boldsymbol{A}$ has linearly independent columns, there are no non-trivial solutions to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. Thus the fact that there are no non-trivial solutions to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ means that $\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x} \neq \boldsymbol{0}, \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} > \boldsymbol{0}$. Note that since $\boldsymbol{A}^T\boldsymbol{A}$ is positive-definite, Gaussian elimination can be carried out on $\boldsymbol{A}^T\boldsymbol{A}$ without row exchanges.

Elementary transformations

- One of the steps of Gaussian elimination is the the subtraction of a multiple of a given row from a row below it. This step can be achieved by pre-multiplication of the given matrix by an elementary matrix. An elementary matrix is like an identity matrix except that one of the entries below the diagonal is allowed to be non-zero.
- To show how the process of elimination works using an elementary matrix consider the matrix $A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$ and assume that we want to subtract two times the first row from the second row.

Elementary transformations

This can be accomplished by the following elementary matrix $\boldsymbol{E} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ so that the product

$$\begin{aligned} EA &= \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} - 2a_{11} & a_{22} - 2a_{12} & a_{23} - 2a_{13} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \end{aligned}$$

Product of elementary transformations

- A series of Gaussian elimination steps can be represented as a product of elementary transformations acting on $\boldsymbol{A}$: $\boldsymbol{E}_m\boldsymbol{E}_{m-1}\ldots\boldsymbol{E}_1\boldsymbol{A}$.
- The product of lower triangular matrices can be seen to be lower triangular, and the inverse of a lower triangular matrix can also be seen as a lower triangular matrix.
- Thus the action of Gaussian elimination operations can be seen in the following terms $\boldsymbol{L}^{-1}\boldsymbol{A} = \boldsymbol{U}$ where the product of the elementary transformations is represented as the inverse of a lower triangular matrix for notational convenience, and the right hand side U is an upper triangular matrix. Thus we have $\boldsymbol{A} = \boldsymbol{LU}$.

Final argument

- Returning to our problem we are performing Gaussian elimination on the matrix $\boldsymbol{A}^T\boldsymbol{A}$ where $\boldsymbol{A}$ contains the basis vectors as its columns. Upon Gaussian elimination on the augmented matrix we reduce $[\boldsymbol{A}^T\boldsymbol{A}|\boldsymbol{A}^T]$ to get $[\boldsymbol{U}|\boldsymbol{L}^{-1}\boldsymbol{A}^T$ where $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{LU}$.
- Now we shall show that $\boldsymbol{Q}^T = \boldsymbol{L}^{-1}\boldsymbol{A}^T$ is an orthogonal matrix whose rows are orthogonal.
- Consider $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{L}^{-1}\boldsymbol{A}^T\boldsymbol{A}(\boldsymbol{L}^{-1})^T = \boldsymbol{U}(\boldsymbol{L}^{-1})^T =$ some upper triangular matrix
- But $\boldsymbol{Q}^T\boldsymbol{Q}$ is a symmetric matrix and can only be upper triangular if it is diagonal. Therefore $\boldsymbol{Q}$ is an orthogonal matrix whose columns are orthogonal. They can be normalized to obtain an orthonormal basis.

Matrix decompositions

- We studied vectors and how to manipulate them in preceding lectures.
- Mappings and transformations of vectors can be conveniently described in terms of operations performed by matrices.
- In this lecture we shall study three aspects of matrices: how to summarize matrices, how matrices can be decomposed, and how the decompositions can be used for matrix approximations.

Characteristic polynomial

- For $\lambda \in \mathbb{R}$ and $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ we can define $p_{\boldsymbol{A}}(\lambda) = \det(A - \lambda\boldsymbol{I})$ and show that it can be written as $c_0 + c_1\lambda + \ldots c_{n-1}\lambda^n + (-1)^n\lambda^n$ where $c_0, c_1 \ldots c_{n-1} \in R$.
- We can show that $c_0 = \det(\boldsymbol{A})$ and $c_{n-1} = (-1)^{n-1}tr(\boldsymbol{A})$
- To see that $c_0 = \det(\boldsymbol{A})$, set $\lambda = 0$ in $\det(\boldsymbol{A} - \lambda\boldsymbol{I})$ to get $p_{\boldsymbol{A}}(0) = \det(\boldsymbol{A}) = c_0$
- The formula for c_{n-1} takes a little bit of work - let us expand a 3×3 determinant $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = \begin{vmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{vmatrix}$

Characteristic polynomial

- ~~Expanding the determinant along the first row we~~ see that the $(a_{11} - \lambda)C_{11}$ term contains the product $\prod_{i=1}^{i=3}(a_{ii} - \lambda)$ which contains the powers λ^3 and λ^2. The other contributors to the determinant i.e $a_{12}C_{12}$ and $a_{12}C_{13}$ expand into terms where the maximum power of $\lambda = 1$.
- Carrying this analogy over to the general case of $n > 3$ we see that expanding along the first row the first contributor to the determinant will have the term $\prod_{i=1}^{i=n}(a_{ii} - \lambda)$ and subsequent contributors will have a maximum power of λ^{n-2} as the minors for each such contributor will kill off a term containing λ in a given row and column.

Characteristic polynomial

- Thus in the determinant expansion to obtain the characteristic polynomial we see that coefficient to λ^{n-1} can only come from the expansion of $\prod_{i=1}^{i=n}(a_{ii}-\lambda)$ and can be seen to be seen to be $(-1)^{n-1}\sum_{i=1}^{i=n} a_{ii} = (-1)^{n-1}tr(\boldsymbol{A})$.
- As a corollary to this argument we can see that the coefficient to λ^n in the characteristic polynomial is $(-1)^n$
- We will use the characteristic polynomial to compute eigenvalues and eigenvectors.

Eigenvalues and eigenvectors

- Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ be a square matrix. 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 λ if $\boldsymbol{Ax} = \lambda\boldsymbol{x}$. This equation is called the eigenvalue equation.
- The following statements are equivalent:
  - λ is an eigenvalue of $\boldsymbol{A} \in \mathbb{R}^{n\times n}$.
  - There exists an $\boldsymbol{x} \in \mathbb{R}^n \setminus 0$ with $\boldsymbol{Ax} = \lambda\boldsymbol{x}$, or equivalently, $(\boldsymbol{A} - \lambda\boldsymbol{I}_n)\boldsymbol{x} = \boldsymbol{0}$ can be solved non-trivially, i.e., $\boldsymbol{x} \neq \boldsymbol{0}$.
  - $\text{rank}(\boldsymbol{A} - \lambda\boldsymbol{I}_n) < n$.
  - $\det(\boldsymbol{A} - \lambda\boldsymbol{I}_n) = 0$.
- If $\boldsymbol{x}$ is an eigenvector corresponding to a particular eigenvalue λ, $c\boldsymbol{x}, c \in \mathbb{R} \setminus 0$ is also an eigenvector.

Eigenvalues and eigenvectors - example

- Consider the matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. The characteristic polynomial $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = (1-\lambda)^2 - 1$ and setting it to zero gives us the roots of the charactristic polynomial: $(1-\lambda)^2 - 1 = 0$ has roots $\lambda = 2, 0$.
- What are the eigenvectors? For $\lambda = 0$ we solve for $\boldsymbol{Ax} = 0\boldsymbol{x}$, so we find the nullspace of the matrix $\boldsymbol{A}$. Using Gaussian elimination we convert $\boldsymbol{Ax} = \boldsymbol{0}$ to $\boldsymbol{Ux} = \boldsymbol{0}$ where $\boldsymbol{U} = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}$. Thus we discover the eigenvector $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ for $\lambda = 0$.
- Similarly we discover the eigenvector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ for $\lambda = 2$.

Eigenvalues and eigenvectors - example

- The general procedure to find eigenvalues and eigenvectors is to first find the roots of the characteristic polynomials and then find the nullspaces of the matrices $\boldsymbol{A} - \lambda\boldsymbol{I}$ for the different roots λ.
- Does every $n \times n$ matrix have a full set of eigenvectors, i.e n eigenvectors?
- Look at $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$. What are its eigenvalues and eigenvectors?
- **Point to ponder** Looking at the equation $\boldsymbol{Ax} = \lambda\boldsymbol{x}$ it seems that the action of $\boldsymbol{A}$ on $\boldsymbol{x}$ is to preserve the direction of $\boldsymbol{x}$ but scale it up or down according to λ. Does this mean that a rotation matrix has no eigenvalues and eigenvectors?

Some additional properties

- λ is an eigenvalue of $\boldsymbol{A}$ if and only if λ is a root of the characteristic polynomial $p_{\boldsymbol{A}}(\lambda)$ of $\boldsymbol{A}$. This can be easily seen as a consequence of the definition of $p_{\boldsymbol{A}}(\lambda)$.
- For $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the set of eigenvectors corresponding to an eigenvalue λ spans a subspace of $\mathbb{R}^n$ called the Eigenspace of $\boldsymbol{A}$ with respect to λ and is denoted by E_λ.
- The set of all eigenvalues of $\boldsymbol{A}$ is called the spectrum of $\boldsymbol{A}$.
- Look at the eigenvalues and eigenspace of the $n \times n$ identity matrix $\boldsymbol{I}_n$. It has one eigenvalue $\lambda = 1$ and the eigenspace is $\mathbb{R}^n$. Every canonical vector is a basis vector for the eigenspace.

Some additional properties

- A matrix and its transpose have the same eigenvalues. To see this, first note that $\det(\boldsymbol{A}) = \det(\boldsymbol{A}^T)$. Then $\det(\boldsymbol{A} - \lambda I) = \det((\boldsymbol{A} - \lambda\boldsymbol{I})^T) = \det(\boldsymbol{A}^T - \lambda\boldsymbol{I}^T) = \det(\boldsymbol{A}^T - \lambda\boldsymbol{I})$. The last expression in the chain of equalities is the characteristic polynomial for $p_{\boldsymbol{A}^T}(\lambda)$. Thus we have $p_{\boldsymbol{A}}(\lambda) = p_{\boldsymbol{A}^T}(\lambda)$ which means the characteristic polynomials are equal and so the roots of the polynomials or the eigenvalues must be equal.
- The eigenspace E_λ is the nullspace of $\boldsymbol{A} - \lambda\boldsymbol{I}$.
- Symmetric, positive-definite matrices always have positive, real eigenvalues.

Some theorems

- The eigenvectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots \boldsymbol{x}_n$ of a $n \times n$ matrix $\boldsymbol{A}$ with n **distinct** eigenvalues are linearly independent $\rightarrow$ why?
- Given a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ we can show that $\boldsymbol{A}^T\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is a symmetric, positive-definite matrix when the rank of $\boldsymbol{A} = n$. Why is this true? Clearly $\boldsymbol{A}^T\boldsymbol{A}$ is a symmetric matrix and it is positive definite since $\boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{Ax} = \|\boldsymbol{Ax}\|^2 > 0 \ \ \forall \boldsymbol{x} \in \mathbb{R}^n \setminus 0$ since the nullspaces of $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}$ are the same, and $\boldsymbol{A}$ is a full column rank matrix.
- The matrix $\boldsymbol{A}^T\boldsymbol{A}$ is important in machine learning since it figures in the least-squares solution to a data matrix represented as $\boldsymbol{A}$ where n represents the number of features and m is the number of data vectors.

Spectral theorem

Theorem: If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is symmetric there exists an orthonormal basis of the corresponding vector space V consisting of the eigenvectors of $\boldsymbol{A}$, and each eigenvalue is real.
Proof: We will not attempt a full proof of this theorem but provide some intuitions about why it is true. The theorem relies on the following three statements, shown in the next slide.

Spectral theorem

- All roots of the characteristic polynomial $p_{\boldsymbol{A}}(\lambda)$ are real.
- For each eigenvalue λ we can compute an orthonormal basis for its eigenspace. We can string together the orthonormal bases for the different eigenvalues of $\boldsymbol{A}$ to come up with the vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \ldots$
- The dimension of the eigenspace E_λ, called its geometric multiplicity, is the same as the algebraic multiplicity of λ which is the number of times λ appears as a root of the characteristic polynomial.
- All the basis vectors from the different Eigenspaces combine to provide an orthonormal basis for $\mathbb{R}^n$.

Complex vectors

- In the old formulation with real vectors, length-squared according to the Euclidean norm was $x_1^2 + x_2^2 + \ldots x_n^2$. If the x_i are complex we should take length-squared to be $|x_1|^2 + |x_2|^2 + \ldots |x_n|^2$ where $|.|$ denotes modulus. For the complex number $a + bi$, the modulus is $\sqrt{(a+bi)(a-bi)} = \sqrt{a^2 + b^2}$
- For complex vectors we would like to preserve the idea as possible that $\|x\|^2 = \boldsymbol{x}^T\boldsymbol{x}$. If we keep the old definition of inner product for complex vectors we will not get a real number as length as shown in the next bullet.
- Let $\boldsymbol{x} = \begin{bmatrix} 1+i \\ 2+i \end{bmatrix}$. We have $\boldsymbol{x}^T\boldsymbol{x} = (1+i)^2 + (2+i)^2 = 1 + 2i + i^2 + 4 + 4i + i^2 = 6i + 3$.

Hermitian matrices

- We modify the inner product between two complex vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ to $\boldsymbol{x}^H\boldsymbol{y}$, where $\boldsymbol{x}^H = \overline{\boldsymbol{x}}^T$.
- Now $\boldsymbol{x}^H\boldsymbol{x} = \overline{x_1}x_1 + \ldots \overline{x_n}x_n = \|x\|^2$ according to the new definition of length.
- A Hermitian matrix is a generalization of a symmetric matrix.
- Instead of requiring $\boldsymbol{A}^T = \boldsymbol{A}$, we say a matrix is Hermitian if it is equal to its conjugate-transpose, ie $\boldsymbol{A}$ is a Hermitian matrix if $\boldsymbol{A}^H = \boldsymbol{A}$ or $\overline{\boldsymbol{A}}^T = \boldsymbol{A}$
- As an example consider the matrix $\boldsymbol{A} = \begin{bmatrix} 1 & 3-i \\ 3+i & 4 \end{bmatrix}$. It is a Hermitian matrix since $\boldsymbol{A}^H = \begin{bmatrix} 1 & 3-i \\ 3+i & 4 \end{bmatrix} = \boldsymbol{A}$.

Spectral theorem

We shall now show that all eigenvalues for a symmetric matrix are real. Let $\boldsymbol{Ax} = \lambda\boldsymbol{x}$. Then premultiplying with $\boldsymbol{x}^H$ on both sides we have $\boldsymbol{x}^H\boldsymbol{Ax} = \lambda\boldsymbol{x}^H\boldsymbol{x}$
Now $\boldsymbol{x}^H\boldsymbol{Ax}$ is a 1×1 matrix. Taking the Hermitian of this matrix we have $(\boldsymbol{x}^H\boldsymbol{Ax})^H = \boldsymbol{x}^H\boldsymbol{A}^H\boldsymbol{x} = \boldsymbol{x}^H\boldsymbol{Ax}$, so the Hermitian of the matrix is itself which means that the matrix is real.
On the right hand side we note that $\boldsymbol{x}^H\boldsymbol{x}$ is real, so this means that λ must be real.

Spectral theorem

Let us show that eigenvectors belonging to different eigenvalues are orthogonal. Let $\boldsymbol{Ax} = \lambda\boldsymbol{x}$ and $\boldsymbol{Ay} = \mu\boldsymbol{y}$. Then we have

$$\begin{aligned} \boldsymbol{y}^H\boldsymbol{Ax} &= \lambda\boldsymbol{y}^H\boldsymbol{x} \\ \boldsymbol{x}^H\boldsymbol{Ay} &= \mu\boldsymbol{x}^H\boldsymbol{y} \end{aligned}$$

But $\boldsymbol{x}^H\boldsymbol{Ay} = (\boldsymbol{y}^H\boldsymbol{A}^H\boldsymbol{x})^H = (\boldsymbol{y}^H\boldsymbol{Ax})^H = \lambda\boldsymbol{x}^H\boldsymbol{y}$. We already know that $\boldsymbol{x}^H\boldsymbol{Ay} = \mu\boldsymbol{x}^H\boldsymbol{y}$. This means $\lambda\boldsymbol{x}^H\boldsymbol{y} = \mu\boldsymbol{x}^H\boldsymbol{y}$. Since $\lambda \neq \mu$, this must mean $\boldsymbol{x}^H\boldsymbol{y} = 0$.
This shows that eigenvectors corresponding to different eigenvalues are orthogonal.

Spectral theorem

- So we see that the eigenvalues of a symmetric matrix are real and eigenvectors belonging to different eigenvalues are orthogonal.
- This suggests that one can string together all the orthonormal bases for the different eigenvalues and get an orthonormal basis for $\mathbb{R}^n$.
- But who is to say that when we string together the basis vectors for all the eigenvalues, we will have enough vectors to describe $\mathbb{R}^n$? We need n basis vectors and might end up having fewer than n vectors.
- If the eigenvalues are all different, we can see that we will indeed have enough basis vectors. But what about when there are repeating eignevalues?

Spectral theorem

- We need one more piece to complete the puzzle and show that we will have enough eigenvectors to complete the orthonormal basis - this part we shall not prove!
- As a consequence of the spectral theorem we can write a real symmetric matrix $\boldsymbol{A}$ as $\boldsymbol{A} = \boldsymbol{Q}\Lambda\boldsymbol{Q}^T$ where $\boldsymbol{Q}$ is an orthonormal basis (think orthonormal basis vectors for example), and Λ is a diagonal matrix consisting of non-zero entries only along the diagonal.
- The spectral theorem can be used in a machine learning context since we can take the data matrix $\boldsymbol{A}$ and create a symmetric matrix out of it - $\boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^T$ which are both used in Singular-Value Decomposition and PCA.

Trace and eigenvalues

- We can show that the sum of the eigenvalues of a matrix is equal to the trace of the matrix, i.e $\sum_{i=1}^{i=n}\lambda_i = \sum_{i=1}^{i=n} a_{ii}$. To see why this is true, note that the characteristic polynomial $p_{\boldsymbol{A}}(\lambda)$ can be written as $\prod_{i=1}^{i=n}(\lambda_i - \lambda)$. The coefficient to λ^{n-1} in this expansion is $(-1)^{n-1}\sum_{i=1}^{i=n}\lambda_i$. Early on in this lecture we showed from a direct expansion of the determinant that the coefficient of λ^{n-1} is $(-1)^{n-1}\sum_{i=1}^{i=n} a_{ii}$. Thus we have our result.
- The product of all eigenvalues is the determinant of the matrix, i.e $\det(\boldsymbol{A}) = \prod_{i=1}^{i=n}\lambda_i$. To see why this is true, let us once again look at the factorisation of $p_{\boldsymbol{A}}(\lambda)$ as $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = p_{\boldsymbol{A}}(\lambda) = \prod_{i=1}^{i=n}(\lambda_i - \lambda)$. Setting $\lambda = 0$ in this equation gives the result.

Lecture 5

Math Foundations Team

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Introduction

- In the previous lecture, we discussed eigenvalues and eigenvectors of matrices
- In this lecture, we will look at two related methods for factorizing matrices into canonical forms.
- The first one is known as Eigenvalue decompostion. It uses the concepts of eigenvalues and eigenvectors to generate the decomposition
- The second method known as singular value decomposition or SVD is applicable to all matrices

Diagonal Matrices

- A diagonal matrix is a matrix that has value zero on all off diagonal elements.

$$\mathcal{D} = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}$$

- For a diagonal matrix $\mathcal{D}$, the determinant is the product of its diagonal entries.
- A matrix power $\mathcal{D}^k$ is given by each diagonal element raised to the power k.
- Inverse of a diagonal matrix is obtained by taking inverse of non-zero diagonal entry.

Diagonalizable Matrices

- A matrix $A \in \mathbb{R}^{n \times n}$ is diagonalizable if there exists an invertible matrix $P \in \mathbb{R}^{n \times n}$ and a diagonal matrix $\mathcal{D}$ such that $\mathcal{D} = \mathbf{P}^{-1}\mathbf{AP}$
- In the definition of diagonalization, it is required that $\mathbf{P}$ is an invertible matrix. Assume $p_1, p_2, ..., p_n$ are the n columns of $\mathbf{P}$
- Rewriting we get $\mathbf{AP} = \mathbf{PD}$. By observing that $\mathcal{D}$ is a diagonal matrix, we can simplify as

$$\mathcal{A}p_i = \lambda_i p_i$$

where λ_i is the i^{th} diagonal entry in $\mathcal{D}$.

Diagonalizable Matrix

- Consider a square matrix

$$\mathcal{A} = \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix}$$

- Consider the invertible matrix

$$\mathbf{P} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}$$

- Now consider the product $\mathbf{P}^{-1}\mathbf{AP}$ as follows

$$\begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 1 & 4 \\ 2 & 3 \end{bmatrix} \cdot \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 5 \end{bmatrix}$$

Eigendecomposition of a matrix

- Recall the existence of eigenvalues and eigenvectors for square matrices
- Eigenvalues can be used to create a matrix decomposition known as Eigenvalue Decomposition
- A square matrix $\mathcal{A} \in \mathbb{R}^{n \times n}$ can be factored into

$$\mathbf{A} = \mathbf{PDP}^{-1}$$

- where $\mathcal{P}$ is an invertible matrix of eigenvectors of A assuming we can find n eigenvectors that form a basis of $\mathbb{R}^n$
- and $\mathcal{D}$ is a diagonal matrix whose diagonal entries are the eigenvalues of $\mathcal{A}$

Example of Eigendecomposition

Let us compute the eigendecomposition of the matrix A

$$\mathbf{A} = \begin{bmatrix} 2.5 & -1 \\ -1 & 2.5 \end{bmatrix}$$

- Step 1: Find the eigenvalues and eigenvectors

$$\mathbf{A} - \lambda\mathbf{I} = \begin{bmatrix} 2.5 - \lambda & -1 \\ -1 & 2.5 - \lambda \end{bmatrix}$$

- The characteristic equation is given by $\det(\mathbf{A} - \lambda\mathbf{I}) = \mathbf{0}$
- This leads to the equation $\lambda^2 - 5\lambda + \frac{21}{4} = 0$
- Solving the quadratic equation gives us $\lambda_1 = 3.5$ and $\lambda_2 = 1.5$

Example of Eigendecomposition

- The eigenvector corresponing to $\lambda_1 = 3.5$ is derived as $p_1 = \begin{bmatrix} -\frac{1}{\sqrt{2}}) \\ \frac{1}{\sqrt{2}} \end{bmatrix}$
- The eigenvector corresponing to $\lambda_1 = 1.5$ is derived as $p_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$
- Step 2 : Construct the matrix $\mathbf{P}$ to diagonalize $\mathbf{A}$

$$\mathbf{P} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Example of Eigendecomposition

- The inverse of matrix P is given by

$$\mathbf{P^{-1}} = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- The eigendecompostion of the matrix A is given by

$$\mathbf{A} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 3.5 & 0 \\ 0 & 1.5 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

- In summary we have obtained the required matrix factorization using eigenvalues and eigenvectors.

Symmetric Matrices and Diagonalizability

- Recall that a matrix A is called symmetric matrix if $\mathbf{A} = \mathbf{A^T}$

$$\mathbf{A} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}$$

- A Symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ can always be diagonalized.
- This follows directly from the spectral theorem discussed in previous lecture
- Moreover the spectral theorem states that we can find an orthogonal matrix $\mathbf{P}$ of eigenvectors of $\mathcal{A}$.

Motivation for Singular Value Decomposition

- The singular value decomposition or (SVD) of a matrix is a central matrix decomposition method in linear algebra.
- The eigenvalue decomposition is applicable to square matrices only.
- The singular value decomposition exists for all rectangular matrices
- SVD involves writing a matrix as a product of three matrices $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V^T}$.
- The three component matrices are derived by applying eigenvalue decomposition discussed previously

Singular Value Decomposition Theorem

- Let $\mathbf{A} \in \mathbb{R}^{\mathbf{m} \times \mathbf{n}}$ be a rectangular matrix. Assume that $\mathbf{A}$ has rank r.
- The Singular value decomposition of $\mathcal{A}$ is defined as

$$\mathbf{A} = \mathbf{U\Sigma V^T}$$

- $\mathbf{U} \in \mathbb{R}^{m \times m}$ is an orthogonal matrix with column vectors u_i where $i = 1, ..., m$
- $\mathbf{V} \in \mathbb{R}^{n \times n}$ is an orthogonal matrix with column vectors v_j where $j = 1, ..., n$
- $\mathbf{\Sigma}$ is an m $\times$ n matrix with $\mathbf{\Sigma_{ii}} = \sigma_\mathbf{i} > \mathbf{0}$
- The diagonal entries $\sigma_i, i = 1, ..., r$ of $\mathbf{\Sigma}$ are called the singular values.
- By convention, the singular values are ordered i.e $\mathbf{\Sigma}_{ii} > \mathbf{\Sigma}_{jj}$ where $i < j$.

Properties of Σ

- The singular value matrix $\mathbf{\Sigma}$ is unique.
- Observe that the $\mathbf{\Sigma} \in \mathbb{R}^{\mathbf{m} \times \mathbf{n}}$matrix is rectangular. In particular, $\mathbf{\Sigma}$ is of the same size as $\mathcal{A}$.
- This means that $\mathbf{\Sigma}$ has a diagonal submatrix that contains the singular values and needs additional zero padding.
- Specifically, if $m > n$, then the matrix $\mathbf{\Sigma}$ has diagonal structure up to row n and then consists of zero rows.
- If $m < n$, the matrix Σ has a diagonal structure up to column m and columns that consist of 0 from $m + 1$ to n.

Construction of V

- It can be observed that

$$\mathbf{A^TA} = \mathbf{V\Sigma^T\Sigma V^T}$$

- Since A^TA has the following eigendecomposition

$$\mathbf{A^TA} = \mathbf{P\mathcal{D}P^T}$$

- Therefore, the eigenvectors of $\mathbf{A^TA}$ that compose $\mathbf{P}$ are the right-singular vectors $\mathbf{V}$ of $\mathcal{A}$.
- The eigenvalues of $\mathcal{A}^\mathbf{T}\mathbf{A}$ are the squared singular values of $\mathbf{\Sigma}$

Construction of U

- It can be observed that

$$\mathbf{AA^T} = \mathbf{U\Sigma V^T V\Sigma^T U^T}$$

- Since $\mathbf{AA^T}$ has the following eigendecomposition

$$\mathbf{AA^T} = \mathbf{SDS^T}$$

- Therefore, the eigenvectors of $\mathbf{AA^T}$ that compose $\mathbf{S}$ are the left-singular vectors $\mathbf{U}$ of $\mathcal{A}$

Construction of U continued

- $\mathcal{A} = \mathbf{U\Sigma V^T}$ can be rearranged to obtain a simple formulation for u_i
- By postmultiplying by $\mathbf{V}$ we get $\mathbf{AV} = \mathbf{U\Sigma V^T V}$
- By observing that $\mathbf{V}$ is orthogonal we obtain a simple form

$$\mathbf{AV} = \mathbf{U\Sigma}$$

- This is equivalent to the following

$$u_i = \frac{1}{\sigma_i}\mathcal{A}v_i \quad \forall i = 1, 2, ..., r$$

Computing Singular Value Decomposition 1

- We want to find SVD of the following rectangular matrix $\mathcal{A}$

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}$$

- Let us consider the matrix $\mathcal{A}^T\mathcal{A}$ derived from $\mathcal{A}$ given by

$$\mathcal{A}^T\mathcal{A} = \begin{bmatrix} 5 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

- It is a symmetric matrix

Computing Singular Value Decomposition 2

- Derive the eigendecomposition of $\mathcal{A}^T\mathcal{A}$ in the form PDP^T
- The matrix $\mathbf{P}$ is given by

$$\mathbf{P} = \begin{bmatrix} \frac{-5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{6}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{6}} \end{bmatrix}$$

- The matrix $\mathcal{D}$ is given by

$$\mathcal{D} = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Computing Singular Value Decomposition 3

Now we construct the singular value matrix $\mathbf{\Sigma}$

- The matrix $\mathbf{\Sigma}$ has the dimension same as $\mathcal{A}$. In this case $\mathbf{\Sigma}$ is hence a 2×3 matrix.
- The diagonal entries of submatrix is obtained by taking square root of 6 and 1 respectively
- Singular-value matrix $\mathbf{\Sigma}$ is given by:

$$\mathbf{\Sigma} = \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

- The last column is a column of zeros only

Computing Singular Value Decomposition 4

Left singular vectors as the normalized image of the right singular vectors. Recall that $u_i = \frac{1}{\sigma_i}\mathbf{A}v_i$

- The first vector

$$u_1 = \frac{1}{\sigma_1}\mathcal{A}v_1 = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} \end{bmatrix}$$

- The second vector

$$u_2 = \frac{1}{\sigma_2}\mathcal{A}v_2 = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix}$$

Final Step : Combining U, Σ and V

We compile all the three matrices together to generate the SVD

- $$\mathcal{A} = \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{6}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{6}} \end{bmatrix}^T$$
- The matrix $\mathbf{U}$ is an 2×2 matrix satifying orthogonality property.
- The matrix $\mathbf{V}$ is an 3×3 matrix satifying orthogonality property.

Comparing SVD and EVD

- The left-singular vectors of $\mathcal{A}$ are eigenvectors of $\mathbf{AA^T}$
- The right-singular vectors of $\mathcal{A}$ are eigenvectors of $\mathcal{A}^T\mathcal{A}$
- The non-zero singular values of $\mathcal{A}$ are the square roots of the nonzero eigenvalues of $\mathcal{A}^T\mathcal{A}$.
- The SVD always exists for any matrix in $\mathbb{R}^{m\times n}$
- The eigendecomposition is only defined for square matrices in $\mathbb{R}^{n\times n}$ and only exists if we can find a basis of eigenvectors of $\mathbb{R}^n$

Comparing SVD and EVD

- The vectors in the eigendecomposition matrix $\mathbf{P}$ are not necessarily orthogonal.
- On the other hand, the vectors in the matrices $\mathbf{U}$ and $\mathbf{V}$ in the SVD are orthonormal.
- Both the eigendecomposition and the SVD are compositions of three linear mappings:
- A key difference between the eigendecomposition and the SVD is that in the SVD, domain and codomain can be of different dimensions
- In the SVD, the left and right singular vector matrices $\mathbf{P}$ and $\mathbf{P}$ are generally not inverse of each other.

Comparing SVD and EVD 3

- In the eigendecomposition, the matrices in decomposition are inverse of each other
- In the SVD, the entries in the diagonal matrix $\mathbf{\Sigma}$ are all real and nonnegative,
- In eigendecomposition diagonal matrix entries need not be real always.
- The leftsingular vectors of $\mathcal{A}$ are eigenvectors of $\mathcal{A}\mathcal{A}^T$
- The rightsingular vectors of $\mathcal{A}$ are eigenvectors of $\mathcal{A}^T\mathcal{A}$.

Matrix Approximation

- We considered the SVD as a way to factorize $\mathcal{A} = \mathbf{U\Sigma V^T}$ into the product of three matrices, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $\mathbf{\Sigma}$ contains the singular values on its main diagonal.
- Instead of doing the full SVD factorization, we will now investigate how the SVD allows us to represent a matrix $\mathcal{A}$ as a sum of simpler matrices $\mathcal{A}_i$
- This representation which lends itself to a matrix approximation scheme that is cheaper to compute than the full SVD.

Matrix Approximation continued

- A matrix $\mathcal{A} \in \mathbb{R}^{m \times n}$ of rank r can be written as a sum of rank-1 matrices so that $\mathcal{A} = \sum_{i=1}^{r} \sigma_i u_i v_i^T$
- The diagonal structure of the singular value matrix $\mathbf{\Sigma}$ multiplies only matching left and right singular vectors $u_i v_i^T$ and scales them by the corresponding singular value σ_i.
- All terms $\sigma_i u_i v_i^T$ vanish for $i \neq j$ because $\mathbf{\Sigma}$ is a diagonal matrix.
- Any term for $i > r$ would vanish because the corresponding singular value is 0.

Rank k Approximation

- We summed up the r individual rank-1 matrices to obtain a rank r matrix $\mathcal{A}$.
- If the sum does not run over all matrices A_i $i = 1, ..., r$ but only up to an intermediate value k we obtain a rank-k approximation
- The approximation represented by $\hat{\mathcal{A}}(k)$ is defined as follows

$$\hat{\mathcal{A}}(k) = \sum_{i=1}^{k} \sigma_i u_i v_i^T$$

- To measure the difference between $\mathcal{A}$ and its rank-k approximation we need the notion of a norm which is introduced next

Spectral Norm of a matrix

- We introduce the notation of a subscript in the matrix norm
- Spectral Norm of a Matrix. For $x \in \mathbb{R}^n$, $x \neq \mathbf{0}$, the spectral norm norm of a matrix $\mathcal{A} \in \mathbb{R}^{m \times n}$ is defined as

$$\|\mathcal{A}\|_2 = \max_x \frac{\|\mathcal{A}x\|_2}{\|x\|_2}$$

where $\|y\|_2$ is the euclidean norm of y

- Theorem : The spectral norm of a matrix $\mathcal{A}$ is its largest singular value

Example : Spectral Norm of a matrix

- Example : Consider the following matrix $\mathcal{A}$

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- Singular value decomposition of this matrix will provide the matrix Σ as follows

$$\mathbf{\Sigma} = \begin{bmatrix} 5.465 & 0 \\ 0 & 0.366 \end{bmatrix}$$

- The 2 singular values are 5.4650 and 0.366.
- By definition the spectral norm is the largest singular value.
- Hence, the spectral norm is 5.4650

Lecture 6

Math Foundations Team

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Introduction

Many algorithms in machine learning optimize an objective function with respect to a set of desired model parameters that control how well a model explains the data: Finding good parameters can be phrased as an optimization problem.

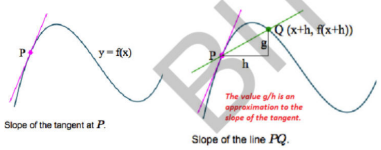
Examples include: linear regression, where we look at curve-fitting problems and optimize linear weight parameters to maximize the likelihood; neural-network auto-encoders for dimensionality reduction and data compression.

Differentiation of Univariate Functions

For $h > 0$, the derivative of f at x is defined as the limit

$$\frac{df}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

The derivative of f points in the direction of steepest ascent of f.



Derivative of a Polynomial

To compute the derivative of $f(x) = x^n$ $n \in N$ using the definition

$$\begin{aligned}\frac{df}{dx} &= \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \\ &= \lim_{h \to 0} \frac{(x+h)^n - x^n}{h} \\ &= \lim_{h \to 0} \frac{\sum_{i=0}^{n} \binom{n}{i} x^{n-i} h^i - x^n}{h} \\ &= \lim_{h \to 0} \frac{\sum_{i=1}^{n} \binom{n}{i} x^{n-i} h^i}{h}\end{aligned} \tag{2}$$

Derivative of a Polynomial

$$\begin{aligned}\frac{df}{dx} &= \lim_{h \to 0} \sum_{i=1}^{n} \binom{n}{i} x^{n-i} h^{i-1} \\ &= \lim_{h \to 0} \binom{n}{1} x^{n-1} + \lim_{h \to 0} \sum_{i=2}^{n} \binom{n}{i} x^{n-i} h^{i-1} \\ &= n x^{n-1}\end{aligned} \tag{3}$$

Taylor polynomial

The Taylor polynomial is a representation of a function f as an finite sum of terms. These terms are determined using derivatives of f evaluated at x_0.

Definition: The Taylor polynomial of degree n of $f : \mathbb{R} \to \mathbb{R}$ at x_0 is defined as

$$T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \tag{4}$$

where $f^{(k)}(x_0)$ is the kth derivative of f at x_0 which we assume exists.

Taylor series

Definition: The Taylor series of smooth (continuously differentiable infinite many times) function $f : \mathbb{R} \to \mathbb{R}$ at x_0 is defined as

$$T_\infty(x) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \tag{5}$$

For $x_0 = 0$, we obtain the Maclaurin series as a special instance of the Taylor series.

Remark: In general, a Taylor polynomial of degree n is an approximation of a function, which does not need to be a polynomial. The Taylor polynomial is similar to f in a neighborhood around x_0. However, a Taylor polynomial of degree n is an exact representation of a polynomial f of degree $k \leq n$ since all derivatives $f^{(i)} = 0$, for $i > k$.

Taylor Polynomial example

Consider the polynomial $f(x) = x^4$. Find the Taylor polynomial T_6 evaluated at $x_0 = 1$.

We compute $f^{(k)}(1)$ for $k = 0, 1, 2..., 6$

$f(1) = 1$, $f'(1) = 4$, $f''(1) = 12$, $f^{(3)}(1) = 24$, $f^{(4)}(1) = 24$, $f^{(5)}(1) = 0$, $f^{(6)}(1) = 0$. The desired Taylor polynomial is

$$\begin{aligned}T_6(x) &= \sum_{k=0}^{6} \frac{f^{(k)}(x_0)}{k!} (x - x_0)^k \\ &= 1 + 4(x-1) + 12(x-1)^2 + 24(x-1)^3 + 24(x-1)^4 \\ &= x^4 = f(x)\end{aligned} \tag{6}$$

we obtain an exact representation of the original function.

Taylor Series example

Consider the smooth function $f(x) = sin(x) + cos(x)$. We compute Taylor series expansion of f at $x_0 = 0$, which is the Maclaurin series expansion of f. We obtain the following derivatives:
$f(0) = sin(0) + cos(0) = 1$
$f^{'}(0) = cos(0) - sin(0) = 1$
$f^{''}(0) = -sin(0) - cos(0) = -1$
$f^{(3)}(0) = -cos(0) + sin(0) = -1$
$f^{(4)}(0) = sin(0) + cos(0) = f(0) = 1$
The coefficients in our Taylor series are only ± 1 (since $sin(0) = 0$), each of which occurs twice before switching to the other one. Furthermore, $f^{(k+4)}(0) = f^{k}(0)$

Taylor Series example

Therefore, the full Taylor series expansion of f at $x_0 = 0$ is given by

$$\begin{aligned} T_\infty(x) &= \sum_{k=0}^{\infty} \frac{f^{(k)}(x_0)}{k!}(x - x_0)^k \\ &= 1 + x - \frac{1}{2!}x^2 - \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 - \dots \\ &= 1 - \frac{1}{2!}x^2 + \frac{1}{4!}x^4 \mp \dots x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 \mp \dots \\ &= \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k)!}x^{2k} + \sum_{k=0}^{\infty}(-1)^k \frac{1}{(2k+1)!}x^{2k+1} \\ &= cos(x) + sin(x) \end{aligned} \tag{7}$$

Differentiation Rules

We denote the derivative of f by $f^{'}$

- Product Rule: $(f(x)g(x))^{'} = f^{'}(x)g(x) + f(x)g^{'}(x)$
- Sum Rule: $(f(x) + g(x))^{'} = f^{'}(x) + g^{'}(x)$
- Quotient Rule: $(\frac{f(x)}{g(x)})^{'} = \frac{f^{'}(x)g(x) - f(x)g^{'}(x)}{(g(x))^2}$
- Chain Rule: $(g(f(x))^{'} = (g \circ f)^{'}(x) = g^{'}(f(x))f^{'}(x)$

Example: Chain Rule

Compute the derivative of function $h(x) = (2x + 1)^4$
$h(x) = (2x + 1)^4 = g(f(x))$
$f(x) = 2x + 1,$
$g(f) = f^4$
Derivatives of f and g are
$f^{'}(x) = 2$
$g^{'}(f) = 4f^3$
$h^{'}(x) = g^{'}(f)f^{'}(x) = (4f^3).2 = 8(2x + 1)^3$

Partial Differentiation and Gradients

Differentiation applies to functions f of a scalar variable $x \in R$. In the following, we consider the general case where the function f depends on one or more variables $x \in R^n$, e.g.,$f(x) = f(x_1, x_2)$. The generalization of the derivative to functions of several variables is the gradient. We find the gradient of the function f with respect to x by varying one variable at a time and keeping the others constant. The gradient is then the collection of these partial derivatives.

Partial derivatives and Gradients

Definition: For a function $f : \mathbb{R}^n \to \mathbb{R}$, $x \to f(x)$, $x \in R^n$ of n variables $x_1, \dots, x_n$ we define the *partial derivatives* as

$$\begin{aligned} \frac{\partial f}{\partial x_1} &= \lim_{h \to 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{h} \\ \frac{\partial f}{\partial x_2} &= \lim_{h \to 0} \frac{f(x_1, x_2 + h, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{h} \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= \lim_{h \to 0} \frac{f(x_1, x_2, \dots, x_n + h) - f(x_1, x_2, \dots, x_n)}{h} \end{aligned}$$

We collect them in the row vector called the gradient of f or Jacobian

$$\Delta_x f = gradf = \frac{df}{dx} = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, \dots, \frac{\partial f(x)}{\partial x_n}\right] \tag{8}$$

Example 1: Find the partial derivatives of $f(x, y) = (x + 2y^3)^2$

$$\frac{\partial f(x,y)}{\partial x} = 2(x + 2y^3)\frac{\partial (x + 2y^3)}{\partial x} = 2(x + 2y^3) \tag{9}$$

$$\frac{\partial f(x,y)}{\partial y} = 2(x + 2y^3)\frac{\partial (x + 2y^3)}{\partial y} = 12y^2(x + 2y^3) \tag{10}$$

here we used the chain rule to compute the partial derivatives.

Example 2

Find the partial derivatives of $f(x_1, x_2) = x_1^2 x_2 + x_1 x_2^3$

$$\frac{\partial f(x_1, x_2)}{\partial x_1} = 2x_1 x_2 + x_2^3 \tag{11}$$

$$\frac{\partial f(x_1, x_2)}{\partial x_2} = x_1^2 + 3x_1 x_2^2 \tag{12}$$

So the gradient is then

$$\frac{df}{dx} = \left[\frac{\partial f(x_1, x_2)}{\partial x_1}, \frac{\partial f(x_1, x_2)}{\partial x_2}\right] = [2x_1 x_2 + x_2^3, x_1^2 + 3x_1 x_2^2] \in \mathbb{R}^{1 \times 2} \tag{13}$$

Basic rules of partial differentiation

When we compute derivatives with respect to vectors $x \in \mathbb{R}^n$ we need to pay attention: Our gradients now involve vectors and matrices, and matrix multiplication is not commutative i.e., the order matters.

$$\text{Product rule: } \frac{\partial}{\partial x}(f(x)g(x)) = \frac{\partial f}{\partial x}g(x) + f(x)\frac{\partial g}{\partial x} \tag{14}$$

$$\text{Sum rule: } \frac{\partial}{\partial x}(f(x) + g(x)) = \frac{\partial f}{\partial x} + \frac{\partial g}{\partial x} \tag{15}$$

$$\text{chain rule: } \frac{\partial}{\partial x}(g \circ f)(x) - \frac{\partial}{\partial x}(g(f(x)) - \frac{\partial g}{\partial f}\frac{\partial f}{\partial x} \tag{16}$$

Chain Rule

Consider a function $f : \mathbb{R} \to \mathbb{R}$ of two variables x_1, x_2. Furthermore, $x_1(t)$ and $x_2(t)$ are themselves functions of t.

To compute the gradient of f with respect to t, we need to apply the chain rule for multivariate functions as

$$\frac{df}{dt} = \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t}\end{bmatrix} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{17}$$

where d denotes the gradient and ∂ partial derivatives.

Example

Consider $f(x_1, x_2) = x_1^2 + 2x_2$, where $x_1 = \sin t$ and $x_2 = \cos t$ then

$$\begin{aligned}\frac{df}{dt} &= \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \\ &= 2\sin t\frac{\partial \sin t}{\partial t} + 2\frac{\partial \cos t}{\partial t} \\ &= 2\sin t\cos t - 2\sin t = 2\sin t(\cos t - 1)\end{aligned}$$

is the corresponding derivative of f with respect to t.

If $f(x_1, x_2)$ is a function of x_1 and x_2, where $x_1(s, t)$ and $x_2(s, t)$ are themselves functions of two variables s and t, the chain rule yields the partial derivatives:

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial s} \tag{18}$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial t} \tag{19}$$

and the gradient is obtained by the matrix multiplication

$$\begin{aligned}\frac{df}{d(s,t)} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial (s,t)} \\ &= \begin{bmatrix}\frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2}\end{bmatrix}\begin{bmatrix}\frac{\partial x_1}{\partial s} & \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial s} & \frac{\partial x_2}{\partial t}\end{bmatrix}\end{aligned}$$

Gradients of Vector-Valued Functions

We have discussed partial derivatives and gradients of functions $f : \mathbb{R}^n \to \mathbb{R}$ mapping to the real numbers. Now we will generalize the concept of the gradient to vector-valued functions $f : \mathbb{R}^n \to \mathbb{R}^m$, where $n \geq 1$ and $m > 1$.
For a function $f : \mathbb{R}^n \to \mathbb{R}^m$ and a vector $x = [x_1, \ldots, x_n]^T$ corresponding vector of function values is given as

$$f(x) = \begin{bmatrix}f_1(x) \\ \vdots \\ f_m(x)\end{bmatrix} \in \mathbb{R}^m \tag{20}$$

where each $f_i : \mathbb{R}^n \to \mathbb{R}$

Gradients of Vector-Valued Functions

Therefore, the partial derivative of a vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ w.r.t. $x_i \in R$, $i = 1, \ldots n$ is given as the vector

$$\begin{aligned}\frac{\partial f}{\partial x_i} &= \begin{bmatrix}\frac{\partial f_1}{\partial x_i} \\ \vdots \\ \frac{\partial f_m}{\partial x_i}\end{bmatrix} \\ &= \begin{bmatrix}\lim_{h\to 0}\frac{f_1(x_1,\ldots,x_{i-1},x_i+h,x_{i+1},\ldots,x_n)-f_1(x)}{h} \\ \vdots \\ \lim_{h\to 0}\frac{f_m(x_1,\ldots,x_{i-1},x_i+h,x_{i+1},\ldots,x_n)-f_m(x)}{h}\end{bmatrix} \in \mathbb{R}^m\end{aligned}$$

Gradients of Vector-Valued Functions

We know that the gradient of f with respect to a vector is the row vector of the partial derivatives. Every partial derivative $\frac{\partial f}{\partial x_i}$ is itself a column vector. Therefore, we obtain the gradient of $f : \mathbb{R}^n \to \mathbb{R}^m$ with respect to $x \in \mathbb{R}^n$ by collecting these partial derivatives:

$$\begin{aligned}\frac{df(x)}{dx} &= \begin{bmatrix}\frac{\partial f(x)}{\partial x_1} & \cdots & \frac{\partial f(x)}{\partial x_n}\end{bmatrix} \\ &= \begin{bmatrix}\frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & & \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n}\end{bmatrix} \in \mathbb{R}^{m\times n}\end{aligned}$$

Example 1: Gradients of Vector-Valued Functions

Given $f(x) = Ax$, $f(x) \in \mathbb{R}^M$, $A \in \mathbb{R}^{M\times N}$, $x \in \mathbb{R}^N$
Since $f : \mathbb{R}^N \to \mathbb{R}^M$, it follows that $df/dx \in \mathbb{R}^{M\times N}$. To compute the gradient we determine the partial derivatives of f w.r.t x_j:

$$f_i(x) = \sum_{i=1}^{N} A_{ij}x_j \implies \frac{\partial f_i}{\partial x_j} = A_{ij} \tag{21}$$

We obtain the gradient using Jacobian

$$\frac{df}{dx} = \begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & & \\ \frac{\partial f_M}{\partial x_1} & \cdots & \frac{\partial f_M}{\partial x_N}\end{bmatrix} = \begin{bmatrix}A_{11} \ldots A_{1N} \\ \vdots \\ A_{M1} \ldots A_{MN}\end{bmatrix} = A \in \mathbb{R}^{M\times N} \tag{22}$$

Example 2: Gradients of Vector-Valued Functions

Consider the function $h : \mathbb{R} \to \mathbb{R}, \quad h(t) = (f \circ g)(t)$ with $f(x) = exp(x_1 x_2^2)$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = g(t) = \begin{bmatrix} t \cos t \\ t \sin t \end{bmatrix} \tag{23}$$

and compute the gradient of h w.r.t. t. Since $f : \mathbb{R}^2 \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}^2$ we note that

$$\frac{\partial f}{\partial x} \in \mathbb{R}^{1\times 2} \text{ and } \frac{\partial g}{\partial t} \in \mathbb{R}^{2\times 1} \tag{24}$$

The desired gradient is computed by applying the chain rule:

$$\begin{aligned} \frac{dh}{dt} &= \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} = \begin{bmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \end{bmatrix} \\ &= \begin{bmatrix} exp(x_1 x_2^2) x_2^2 & 2exp(x_1 x_2^2) x_1 x_2 \end{bmatrix} \begin{bmatrix} \cos t - t \sin t \\ \sin t + t \cos t \end{bmatrix} \\ &= exp(x_1 x_2^2)(x_2^2(\cos t - t \sin t) + 2x_1 x_2(\sin t + t \cos t)) \end{aligned}$$

where $x_1 = t \cos t$ and $x_2 = t \sin t$;

Introduction

- In last lecture, we discussed about differentiation of univariate functions, partial differentiation, gradients and gradients of vector valued functions.
- Now we will look into gradients of matrices and some useful identities for computing gradients.
- Finally, we will discuss back propagation and automatic differentiation.

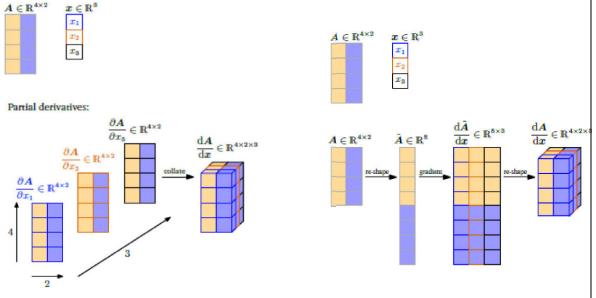
Gradients of Matrices

The gradient of an $m \times n$ matrix A with respect to a $p \times q$ matrix B, the resulting Jacobian would be an $(m \times n) \times (p \times q)$, i.e., a four-dimensional tensor J, whose entries are given as

$$J_{ijkl} = \frac{\partial A_{ij}}{\partial B_{kl}}$$

Since, we can consider $\mathbb{R}^{m\times n}$ as $\mathbb{R}^{mn}$, we can shape our matrix into vectors of length mn and pq respectively. The gradient using mn vectors results in a Jacobian of size $mn \times pq$

Gradients of Matrices



Gradients of Matrices

Let $f = Ax$ where $A \in \mathbb{R}^{m\times n}$, and $x \in \mathbb{R}^n$, then

$$\frac{\partial f}{\partial A} \in \mathbb{R}^{m\times(m\times n)}$$

By definition

$$\frac{\partial f}{\partial A} = \begin{bmatrix} \frac{\partial f_1}{\partial A} \\ \vdots \\ \frac{\partial f_m}{\partial A} \end{bmatrix}, \frac{\partial f_i}{\partial A} \in \mathbb{R}^{1\times(m\times n)}$$

Gradients of Matrices

Now, we have

$$f_i = \sum_{j=1}^{n} A_{ij}x_j, i = 1, \cdots, m.$$

Therefore, by taking partial derivatives with respect to A_{iq}

$$\frac{\partial f_i}{\partial A_{iq}} = x_q.$$

Hence, i^{th} row becomes

$$\frac{\partial f_i}{\partial A_{i,:}} = x^T \in \mathbb{R}^{1\times 1\times n}$$

$$\frac{\partial f_i}{\partial A_{k,:}} = 0^T \in \mathbb{R}^{1\times 1\times n}, \text{for} k \neq i$$

Hence, by stacking the partial derivatives, we get

$$\frac{\partial f_i}{\partial A_{k,:}} = \begin{bmatrix} 0^T \\ \vdots \\ x^T \\ \vdots \\ 0^T \end{bmatrix} \in \mathbb{R}^{1\times m\times n}$$

Gradients of Matrices with respect to Matrices

Let $B \in \mathbb{R}^{m\times n}$ and $f : \mathbb{R}^{m\times n} \to \mathbb{R}^{n\times n}$ with

$$f(B) = B^T B =: K \in \mathbb{R}^{n\times n}$$

Then, we have

$$\frac{\partial K}{\partial B} \in \mathbb{R}^{(n\times n)\times(m\times n)}.$$

Moreover

$$\frac{\partial K_{pq}}{\partial B} \in \mathbb{R}^{1\times(m\times n)}, \text{for} p, q = 1, \cdots n$$

where K_{pq} is the $(p, q)^{th}$ entry of $K = f(B)$

Gradients of Matrices with respect to Matrices

Let i^{th} column of B be b_i, then

$$K_{pq} = r_p^T r_q = \sum_{l=1}^{m} B_{lp}B_{lq}$$

Computing the partial derivative, we get

$$\frac{\partial K_{pq}}{\partial B_{ij}} = \sum_{l=1}^{m} \frac{\partial}{\partial B_{ij}} B_{lp}B_{lq} = \partial_{pqij}$$

Gradients of Matrices with respect to Matrices

Clearly, we have

$$\begin{aligned} \partial_{pqij} &= B_{iq} \quad \text{if } j = p, p \neq q \\ \partial_{pqij} &= B_{ip} \quad \text{if } j = q, p \neq q \\ \partial_{pqij} &= 2B_{iq} \quad \text{if } j = p, p = q \\ \partial_{pqij} &= 0 \quad \text{otherwise} \end{aligned}$$

where $p, q, j = 1, \cdots, n$ $i = 1, \cdots, m$

Useful Identities for Computing Gradients

- $\frac{\partial}{\partial X} f(X)^T = (\frac{\partial f(X)}{\partial X})^T$
- $\frac{\partial}{\partial X} tr(f(X)) = tr(\frac{\partial f(X)}{\partial X})$
- $\frac{\partial}{\partial X} det(f(X)) = det(f(X))tr(f(X)^{-1}\frac{\partial f(X)}{\partial X})$
- $\frac{\partial}{\partial X} f(X)^{-1} = -f(X)^{-1}\frac{\partial f(X)}{\partial X} f(X)^{-1}$

Useful Identities for Computing Gradients

- $\frac{\partial a^T X^{-1} b}{\partial X} = -(X^{-1})^T ab^T (X^{-1})^T$
- $\frac{\partial x^T a}{\partial x} = a^T$
- $\frac{\partial a^T x}{\partial x} = a^T$
- $\frac{\partial a^T Xb}{\partial X} = ab^T$
- $\frac{\partial x^T B}{\partial x} = x^T(B + B^T)$
- $\frac{\partial}{\partial s}(x - As)^T W(x - As) = -2(x - As)^T WA$

for symmetric W.

Backpropagation and Automatic Differentiation

Consider the function

$$f(x) = \sqrt{x^2 + exp(x^2)} + cos(x^2 + exp(x^2))$$

Taking dervatives

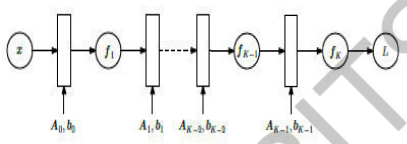
$$\begin{aligned}\frac{df}{dx} &= \frac{2x + 2xexp(x^2)}{2\sqrt{x^2 + exp(x^2)}} - sin(x^2 + exp(x^2))(2x + 2xexp(x^2)) \\ &= 2x(\frac{1}{2\sqrt{x^2 + exp(x^2)}} - sin(x^2 + exp(x^2)))(1 + exp(x^2))\end{aligned}$$

Motivation

- The implementation of the gradient could be significantly more expensive than computing the function, which imposes unnecessary overhead where we get such lengthy expressions.
- We need an efficient way to compute the gradient of an error function with respect to the parameters of the model.
- For training deep neural network models, the backpropagation algorithm is one such method.

Backpropagation and Automatic Differentiation

In neural networks with multiple layers



$$f_i(x_{i-1}) = \sigma(A_{i-1}x_{i-1} + b_{i-1})$$

where x_{i-1} is the output of layer $i-1$ and σ is an activation function.

Backpropagation

To train these model, the gradient of the loss function L with respect to all model parameters $\theta_j = \{A_j, b_j\}, j = 1, \cdots, K$ and inputs of each layer needs to be computed. Consider,

$$\begin{aligned}f_0 &:= x \\ f_i &:= \sigma_i(A_{i-1}f_{i-1} + b_{i-1}), i = 1, \cdots, K.\end{aligned}$$

We have to find $\theta_j = \{A_j, b_j\}, j = 1, \cdots, K-1$ such that

$$L(\theta) = ||y - f_K(\theta, x)||^2$$

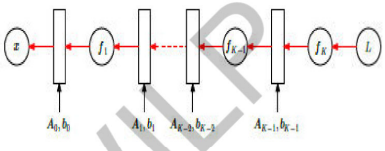
is minimum where $\theta = \{A_0, b_0, \cdots, A_{K-1}, b_{K-1}\}$

Backpropagation

Using the chain rule, we get

$$\begin{aligned}\frac{\partial L}{\partial \theta_{K-1}} &= \frac{\partial L}{\partial f_K}\frac{\partial f_K}{\partial \theta_{K-1}} \\ \frac{\partial L}{\partial \theta_{K-2}} &= \frac{\partial L}{\partial f_K}\frac{f_K}{f_{K-1}}\frac{\partial f_{K-1}}{\partial \theta_{K-2}} \\ \frac{\partial L}{\partial \theta_{K-3}} &= \frac{\partial L}{\partial f_K}\frac{f_K}{f_{K-1}}\frac{\partial f_{K-1}}{\partial f_{K-2}}\frac{\partial f_{K-2}}{\partial \theta_{K-3}} \\ \frac{\partial L}{\partial \theta_i} &= \frac{\partial L}{\partial f_K}\frac{f_K}{f_{K-1}}\cdots\frac{\partial f_{i+2}}{\partial f_{i+1}}\frac{\partial f_{i+1}}{\partial \theta_i}\end{aligned}$$

Backpropagation



If the partial derivatives $\frac{\partial L}{\partial \theta_{i+1}}$ are computed, then the computation can be reused to compute $\frac{\partial L}{\partial \theta_i}$.

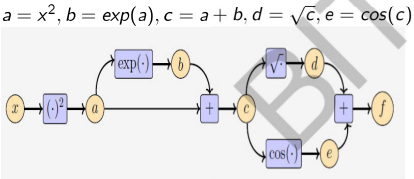
Example

Consider the function

$$f(x) = \sqrt{x^2 + exp(x^2)} + cos(x^2 + exp(x^2))$$

Let

$a = x^2, b = exp(a), c = a + b, d = \sqrt{c}, e = cos(c) \Rightarrow f = d + e$



Example

$$\begin{aligned}\Rightarrow \frac{\partial a}{\partial x} &= 2x \\ \frac{\partial b}{\partial a} &= exp(a) \\ \frac{\partial c}{\partial a} &= 1 = \frac{\partial c}{\partial b} \\ \frac{\partial d}{\partial c} &= \frac{1}{2\sqrt{c}} \\ \frac{\partial e}{\partial c} &= -sin(c) \\ \frac{\partial f}{\partial d} &= 1 = \frac{\partial f}{\partial e}\end{aligned}$$

Example

Thus, we have

$$\begin{aligned}\frac{\partial f}{\partial c} &= \frac{\partial f}{\partial d}\frac{\partial d}{\partial c} + \frac{\partial f}{\partial e}\frac{\partial e}{\partial c} \\ \frac{\partial f}{\partial b} &= \frac{\partial f}{\partial c}\frac{\partial c}{\partial b} \\ \frac{\partial f}{\partial a} &= \frac{\partial f}{\partial b}\frac{\partial b}{\partial a} + \frac{\partial f}{\partial c}\frac{\partial c}{\partial a} \\ \frac{\partial f}{\partial x} &= \frac{\partial f}{\partial a}\frac{\partial a}{\partial x}\end{aligned}$$

Example

Substituting the results, we get

$$\begin{aligned}\frac{\partial f}{\partial c} &= 1.(\frac{1}{2\sqrt{c}} + 1).(-sin(c)) \\ \frac{\partial f}{\partial b} &= \frac{\partial f}{\partial c}.1 \\ \frac{\partial f}{\partial a} &= \frac{\partial f}{\partial b}exp(a) + \frac{\partial f}{\partial c}.1 \\ \frac{\partial f}{\partial x} &= \frac{\partial f}{\partial a}2x\end{aligned}$$

Thus, the computation for calculating the derivative is of similar complexity as the computation of the function itself.

Formalization of Automatic Differentiation

Let $x_1, \cdots, x_d$: input variables.
$x_{d+1}, \cdots, x_{D-1}$: intermediate variables.
x_D : output variable, then we have,

$$x_i = g_i(x_{Pa(x_i)})$$

Note that g_is are elementary functions and are also called as forward propagation function and $x_{Pa(x_i)}$ is the set of parent nodes of variable x_i in the graph.

Formalization of Automatic Differentiation

Now,

$$f = x_D \Rightarrow \frac{\partial f}{\partial_D} = 1$$

For other variables, using chain rule, we get

$$\frac{\partial f}{\partial x_i} = \sum_{x_j : x_i \in Pa(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial x_i} = \sum_{x_j : x_i \in Pa(x_j)} \frac{\partial f}{\partial x_j}\frac{\partial g_i}{\partial x_i}$$

The last equation is the back propagation of the gradient through the computation graph. For neural network training, we back propagate the error of the prediction with respect to the label.

Introduction

- Till now we have discussed about Taylor/Maclaurian series, Partial Derivatives and Gradients.
- Now we are interested in Higher order Derivatives.
- Multivariate Taylor Series and its uses in the expansion of a function with multivariables.

Higher-Order Derivatives

Consider a function $f : \mathbb{R}^2 \to \mathbb{R}$

Notations for Higher-Order Partial Derivatives:

$\frac{\partial^2 f}{\partial x^2}$: Second Partial Derivative of x w.r.t. x

$\frac{\partial^n f}{\partial x^n}$: n^{th} Partial Derivative of x w.r.t. x

$\frac{\partial^2 f}{\partial y \partial x} = \frac{\partial}{\partial y}(\frac{\partial f}{\partial x})$: is the partial derivative obtained by first partial differentiating with respect to x and then with respect to y.

$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}(\frac{\partial f}{\partial y})$: is the partial derivative obtained by first partial differentiating by y and then x.

Hessian Matrix

The Hessian is the collection of all second-order partial derivatives. If f(x, y) is a twice (continuously) differentiable function, then $\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial^2 f}{\partial y \partial x}$ i.e., the order of differentiation does not matter, and the corresponding Hessian matrix

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

is symmetric. The Hessian is denoted as $\nabla^2_{x,y} f(x, y)$

Linearization and Multivariate Taylor Series

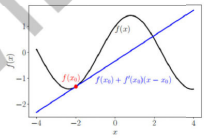
The gradient ∇f of a function f is often used for a locally linear approximation of f around x_0 :

$$f(x) \approx f(x_0) + (\nabla_x f)(x_0)(x - x_0) \tag{1}$$

Here $(\nabla_x f)(x_0)$ is the gradient of f with respect to x, evaluated atx_0. Figure illustrates the linear approximation of a function f at an input x_0. The original function is approximated by a straight line.

Linearization and Multivariate Taylor Series...

This approximation is locally accurate, but the farther we move away from x_0 the worse the approximation gets. Equation (1) is a special case of a multivariate Taylor series expansion of f at x_0, where we consider only the first two terms. We discuss the more general case in the following, which will allow for better approximations.



Linear approximation of a function. The original function $f$is linearized at $x_0 = -2$ using a first-order Taylor series expansion.

Multivariate Taylor Series

Consider a function $f : \mathbb{R}^D \to \mathbb{R}$, $x \to f(x)$,

$$x \in \mathbb{R}^D$$

that is smooth at x_0. When we define the difference vector $\delta := x - x_0$ the multivariate Taylor series of f at (x_0) is defined as multivariate Taylor series

$$f(x) = \sum_{k=0}^{\infty} \frac{D_x^k f(x_0)}{k!} \delta^k \tag{2}$$

where $D_x^k f(x_0)$ is the k^{th} (total) derivative of f with respect to x, evaluated at x_0.

Taylor Polynomial

The Taylor polynomial of degree n of Taylor polynomial f at x_0 contains the first n + 1 components of the series in (2) and is defined as

$$T_n(x) = \sum_{k=0}^{n} \frac{D_x^k f(x_0)}{k!} \delta^k \tag{3}$$

In (2) and (3), we used the slightly sloppy notation of δ^k, which is not defined for vectors

$$x \in \mathbb{R}^D,$$

$D > 1$, and $k > 1$. Note that both $D_x^k f$ and δ^k are k^{th} order tensors, i.e., k-dimensional arrays.

Taylor Polynomial...

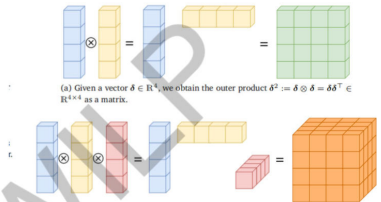
kth-order tensor $\boldsymbol{\delta}^k \in \mathbb{R}^{\overbrace{D \times D \times \ldots \times D}^{k \text{ times}}}$ is obtained as a k-fold outer product, denoted by $\otimes$, of the vector $\boldsymbol{\delta} \in \mathbb{R}^D$. For example,

$$\boldsymbol{\delta}^2 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top, \quad \boldsymbol{\delta}^2[i,j] = \delta[i]\delta[j]$$

$$\boldsymbol{\delta}^3 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta}, \quad \boldsymbol{\delta}^3[i,j,k] = \delta[i]\delta[j]\delta[k].$$

Taylor Polynomial...

.



(a) Given a vector $\boldsymbol{\delta} \in \mathbb{R}^4$, we obtain the outer product $\boldsymbol{\delta}^2 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} = \boldsymbol{\delta}\boldsymbol{\delta}^\top \in \mathbb{R}^{4\times4}$ as a matrix.

(b) An outer product $\boldsymbol{\delta}^3 := \boldsymbol{\delta} \otimes \boldsymbol{\delta} \otimes \boldsymbol{\delta} \in \mathbb{R}^{4\times4\times4}$ results in a third-order tensor ("three-dimensional matrix"), i.e., an array with three indexes.

$$D_x^k f(\boldsymbol{x}_0)\boldsymbol{\delta}^k = \sum_{i_1=1}^{D} \cdots \sum_{i_k=1}^{D} D_x^k f(\boldsymbol{x}_0)[i_1, \ldots, i_k]\delta[i_1] \cdots \delta[i_k]$$

Taylor Polynomial...

In general, we obtain the terms in the Taylor series, where $D_x^k f(x_0)\delta^k$ contains k^{th} order polynomials.Now that we defined the Taylor series for vector fields, let us explicitly write down the first terms $D_x^k f(x_0)\delta^k$ of the Taylor series expansion for

$k = 0, \ldots, 3$ and $\boldsymbol{\delta} := \boldsymbol{x} - \boldsymbol{x}_0$:

$$k = 0 : D_x^0 f(\boldsymbol{x}_0)\boldsymbol{\delta}^0 = f(\boldsymbol{x}_0) \in \mathbb{R}$$

$$k = 1 : D_x^1 f(\boldsymbol{x}_0)\boldsymbol{\delta}^1 = \underbrace{\nabla_x f(\boldsymbol{x}_0)}_{1\times D}\underbrace{\boldsymbol{\delta}}_{D\times 1} = \sum_{i=1}^{D} \nabla_x f(\boldsymbol{x}_0)[i]\delta[i] \in \mathbb{R}$$

$$k = 2 : D_x^2 f(\boldsymbol{x}_0)\boldsymbol{\delta}^2 = \mathrm{tr}(\underbrace{\boldsymbol{H}(\boldsymbol{x}_0)}_{D\times D}\underbrace{\boldsymbol{\delta}}_{D\times 1}\underbrace{\boldsymbol{\delta}^\top}_{1\times D}) = \boldsymbol{\delta}^\top \boldsymbol{H}(\boldsymbol{x}_0)\boldsymbol{\delta}$$

$$= \sum_{i=1}^{D}\sum_{j=1}^{D} H[i,j]\delta[i]\delta[j] \in \mathbb{R}$$

$$k = 3 : D_x^3 f(\boldsymbol{x}_0)\boldsymbol{\delta}^3 = \sum_{i=1}^{D}\sum_{j=1}^{D}\sum_{k=1}^{D} D_x^3 f(\boldsymbol{x}_0)[i,j,k]\delta[i]\delta[j]\delta[k] \in \mathbb{R}$$

Here, $\boldsymbol{H}(\boldsymbol{x}_0)$ is the Hessian of f evaluated at $\boldsymbol{x}_0$.

Taylor Series Expansion of a Function with Two Variables

Consider the function $f(x, y) = x^2 + 2xy + y^3$.
We want to compute the Taylor series expansion of f at $(x_0, y_0) = (1, 2)$.
Before we start, let us discuss what to expect: The function in $f(x, y)$ is a polynomial of degree 3. We are looking for a Taylor series expansion,which itself is a linear combination of polynomials. Therefore, we do not expect the Taylor series expansion to contain terms of fourth or higher order to express a third-order polynomial. This means that it should be sufficient to determine the first four terms of $f(x) = \sum_{k=0}^{\infty} \frac{D_x^k f(x_0)}{k!}\delta^k$ for an exact alternative representation of $f(x, y)$. To determine the Taylor series expansion, we start with the constant term and the first-order derivatives, which are given by $f(1, 2) = 13$

Taylor Series Expansion of a Function with Two Variables...

$$\frac{\partial f}{\partial x} = 2x + 2y \implies \frac{\partial f}{\partial x}(1,2) = 6$$
$$\frac{\partial f}{\partial y} = 2x + 3y^2 \implies \frac{\partial f}{\partial y}(1,2) = 14\,.$$

Therefore, we obtain

$$D^1_{x,y} f(1,2) = \nabla_{x,y} f(1,2) = \begin{bmatrix} \frac{\partial f}{\partial x}(1,2) & \frac{\partial f}{\partial y}(1,2) \end{bmatrix} = \begin{bmatrix} 6 & 14 \end{bmatrix} \in \mathbb{R}^{1\times 2}$$

such that

$$\frac{D^1_{x,y} f(1,2)}{1!}\boldsymbol{\delta} = \begin{bmatrix} 6 & 14 \end{bmatrix} \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} = 6(x-1) + 14(y-2)\,.$$

Therefore, we obtain

$$D^1_{x,y} f(1,2) = \nabla_{x,y} f(1,2) = \begin{bmatrix} \frac{\partial f}{\partial x}(1,2) & \frac{\partial f}{\partial y}(1,2) \end{bmatrix} = \begin{bmatrix} 6 & 14 \end{bmatrix} \in \mathbb{R}^{1\times 2}$$

Taylor Series Expansion of a Function with Two Variables...

When we collect the second-order partial derivatives, we obtain the Hessian

$$\boldsymbol{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 2 & 6y \end{bmatrix},$$

such that

$$\boldsymbol{H}(1,2) = \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \in \mathbb{R}^{2\times 2}.$$

Therefore, the next term of the Taylor-series expansion is given

$$\begin{aligned} \frac{D^2_{x,y} f(1,2)}{2!}\boldsymbol{\delta}^2 &= \frac{1}{2}\boldsymbol{\delta}^\top \boldsymbol{H}(1,2)\boldsymbol{\delta} \\ &= \frac{1}{2}\begin{bmatrix} x-1 & y-2 \end{bmatrix} \begin{bmatrix} 2 & 2 \\ 2 & 12 \end{bmatrix} \begin{bmatrix} x-1 \\ y-2 \end{bmatrix} \\ &= (x-1)^2 + 2(x-1)(y-2) + 6(y-2)^2\,. \end{aligned}$$

Taylor Series Expansion of a Function with Two Variables...

The third-order derivatives are obtained as

$$D^3_{x,y} f = \begin{bmatrix} \frac{\partial \boldsymbol{H}}{\partial x} & \frac{\partial \boldsymbol{H}}{\partial y} \end{bmatrix} \in \mathbb{R}^{2\times 2\times 2},$$

$$D^3_{x,y} f[:,:,1] = \frac{\partial \boldsymbol{H}}{\partial x} = \begin{bmatrix} \frac{\partial^3 f}{\partial x^3} & \frac{\partial^3 f}{\partial x^2 \partial y} \\ \frac{\partial^3 f}{\partial x \partial y \partial x} & \frac{\partial^3 f}{\partial x \partial y^2} \end{bmatrix},$$

$$D^3_{x,y} f[:,:,2] = \frac{\partial \boldsymbol{H}}{\partial y} = \begin{bmatrix} \frac{\partial^3 f}{\partial y \partial x^2} & \frac{\partial^3 f}{\partial y \partial x \partial y} \\ \frac{\partial^3 f}{\partial y^2 \partial x} & \frac{\partial^3 f}{\partial y^3} \end{bmatrix}.$$

Taylor Series Expansion of a Function with Two Variables...

Since most second-order partial derivatives in the Hessian, are constant, the only nonzero third-order partial derivative is $\frac{\partial^3 f}{\partial y^3} = 6 \implies \frac{\partial^3 f}{\partial y^3}(1,2) = 6$ Higher-order derivatives and the mixed derivatives of degree 3 (e.g., $\frac{\partial^3 f}{\partial x^2 \partial y}$) vanish, such that

$$D^3_{x,y} f[:,:,1] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad D^3_{x,y} f[:,:,2] = \begin{bmatrix} 0 & 0 \\ 0 & 6 \end{bmatrix}$$

and

$$\frac{D^3_{x,y} f(1,2)}{3!}\boldsymbol{\delta}^3 = (y-2)^3\,,$$

Taylor Series Expansion of a Function with Two Variables...

which collects all cubic terms of the Taylor series. Overall, the (exact) Taylor series expansion of f at $(x_0, y_0) = (1,2)$ is

$$\begin{aligned} f(x) &= f(1,2) + D^1_{x,y} f(1,2)\boldsymbol{\delta} + \frac{D^2_{x,y} f(1,2)}{2!}\boldsymbol{\delta}^2 + \frac{D^3_{x,y} f(1,2)}{3!}\boldsymbol{\delta}^3 \\ &= f(1,2) + \frac{\partial f(1,2)}{\partial x}(x-1) + \frac{\partial f(1,2)}{\partial y}(y-2) \\ &\quad + \frac{1}{2!}\left(\frac{\partial^2 f(1,2)}{\partial x^2}(x-1)^2 + \frac{\partial^2 f(1,2)}{\partial y^2}(y-2)^2 \right. \\ &\quad \left. + 2\frac{\partial^2 f(1,2)}{\partial x \partial y}(x-1)(y-2)\right) + \frac{1}{6}\frac{\partial^3 f(1,2)}{\partial y^3}(y-2)^3 \\ &= 13 + 6(x-1) + 14(y-2) \\ &\quad + (x-1)^2 + 6(y-2)^2 + 2(x-1)(y-2) + (y-2)^3\,. \end{aligned}$$

Lecture 9

Math Foundations Team

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Introduction

- We will look at continuous optimization concepts in this lecture.
- There are two main branches of continuous optimization - constrained and unconstrained optimization.
- We seek the minimum of an objective function which we assume is differentiable.
- This is like finding the valleys of the objective function, and since the objective function is differentiable, the gradient tells us the direction to move to get the maximum increase in the objective function

Motivation

Consider the data in the given table

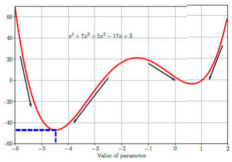
x	y
1	3.1
2	4.9
3	7.3
4	9.1

(1)

Easiest model of y one can think of is $y = ax + b$. Let $\hat{y} = ax + b$ be the y predicted. Our aim is to find a and b such that the difference between actual y and the predicted y is minimum. So the loss function can be defined as $L(a, b) = \sum_{i=1}^{4}(y - (ax + b))^2$ Then the problem will be to find a and b such that $L(a, b)$ is minimized which is an optimization problem.

Unconstrained Optimization

- We move in the direction of the negative gradient to decrease the objective function.
- We move until we encounter a point at which the gradient is zero.



Example

- Let $l(x) = x^4 + 7x^3 + 5x^2 - 17x + 3$.
- The gradient is $\frac{dl(x)}{dx} = 4x^3 + 21x^2 + 10x - 17$.
- Setting the gradient to zero identifies points corresponding to a local minimum or local maximum - there are three such points since this is a cubic equation.
- The second derivative is $12x^2 + 42x + 10$

Optimization using gradient descent

- For low-order polynomials we can solve the equations analytically and find points at which the gradient is zero.
- Consider the problem of solving for the minimum of a real-valued function $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ where $f : R^d \rightarrow R$ is an objective loss function.
- We assume our function f is differentiable but that the minimum cannot be found analytically in closed form.
- The main idea of gradient-descent is to take a step from the current point of magnitude proportional to the negative gradient of the function at the current point.

Optimization using gradient descent

- if $\boldsymbol{x}_1 = \boldsymbol{x}_0 - \alpha((\nabla f)(\boldsymbol{x}_0))^T$ for a small step-size $\alpha > 0$ then $f(\boldsymbol{x}_1) \leq f(\boldsymbol{x}_0)$.
- start at some initial point $\boldsymbol{x}_0$ and then iterate according to $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha_i((\nabla f)(\boldsymbol{x}_i))^T$
- For a suitable step-size α_i, the sequence of points $f(\boldsymbol{x}_0) \geq f(\boldsymbol{x}_1) \geq \ldots$ converges to some local minimum.
- α is also called as the learning-rate

Example

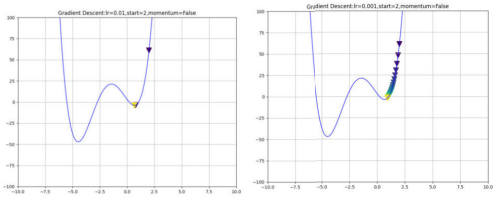


Figure: **left**: with a learning rate of 0.01, local minimum is reached within a couple of steps. **right**: When learning rate is reduced to 0.001, we need relatively more steps to reach the local minimum

Example

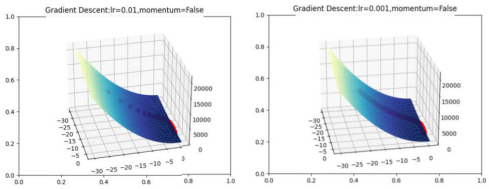


Figure: **left**: with a learning rate of 0.01, minimum is reached. **right**: When learning rate is reduced to 0.001, we need relatively more steps to reach the minimum

Batch gradient descent

- Consider a machine learning problem consisting of loss functions incurred at N data points.
- Let the loss function at the ith data point be $L_i(\theta)$
- Total loss $L(\theta) = \sum_{i=1}^{i=N} L_i(\theta)$.
- Here θ is the parameter vector of interest
- The standard gradient descent procedure is a batch optimization method $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \alpha_i \sum_{i=1}^{i=N} \nabla L_i(\boldsymbol{\theta}_i)^T$.

Mini-batch stochastic gradient

- Let S be a subset of the indices $\{1, 2, \ldots N\}$.
- The set S of data points can be treated as a sample and a sample-centric objective function can be constructed as follows:

$$L_S(\theta) = \sum_{i \in S} L_i(\theta)$$

- The update equation in case of mini-batch stochastic gradient descent can be written as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \alpha_i \sum_{i \in S} \nabla L_i(\boldsymbol{\theta}_i)^T$$

- This approach is referred to as mini-batch stochastic gradient.

Stochastic Gradient Descent

- In the extreme case S can contain only one index chosen at random, and the approach is then called as stochastic gradient descent.
- The key idea in stochastic gradient descent is that the gradient of the sample-specific objective function is an excellent approximation of the true gradient.
- We can show that when the learning rate decreases at a suitable rate and some mild assumptions can be made, stochastic gradient descent almost surely converges to a local minimum.

Example

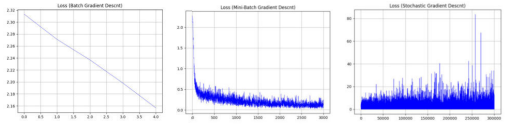
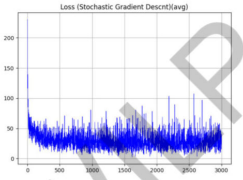


Figure: loss vs num_updates (num epochs:5, dataset:MNIST, layers: lin-relu-lin-relu-lin-relu, loss:crossEntropy, opt:Adam) **left**: Batch Gradient Descent. Entire data is used for every update (thus, 5 epochs results in 5 updates). **right**: Stochastic Gradient Descent. Every update is done based on single sample only. **centre**: Minibatch Gradient Descent. Every update is done usnig a batch of 100 samples.

Example



method	num epochs	accuracy
sgd	10 epoch	.926
sgd	5 epoch	.911
mbsgd	5 epoch	.969
bgd	5 epoch	.514

Figure: **left**: Though the loss update is done for every sample in SGD, this plot shows the loss averaged over 100 such updates. **right**: A summary of measured accuracy for various methods

Learning rate Algorithm 1 : Decay

- How are we to decide the value of the learning rate?
- What happens if we choose a large value for the learning rate and let it be constant? In this case, the algorithm might come close to the optimal answer in the very first iteration but it will then oscillate around the optimal point.
- What happens if we choose a small value for the learning rate and let it be constant? In this case, it will take a very long time for the algorithm to converge to the optimal point.

Learning rate Algorithm 1 : Decay

- Choose a variable learning-rate - large initially but decaying with time.
- This will enable the algorithm to make large strides towards the optimal point and then slowly converge.
- With a learning-rate dependent on time, the update step becomes $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha_t \nabla L$.

Learning rate Algorithm 1 : Decay

- The two most common decay functions are exponential decay and inverse decay, expressed mathematically as follows:

$$\text{exponential decay: } \alpha_t = \alpha_0 e^{-kt}$$
$$\text{inverse decay: } \alpha_t = \frac{\alpha_0}{1+kt}$$

- In both of the above functions k controls the rate of decay.
- Another kind of decay function is the step decay where we reduce the learning rate by a constant factor every few steps of gradient descent.

Learning rate Algorithm 2 : Line search

- Line search uses the optimum step-size directly in order to provide the best improvement.
- It is rarely used in vanilla gradient descent because of its computational expense, but is helpful in some specialized variations of gradient descent.
- Let $L(\theta)$ be the function being optimized, and let $\boldsymbol{d}_t = -\nabla L(\boldsymbol{\theta}_t)$.
- The update step is $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha_t \boldsymbol{d}_t$.
- In line search the learning rate α_t is chosen at the t^{th} step so as to minimize the value of the objective function at $\boldsymbol{\theta}_{t+1}$.
- Therefore the step-size α_t is computed as $\alpha_t = \min_\alpha L(\boldsymbol{\theta}_t + \alpha \boldsymbol{d}_t)$.

Line Search example

Iteration i of gradient descent $f(x_1, x_2) = x_1^2 + 3x_2^2$

1. At iteration i, $\nabla f(x_1, x_2) = [2x_1, 6x_2]$
2. $H(\alpha) = f(\boldsymbol{x} - \alpha((\nabla f)(\boldsymbol{x}))^T) = (1-2\alpha)^2 x_1^2 + 3(1-6\alpha)^2 x_2^2$.
3. $H'(\alpha) = -4(1-2\alpha)x_1^2 - 36(1-6\alpha)x_2^2 = 0$.
4. Step Size : $\alpha = \frac{x_1^2 + 9x_2^2}{2x_1^2 + 54x_2^2}$.

Line search Algorithms

- One question remains - how do we perform the optimization $\min_\alpha L(\boldsymbol{\theta}_t + \alpha \boldsymbol{d}_t)$?
- An important property that we exploit of typical line-search settings is that the objective function is a unimodal function of α.
- This is especially true if we do not use the original objective function but quadratic or convex approximations of it.
- The first step in optimization is to identify a range $[0, \alpha_{\text{max}}]$ in which to perform the search for the optimum α.

Line search Algorithms

- We can sweep evaluate the objective function values at geometrically increasing values of α.
- It is then possible to narrow the search interval by using binary-search, golden-section search method, or the Armijo rule.
- The first two of these methods are exact methods and need for the objective function to be unimodal in α, and the last of the methods is an inexact method that does not rely on unimodality.
- The Armijo rule has broader applicability than either the binary search or golden-section search methods. It will be part of Assignment 2.

Line search Algorithms -Binary search

- Initialize the search interval $[a, b] = [0, \alpha_{\text{max}}]$.
- Evaluate the objective function at $\frac{a+b}{2}$ and $\frac{a+b+\epsilon}{2}$
- Find out whether the function is increasing or decreasing at $\frac{a+b}{2}$. Here ϵ is a small value such as 10^{-6}.
- If the objective function is found to be increasing at $\frac{a+b}{2}$, we narrow the interval to $[a, \frac{a+b+\epsilon}{2}]$ and continue the search.
- Otherwise we narrow the interval to $[\frac{a+b}{2}, b]$ and continue the search.

Line search Algorithms-Golden-section search

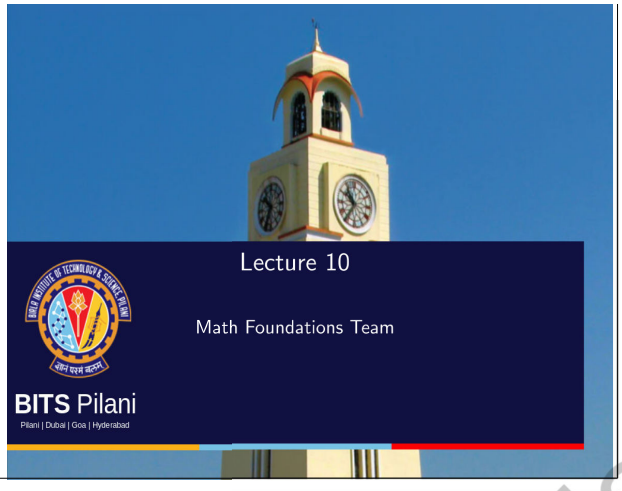
- Initialize the search interval to $[a, b] = [0, \alpha_{\text{max}}]$.
- we use the fact that for any mid-samples m_1, m_2 in the region $[a, b]$ where $a < m_1 < m_2 < b$, at least one of the intervals $[a, m_1]$ or $[m_2, b]$ can be dropped. Sometimes we can go so far as to drop $[a, m_2]$ and $[m_1, b]$.
- When $\alpha = a$ yields the minimum for the objective function, i.e $H(\alpha)$, we can drop the interval $(m_1, b]$.
- Similarly when $\alpha = b$ yields the minimum for $H(\alpha)$ we can drop the interval $[a, m_2)$. When $\alpha = m_1$ is the value at which the minimum is achieved we can drop $(m_2, b]$.
- When $\alpha = m_2$ is the value at which the minimum is achieved we can drop $[a, m_1)$.

Line search Algorithms-Golden-section search

- The new bounds on the search interval $[a, b]$ are reset based on the exclusions mentioned in the previous slide.
- At the end of the process we are left with an interval containing 0 or 1 evaluated point.
- If we have an interval containing no evaluated point, we select a random point $\alpha = p$ in the reset interval $[a, b]$, and then another point q in the larger of the intervals $[a, p]$ and $[p, b]$.
- On the other hand if we are left with an interval $[a, b]$ containing a single evaluated point $\alpha = p$, then we select $\alpha = q$ in the larger of the intervals $[a, p]$ and $[p, b]$.
- This yields another four points on which to continue the golden-section search. We continue until we achieve the desired accuracy.

When do we use line search?

- The line-search method can be shown to converge to a local optimum, but it is computationally expensive. For this reason, it is rarely used in vanilla gradient descent.
- Some methods like Newton's method, however, require exact line search.
- Fast inexact methods like Armijo's rule are used in vanilla gradient descent.
- One advantage of using exact line search is that fewer steps are needed to achieve convergence to a local optimum. This might more than compensate for the computational expense of individual steps.



Introduction

- We will look at nonlinear optimization concepts in this lecture.
- We already know how to compute gradient, but there are some minutiae of gradient descent that we need to address.
- Machine learning algorithms depend heavily on the correctness of the gradient since if the gradient is computed erroneously, the algorithms might fail to find the local or global optimum.
- We will also look into some challenges in non-linear optimization.

Optimization in machine learning

- A common loss function is $L(\boldsymbol{\theta}) = \sum_{i=1}^{i=n} \|\boldsymbol{\theta}^T \boldsymbol{x}_i - y_i\|^2$.
- Here $\boldsymbol{x}_i^T$ is the ith row vector in a matrix $\mathbf{X}$ consisting of n row vectors where each row vector represents a training point
- y_i contains the real-valued observation of the i^{th} training point.
- The above loss function occurs in least squared regression
- It represents the sum of squared differences between the observed values y_i in the data and the predicted values $\hat{y}_i = \boldsymbol{\theta}^T \boldsymbol{x}_i$.

Optimization of Additionally separable sum

- We can write the total objective function as

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{n} L_i(\boldsymbol{\theta})$$

- This type of linear separability is useful since it enables the use of techniques like stochastic gradient descent and mini-batch stochastic gradient descent.
- The idea here is that we can replace the gradient of the entire objective function with a sampled approximation.

Overfitting in machine learning

- In traditional optimization, we focus on updating parameters so that the objective function is minimized as much as possible.
- In machine learning, minimization of the objective function is performed over training data but the model is applied on test data which is unseen.
- We need to avoid the problem of overfitting the training data.

Overfitting in machine learning

- Suppose we have 4-dimensional data and one dependent variable, i,e output that is a function of the four inputs.
- Let the input parameters be x_1, x_2, x_3, x_4 and the output be y.
- We seek to learn parameters w_1, w_2, w_3, w_4, w_5 such that our prediction expression $\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5$ gives a good prediction.
- We would like to minimize the squared error $\sum_{i=1}^{n} \|y - \hat{y}\|_2^2$

Example of overfitting

- Consider the following data on 4 variables x_1, x_2, x_3, x_4 and associated output variable y.
- Let us say that this is a sample of real-life data where the output $y \approx x_1$.

x_1	x_2	x_3	x_4	y
61	2	3	0.1	49
40	0	4	0.5	40
68	0	10	1.0	70

Example of overfitting

- Minimizing squared error, we notice that one good solution is $w_1 = 1, w_2 = w_3 = w_4 = 0, w_5 = 0$.
- This solution does not give zero squared error with respect to the actual observations but gives an error close to zero.
- The solution $w_1 = 1, w_2 = w_3 = w_4 = 0, w_5 = 0$ is a very good one since it captures the real-life relationship between the output variable y and x_1.

Example of overfitting

- Consider $w_1 = 0, w_2 = 7, w_3 = 5, w_4 = 0, w_5 = 20$. This solution gives zero training error.
- It is a very poor solution since there is no dependence of the output variable y on x_1 while we know that there is actually a strong dependence between y and x_1.
- Therefore it will incur a high error on test-data.
- This example illustrates the idea that minimizing the loss function to the greatest extent may not be a good thing since the model may then perform poorly on real-life data.

Feature processing

- The loss function can have vastly different sensitivities to different model parameters and this can have an impact in the learning process.
- Consider a model where a person's wealth y is modeled in terms of his age x_1 and number of years of college education x_2.The formula for wealth y is $y = w_1x_1^2 + w_2x_2^2$.
- Age $[0, 100]$, number of years in college education $[0, 10]$.
- We have $\frac{\partial y}{\partial w_1} = x_1^2$ and $\frac{\partial y}{\partial w_2} = x_2^2$.
- Since x_1 and x_2 are generally very different in magnitude, we take small steps in respect of w_2 and large steps in respect of w_1.

Feature processing

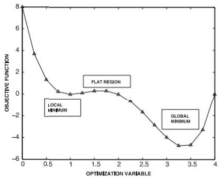
- Taking small steps in w_2 and large steps in w_1 will make us go steadily towards the optimal value for w_2 but oscillate with respect to the optimal value of w_1, overshooting the target each time.
- This makes convergence very slow.
- It is therefore helpful to have features with similar variance.

Feature processing

- Two techniques used for achieving similar variance are mean-centering and feature normalization.
- In case of mean-centering a vector of column-wise means is subtracted from each data point.
- In case of feature normalization, each feature value is divided by its standard deviation.
- In case of min-max normalization we scale the jth feature of the ith data point as follows: $x_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j}$.

Challenges in Gradient-Based Optimization

1. Flat regions and local optima
2. Different levels of curvature in different directions



(a) Local optima with flat regions

Local optima and flat regions

Consider a function defined as follows

- $F(x) = (x-1)^2((x-3)^2 - 1)$.
- $F'(x) = 2(x-1)((x-1)(x-3) + (x-3)^2 - 1) = 0$.
- The solutions to this equation are $x = 1, x = \frac{5-\sqrt{3}}{2} = 1.634, x = \frac{5+\sqrt{3}}{2} = 3.366$.

Local optima and flat regions

- We can show that the first and third roots are minima since $F''(x) > 0$ at these points while the second point is a maximum since $F''(x) < 0$.
- $F(1) = 0$, $F(\frac{5-\sqrt{3}}{2}) = 0.348$, $F(\frac{5+\sqrt{3}}{2}) = -4.848$.
- If we start gradient descent from any point less than 1.634, we will arrive only at a local minimum.
- We might never arrive at a global minimum if we keep choosing wrong starting points.
- The problem becomes worse with high-dimensionality.

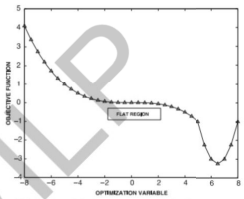
Local optima and flat regions

- Consider $F(x_1, x_2, \ldots x_d) = A_1(x_1) + A_2(x_2) + \ldots + A_n(x_n)$.
- Let $A_i(x_i)$ have k_i local minima.
- Setting $\frac{\partial F}{\partial x_i} = 0 \ \forall i$, we note that any point $(x_1^*, x_2^*, \ldots, x_d^*)$, where x_i^* is a local minima of the function $A_i(x_i)$, is a solution to $\frac{\partial F}{\partial x_i} = 0$.

Local optima and flat regions

- This is because $\frac{\partial F}{\partial x_i}|_{(x_1^*, x_2^*, \ldots x_d^*)} = A_i'(x_i^*) = 0$ since x_i^* is a local minimum of $A_i(x_i)$.
- There are $\prod_{i=1}^{i=d} k_i$ local minima for the function $F(x_1, x_2, \ldots, x_d)$, which is very large number of points.
- Gradient descent could be stuck at any one of these points which might be far from the global optimum.

Local optima and flat regions



(b) Only global optimum with flat region

- Another problem to contend with is the presence of flat regions where the gradient is close to zero.

Local optima and flat regions

- Flat regions are problematic because the speed of descent depends on the magnitude of the gradient, given a fixed learning rate.
- The optimization process will take a long time to cross a flat region of space which will make convergence slow.

Different levels of curvature

- In multi-dimensional settings, the components of the gradient with respect to different parameters can vary widely. This will cause convergence problems since there is oscillation in the update step with respect to some components and a steady movement with respect to other components.
- Consider the simplest possible case of a bowl-like convex, quadratic objective function with a single global minimum - $L = x^2 + y^2$ represents a perfectly circular bowl, and the function $L = x^2 + 4y^2$.
- We shall show contour plots of both functions and how gradient descent performs on finding the minimum of the two functions.

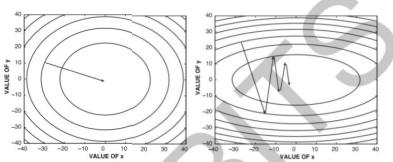
Different levels of curvature

- What is the qualitative difference between $L = x^2 + y^2$ and $L = x^2 + 4y^2$. Intuitively one looks symmetric in x and y, and the other is not.
- The second loss function is more sensitive to changes in y as compared to x - it looks like an elliptical bowl. The specific sensitivity depends on the position of x, y.
- Looking at the second-order derivatives we can see that for the second function $\frac{\partial^2 L}{\partial^2 x^2}$, and $\frac{\partial^2 L}{\partial^2 y^2}$ are very different.

Different levels of curvature

- The second-order derivative measures the rate of change of the gradient - a high second-order derivative means high curvature.
- From the point of view of gradient descent we want moderate curvature in all dimensions as it would means that the gradient does not change too much in some dimensions compared to others.
- We can then make gradient-descent steps of large sizes.

Different levels of curvature



(a) Loss function is circular bowl $L = x^2 + y^2$

(b) Loss function is elliptical bowl $L = x^2 + 4y^2$

Figure 5.2: The effect of the shape of the loss function on steepest-gradient descent

Different levels of curvature

- In case of the perfect bowl, a sufficiently large step-size from any point can take us directly to the optimum of the function in one-step, since the gradient at any point points towards the optimum of the function.
- This is not true for the elliptical bowl, the gradient at any point does not point to the optimum of the function.

Contour plots

- Note that the gradient at any point is orthogonal to the contour line at that point.
- This because the dot product of the gradient ∇F and a small displacement $\delta \boldsymbol{x}$ along the contour line gives the change in the value of the function along the displacement $\boldsymbol{x}$.
- Since the function remains constant along the contour line, $\nabla F.\boldsymbol{x} = 0$

Differential curvature

- A closer look at the contour plot for the elliptical bowl case shows that in the y-direction, we see oscillatory movement as in each step we correct the mistake of overshooting made in the previous step. The gradient component along the y-direction is more than the component along the x-direction.
- Along the x-direction, we make small movements towards the optimum x-value. Overall, after many training steps we find that we have made little progress to the optimum.
- It needs to be kept in mind that the path of steepest descent in most objective functions is only an instantaneous direction of best improvement, and is not the correct direction of descent in the longer term.

Revisiting feature normalization

- We show how to address in some measure the differential curvature problem by feature normalization.
- Consider the following toy dataset, where the two input attributes are x_1 and x_2 , and the output attribute is y.
- We intend to find a relationship of the form $y = w_1 x_1 + w_2 x_2$ from the data. The coefficients w_1 and w_2 are found using gradient descent on the loss function computed from the data.

x_1	x_2	y
0.1	25	7
0.8	10	1
0.4	10	4

Revisiting feature normalization

- Loss function:

$$J(\boldsymbol{w}) = (0.1w_1+25w_2-7)^2+(0.8w_1+10w_2-1)^2+(0.4w_1+10w_2-4)^2$$

- Objective function is much more sensitive to w_2 than w_1
- One way to get around this issue is to standardize each column to zero mean and unit variance
- The coefficients for w_1 and w_2 will become much more similar, and differential curvature will be reduced.

Constrained Optimization

- Optimization problems are of 2 types
  1. Unconstrained Optimization
  2. Constrained Optimization
- We discussed algorithms to solve unconstrained optimization
- How do we find the solution to an optimization problem with constraints?
- Example of a constrained optimization problem

$$\begin{aligned} &\textit{maximize} && f(x,y) = x^2 y \\ &\text{subject to} && g(x,y) : x^2 + y^2 = 1 \end{aligned}$$

Constrained Optimization : Lagrange Multiplier Method

- Constrained maximization (minimization) problem is rewritten as a Lagrange function whose optimal point is a saddle point, i.e. a global maximum (minimum)
- Lagrange function use Lagrange multipliers as a strategy for finding the local maxima and minima of a function subject to constraints
- Maximum of $f(x,y)$ under constraint $g(x,y)$ is obtained when their gradients point to same direction
- Introduce a Lagrange multiplier λ for the equality constraint Mathematically, $\nabla f(x,y) = \lambda \nabla g(x,y)$

Constrained Optimization : Lagrange Multiplier Method

Consider the following optimization problem

$$\begin{aligned} &\text{maximize } xy \\ &\text{subject to } x + y = 6 \end{aligned}$$

- The Lagrangian is $L(x,y) = xy - \lambda(x+y-6)$

$$\left.\begin{aligned} \frac{\partial L(x,y)}{\partial \lambda} &= x + y - 6 = 0 \\ \frac{\partial L(x,y)}{\partial x} &= y - \lambda = 0 \\ \frac{\partial L(x,y)}{\partial y} &= x - \lambda = 0 \end{aligned}\right\} \Rightarrow x = y = \lambda$$
$$\Rightarrow x = y = 3$$

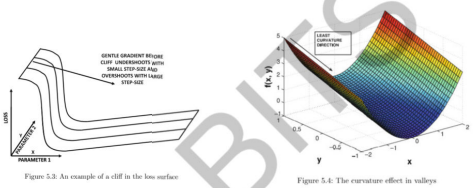
- x and y values remain same even if you take $+\lambda$ or $-\lambda$ for equality constraint

Introduction

- We will take a deeper look at some challenges in non-linear optimization, continuing on from the previous lecture.
- First we will look at the problem of different levels of curvature in different directions.
- We will need to figure out strategies to deal with the above situations to design a good optimization algorithm.
- Second we will look into how to solve constrained optimization problem.

Examples of high curvature surfaces

- Two examples of high curvature surfaces are cliffs and valleys.



(a) (b)

Difficult topologies

- In Figure (a), the partial derivative with respect to x changes drastically as we go along the axis of x.
- A modest learning rate will cause minimal reduction in the value of the objective function in the gently sloping regions.
- The same modest learning rate in the high-sloping regions will cause us to overshoot the optimal value in those regions.
- In Figure (b), there is gentle slope along the y-direction and a U-shape in the x-direction.
- The gradient descent method will bounce violently along the steep sides of the valley while not making much progress along the x-axis.

Gradient descent with momentum

- Gradient descent may be slow if the curvature of the function is such that the gradient descent steps hop between the walls of the valley of contours and approaches the optimum slowly.
- If we endow the optimization procedure with memory, we can improve convergence.
- We use an additional term in the step-update to remember what happened in the previous iteration, so that we can dampen oscillations and speed up convergence.
- This is a momentum term - the name momentum comes from a comparison to a rolling ball whose direction becomes more and more difficult to change as its velocity increases.

Gradient descent with momentum

The gradient descent with momentum has the following iteration

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \alpha_i((\nabla f)(\boldsymbol{x}_i))^T + v_i$$

- $v_i = \beta(\boldsymbol{x}_i - \boldsymbol{x}_{i-1})$ and $v_0 = 0$
- $\beta \in [0, 1]$ is referred to as the momentum parameter or the friction parameter.
- Momentum-based methods attack the issues of flat-regions, cliffs and valleys by emphasizing medium-term to long-term directions of consistent movement.

Momentum-based learning

- An aggregated measure of feedback is used to reinforce movement along certain directions and speed up gradient descent.
- Momentum-based learning accelerates gradient descent since the algorithm moves quicker in the direction of the optimal solution.
- The useless sideways oscillations as they get cancelled out during the averaging process.
- The momentum term is useful where only approximate gradient is known as the momentum term averages out the noisy estimates of the gradient.

Momentum-based learning

- The concept of momentum can be illustrated by a marble rolling down a hill that has a number of "local" distortions like potholes, ditches etc. The momentum of the marble causes it to navigate local distortions and emerge out of them.

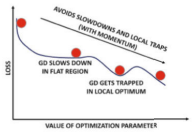


Figure 5.5: Effect of momentum in navigating complex loss surfaces. The annotation "GD" indicates pure gradient descent without momentum. Momentum helps the optimization process retain speed in flat regions of the loss surface and avoid local optima

Momentum-based learning

Momentum increases the relative component of the gradient in the correct direction

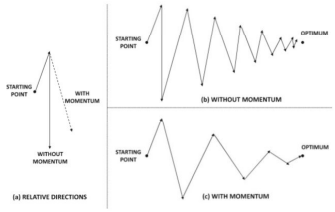


Figure 5.6: Effect of momentum in smoothing zigzag updates

AdaGrad

- AdaGrad algorithm keep track of the aggregated squared magnitude of the partial derivative with respect to each parameter.
- At each iteration of Adagrad

$$A_i \leftarrow A_i + (\frac{\partial J}{\partial w_i})^2$$
$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}}\frac{\partial J}{\partial w_i}, \quad \forall i$$

- A_i measures only the historical magnitude of the gradient rather than the sign. To avoid ill-conditioning, $\epsilon = 10^{-8}$ can be added to A_i.

AdaGrad

- If the gradient takes values +100 and -100 alternatively, A_i will be large and the update step along the parameter in question will be small.
- If the gradient takes a value 0.1 consistently, A_i will not be as large as before and the update step will be comparatively larger.
- With the passage of time absolute movements along all components will slow down because A_i is monotonically increasing with time.
- AdaGrad suffers from the problem of not making much progress after a while. The fact that A_i is aggregated over the entire history of partial derivatives may make the method stale.

RMSProp

- At each iteration of RMSProp

$$A_i \leftarrow \rho A_i + (1-\rho)(\frac{\partial J}{\partial w_i})^2$$
$$w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}}\frac{\partial J}{\partial w_i}, \quad \forall i$$

- It uses exponential averaging. scaling factor A_i does not constantly increase. Note that $\rho \in (0, 1)$.
- In RMSProp the importance of ancient gradients decays exponentially with time as the gradient from t steps before is weighted by ρ^t.

Adam Algorithm

- At each iteration of Adam Algorithm

$$A_i \leftarrow \rho A_i + (1-\rho)(\frac{\partial J}{\partial w_i})^2$$
$$F_i \leftarrow \rho_f F_i + (1-\rho_f)\frac{\partial J}{\partial w_i}$$
$$w_i \leftarrow w_i - \alpha_t \frac{F_i}{\sqrt{A_i}}, \quad \forall i$$

- $\alpha_t = \alpha \frac{\sqrt{1-\rho^t}}{1-\rho_f^t}$,
- $\rho \in (0,1)$ and ρ_f is called decay parameter.

Adam

- Adam incorporates momentum into the update.
- Adam addresses the initialization bias present in RMSProp. Both F_i and A_i are initialized to zero which causes bias in early iterations.
- The two quantities are affected differently which accounts for the equation for α_t.
- As $t \to \infty$, $\rho^t \to 0$, $\rho_f^t \to 0$ and $\alpha_t \to \alpha$ since $\rho, \rho_f \in (0,1)$.
- The default suggested value for ρ_f and ρ are 0.9 and 0.999 respectively.

Constrained optimization and Lagrange multipliers

- Consider the following problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x}), f : \mathbb{R}^D \to \mathbb{R}$, subject to additional constraints - so we are looking at a minimization problem except that the set of all $\boldsymbol{x}$ over which minimization is performed is not all of $\mathbb{R}^D$.
- The constrained problem becomes
  $\min_{\boldsymbol{x}} f(\boldsymbol{x})$
  subject to $g_i(\boldsymbol{x}) \leq 0 \; \forall i 1, 2, \ldots m$.
- If we convert constrained optimization problem into an unconstrained one, we can use techniques we studied
- We construct $J(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \mathbf{1}(g_i(\boldsymbol{x}))$, where $\mathbf{1}(z) = 0$ for $z \leq 0$ and $\mathbf{1}(z) = \infty$ for $z > 0$.

Constrained optimization and Lagrange multipliers

- The formulation of $J(\boldsymbol{x})$ in the previous slide ensures that the optimal solution to the unconstrained problem is the same as the constrained problem.
- The step-function is also difficult to optimize. Hence, we replace the step-function by a linear function using Lagrange multipliers.
- We create the Lagrangian of the given constrained optimization problem as follows:
  $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \lambda_i g_i(\boldsymbol{x}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{g}(\boldsymbol{x})$, where $\lambda_i \geq 0$ for all i.

Primal and dual problems

- The primal problem is $\min f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0, 1 \leq i \leq m$. Optimization is performed over the primal variables $\boldsymbol{x}$.
- The associated Lagrangian dual problem is $\max_{\boldsymbol{\lambda} \in \mathbb{R}^m} \mathfrak{D}(\boldsymbol{\lambda})$ subject to $\boldsymbol{\lambda} \geq 0$ where $\boldsymbol{\lambda}$ are dual variables.
- $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.
- The following minimax inequality holds over two arguments $\boldsymbol{x}, \boldsymbol{y}$: $\max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \phi(\boldsymbol{x}, \boldsymbol{y}) \leq \min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \phi(\boldsymbol{x}, \boldsymbol{y})$.

Minimax inequality

- Why is this inequality true?
- Assume that $\boldsymbol{x}, \boldsymbol{y}$: $\max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}_A, \boldsymbol{y}_A)$ and $\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} \phi(\boldsymbol{x}, \boldsymbol{y}) = \phi(\boldsymbol{x}_B, \boldsymbol{y}_B)$.
- Fixing $\boldsymbol{y}$ at $\boldsymbol{y}_A$ we see that the inner operation on the left hand side of the minimax inequality is a min operation over $\boldsymbol{x}$ and returns $\boldsymbol{x}_A$. Thus we have $\phi(\boldsymbol{x}_A, \boldsymbol{y}_A) \leq \phi(\boldsymbol{x}_B, \boldsymbol{y}_A)$.
- Fixing $\boldsymbol{x}$ at $\boldsymbol{x}_B$ we see that the inner operation on the right hand side of the minimax inequality is a max operation over $\boldsymbol{y}$ and returns $\boldsymbol{y}_B$. Thus we have $\phi(\boldsymbol{x}_B, \boldsymbol{y}_B) \geq \phi(\boldsymbol{x}_B, \boldsymbol{y}_A)$.
- From the above we conclude that $\phi(\boldsymbol{x}_B, \boldsymbol{y}_B) \geq \phi(\boldsymbol{x}_A, \boldsymbol{y}_A)$.

Minimax inequality

- The difference between $J(\boldsymbol{x})$ and the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is that the indicator function is relaxed to a linear function.
- When $\boldsymbol{\lambda} \geq 0$, the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a lower bound on $J(\boldsymbol{x})$.
- The maximum of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ is $J(\boldsymbol{x})$ - if the point $\boldsymbol{x}$ satisfies all the constraints $g_i(\boldsymbol{x}) \leq 0$, then the maximum of the Lagrangian is obtained at $\boldsymbol{\lambda} = 0$ and it is equal to $J(\boldsymbol{x})$.
- If one or more constraints is violated such that $g_i(\boldsymbol{x}) > 0$, then the associated Lagrangian coefficient λ_i can be taken to be ∞ so as to equal $J(\boldsymbol{x})$.

Minimax inequality

- From the previous slide, we have $J(\boldsymbol{x}) = \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.
- Our original constrained optimization problem boiled down to minimizing $J(\boldsymbol{x})$, in other words we are looking at $\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$
- Using the minimax inequality we see that $\min_{\boldsymbol{x} \in \mathbb{R}^d} \max_{\boldsymbol{\lambda} \geq 0} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \geq \max_{\boldsymbol{\lambda} \geq 0} \min_{\boldsymbol{x} \subset \mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$.
- This is known as weak duality. The inner part of the right hand side of the inequality is $\mathfrak{D}(\boldsymbol{\lambda})$.

Lagrangian formulation

- In contrast to the original formulation $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is an unconstrained optimization problem for a given value of $\boldsymbol{\lambda}$.
- We observe that $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is a point-wise minimum of affine functions and hence $\mathfrak{D}(\boldsymbol{\lambda})$ is concave even though $f()$ and $g()$ may be nonconvex.
- We have obtained a Lagrangian formulation for a constrained optimization problem where the constraints are inequalities. What happens when some constraints are equalities?

Modeling equality constraints

- Suppose the problem is $\min_{\boldsymbol{x}} f(\boldsymbol{x})$ subject to $g_i(\boldsymbol{x}) \leq 0$ for all $1 \leq i \leq m$ and $h_j(\boldsymbol{x}) = 0$ for $1 \leq j \leq n$.
- We model the equality constraint $h_j(\boldsymbol{x}) = 0$ with two inequality constraints $h_j(\boldsymbol{x}) \geq 0$ and $h_j(\boldsymbol{x}) \leq 0$.
- The resulting Lagrange multipliers are then unconstrained.
- The Lagrange multipliers for the original inequality constraints are non-negative while those corresponding to the equality constraints are unconstrained.

Convex optimization

- We are interested in a class of optimization problems where we can guarantee global optimality.
- When $f()$, the objective function, is a convex function and $g()$ and $h()$ are convex functions, we have a convex optimization problem.
- In this setting we have strong duality - the optimal solution of the primal problem is equal to the optimal solution of the dual problem.
- What is a convex function?

Convex function

- First we need to know what is a convex set. A set C is a convex set if for any $x, y \in C$, $\theta x + (1-\theta)y \in C$.
- For any two points lying in the convex set, a line joining them lies entirely in the convex set.
- Let a function $f : \mathbb{R}^d \rightarrow R$ be a function whose domain is a convex set C.
- The function is a convex function if for any $\boldsymbol{x}, \boldsymbol{y} \in C$, $f(\theta\boldsymbol{x} + (1-\theta)\boldsymbol{y}) \leq \theta f(\boldsymbol{x}) + (1-\theta)f(\boldsymbol{y})$
- Another way of looking at a convex function is to use the gradient: for any two points $\boldsymbol{x}$ and $\boldsymbol{y}$, we have $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla_{\boldsymbol{x}} f(\boldsymbol{x})(\boldsymbol{y} - \boldsymbol{x})$.

Example

- The negative entropy, a useful function in Machine Learning, is convex: $f(x) = xlog_2 x$ for $x > 0$.
- First let us check if $f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$. Take $x = 2$, $y = 4$, and $\theta = 0.5$ to get $f(0.5 * 2 + 0.5 * 4) = f(3) = 3log_2 3 \approx 4.75$. Then $\theta f(2) + (1-\theta)f(4) = 0.5 * 2log_2 2 + 0.5 * 4log_2 4 = log_2 32 = 5$. Therefore the convexity criterion is satisfied for these two points.
- Let us now use the gradient criterion. We have $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = log_2 x + x\frac{1}{xlog_e 2}$. $f(2) + \nabla f(2) * (4-2) = 2log_2 2 + (log_2 2 + \frac{1}{log_e 2} * 2 \approx 6.9$. We see that $f(4) = 4log_2 4 = 8$ which shows that the gradient criterion is also satisfied.

Linear programming

- Let us look at a convex optimization problem where the objective function and constraints are all linear.
- Such a convex optimization problem is called a linear programming problem.
- We can express a linear programming problem as $\min_{\boldsymbol{x}} \boldsymbol{c}^T\boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$ where $\boldsymbol{A} \in \mathbb{R}^{m\times d}$ and $b \in \mathbb{R}^{m\times 1}$.
- The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = c^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$ where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of non-negative Lagrangian multipliers.
- We can rewrite the Lagrangian as $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{b}$.

Linear programming

- Taking the derivative of the Lagrangian with respect to $\boldsymbol{x}$ and setting it to zero we get $\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda} = 0$.
- Since $\mathfrak{D}(\lambda) = \min_{\boldsymbol{x}\in\mathbb{R}^d} \mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$, plugging in the above equation gives $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^T\boldsymbol{b}$.
- We would like to maximize $\mathfrak{D}(\boldsymbol{\lambda})$, subject to the constraint $\boldsymbol{\lambda} \geq 0$.
- Thus we end up with the following problem:

$$\begin{aligned} max_{\boldsymbol{\lambda}\in\mathbb{R}^m} & -\boldsymbol{\lambda}^T\boldsymbol{b} \\ \text{subject to } & \boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda} = 0 \\ & \boldsymbol{\lambda} \geq 0 \end{aligned}$$

Linear programming

- We can solve the original primal linear program or the dual one - the optimum in each case is the same.
- The primal linear program is in d variables but the dual is in m variables, where m is the number of constraints in the original primal program.
- We choose to solve the primal or dual based on which of m or d is smaller.

Quadratic programming

- We now consider the case of a quadratic objective function subject to affine constraints:

$$min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} \text{ subject to}$$
$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

- Here $\boldsymbol{A} \in \mathbb{R}^{m\times d}, b \in \mathbb{R}^m, c \in \mathbb{R}^d$

Quadratic programming

- The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$.
- Rearranging the above we have $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{b}$
- Taking the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ and setting it equal to zero gives $\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) = 0$.
- If we take $\boldsymbol{Q}$ to be invertible, we have $\boldsymbol{x} = -\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})$.
- Plugging this value of $\boldsymbol{x}$ into $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ gives us $\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$.
- This gives us the dual optimization problem: $max_{\boldsymbol{\lambda}\in\mathbb{R}^m} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$ subject to $\boldsymbol{\lambda} \geq \boldsymbol{0}$.

Lecture 12

Math Foundations Team

BITS Pilani

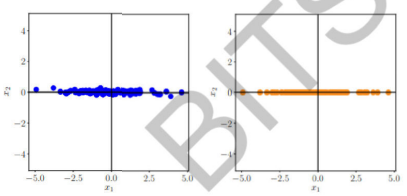
Pilani | Dubai | Goa | Hyderabad

Introduction

- We will look at principle components analysis and dimension reduction in this lecture.
- High-dimensional data is hard to visulaize and interpret, can we project this data into lower dimensions while preserving the semantics of the data so as to draw the same conclusions as if we interpreted the higher dimensional data?
- Higher dimensional data is often overcomplete, in that there are redundant dimensions which can be explained by a combination of other dimensions.
- Dimensions in higher-dimensional data might be correlated, so the actual data may have an intrinsic lower-dimensional structure

Principle components analysis

- PCA is a technique for linear dimensionality reduction. It was first proposed by Pearson in 1900 and was independently rediscovered by Hotelling in 1933.



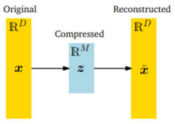
(a) Dataset with x_1 and x_2 coordinates. (b) Compressed dataset where only the x_1 coordinate is relevant.

Problem setting

- **Given :** Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}, \boldsymbol{x}_n \in \mathbb{R}^D$ an independent, identically distributed dataset, with mean $\boldsymbol{0}$.
- Thus, the data covariance matrix $\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{n=N} \boldsymbol{x_n}\boldsymbol{x_n}^T$.
- **Aim :** To find projections $\tilde{\boldsymbol{x}_n} \in U \subseteq \mathbb{R}^D$ of datapoints $\boldsymbol{x}_n \in \mathbb{R}^D$ which are as similar as possible to the original datapoints but $\dim(U) = M < D$.
- We are looking for a lower-dimensional compressed representation $\boldsymbol{z}_n$ of $\boldsymbol{x}_n$ such that $\boldsymbol{z}_n = \boldsymbol{B}^T\boldsymbol{x}_n$ where the projection matrix $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m] \in \mathbb{R}^{D\times M}$.
- The columns of B are orthonormal which means $\boldsymbol{b}_i^T\boldsymbol{b}_j = 0$ when $i \neq j$ and $\boldsymbol{b}_i^T\boldsymbol{b}_i = 1$.

Problem setting

- The figure below shows how $\boldsymbol{z}$ represents the lower-dimensional representation of the compressed data $\tilde{\boldsymbol{x}}$ and plays the role of a bottleneck which controls the information flow between $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$.



- There exists a linear relationship between the original data $\boldsymbol{x}$, its low-dimensional code $\boldsymbol{z}$ and the compressed data $\tilde{\boldsymbol{x}}$: $\boldsymbol{z} = \boldsymbol{B}^T\boldsymbol{x}$, and $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z}$ for a suitable matrix $\boldsymbol{B}$.

Maximum variance perspective

- We can interpret information content in the data as how "space-filling" it is and describe the information contained in the data by looking at the spread of the data.
- We can capture spread of the data using the concept of variance.
- PCA can then be viewed as a dimensionality reduction algorithm that maximizes the variance in the low-dimensional representation of the data to retain as much information as possible.
- Mathematically our aim is to find a matrix $\boldsymbol{B}$ so that we can retain as much information as possible by projecting the data on the columns $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_M$ of the matrix.

Centred data

- If $\boldsymbol{\mu}$ is the mean of the data. Centred data means that we work with data columns $\boldsymbol{x} - \boldsymbol{\mu}$, rather than the original columns $\boldsymbol{x}$.
- Note that $\mathbb{V}_{\boldsymbol{z}}(\boldsymbol{z}) = \mathbb{V}_{\boldsymbol{x}}(\boldsymbol{B}^T(\boldsymbol{x} - \boldsymbol{\mu})) = \mathbb{V}_{\boldsymbol{x}}(\boldsymbol{B}^T\boldsymbol{x} - \boldsymbol{B}^T\boldsymbol{\mu}) = \mathbb{V}_{\boldsymbol{x}}(\boldsymbol{B}^T\boldsymbol{x})$.
- Thus by considering $\boldsymbol{x} - \boldsymbol{\mu}$, the variance does not change. Therefore we assume that the data has a mean of $\boldsymbol{0}$ for this lecture.
- Letting the mean be $\mathbb{E}_{\boldsymbol{x}}(\boldsymbol{x}) = \boldsymbol{0}$ means $\mathbb{E}_{\boldsymbol{z}}(\boldsymbol{z}) = \mathbb{E}_{\boldsymbol{x}}(\boldsymbol{B}^T\boldsymbol{x}) = \boldsymbol{B}^T\mathbb{E}_{\boldsymbol{x}}(\boldsymbol{x}) = \boldsymbol{0}$
- And the data covariance matrix $\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{n=N} \boldsymbol{x_n}\boldsymbol{x_n}^T$.

Direction with maximal variance

- We maximize the variance of the low-dimensional code by following a sequential approach.
- **Aim 1 :** To maximize the variance V_1 of the first coordinate z_{1n} of $\boldsymbol{z} \in R^M$
- i.e to maximize $V_1 = \mathbb{V}(z_1) = \frac{1}{N}\sum_{n=1}^{n=N} z_{1n}^2$ since the data $\boldsymbol{x}$ is independent.
- Now $z_{1n} = \boldsymbol{b}_1^T\boldsymbol{x}_n$, and can be viewed as the orthogonal projection of $\boldsymbol{x}_n$ onto the one-dimensional subspace spanned by $\boldsymbol{b}_1$.

Direction with maximal variance

$$
\begin{aligned}
\text{Then we have } V_1 &= \frac{1}{N}\sum_{n=1}^{n=N}(\boldsymbol{b}_1^T\boldsymbol{x}_n)^2 \\
&= \frac{1}{N}\sum_{n=1}^{n=N}\boldsymbol{b}_1^T\boldsymbol{x}_n\boldsymbol{x}_n^T\boldsymbol{b}_1 \\
&= \boldsymbol{b}_1^T\Big(\sum_{n=1}^{n=N}\boldsymbol{x}_n\boldsymbol{x}_n^T\Big)\boldsymbol{b}_1 \\
&= \boldsymbol{b}_1^T\boldsymbol{S}\boldsymbol{b}_1.
\end{aligned}
$$

Arbitrarily increasing the magnitude of the vector $\boldsymbol{b}_1$ wil increase the variance - so we seek to maximize the variance subject to $\|\boldsymbol{b}_1\| = 1$.

Direction with maximal variance

- Therefore to find the direction $\boldsymbol{b}_1$ that maximizes variance can be set up as a constrained optimization problem

$$\max \boldsymbol{b}_1^T\boldsymbol{S}\boldsymbol{b}_1 \text{ subject to } \|\boldsymbol{b}_1\| = 1$$

- To solve this problem we set up the Lagrangian $\mathfrak{L}(\boldsymbol{x}, \lambda) = \boldsymbol{b}_1^T\boldsymbol{S}\boldsymbol{b}_1 + \lambda(1 - \boldsymbol{b}_1^T\boldsymbol{b}_1)$.

Solving the Lagrangian

- To solve the Lagrangian, let $\frac{\partial \mathbb{L}}{\partial \lambda} = 0$ and $\frac{\partial \mathbb{L}}{\partial \boldsymbol{b}_1} = \boldsymbol{0}$.
- So, we get

$$
\begin{aligned}
\frac{\partial \mathbb{L}}{\partial \lambda} &= 1 - \boldsymbol{b}_1^T\boldsymbol{b}_1 = 0 \\
\frac{\partial \mathbb{L}}{\partial \boldsymbol{b}_1} &= 2\boldsymbol{b}_1^T\boldsymbol{S} - 2\lambda\boldsymbol{b}_1^T = 0
\end{aligned}
$$

- On simplification, we have $\boldsymbol{S}\boldsymbol{b}_1 = \lambda\boldsymbol{b}_1$ and $\boldsymbol{b}_1^T\boldsymbol{b} = 1$.
- Thus we find that the direction $\boldsymbol{b}_1$ we seek is an eigenvector of the covariance matrix $\boldsymbol{S}$ and λ is its corresponding eigenvalue.

First principal component

- Putting the result of the previous slide into the objective function of the constrained optimization problem ie. $\max \boldsymbol{b}_1^T\boldsymbol{S}\boldsymbol{b}_1$ we have $\boldsymbol{b}_1^T\boldsymbol{S}\boldsymbol{b}_1 = \boldsymbol{b}_1^T\lambda\boldsymbol{b}_1 = \lambda$.
- Our objective function boils to maximizing λ which means we are looking for the eigenvector of $\boldsymbol{S}$ that corresponds to its largest eigenvalue.
- This is the first principal component.
- Let us now examine the inner workings of the Lagrangian method.

M-dimensional subspace with maximum variance

- Assume that we have found the first $m-1$ principal components as the $m-1$ eigenvectors of S that are associated with the largest $m-1$ eigenvalues of S.
- Since S is symmetric we can use the spectral theorem to use the $m-1$ eigenvectors to construct an orthonormal basis of an $m-1$-dimensional basis of $\mathbb{R}^D$.
- The mth principal component can be found by subtracting from the data the contribution of the first $m-1$ components $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_{m-1}$. Essentially we are trying to find principal components that compress the remainder of the information.

M-dimensional subspace with maximum variance

- $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{D \times N}$ contains the data points $\boldsymbol{x}_k$ as column vectors.
- Then, new data matrix $\hat{\boldsymbol{X}}$ is given by

$$\begin{aligned} \hat{\boldsymbol{X}} &= \boldsymbol{X} - \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^T \boldsymbol{X} \\ &= \boldsymbol{X} - \boldsymbol{B}_{m-1} \boldsymbol{X} \end{aligned}$$

- Here $\boldsymbol{B}_{m-1} = \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^T$ is a projection matrix that projects $\boldsymbol{X}$ onto the subspace spanned by $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_{m-1}$.

M-dimensional subspace with maximum variance

- To find the mth principal component we maximize

$$\begin{aligned} V_m = \mathbb{V}[z_m] &= \frac{1}{N} \sum_{n=1}^{n=N} z_{mn}^2 \\ &= \frac{1}{N} \sum_{n=1}^{n=N} (\boldsymbol{b}_m^T \hat{\boldsymbol{x}}_n)^2 \\ &= \boldsymbol{b}_m^T \hat{\boldsymbol{S}} \boldsymbol{b}_m. \end{aligned}$$

- Here $\hat{\boldsymbol{S}}$ is the data covariance matrix of the transformed data set represented by $\hat{\boldsymbol{x}} = [\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots \hat{\boldsymbol{x}}_N]$.

M-dimensional subspace with maximum variance

- As before we set up a constrained optimization problem to find the first principal component, and establish that the optimal solution $\boldsymbol{b}_m$ is the eignevector of $\hat{\boldsymbol{S}}$ that corresponds to the largest eigenvalue.
- We now establish that $\boldsymbol{b}_m$ is an eigenvector of the original data matrix $\boldsymbol{X}$.
- More generally the sets of eigenvectors for $\hat{\boldsymbol{S}}$ and $\boldsymbol{S}$ are the same.

Eigenvectors of S and $\hat{S}$

- We now show that the eigenvectors of $\boldsymbol{S}$ and $\hat{\boldsymbol{S}}$ are the same.
- Let $\boldsymbol{b}_i$ be an eigenvector of $\boldsymbol{S}$, i.e $\boldsymbol{S}\boldsymbol{b}_i = \lambda \boldsymbol{b}_i$.
- Now we can write

$$\begin{aligned} \hat{\boldsymbol{S}}\boldsymbol{b}_i &= \frac{1}{N}(\boldsymbol{X} - \boldsymbol{B}_{m-1}\boldsymbol{X})(\boldsymbol{X} - \boldsymbol{B}_{m-1}\boldsymbol{X})^T \boldsymbol{b}_i \\ &= (\boldsymbol{S} - \boldsymbol{S}\boldsymbol{B}_{m-1}^T - \boldsymbol{B}_{m-1}\boldsymbol{S} + \boldsymbol{B}_{m-1}\boldsymbol{S}\boldsymbol{B}_{m-1}^T)\boldsymbol{b}_i \\ &= (\boldsymbol{S} - \boldsymbol{S}\boldsymbol{B}_{m-1} - \boldsymbol{B}_{m-1}\boldsymbol{S} + \boldsymbol{B}_{m-1}\boldsymbol{S}\boldsymbol{B}_{m-1})\boldsymbol{b}_i \end{aligned}$$

- Note that in the last line we have used the fact that $\boldsymbol{B}_{m-1}$ is a projection matrix and is therefore symmetric.

Eigenvectors of S and $\hat{S}$

- **Case 1:** $i \geq m$.
- $\boldsymbol{b}_i$ is an eigenvector not among the first $m-1$ components.
- Since $\boldsymbol{B}_{m-1} = \sum_{i=1}^{m-1} \boldsymbol{b}_i \boldsymbol{b}_i^T$ and $\boldsymbol{b}_m$ is orthogonal to the $\boldsymbol{b}_i, 1 \leq i \leq m-1$, we have $\boldsymbol{B}_{m-1}\boldsymbol{b}_i = 0$.
- Plugging this into the last equation on the previous slide, we see that $\hat{\boldsymbol{S}}\boldsymbol{b}_i = (\boldsymbol{S} - \boldsymbol{B}_{m-1}\boldsymbol{S})\boldsymbol{b}_i = \boldsymbol{S}\boldsymbol{b}_i = \lambda_i \boldsymbol{b}_i$.
- Thus $\boldsymbol{S}\boldsymbol{b}_m = \lambda_m \boldsymbol{b}_m$. λ_m is the mth largest eigenvalue of $\boldsymbol{S}$ and is also the largest eigenvalue of $\hat{\boldsymbol{S}}$ because of the way the constrained optimization problem is set up.

Eigenvectors of S and $\hat{S}$

- **Case 2:** $i \leq m-1$.
- We have $\boldsymbol{B}_{m-1}\boldsymbol{b}_i = \sum_{j=1}^{m-1} \boldsymbol{b}_j \boldsymbol{b}_j^T \boldsymbol{b}_i = \boldsymbol{b}_i$.
- Plugging this into $\hat{\boldsymbol{S}}\boldsymbol{b}_i = (\boldsymbol{S} - \boldsymbol{S}\boldsymbol{B}_{m-1} - \boldsymbol{B}_{m-1}\boldsymbol{S} + \boldsymbol{B}_{m-1}\boldsymbol{S}\boldsymbol{B}_{m-1})\boldsymbol{b}_i$, we get $\boldsymbol{S}\boldsymbol{b}_i = \boldsymbol{0}$.
- Thus the vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots \boldsymbol{b}_{m-1}$ are eigenvectors for $\hat{\boldsymbol{S}}$ which are associated with the eigenvalue 0.

- Since $V_m = \boldsymbol{b}_m^T \boldsymbol{S} \boldsymbol{b}_m = \lambda_m$, we see that the variance of the data projected onto the mth principal component is λ_m.
- To find an M-dimensional susbspace that retains as much information as possible, PCA tells us to choose the columns of the matrix $\boldsymbol{B}$ as the M eigenvectors of the data covariance matrix $\boldsymbol{S}$ that have the largest eigenvalues.

Projection perspective

- We derived the PCA as an algorithm that maximizes the variance in the projected space to retain as much information as possible.
- Now we can also derive the PCA using a projection perspective to minimize the average reconstruction error. The original data is modeled as $\boldsymbol{x}_n$ and the reconstruction is modeled as $\tilde{\boldsymbol{x}}_n$. We seek to minimize the distance between $\boldsymbol{x}_n$ and $\boldsymbol{x}_n$.

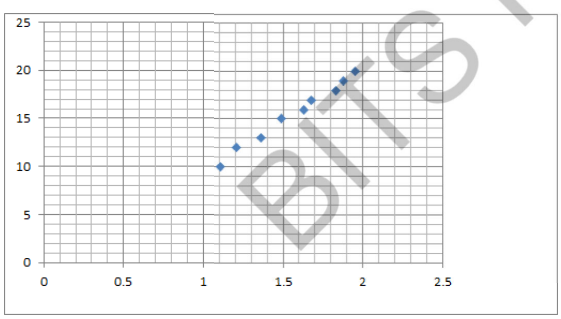
An Example

Consider the data given below:

x1	x2	x3	x4	x5	x6	x7	x8	x9	Mean
1.11	1.21	1.36	1.49	1.63	1.68	1.83	1.88	1.95	1.571111111
10	12	13	15	16	17	18	19	20	15.55555556

An Example

Data when plotted, we get



After subtracting mean from the data, then we get

$$S = \frac{1}{9}XX' = \begin{pmatrix} 0.079498765 & 0.888271605 \\ 0.888271605 & 10.02469136 \end{pmatrix}$$

The largest eigenvalue of S is 10.103 and the corresponding eigenvector with unit norm (first principal component) is $b_1 = [0.088269, 0.996097]^T$.

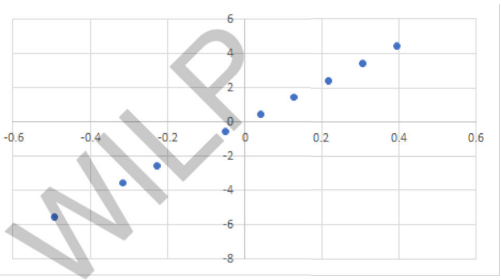
The compressed or the reduced data is then given by $z = b_1^T X$

z1	z2	z3	z4	z5	z6	z7	z8	z9
-5.57	-3.57	-2.56	-0.56	0.45	1.45	2.46	3.46	4.46

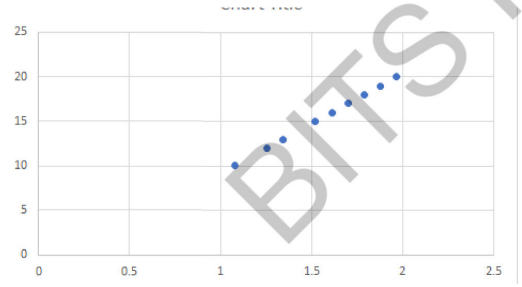
We can project the data onto the principal subspace by $\tilde{X} = b_1 z$. To obtain our projection in the original data space (i.e., before standardization), we need to undo the standardization.

$$\tilde{X} = \tilde{X} + \mu$$

Projected data onto the Principal Subspace



Projected data onto the Principal Subspace in the original data space



Introduction

- In the previous lecture, we discussed dimensionality reduction using PCA
- In this lecture, we will see the use of linear algebra in practical implementation of PCA.
- We will also study the challenges encountered when PCA is used in problems of larger dimensions.
- Finally, we elaborate the key steps of PCA in practice.

Revision of PCA problem

- We derived the matrix **B** used in generation of lower-dimensional representation $\boldsymbol{z}$ and the compressed data $\tilde{\boldsymbol{x}}$.
- The data covariance matrix **S** was used to derive **B**
- Recall that linear relationship connecting the original data $\boldsymbol{x}$, its low-dimensional code $\boldsymbol{z}$ and the compressed data $\tilde{\boldsymbol{x}}$: $\boldsymbol{z} = \boldsymbol{B}^T\boldsymbol{x}$, and $\tilde{\boldsymbol{x}} = \boldsymbol{B}\boldsymbol{z}$.

Recall : Example

Consider the data given below:

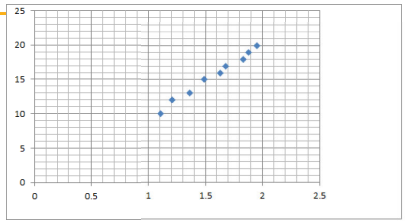
x1	x2	x3	x4	x5	x6	x7	x8	x9	Mean
1.11	1.21	1.36	1.49	1.63	1.68	1.83	1.88	1.95	1.571
10	12	13	15	16	17	18	19	20	15.555

$$S = \frac{1}{9}XX^T = \begin{bmatrix} 0.079 & 0.888 \\ 0.888 & 10.024 \end{bmatrix}$$

The largest eigenvalue of S is $\lambda = 10.103$.

The corresponding eigenvector is $\begin{bmatrix} 0.088 \\ 0.996 \end{bmatrix}$.

Data Plot



The compressed or the reduced data is then given by $z = b_1^T X$

z1	z2	z3	z4	z5	z6	z7	z8	z9
-5.57	-3.57	-2.56	-0.56	0.45	1.45	2.46	3.46	4.46

Eigenvector Computation and Low Rank Approximation

- In the previous sections, we obtained the basis of the principal subspace as the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix.

$$S = \frac{1}{N}\sum_{i=1}^{N} x_n x_n^T$$

- Equivalently we get

$$S = \frac{1}{N}XX^T$$

- Note that $X = [x_1, ..., x_N] \in \mathbb{R}^{D\times N}$
- Note that X is a $D \times N$ matrix,

Eigenvector Computation and Low Rank Approximation

- To get the eigenvalues and the corresponding eigenvectors of S, we can follow two approaches
- We can perform an eigen-decomposition and compute the eigenvalues and eigenvectors of S directly.
- We can also use a singular value decomposition.
- Since S is symmetric and factorizes into XX^T, the eigenvalues of S are the squared singular values of X.
- Assume the SVD of X as $X = U\Sigma V^T$. Then

$$S = \frac{1}{N}XX^T = \frac{1}{N}U\Sigma\Sigma^T U^T$$

Example of PCA Revisited

Consider the data given below:

x1	x2	x3	x4	x5	x6	x7	x8	x9	Mean
1.11	1.21	1.36	1.49	1.63	1.68	1.83	1.88	1.95	1.57
10	12	13	15	16	17	18	19	20	15.55

Consider $X = U\Sigma V^T$

$$U = \begin{bmatrix} 0.0883 & 0.9961 \\ 0.9961 & -0.0883 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 9.535 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.084 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Eigenvector Computation and Low Rank Approximation

- The columns of U are the eigenvectors of S.
- The eigenvalues λ_d of S are related to the singular values of X via

$$\lambda_d = \frac{\sigma_d^2}{N}$$

- This relationship between the eigenvalues of S and the singular values of X provides the connection between the maximum variance view and the singular value decomposition.

Best Rank-M approximation

- To maximize the variance of the projected data PCA chooses the columns of U to be the eigenvectors that are associated with the M largest eigenvalues of the data covariance matrix S
- The Eckart-Young theorem offers a direct way to estimate the low-dimensional representation.
- Consider the best rank-M approximation of X defined as $\tilde{X}_M$

$$\tilde{X}_M = \text{argmin}_{rank(A)<=M} \|X - A\|_2$$

Best Rank-M approximation

- The Eckart-Young theorem states that the best rank M approximation $\tilde{X}_M$ is given by truncating the SVD at the top-M singular value.

$$\tilde{X}_M = U_M \Sigma_M V_M^T$$

- U_M is an orthogonal matrix
- V_M is an orthogonal matrix
- Σ_M has M largest singular values of X as diagonal entries

Practical Aspects of PCA

- Finding eigenvalues and eigenvectors is also important in other fundamental machine learning methods that require matrix decompositions
- In theory we can solve for the eigenvalues as roots of the characteristic polynomial.
- However for matrices larger than 4 by 4 this is not possible because we would need to find the roots of polynomial of degree 5 or higher.

Practical Aspects of PCA

- However the Abel-Ruffini theorem states that there exists no algebraic solution to this problem for polynomials of degree 5 or more.
- Therefore, in practice, solve for eigenvalues or singular values using iterative methods, which are implemented in all modern packages for linear algebra
- In many applications we only require a few eigenvectors.

Practical Aspects of PCA

- It would be wasteful to compute the full decomposition,and then discard all eigenvectors with eigenvalues that are beyond the first few.
- It turns out that if we are interested in only the first few eigenvectors (with the largest eigenvalues), then iterative processes, which directly optimize these eigenvectors, are computationally more efficient than a full eigen-decomposition

Practical Aspects of PCA

- In the extreme case of only needing the first eigenvector, a simple method called the power iteration is very efficient.
- Power iteration chooses a random vector x_0 that is not in the null space of S and follows the iteration for $k = 0, 1, \ldots$

$$x_{k+1} = \frac{Sx_k}{\|Sx_k\|}$$

- This sequence of vectors converges to the eigenvector associated with the largest eigenvalue of S.

PCA in High Dimension

1. In order to do PCA, we need to compute the data covariance matrix.
2. In D dimensions, the data covariance matrix is a $D \times D$ matrix.
3. Computing the eigenvalues and eigenvectors of this matrix is computationally expensive as it scales cubically in D.
4. Therefore, PCA, as we discussed earlier, will be infeasible in very high dimensions.
5. In the following, we provide a solution to this problem for the case that we have substantially fewer data points than dimensions, i.e., $N << D$

PCA in High Dimension

- Assume we have a centered dataset $x_1, ..., x_N$,where $x_i \in \mathbb{R}^D$.
- Then the data covariance matrix is given as $S = \frac{1}{N} XX^T$
- Consider the eigenvectors equation of S

$$Sb_m = \lambda_m b_m$$

- By substituting the definition of S

$$\frac{1}{N} XX^T b_m = \lambda_m b_m$$

PCA in High Dimension

- Multiply by X^T, we get

$$\frac{1}{N} X^T XX^T b_m = \lambda_m X^T b_m$$

- If $c_m = X^T b_m$, then

$$\frac{1}{N} X^T Xc_m = \lambda_m c_m$$

- The nonzero eigenvalues of XX^T is same as the nonzero eigenvalues of $X^T X$.

PCA in High Dimension

- Now that we have the eigenvectors of $\frac{1}{N} X^T X$,

$$\frac{1}{N} X^T Xc_m = \lambda_m c_m$$

- We need to derive the eigenvectors of XX^T, which we still need for PCA
- Multiply by X to get

$$\frac{1}{N} XX^T Xc_m = \lambda_m Xc_m$$

- Here, we recover the data covariance matrix again.
- This means that we recover Xc_m as an eigenvector of S.

Key Steps of PCA in Practice

In the following, we will go through the individual steps of PCA using a running example.

- We are given a two dimensional dataset
- we want to use PCA to project it onto a one-dimensional subspace.
- The key steps are given below
  - Mean subtraction
  - Standardization
  - Eigen-decomposition of the covariance matrix
  - Projection

Key Steps of PCA in Practice

Mean subtraction

1. We start by centering the data by computing the mean of the dataset and subtracting it from every single data point.
2. This ensures that the dataset has mean 0.
3. Mean subtraction is not strictly necessary but reduces the risk of numerical problems

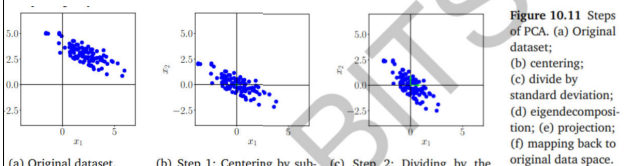
Key Steps of PCA in Practice

Standardization

1. Divide the data points by the standard deviation σ_d of the dataset for every dimension d.
2. Now the data is unit free, and it has variance 1 along each axis, which is indicated by the standardization
3. This step completes the standardization of the data.

Key Steps of PCA in Practice

Figure: PCA



(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.

Figure 10.11 Steps of PCA. (a) Original dataset; (b) centering; (c) divide by standard deviation; (d) eigendecomposition; (e) projection; (f) mapping back to original data space.

Key Steps of PCA in Practice

Eigen-decomposition of the covariance matrix

1. Compute the data covariance matrix
2. Compute its eigenvalues and corresponding eigenvectors.
3. Since the covariance matrix is symmetric, the spectral theorem states that we can find an orthonormal basis of eigenvectors.
4. The eigenvectors are scaled by the magnitude of the corresponding eigenvalue.

Key Steps of PCA in Practice

Projection

1. We can project any data point $x_* \in \mathbb{R}^d$ onto the principal subspace:
2. To get this right, we need to standardize x_* using the mean and standard deviation of the training data in the d th dimension

$$x_*^{(d)} = \frac{x_*^{(d)} - \mu_d}{\sigma_d}, \quad d = 1, ..., D$$

3. Here $x_*^{(d)}$ is the d th component of x_*.

Key Steps of PCA in Practice

1. We obtain the projection as

$$\tilde{x} = BB^T x_*$$

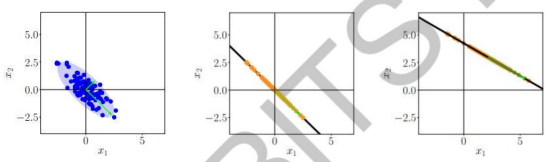
2. The coordinates are

$$z_* = B^T x_*$$

with respect to the basis of the principal subspace.

3. Here, B is the matrix that contains the eigenvectors that are associated with the largest eigenvalues of the data covariance matrix as columns.

Key Steps of PCA in Practice



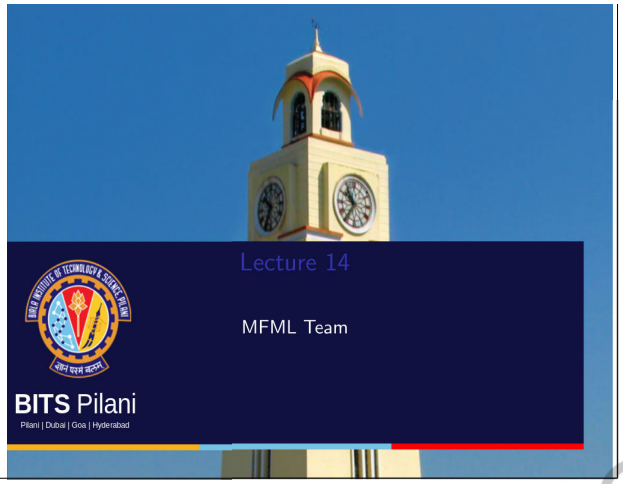
(d) Step 3: Compute eigenvalues and eigenvectors (arrows) of the data covariance matrix (ellipse).

(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).

Summary of PCA

1. We derived PCA by maximizing the variance in the projected space
2. We took high-dimensional data $x \in \mathbb{R}^D$ and used a matrix B to find a lower-dimensional representation $z \in \mathbb{R}^M$
3. The columns of B are the eigenvectors of the data covariance matrix S that are associated with the largest eigenvalues.
4. Once we have a low-dimensional representation z, we can get a high-dimensional version of it as Bz.



Agenda

Mathematical preliminaries for Support Vector Machines

- Constrained optimization and Lagrange multipliers.
- Primal and dual problems and how their solutions are related
- Karash-Kuhn-Tucker conditions.
- Definition of Kernel Functions
- Linear Classifiers

Optimization problem

We shall work with the following optimization problem:

$$\min f(\boldsymbol{x}) \text{ subject to}$$
$$g_i(\boldsymbol{x}) \leq 0 \ \forall i \in [m]$$
$$h_j(\boldsymbol{x}) = 0 \ \forall j \in [p]$$

Optimization problem : Lagrangian

The Lagrangian associated with this optimization problem is

- $$\min f(\boldsymbol{x}) + \sum_{i=1}^{i=m} \lambda_i g(\boldsymbol{x}) + \sum_{j=1}^{j=p} \nu_j h_j(\boldsymbol{x})$$
- The λ_i's and h_j's are called Lagrange multipliers.

Quadratic programming

Consider the following primal problem:

- We now consider the case of a quadratic objective function subject to affine constraints:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$
$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

- Here $\boldsymbol{A} \in \mathbb{R}^{m\times d}, b \in \mathbb{R}^m, c \in \mathbb{R}^d$

Quadratic programming

- The Lagrangian $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ is given by $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$.
- Rearranging the above we have $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x} - \boldsymbol{\lambda}^T\boldsymbol{b}$
- Taking the derivative of $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ and setting it equal to zero gives $\boldsymbol{Q}\boldsymbol{x} + (\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) = 0$.

Quadratic programming

We will now derive the dual problem

- If we take $\boldsymbol{Q}$ to be invertible, we have $\boldsymbol{x} = \boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})$.
- Plugging this value of $\boldsymbol{x}$ into $\mathfrak{L}(\boldsymbol{x}, \boldsymbol{\lambda})$ gives us $\mathfrak{D}(\boldsymbol{\lambda}) = -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$.
- This gives us the dual optimization problem: $\max_{\boldsymbol{\lambda}\in\mathbb{R}^m} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$ subject to $\boldsymbol{\lambda} \geq \boldsymbol{0}$.

Summary

The original problem is :

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{c}^T\boldsymbol{x}$$
$$\text{subject to } \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$

The dual problem is

$$\max_{\boldsymbol{\lambda}\geq 0} -\frac{1}{2}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda})\boldsymbol{Q}^{-1}(\boldsymbol{c} + \boldsymbol{A}^T\boldsymbol{\lambda}) - \boldsymbol{\lambda}^T\boldsymbol{b}$$

Weak duality

- Weak duality establishes an inequality connecting primal and dual problems
- Weak duality condition states that the optimal solution of the primal problem is greater than or equal to that of the dual problem.
- In the Quadratic Optimization problem discussed previously , weak duality exists

Strong duality

- Strong duality condition states that the optimal solution of the primal problem is equal to that of the dual problem
- One can solve the dual problem to get the same solution as solving the primal problem.
- In some optimization problems, solving the dual problem may be easier.
- Question: When does strong duality hold?

Slater's condition

- For a primal optimization problem we say that it obeys Slater's condition if
  1. the objective function f is convex, the constraints g_i are all convex ,the contraint functions h_i are all linear
  2. there exists a point $\bar{\boldsymbol{x}}$ in the interior of the region, i.e $g_i(\bar{\boldsymbol{x}}) < 0$ for all $i \in [m]$ and $h_j(\bar{\boldsymbol{x}}) = 0$ for all $j \in [p]$.
- Suppose Slater's condition holds then we have strong duality.
- Strong duality condition states that the optimal solution of the primal problem is equal to that of the dual problem

Example of Slater's condition

We will consider an optimization problem as given below

$$\min\ x^2 + y^2$$
$$\text{st } x + y - 3 \leq 0$$

- Here $f(x, y) = x^2 + y^2$ is a convex function and $g(x, y) = x + y - 3$ is a convex function
- We can find a point that satisfies the condition $x + y - 3 < 0$
- Slaters condition is satisfied

KKT conditions

$$\min f(\boldsymbol{x}) \quad \text{st} \quad g_i(\boldsymbol{x}) \leq 0 \; \forall i \in [m], \quad h_j(\boldsymbol{x}) = 0 \; \forall j \in [p]$$

We say that $\boldsymbol{x}^*$ and $(\lambda^*, \nu^*) \in \mathbb{R}^m \times \mathbb{R}^p$ respect the Karash-Kuhn-Tucker conditions if:

1. $g_i(\boldsymbol{x}^*) \leq 0 \; \forall i \in [m]$, $h_i(\boldsymbol{x}^*) = 0 \; \forall i \in [p]$.
2. $\lambda_i^* \geq 0 \; \forall i \in [m]$.
3. $\lambda_i^* g_i(\boldsymbol{x}^*) = 0 \; \forall i \in [m]$.
4. $\nabla f(\boldsymbol{x}^*) + \sum_{i=1}^{i=m} \lambda_i^* \nabla g_i(\boldsymbol{x}^*) + \sum_{i=1}^{i=p} \nu_i^* \nabla h_i(\boldsymbol{x}^*) = 0$.

If strong duality holds then any primal optimal solution $\boldsymbol{x}^*$ and dual optimal solution (λ^*, ν^*) satisfy the KKT conditions.

KKT condition

We will consider an optimization problem and will write its KKT conditions

$$\begin{aligned} &\min \; x^2 + y^2 \\ &\text{st } x + y - 3 \leq 0 \end{aligned}$$

- Here $f(x, y) = x^2 + y^2$ and $g(x, y) = x + y - 3$

1. $x + y - 3 \leq 0$
2. $\lambda \geq 0$
3. $\lambda(x + y - 3) = 0$
4. $\nabla f + \lambda \nabla g = \mathbf{0}$

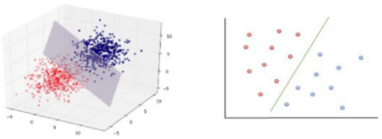
Classification Problem in Machine Learning

- Classification of data into different classes is one of the primary problems in machine learning
- Binary classification involves classifying data into exactly 2 classes
- There exists different algorithms for binary classification
- We will discuss a model called Support Vector Machine.
- SVM is a linear classifer model for binary classification

Linear Classifier

$\mathbf{w}^T\mathbf{x} = 0$ Hyperplane

$y = ax + b$ Line

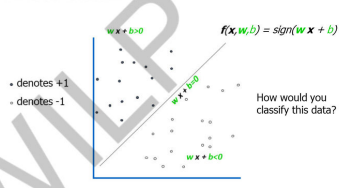


Linear Classifier and Hyperplane

- Consider line $w^T x + b = 0$. Let x_a and x_b lie on this line. So $w^T x_a + b = 0$ and $w^T x_b + b = 0$.
- This means $w^T(x_a - x_b) = 0$. $x_a - x_b$ lies on the line and is directed from x_b to x_a.
- Hence w is orthogonal to $x_a - x_b$ and in turn, to the line.

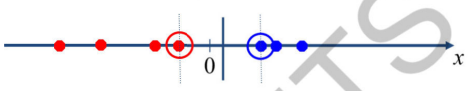
Linear Classifer for Binary Classification

Linear Classifiers



Two examples of data

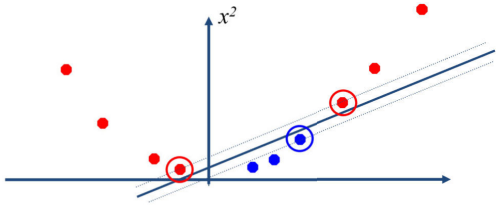
Dataset that are linearly separable with some noise



Dataset is not linearly separable

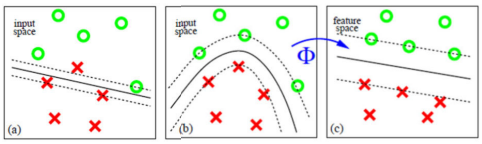
Mapping of Data

mapping data to a higher-dimensional space:



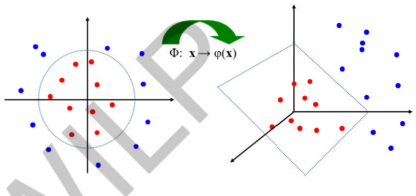
Mapping of Data

Find a feature space



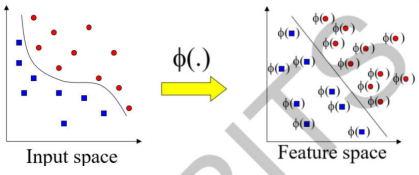
If every data point is mapped into high-dimensional space via some transformation $\phi : x \rightarrow \phi(x)$

Feature spaces



- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable.

Transforming the Data



- Computation in the feature space can be costly because it is high dimensional.
- The feature space is typically infinite-dimensional.
- The kernel trick using kernel functions comes to rescue

Kernel Functions

- Kernel is a continuous function $K(x, y)$
- Kernel takes two arguments x and y
- x and y could be real numbers, functions, vectors, etc
- $K(x, y)$ maps x and y to a real value
- Kernel value is independent of the order of the arguments, i.e.,

$$K(x, y) = K(y, x)$$

Kernel Functions

- A kernel function is some function that corresponds to an inner product in some expanded feature space.

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

- Linear classifier relies on dot product between vectors $x_i^T x_j$
- If every data point is mapped into high-dimensional space via some transformation $\phi : x \rightarrow \phi(x)$, the dot product becomes: $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
- For some functions $K(x_i, x_j)$ checking $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is difficult.
- Mercer's theorem: Every positive-semidefinite symmetric function is a kernel function.

Kernel Functions Construction

1) We can *construct kernels from scratch*:

- For any $\varphi : \mathcal{X} \rightarrow \mathbb{R}^m$, $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathbb{R}^m}$ is a kernel.
- If $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a *distance function*, i.e.
  - $d(x, x') \geq 0$ for all $x, x' \in \mathcal{X}$,
  - $d(x, x') = 0$ only for $x = x'$,
  - $d(x, x') = d(x', x)$ for all $x, x' \in \mathcal{X}$,
  - $d(x, x') \leq d(x, x'') + d(x'', x')$ for all $x, x', x'' \in \mathcal{X}$,

then $k(x, x') := \exp(-d(x, x'))$ is a kernel.

Kernel Functions Construction

2) We can *construct kernels from other kernels*:

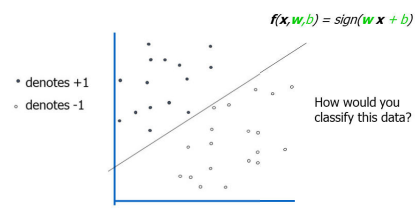
- if k is a kernel and $\alpha > 0$, then αk and $k + \alpha$ are kernels.
- if k_1, k_2 are kernels, then $k_1 + k_2$ and $k_1 \cdot k_2$ are kernels.

Examples of Kernels

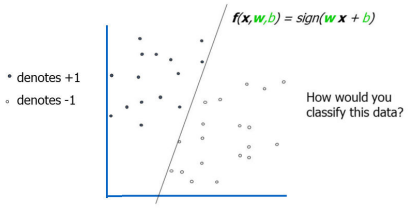
- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial of power p : $K(x_i, x_j) = (1 + x_i^T x_j)^p$
- Sigmoid: $K(x_i, x_j) = tanh(\beta_0 x_i^T x_j + \beta_1)$

Linear Classifiers Revisited

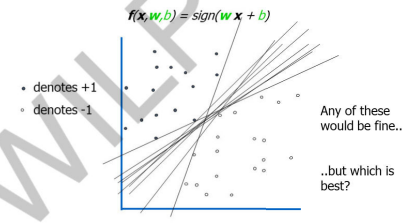
Linear Classifiers



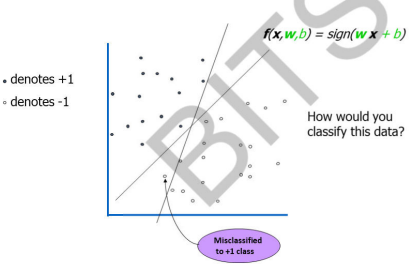
Linear Classifiers



Linear Classifiers



Linear Classifiers



Support Vector Machines

MFML Team

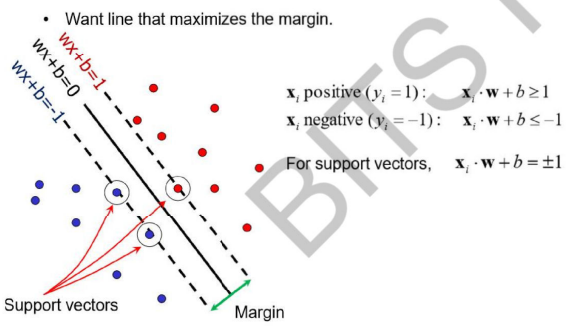
BITS Pilani
Pilani Campus

Topics to be covered

- Linear Classifiers
- Maximum Margin Classification
- Linear SVM
- SVM optimization problem
- Soft Margin SVM

Support Vector Machines

- Want line that maximizes the margin.



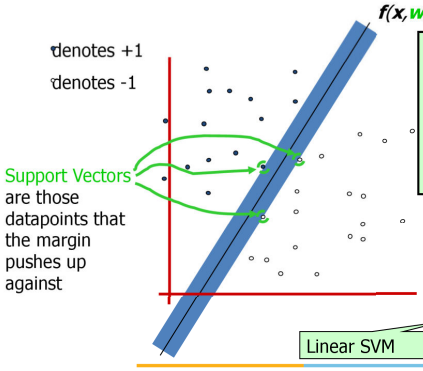
$\mathbf{x}_i$ positive $(y_i = 1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

For support vectors, $\mathbf{x}_i \cdot \mathbf{w} + b = \pm 1$

Maximum Margin

$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}\,\mathbf{x} + b)$



1. If hyperplane is oriented such that it is close to some of the points in your training set, new data may lie on the wrong side of the hyperplane, even if the new points lie close to training examples of the correct class.
2. Solution is maximizing the margin

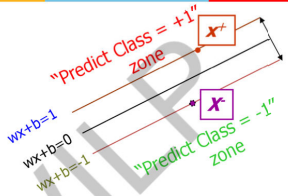
The maximum margin linear classifier is the linear classifier with the, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

Support Vectors

- Geometric description of SVM is that the max-margin hyperplane is completely determined by those points that lie nearest to it.
- Points that lie on this margin are the support vectors.
- The points of our data set which if removed, would alter the position of the dividing hyperplane

Linear SVM Mathematically



M=Margin Width

What we know:

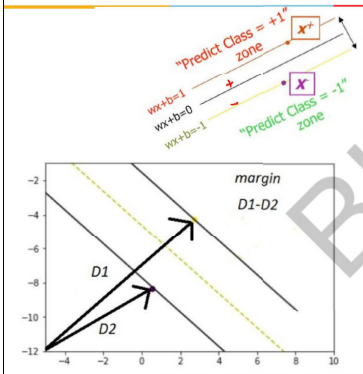
- $\mathbf{w} \cdot \mathbf{x}^+ + b = +1$
- $\mathbf{w} \cdot \mathbf{x}^- + b = -1$

Distance between lines given by solving linear equation:

Maximize margin: $M = \frac{2}{||w||}$

Equivalent to minimize: $\frac{1}{2}||w||^2$

Linear SVM Mathematically



M=Margin Width

$D1 = w^T x+b=1$ $\quad w^T x+b-1=0$

$D2 = w^T x+b=-1$ $\quad w^T x+b+1=0$

$w^T x+b-1 - w^T x+b+1$

Solve algebraically

$\frac{2}{|w|}$

Optimization Problem

1. Maximize margin $2/||\mathbf{w}||$
2. Correctly classify all training data points:

$\mathbf{x}_i$ positive $(y_i = 1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$

$\mathbf{x}_i$ negative $(y_i = -1)$: $\quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$

Quadratic optimization problem:

Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$ is minimized;
and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$
$+1(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$
$-1(\mathbf{w}^T \mathbf{x}_i + b) \leq 1$
same as $(\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2$ is minimized; Type equation here.
and for all $\{(\mathbf{x}_i, y_i)\}$: $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$ ← Primal

- **Need to optimize a *quadratic* function subject to *linear inequality* constraints.**
- **All constraints in SVM are linear**
- **Quadratic optimization problems are a well-known class of mathematical programming problems, and many (rather intricate) algorithms exist for solving them.**
- **The solution involves constructing a *unconstrained problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

Recall : Karush–Kuhn–Tucker (KKT) theorem

- max f(x) subject to $g_i(x) = 0$ and $h_j(x) \geq 0$ for all i, j
- Make the Lagrangian function

$$\mathcal{L} = f(x) - \sum_i \lambda_i g_i(x) - \sum_j \mu_j h_j(x)$$

- Necessary conditions to have a minimum are

$$\nabla_x \mathcal{L}(x^*, \lambda^*, \mu^*) = 0$$
$$g_i(x^*) = 0 \text{ for all } i$$
$$h_j(x^*) \geq 0 \text{ for all } j$$
$$\mu_j \geq 0 \text{ for all } j$$
$$\mu_j^* h_j(x^*) = 0 \text{ for all } j$$

Solving the Optimization Problem

- **The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every constraint in the primary problem:**

$$L(w, b, \alpha_i) = \frac{1}{2}||\mathbf{w}||^2 - \Sigma\, \alpha_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1]$$

- **Taking partial derivative with respect to w, $\frac{\partial L}{\partial w} = 0$**
  - $w - \Sigma\, \alpha_i y_i \mathbf{x}_i = 0$
  - $w = \Sigma\, \alpha_i y_i \mathbf{x}_i$
- **Taking partial derivative with respect to b, $\frac{\partial L}{\partial b} = 0$**
  - $-\Sigma\, \alpha_i y_i = 0$
  - $\Sigma\, \alpha_i y_i = 0$

Solving the Optimization Problem

$$L(w, b, \alpha_i) = \frac{1}{2}||\mathbf{w}||^2 - \Sigma\, \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

- Expanding above equation:

$$L(w, b, \alpha_i) = \frac{1}{2}||\mathbf{w}||^2 - \Sigma\, \alpha_i y_i \mathbf{w} \cdot \mathbf{x}_i - \Sigma\, \alpha_i y_i b + \Sigma\, \alpha_i$$

- Substituting $w = \Sigma\, \alpha_i y_i \mathbf{x}_i$ and $\Sigma\, \alpha_i y_i = 0$ in above equation

$$L(w, b, \alpha_i) = \frac{1}{2} (\Sigma_i \alpha_i y_i \mathbf{x}_i)(\Sigma_j \alpha_j y_j \mathbf{x}_j) - (\Sigma_i \alpha_i y_i \mathbf{x}_i)(\Sigma_j \alpha_j y_j \mathbf{x}_j) + \Sigma\, \alpha_i$$

$$L(w, b, \alpha_i) = \Sigma\, \alpha_i - \frac{1}{2} (\Sigma_i \alpha_i y_i \mathbf{x}_i)(\Sigma_j \alpha_j y_j \mathbf{x}_j)$$

$$L(w, b, \alpha_i) = \Sigma\, \alpha_i - \frac{1}{2} (\Sigma_i \Sigma_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j)$$

Support Vectors

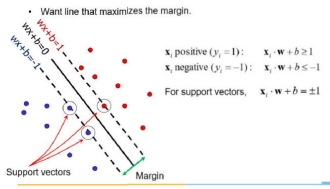
Using KKT conditions : $\alpha_i [y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1] = 0$

For this condition to be satisfied either $\alpha_i = 0$ and $y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1 > 0$
OR $y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1 = 0$ and $\alpha_i > 0$

$L(w, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \Sigma\, \alpha_i [y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1]$

For support vectors: $y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1 = 0$

For all points other than support vectors: $\alpha_i = 0$



Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Learned weight | Support vector

Solving the Optimization Problem

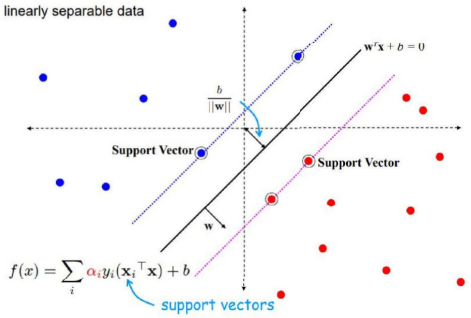
- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$
  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)
- Classification function:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point $\boldsymbol{x}$ and the support vectors $\boldsymbol{x_i}$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x_i} \cdot \boldsymbol{x_j}$ between all pairs of training points)

Substituting w in support vectors function

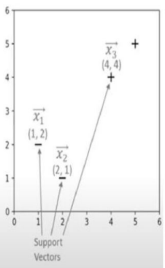


Linear SVM: Numerical Problem

$$\sum_i \alpha_i y_i \vec{x_i} \cdot \vec{x} + b = 1 \quad \text{if } \vec{x} \text{ is +ve support vector}$$
$$\sum_i \alpha_i y_i \vec{x_i} \cdot \vec{x} + b = -1 \quad \text{if } \vec{x} \text{ is −ve support vector}$$
$$\sum_i \alpha_i y_i = 0$$

Let $\beta_i = \alpha_i y_i$

$$\begin{cases} \beta_1 \vec{x_1} \cdot \vec{x_1} + \beta_2 \vec{x_2} \cdot \vec{x_1} + \beta_3 \vec{x_3} \cdot \vec{x_1} + b = -1 & \vec{x} = \vec{x_1} \\ \beta_1 \vec{x_1} \cdot \vec{x_2} + \beta_2 \vec{x_2} \cdot \vec{x_2} + \beta_3 \vec{x_3} \cdot \vec{x_2} + b = -1 & \vec{x} = \vec{x_2} \\ \beta_1 \vec{x_1} \cdot \vec{x_3} + \beta_2 \vec{x_2} \cdot \vec{x_3} + \beta_3 \vec{x_3} \cdot \vec{x_3} + b = 1 & \vec{x} = \vec{x_3} \\ \beta_1 + \beta_2 + \beta_3 = 0 \end{cases}$$

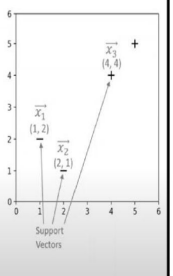


Numerical: Linear SVM

$$\begin{cases} \beta_1 \vec{x_1} \cdot \vec{x_1} + \beta_2 \vec{x_2} \cdot \vec{x_1} + \beta_3 \vec{x_3} \cdot \vec{x_1} + b = -1 \\ \beta_1 \vec{x_1} \cdot \vec{x_2} + \beta_2 \vec{x_2} \cdot \vec{x_2} + \beta_3 \vec{x_3} \cdot \vec{x_2} + b = -1 \\ \beta_1 \vec{x_1} \cdot \vec{x_3} + \beta_2 \vec{x_2} \cdot \vec{x_3} + \beta_3 \vec{x_3} \cdot \vec{x_3} + b = 1 \\ \beta_1 + \beta_2 + \beta_3 = 0 \end{cases}$$

Plug in the values of all the vectors

$$\begin{cases} 5\beta_1 + 4\beta_2 + 12\beta_3 + b = -1 \\ 4\beta_1 + 5\beta_2 + 12\beta_3 + b = -1 \\ 12\beta_1 + 12\beta_2 + 32\beta_3 + b = 1 \\ \beta_1 + \beta_2 + \beta_3 = 0 \end{cases} \Rightarrow \begin{cases} \beta_1 = -0.08 \\ \beta_2 = -0.08 \\ \beta_3 = 0.16 \\ b = -2.2 \end{cases}$$



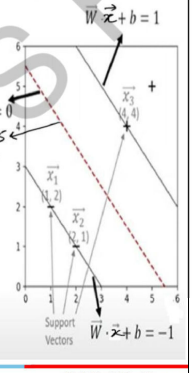
$$\begin{cases} \beta_1 = -0.08 \\ \beta_2 = -0.08 \\ \beta_3 = 0.16 \end{cases}$$

So, hyperplane is essentially those $\vec{x}$ that satisfy the equality.

$$\vec{W} = \sum_i \alpha_i y_i \vec{x_i} \quad \sum_i \beta_i \vec{x_i} = -0.08 \begin{pmatrix} 1 \\ 2 \end{pmatrix} - 0.08 \begin{pmatrix} 2 \\ 1 \end{pmatrix} + 0.16 \begin{pmatrix} 4 \\ 4 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.4 \end{pmatrix}$$

$$\vec{W} \cdot \vec{x} + b = 0$$

So, $\begin{pmatrix} 0.4 \\ 0.4 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 2.2 = 0 \Rightarrow x_1 + x_2 = 5.5$



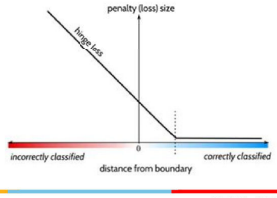
Hinge Loss

Hinge Loss is one of the types of Loss Function, mainly used for maximum margin classification models.

Hinge Loss incorporates a margin or distance from the classification boundary into the loss calculation. Even if new observations are classified correctly, they can incur a penalty if the margin from the decision boundary is not large enough.

$L = \max(0, 1 - y_i (w^T x_i + b))$

0 - for correct classification
1 - for wrong classification



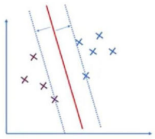
Hinge loss

HINGE LOSS – Numerical Example 1

Correctly classification

- Actual output $y = +1$ Predicted output $y' = 0.5$.
- Hinge Loss = $max(0,\ 1 - y * y')$
  $= max(0,\ 1 - 1 * 0.5)$
  $= 0.5.$

Hinge loss is close to zero for correctly classified sample.



Misclassification

- Actual output $y = -1$. Predicted output $y' = 0.5$.
- Hinge Loss = $max(0,\ 1 - y * y')$
  $= max(0,\ 1 - (-1) * 0.5)$
  $= 1.5.$
- Hinge loss is higher for misclassified sample.

SVM : Estimating w and b

n: no. of data points

- Want to minimize **J(w,b):**

$$J(w,b) = \frac{1}{2}\sum_{j=1}^{d}\left(w^{(j)}\right)^2 + C\sum_{i=1}^{n}\max\left\{0, 1 - y_i\left(\sum_{j=1}^{d} w^{(j)}x_i^{(j)} + b\right)\right\}$$

Empirical loss $L(x_i\ y_i)$

- Compute the gradient $\nabla(j)$ w.r.t. $w^{(j)}$

$$\frac{\partial J(w,b)}{\partial w^{(j)}} = w^{(j)} + C\sum_{i=1}^{n}\frac{\partial L(x_i, y_i)}{\partial w^{(j)}}$$

$$\frac{\partial L(x_i, y_i)}{\partial w^{(j)}} = 0 \quad \text{if } y_i(\mathbf{w}\cdot x_i + b) \geq 1$$

$$= -y_i x_i^{(j)} \quad \text{else}$$

Gradient Descent

Iterate until Convergence

$$Compute: \frac{\partial J}{\partial b} = C\sum_{1}^{n}\frac{\partial L(x_i, y_i)}{\partial b}$$

$$Update: b_{new} \leftarrow b - \eta\frac{\partial J}{\partial b},$$

For j=1…..d

$$Compute: \frac{\partial J}{\partial w^j} = w^j + C\sum_{1}^{n}\frac{\partial L(x_i, y_i)}{\partial w^j}$$

$$Update: w_{new}^j \leftarrow w^j - \eta\frac{\partial J}{\partial w^j},$$

$$b \leftarrow b_{new}, w \leftarrow w_{new}$$

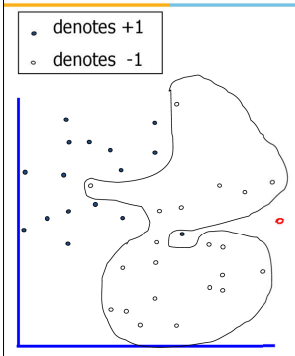
C parameter in cost function

C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.

Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points

Dataset with noise



- **Hard Margin: So far we require all data points be classified correctly**
  - No training error
- **What if the training set is noisy?**

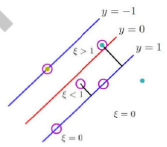
Soft Margin Classification

Slack variables ξ_i **can be added to allow misclassification of difficult or noisy examples.**

What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$



- **Slack variable** as giving the classifier some leniency when it comes to moving around points near the **margin**.
- When C is large, larger slacks penalize the objective function of SVM's more than when C is small.

Soft Margin

$$\min_{\boldsymbol{w}} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

Misclassification cost; # data samples N; Slack variable

The w that minimizes…

Maximize margin; Minimize misclassification

$$\text{subject to} \quad y_i\boldsymbol{w}^T\boldsymbol{x}_i \geq 1 - \xi_i,$$
$$\xi_i \geq 0, \quad \forall i = 1, \ldots, N$$

Hard Margin versus Soft Margin

- **Hard Margin:**

Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$ is minimized and for all $\{(\mathbf{x}_i, y_i)\}$
$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

- **Soft Margin incorporating slack variables:**

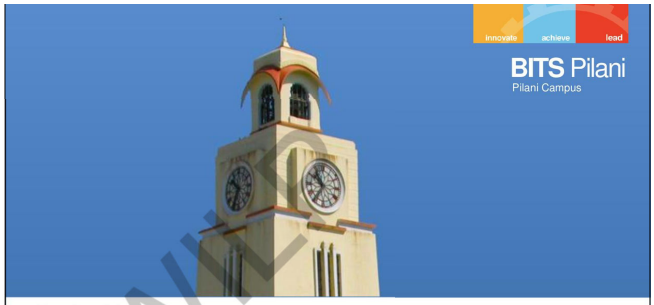
Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum\xi_i$ is minimized and for all $\{(\mathbf{x}_i, y_i)\}$
$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$ and $\xi_i \geq 0$ for all i

- **Parameter C can be viewed as a way to control overfitting.**

Support Vector Machines

MFML Team

BITS Pilani
Pilani Campus



BITS Pilani
Pilani Campus

Text Book(s)

T1	Christopher Bishop: Pattern Recognition and Machine Learning, Springer International Edition
T2	Tom M. Mitchell: Machine Learning, The McGraw-Hill Companies, Inc..

These slides are prepared by the instructor, with grateful acknowledgement of Prof. Tom Mitchell, Prof.. Burges, Prof. Andrew Moore and many others who made their course materials freely available online.

Topics to be covered

- Nonlinear SVM
- Kernel Trick
- SVM Kernels
- XOR problem using non linear SVM

Optimization Problem

Find $\mathbf{w}$ and b such that
$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} = \frac{1}{2}\|\mathbf{w}\|^2$ is minimized;
and for all $\{(\mathbf{x_i}, y_i)\}$: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$

$L(w, b, \alpha_i) = \frac{1}{2}\|\mathbf{w}\|^2 - \Sigma\, \alpha_i [y_i(\mathbf{w}^T\mathbf{x_i} + b) - 1]$

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \frac{1}{2}(\sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j})$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $\alpha_i \geq 0$ for all α_i

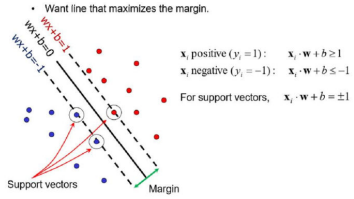
Support Vectors

Using KKT conditions :
$\alpha_i [y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1] = 0$

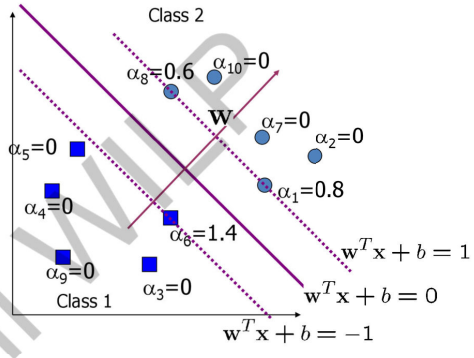
For this condition to be satisfied either $\alpha_i = 0$ and $y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1 > 0$
OR
$y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1 = 0$ and $\alpha_i > 0$

For support vectors:
$y_i (\mathbf{w}^T\mathbf{x_i} + b) - 1 = 0$

For all points other than support vectors:
$\alpha_i = 0$



A Geometrical Interpretation



Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$

Learned weight
Support vector

Solving the Optimization Problem

- Solution: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$
  $b = y_i - \mathbf{w} \cdot \mathbf{x}_i$ (for any support vector)
- Classification function:
$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$
$$= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)$$

If f(x) < 0, classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point ***x*** and the support vectors $\boldsymbol{x}_i$
- (Solving the optimization problem also involves computing the inner products $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$ between all pairs of training points)

Linear SVMs: Overview

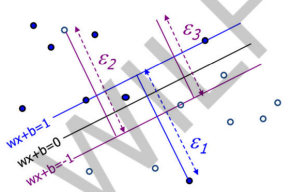
- The classifier is a *separating hyperplane.*
- Most "important" training points are support vectors; they define the hyperplane.
- Quadratic optimization algorithms can identify which training points x_i are support vectors with non-zero Lagrangian multipliers α_i.
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find $\alpha_1 \ldots \alpha_N$ such that
$Q(\alpha) = \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \le \alpha_i \le C$ for all α_i

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$

Soft Margin Classification

Slack variables ξ_i **can be added to allow misclassification of difficult or noisy examples.**

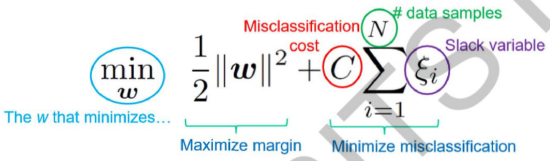


What should our quadratic optimization criterion be?

Minimize

$$\frac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_{k=1}^{R}\varepsilon_k$$

Soft Margin



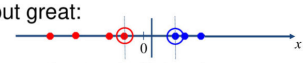
$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

subject to $y_i \boldsymbol{w}^T \boldsymbol{x}_i \ge 1 - \xi_i,$

$\xi_i \ge 0, \quad \forall i = 1, \ldots, N$

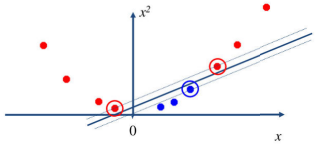
Non-linear SVMs

- Datasets that are linearly separable with some noise soft margin work out great:



- But what are we going to do if the dataset is just too hard?

- How about… mapping data to a higher-dimensional space:



The "Kernel Trick"

- The linear classifier relies on dot product between vectors
  - $\mathbf{x}_i^T \cdot \mathbf{x}_j$
- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the dot product becomes:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$$

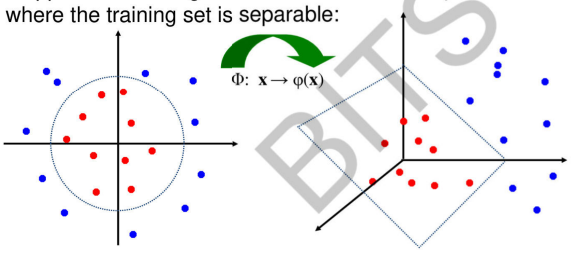
- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

SVM Kernels

- SVM algorithms use a set of mathematical functions that are defined as the kernel.
- Function of kernel is to take data as input and transform it into the required form.
- Different SVM algorithms use different types of kernel functions. Example *linear, nonlinear, polynomial, and sigmoid etc.*

Non-linear SVMs: Feature spaces

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



SVM – Overlapping Class Scenario

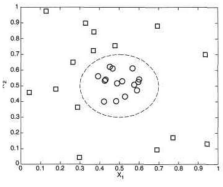
- Data is not separable linearly
- Margin will become inefficient
- Data needs to be transformed from original coordinate space **x** to a new space **Φ(x)**, so that linear decision boundary can be applied
- A non-linear transformation function is needed, like, ex:

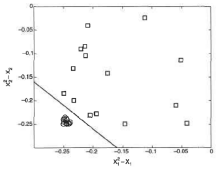
$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

- In the transformed space we can choose w = (w_0, w_1, … , w_4) such that

$$w_4 x_1^2 + w_3 x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

- The linear decision boundary in the transformed space has the following form: w. Φ(x) + b = 0





Non-linear SVMs Mathematically

- The solution is:

$$f(\mathbf{x}) = \Sigma \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

- Optimization techniques for finding α_i's remain the same!

Non-linear SVM using kernel

1. Select a kernel function.
2. Compute pairwise kernel values between labeled examples.
3. Use this "kernel matrix" to solve for SVM support vectors & alpha weights.
4. To classify a new example: compute kernel values between new input and support vectors, apply alpha weights, check sign of output.

Nonlinear SVM - Overview

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the dot product in the feature space.

XOR problem: Not linearly separable

For the XOR problem,

$\underline{x}_1$ $(-1, -1)$ $\longrightarrow$ -1 y_1

$\underline{x}_2$ $(-1, +1)$ $\longrightarrow$ $+1$ y_2

$\underline{x}_3$ $(+1, -1)$ $\longrightarrow$ $+1$ y_3

$\underline{x}_4$ $(+1, +1)$ $\longrightarrow$ -1 y_4

XOR problem: Choosing Kernel

Let us consider the XOR problem using SVMs (Cherkassy, 1998)

Start with a kernel

$$K(\underline{x}, \underline{x}_i) = (1 + \underline{x}^T \underline{x}_i)^2$$

$$\underline{x} = [x_1 \; x_2]^T \qquad \underline{x}_i = [x_{i1} \; x_{i2}]^T$$

$$K(\underline{x}, \underline{x}_i) = \left(1 + (x_1 \; x_2) \cdot \begin{bmatrix} x_{i1} \\ x_{i2} \end{bmatrix}\right)^2$$

$$= (1 + x_1 x_{i1} + x_2 x_{i2})^2$$

$$= 1 + x_1^2 x_{i1}^2 + 2 x_1 x_2 x_{i1} x_{i2} + x_2^2 x_{i2}^2 + 2 x_1 x_{i1} + 2 x_2 x_{i2}$$

Let us express $k(.,.) = \langle \phi(\underline{x}), \phi(\underline{x}_i) \rangle$

$$\phi(\underline{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1 x_2 \;\; x_2^2 \;\; \sqrt{2}\, x_1 \;\; \sqrt{2}\, x_2]^T$$

$$\phi(\underline{x}_i) = [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1} x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}\, x_{i1} \;\; \sqrt{2}\, x_{i2}]^T$$

For the XOR problem,

$(-1, -1)$ $\longrightarrow$ -1

$(-1, +1)$ $\longrightarrow$ $+1$

$(+1, -1)$ $\longrightarrow$ $+1$

$(+1, +1)$ $\longrightarrow$ -1

$K = [K(\underline{x}_i, \underline{x}_j)]$

$K(\underline{x}_i, \underline{x}_j) = \phi^T(\underline{x}_i)\phi(\underline{x}_j)$

Each $\underline{x}_i, \underline{x}_j$

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

Dual of SVM

From the dual problem, the objective $Q(\underline{\alpha})$

$$Q(\underline{\alpha}) = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 - \frac{1}{2}\left(9\alpha_1^2 - 2\alpha_1\alpha_2 - 2\alpha_1\alpha_3 + 2\alpha_1\alpha_4 + 9\alpha_2^2 + 2\alpha_2\alpha_3 - 2\alpha_2\alpha_4 + 9\alpha_3^2 - 2\alpha_3\alpha_4 + 9\alpha_4^2\right)$$

$$\sum_{i=1}^{4}\sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j K(\underline{x}_i, \underline{x}_j)$$

Solving for α

$$\frac{\partial Q(\underline{\alpha})}{\partial \alpha_i} = 0 \qquad i = 1, \cdots, 4$$

Doing this yields

$$\begin{cases} 9\alpha_1 - \alpha_2 - \alpha_3 + \alpha_4 = 1 \\ -\alpha_1 + 9\alpha_2 + \alpha_3 - \alpha_4 = 1 \\ -\alpha_1 + \alpha_2 + 9\alpha_3 - \alpha_4 = 1 \\ \alpha_1 - \alpha_2 - \alpha_3 + 9\alpha_4 = 1 \end{cases}$$

Solving this set of eqns

$$\alpha_{opt,i} = \frac{1}{8} \qquad i = 1, \cdots, 4$$

Now, all 4 inputs $\{\underline{x}_i\}_{i=1}^{4}$ are support vectors

$$Q(\alpha) = \frac{1}{4}$$

$$\frac{1}{2}\|\underline{w}_o\|^2 = \frac{1}{4} \Rightarrow \|\underline{w}_o\| = \frac{1}{\sqrt{2}}$$

$$\underline{w}_o = \sum_{i=1}^{4} \alpha_{o,i}\, y_i\, \phi(\underline{x}_i)$$

$$= \frac{1}{8}\left[-\phi(\underline{x}_1) + \phi(\underline{x}_2) + \phi(\underline{x}_3) - \phi(\underline{x}_4)\right]$$

$$\underline{w}_o = \left[0 \;\; 0 \;\; -\frac{1}{\sqrt{2}} \;\; 0 \;\; 0 \;\; 0\right]^T$$

For +ve support vectors we have

$$\sum_{i=1}^{4} \alpha_i y_i K(x_i, x_j) + b = +1 \qquad (x_j \text{ is +ve support vector})$$

Put $x_j = x_2$ or x_4

$\alpha_i = \frac{1}{8} \quad \forall i = 1$ to 4

$$K = \begin{bmatrix} 9 & 1 & 1 & 1 \\ 1 & 9 & 1 & 1 \\ 1 & 1 & 9 & 1 \\ 1 & 1 & 1 & 9 \end{bmatrix}$$

$$-\frac{K(x_1, x_2)}{8} + \frac{K(x_2, x_2)}{8} + \frac{K(x_3, x_2)}{8} - \frac{K(x_4, x_2)}{8} + b = 1$$

$$-\frac{1}{8} + \frac{9}{8} + \frac{1}{8} - \frac{1}{8} + b = 1$$

$$\boxed{b = 0}$$

Decision boundary of non linear SVM

$$\boxed{w^T \phi(x) + b = 0}$$

The opt. hyperplane is given by

$$\underline{w}_o^T \phi(\underline{x}) = 0$$

$$\begin{bmatrix} 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ \sqrt{2}x_1x_2 \\ x_2^2 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \end{bmatrix} = 0$$

$\Rightarrow x_1 x_2 = 0$ is the decision boundary

Non Linear Decision boundary

$$y = -x_1 x_2$$

x		y
−1	−1	−1
−1	+1	+1
+1	−1	+1
+1	+1	−1