



# Lecture 1

Math Foundations Team



# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

# 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  $R^n$

- ▶ Let's say we have a bunch of mathematical objects and we perform some operations on them. Do we get back similar objects?
- ▶ This leads to the idea of a vector space which underlies much of machine learning.

- ▶ 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.

# Motivating problem



Consider the following problem A company produces products  $N_1, N_2, \dots, N_n$  for which resources  $R_1, R_2, \dots, R_m$  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  $x_j$  units of product  $N_j$  are produced if a total  $b_i$  units of resource  $R_i$  are available, and no resources are left over.

If we produce  $x_1, x_2, \dots, x_n$  units of the products  $N_1, N_2, \dots, N_n$  we need a total of  $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n$  units of resource  $R_i$ . Thus we set up the equation:

$$a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n = b_i$$

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

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n = b_m$$

# Does a linear system always have solutions?



- ▶ A linear system has zero, one or infinitely many solutions
- ▶ Linear regression, a Machine Learning technique, provides an approximate solution to an overconstrained linear system, i.e one with no solution

## Example



Consider the following system of linear equations

$$x_1 + x_2 + x_3 = 3$$

$$x_1 - x_2 + 2x_3 = 2$$

$$2x_1 + 3x_3 = 1$$

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

---

Now consider another modification to the original set of equations

$$x_1 + x_2 + x_3 = 3$$

$$x_1 - x_2 + 2x_3 = 2$$

$$2x_1 + 3x_3 = 5$$

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  $2x_1 = 5 - 3x_3$ .

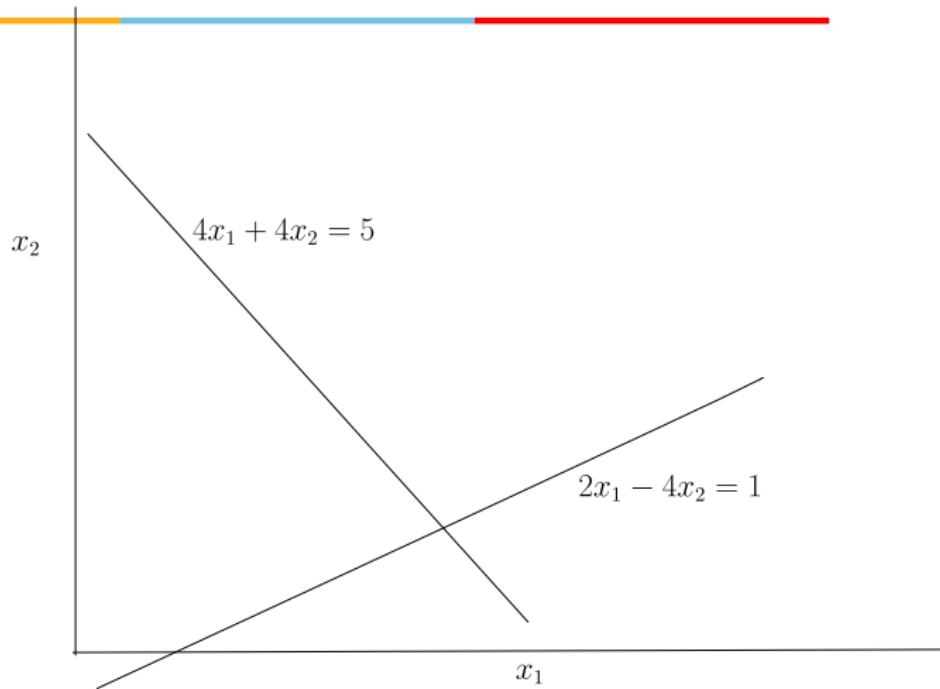
- ▶ Subtracting equation (2) from (1) we get  $2x_2 - x_3 = 1$ , so we can write

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

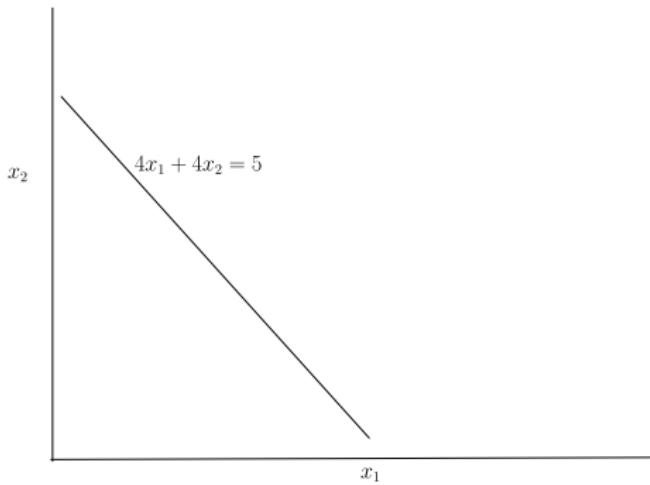
- ▶ For the previous problem we can express the set of infinite solutions in terms of the free variable  $x_3$ .
- ▶ Once  $x_3$  is fixed, the other two variables have to take on specific values - they are known as pivot variables.
- ▶ We will show later how to identify pivot and free variables using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

The solution to linear equations can be given a geometrical interpretation. When equations are given in terms of two variables, the solution to two equations in two variables could be a point (unique solution), a line (infinite solutions) or no solution (parallel lines). The first case is shown in the next slide.

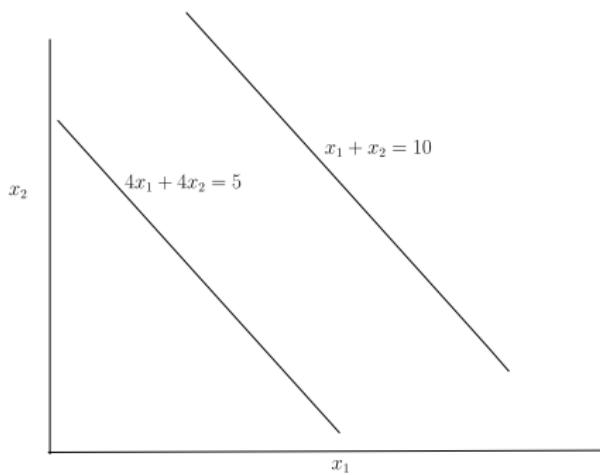
# Geometrical Interpretation



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



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:



- ▶ 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 3 gives parallel line) which means there is no solution.
- ▶ All three constraints or planes may intersect in the same line which means infinite solutions.

---

An  $(m, n)$  matrix  $A$  is a  $mn$  tuple of elements  $a_{ij}$  arranged in  $m$  rows and  $n$  columns as below:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

Note that row and column vectors are also matrices - a row vector is a  $(1, n)$  matrix and a column vector is a  $(m, 1)$  matrix.  $R^{m \times n}$  is the set of all  $(m, n)$  matrices consisting of real numbers as elements

# Matrix addition and multiplication



The sum of two matrices  $A \in R^{m \times n}$ ,  $B \in R^{m \times n}$  is defined as an element-wise sum of the elements of the two matrices:

$$A = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \dots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \dots & a_{2n} + b_{2n} \\ \vdots & & & \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \dots & a_{mn} + b_{mn} \end{bmatrix}$$

- ▶ The product of two matrices  $A \in R^{m \times k}$ ,  $B \in R^{k \times n}$  is defined as the  $m \times n$  matrix  $C = AB$  where the elements  $c_{ij}$  are calculated as follows:  $c_{ij} = \sum_{l=1}^{l=k} a_{il} b_{lj}$
- ▶ Essentially we are multiplying the elements of the  $i$ th row of  $A$  with the  $j$ th column of  $B$ .
- ▶ The product  $BA$  is not defined if  $n \neq m$

- ▶ Associativity:  
 $\forall A \in R^{m \times n}, B \in R^{n \times p}, C \in R^{p \times k} (AB)C = A(BC)$
- ▶ Distributivity:  $\forall A, B \in R^{m \times n}$ ,  
 $\forall C, D \in R^{n \times p}, (A + B)C = AC + BC, A(C + D) = AC + AD$
- ▶ Multiplication with the identity matrix:  $\forall A \in R^{m \times n}$ ,  
 $I_m A = A I_n = A$  where  $I_m$  is the  $m \times m$  identity matrix and  $I_n$  is the  $n \times n$  matrix.

Consider a square matrix  $A \in R^{n \times n}$ . Let matrix  $B \in R^{n \times n}$  exist such that  $AB = I_n$ . Does every matrix  $A \in R^{n \times n}$  possess an inverse?

Let us take a simple  $2 \times 2$  example → under what circumstances does it possess an inverse?

## $2 \times 2$ case



Define a  $2 \times 2$  matrix A as follows:

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

Define matrix B to be

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

The product  $AB$  is

$$AB = \begin{bmatrix} a_{11}a_{22} - a_{12}a_{21} & a_{12}a_{11} - a_{11}a_{12} \\ -a_{21}a_{22} - a_{22}a_{21} & -a_{21}a_{12} + a_{11}a_{22} \end{bmatrix}$$

## $2 \times 2$ case



$$AB = \begin{bmatrix} a_{11}a_{22} - a_{12}a_{21} & 0 \\ 0 & a_{11}a_{22} - a_{12}a_{21} \end{bmatrix} = (a_{11}a_{22} - a_{12}a_{21})I_2$$

$AB = (a_{11}a_{22} - a_{12}a_{21})I_2$ . We can define

$$A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

whenever  $a_{11}a_{22} - a_{12}a_{21} \neq 0$ .

Bottomline is that the inverse of a square matrix exists if and only if its determinant is non-zero.

For  $A \in R^{m \times n}$ , the matrix  $B \in R^{n \times m}$  with  $b_{ij} = a_{ji}$  is called the transpose of  $A$ . We say  $B = A^T$ . Transposition means writing the rows of one matrix as the columns of the other. The following properties can be shown:

$$AA^{-1} = A^{-1}A = I$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

$$(A + B)^{-1} \neq B^{-1} + A^{-1}$$

$$(A^T)^T = A$$

$$(A + B)^T = A^T + B^T$$

$$(AB)^T = B^T A^T$$

# Proof of $(AB)^T = B^T A^T$

---



- ▶ How do we show these properties? Let us look at the last one for example ...
- ▶ The  $(i, j)$ th entry of  $AB$  is obtained by taking the inner product of the  $i$ th row of  $A$  with the  $j$ th column of  $B$ .
- ▶ But the  $i$ th row of  $A$  is the  $i$ th column of  $A^T$ , and the  $j$ th column of  $B$  is the  $j$ th row of  $B^T$ .
- ▶ Thus we can take the inner product of the  $j$ th row of  $B^T$  with the  $i$ th column of  $A^T$  to get the same value for the  $(i, j)$ th entry of  $AB$ . Thus when we compute  $C = B^T A^T$ , we find that  $C_{ji} = (AB)_{ij}$ , so  $C = (AB)^T$ .

# Multiplication by a scalar



- ▶ Let  $A \in R^{m \times n}$  and  $\lambda$  be a scalar. Then  $\lambda A = B$  such that  $B_{ij} = \lambda A_{ij}$ , i.e every element in  $A$  is scaled by  $\lambda$  to get the corresponding element in the scaled matrix  $B$ .
- ▶ For  $\lambda, \psi \in R$  we have
  - ▶ associativity:  $(\lambda\psi)A = \lambda(\psi A)$ ,  $A \in R^{m \times n}$  and  $\lambda(AB) = (\lambda A)B = A(\lambda B)$  where  $A \in R^{m \times n}$  and  $B \in R^{n \times k}$ .
- ▶ distributivity:
  - ▶  $(\lambda A)^T = A^T \lambda^T = A^T \lambda = \lambda A^T$  since  $\lambda = \lambda^T$  for all  $\lambda \in R$ .
  - ▶  $(\lambda + \psi)A = \lambda A + \psi A$
  - ▶  $\lambda(A + B) = \lambda A + \lambda B$

Consider the following system of equations:

$$2x_1 + 3x_2 + 5x_3 = 1$$

$$4x_1 - 2x_2 - 7x_3 = 8$$

$$9x_1 + 5x_2 - 3x_3 = 2$$

In matrix terms we can write this set of equations as

$$\begin{bmatrix} 2 & 3 & 5 \\ 4 & -2 & -7 \\ 9 & 5 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \\ 2 \end{bmatrix}$$

# Solving a system of equations



- ▶ From the previous slide we see that a system of equations can be expressed as  $Ax = b$  where  $A \in R^{m \times n}$ ,  $x$  is a  $n \times 1$  matrix and  $b$  is a  $m \times 1$  matrix.
- ▶ We now look at how to obtain a particular and general solution for a system of equations
- ▶ Let us first look at an example.

# 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  $i$ th 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 **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

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

## 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  $\mathbf{A}$  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

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

# 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 → 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 → we call the procedure Gaussian elimination

- ▶ 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  $\lambda \in R \setminus \{0\}$
- ▶ Adding a row to another row
- ▶ Question → why must any multiplier to a row be non-zero?

# Example to illustrate elementary operations



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

$$-2x_1 + 4x_2 - 2x_3 - x_4 + 4x_5 = -3$$

$$4x_1 - 8x_2 + 3x_3 - 3x_4 + x_5 = 2$$

$$x_1 - 2x_2 + x_3 - x_4 + x_5 = 0$$

$$x_1 - 2x_2 - 3x_4 + 4x_5 = a$$

# 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



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

The notation above is used to convey that we could 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



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

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



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

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.

## Row-echelon form



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

We can revert to the set of equations represented by the augmented matrix as follows. These equations are equivalent to the original set of equations:

$$x_1 - 2x_2 + x_3 - x_4 + x_5 = 0$$

$$x_3 - x_4 + 3x_5 = -2$$

$$x_4 - 2x_5 = 1$$

$$0 = a + 1$$

- ▶ The preceding set of equations cannot be solved when  $a \neq -1$ .
- ▶ The last equation is consistent only for  $a = -1$ .
- ▶ A particular solution is

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}$$

# General Solution



$$x \in R^5 : x = \begin{bmatrix} 2 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix} + \lambda_1 \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \lambda_2 \begin{bmatrix} 2 \\ 0 \\ -2 \\ 1 \\ 1 \end{bmatrix}, \lambda_1, \lambda_2 \in R$$

- ▶ All rows that contain only zeros are at the bottom of the matrix.
- ▶ All rows that contain at least one nonzero element are on top of rows that contain only zeros.
- ▶ Considering only the non-zero rows, the first non-zero element in a given row is called the pivot and is always to the right of the pivot in the row above it.
- ▶ The positions of the pivots in the non-zero rows give rise to a staircase pattern.
- ▶ The variables corresponding to the pivot variables are called basic variables and those corresponding to the non-pivot positions correspond to the free variables.

## 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  $\lambda_3$ . We can see that the third equation establishes  $\lambda_3 = 1$ .
- ▶ The second equation involves only  $\lambda_2$  and  $\lambda_3$ . Plugging the just discovered value of  $\lambda_3$  into the second equation, we can find  $\lambda_2 = -1$ .
- ▶ Now we can plug the values of  $\lambda_2, \lambda_3$  into the first equation to get  $\lambda_1 = 2$

- ▶ We can convert the row-echelon form into a simpler form called the reduced row-echelon form.
- ▶ In reduced row-echelon form, every pivot is equal to 1.
- ▶ The pivot is the only non-zero entry in its column
- ▶ Therefore the pivot columns look like canonical basis vectors of  $R^m$  where the original given matrix  $A$  is a  $R^{m \times n}$  matrix.

## 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:

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

- ▶ 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

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

- ▶ 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 Gaussian Elimination

- ▶ 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 process is shown diagrammatically 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.

# Gaussian elimination diagram



$$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \xrightarrow{\quad} \begin{bmatrix} a & b & c \\ 0 & e' & f' \\ 0 & h' & i' \end{bmatrix} \xrightarrow{\quad} \begin{bmatrix} a & b & c \\ 0 & e' & f' \\ 0 & 0 & i'' \end{bmatrix} \xrightarrow{\quad} \begin{bmatrix} a & 0 & c' \\ 0 & e' & f' \\ 0 & 0 & i'' \end{bmatrix} \downarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \leftarrow \begin{bmatrix} a & 0 & 0 \\ 0 & e' & 0 \\ 0 & 0 & i'' \end{bmatrix}$$

- ▶ Can the Gaussian elimination 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 Gaussian elimination. 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  $i$ th canonical basis vector and find their solutions  $\mathbf{x}$ . Each solution vector constitutes a column in  $A^{-1}$ . Why is this true?

- ▶ Consider the linear system  $\mathbf{A}\mathbf{x} = \mathbf{e}_i$ .
- ▶ Gaussian elimination will convert this system to the equivalent system  $\mathbf{I}_n\mathbf{x} = \mathbf{c}_i$  whose solution is  $\mathbf{x} = \mathbf{c}_i$ .
- ▶ On the other hand, the solution to  $\mathbf{A}\mathbf{x} = \mathbf{e}_i$  is  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{e}_i$ .
- ▶ Since the two systems are equivalent they have the same solution, so  $\mathbf{x} = \mathbf{c}_i = \mathbf{A}^{-1}\mathbf{e}_i$  which means  $\mathbf{c}_i$  is the  $i$ th column of  $\mathbf{A}^{-1}$ .
- ▶ Thus when we create the augmented matrix  $[\mathbf{A}\mathbf{e}_i]$ , Gaussian elimination will convert it into  $[\mathbf{I}_n\mathbf{c}_i]$ .
- ▶ We can solve  $n$  linear systems at once by letting the augmented matrix be  $[ \mathbf{A} \quad | \quad \mathbf{I}_n ]$  which will become  $[\mathbf{I}_n\mathbf{A}^{-1}]$ .



## Lecture 2

Math Foundations Team



**BITS** Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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?

- ▶ 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



- ▶  $(\mathbb{Z}, +)$  is an Abelian group. We can see that all the aforementioned properties hold.
- ▶ What about  $(\mathbb{N}_0, +)$ ? This is not a group since there is no inverse element for an arbitrary element in it.
- ▶  $(\mathbb{Z}, \cdot)$  where  $\mathbb{Z}$  is the set of integers, and  $\cdot$  is product. It has the identity element but there exist elements in it that don't have an inverse.

- ▶ 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  $\cdot$  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  $V = (\mathcal{V}, +, \cdot)$  with operations  $+, \cdot$  such that  $+ : \mathcal{V} \times \mathcal{V} \rightarrow \mathcal{V}$  and  $\cdot : R \times \mathcal{V} \rightarrow \mathcal{V}$  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.

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  $[0, 0, \dots, 0]^T$  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:

$$\mathbf{x} + \mathbf{y} = (x_1, x_2, \dots, x_n) + (y_1, y_2, \dots, y_n) = \\ (x_1 + y_1, x_2 + y_2, \dots, x_n + y_n)$$

- ▶ Multiplication by scalars:

$$\lambda \mathbf{x} = \lambda(x_1, x_2, \dots, x_n) = (\lambda x_1, \lambda x_2, \dots, \lambda x_n), \forall \lambda \in \mathbb{R}, \mathbf{x} \in \mathbb{R}^n.$$

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  $\mathcal{V} = R^{m \text{ times } n}$  with addition and multiplication defined as below:

- ▶ Addition  $\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} + b_{11} & \dots & a_{1n} + b_{1n} \\ & \vdots & \\ a_{n1} + b_{n1} & \dots & a_{nn} + b_{nn} \end{bmatrix}$  is defined element-wise for two matrices  $A, B \in \mathcal{V}$ .

- ▶ Multiplication by scalars:  $\lambda \mathbf{A} = \begin{bmatrix} \lambda a_{11} & \dots & \lambda a_{1n} \\ & \vdots & \\ \lambda a_{n1} & \dots & \lambda a_{nn} \end{bmatrix}$

- ▶ Let  $V = (\mathcal{V}, +, \cdot)$  be a vector space and let  $\mathcal{U} \subseteq \mathcal{V}, \mathcal{U} \neq \Phi$ . then  $U = (\mathcal{U}, +, \cdot)$  is called a vector subspace of  $V$  if  $U$  is a vector space with the vector space operations  $+$  and  $\cdot$  restricted to  $\mathcal{U} \times \mathcal{U}$  and  $\mathbb{R} \times \mathcal{U}$ .
- ▶ 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.

How do we show if  $(\mathcal{U}, +, \cdot)$  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  $\forall \lambda \in R, \forall \mathbf{x}, \mathbf{y} \in \mathcal{U}, \lambda \mathbf{x} \in \mathcal{U}$  and  $\mathbf{x} + \mathbf{y} \in \mathcal{U}$ .

## 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.

- ▶ What about the subset of  $R^2$  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?

- ▶ 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 → can we come up with a set of vectors such that every vector in the vector space can be represented as a sum of ~~one~~ 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.

- ▶ Consider a vector space  $V$  and a set of vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k \in V$ . Then every  $\mathbf{v} \in V$  that is of the form  $\mathbf{v} = \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 \dots + \lambda_k \mathbf{x}_k$  is a linear combination of the vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$ .
- ▶ 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.

- ▶ Consider a vector space  $V$  and  $k$  vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k \in V$ . 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  $\lambda_i$ 's is non-zero, then the vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$  are linearly dependent.
- ▶ Put another way, if the only way we can combine vectors to get the **0**-vector is by letting all the  $\lambda_i$  be zero, then the vectors are linearly independent.

- 
- ▶ 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.

- ▶  $k$  vectors in a vector space are either linearly independent or linearly dependent. There is no third option.
- ▶ If at least one of the vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$  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  $\lambda$  for the 0-vector and zero  $\lambda$ s for all the other vectors to get the linear combination to be zero.
- ▶ If all the vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k$  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.

- ▶ 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?

- ▶ Consider a vector space  $V = (\mathcal{V}, +, \cdot)$  and a set of vectors  $\mathcal{A} = \{x_1, x_2, \dots, \cancel{x}_k\} \subseteq V$ .
- ▶ If every vector  $v \in V$  can be expressed as a linear combination of the vectors  $x_1, x_2, \dots, \cancel{x}_k$ , 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  $V = \text{span}[\mathcal{A}]$ .

# Basis as smallest generating set



- ▶ Consider a vector space  $V = (\mathcal{V}, +, \cdot)$  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  $V$ .

# Basis is not unique



- ▶ Consider the space  $R^3$ . 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  $R^3$  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  $R^3$  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  $R^4$

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

- ▶ Why is the above not a basis?

There are three steps to finding a basis for a vector space. A basis of a subspace  $U = \text{span}(x_1, \dots, x_m) \subseteq R^n$  can be found by executing the following steps:

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

## Example

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

$$x_1 = \begin{bmatrix} 1 \\ 2 \\ -1 \\ -1 \\ -1 \end{bmatrix}, x_2 = \begin{bmatrix} 2 \\ -1 \\ 1 \\ 2 \\ -2 \end{bmatrix}, x_3 = \begin{bmatrix} 3 \\ -4 \\ 3 \\ 5 \\ -3 \end{bmatrix}, x_4 = \begin{bmatrix} -1 \\ 8 \\ -5 \\ -6 \\ 1 \end{bmatrix}$$

- ▶ We would like to check if the vectors  $x_1, x_2, x_3, x_4$  constitute a basis for the subspace  $U$ .
- ▶ We therefore need to check whether the given vectors are linearly independent or not.

## Example



- ▶ We need to check if the only way to get  $\sum_{i=1}^{i=4} \lambda_i \mathbf{x}_i = 0$  is to set  $\lambda_i = 0$ .
- ▶ We set up the matrix  $\mathbf{A}\boldsymbol{\lambda} = 0$  where

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 & -1 \\ 2 & -1 & -4 & 8 \\ -1 & 1 & 3 & -5 \\ -1 & 2 & 5 & -6 \\ -1 & -2 & -3 & 1 \end{bmatrix}$$

- ▶ Now we perform a series of row transformations to turn this matrix into its row-echelon form to discover its pivot columns.

## Example



- We get the following row-echelon form:

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

- From the above the pivot columns are columns 1, 2 and 4. Column 3 can be expressed as  $2 \times$  column 2 -  $1 \times$  column 1, so it is not linearly independent of the other columns.

- ▶ The number of linearly independent rows in a matrix is equal to the number of linearly independent columns and is known as the rank of the matrix.
- ▶ Rank of a matrix  $\mathbf{A}$  is denoted as  $rk(\mathbf{A})$ .
- ▶ The columns of  $\mathbf{A} \in R^{m \times n}$  span a subspace  $U \subseteq R^m$  with  $dim(U) = rk(\mathbf{A})$ .
- ▶ The rows of  $\mathbf{A} \in R^{m \times n}$  span a subspace  $W \subseteq R^n$  with  $dim(W) = rk(\mathbf{A})$ .

- ▶ For all  $\mathbf{A} \in R^{n \times n}$ ,  $\mathbf{A}$  is invertible if and only if  $rk(\mathbf{A}) = n$ .
- ▶ For all  $\mathbf{A} \in R^{m \times n}$ , the linear system  $\mathbf{Ax} = \mathbf{b}$  is solvable if and only if  $rk(\mathbf{A}) = rk(\mathbf{A}|\mathbf{b})$ , where  $\mathbf{A}|\mathbf{b}$  is the augmented matrix.
- ▶ For  $\mathbf{A} \in R^{m \times n}$ , the solution space  $\mathbf{Ax} = \mathbf{0}$  has dimension  $n - rk(\mathbf{A})$ . This is called the kernel or nullspace.

- ▶ Let the number of linearly independent rows in the  $m \times n$  matrix  $A$  be  $r$ , so that  $\text{rowrank}(A) = r$ .
- ▶ This means  $A = BC$  where  $B$  is  $m \times r$  and  $C = r \times n$  since we can express each row of  $A$  as a linear combination of  $r$  independent vectors. Here the basis vectors of the rows of  $A$  are stored in the rows of  $C$  and the combining coefficients for these rows are stored in the rows of  $B$ . To get the  $i$ th row of  $A$  we use combining coefficients from the  $i$ th row of  $B$  and row vectors from the rows of  $C$ .

column

- ▶ We can interpret  $\mathbf{A} = \mathbf{BC}$  in a different way, where the *i*th of  $\mathbf{A}$  is obtained by combining the columns of  $\mathbf{B}$  with combining coefficients coming from the *i*th column of  $\mathbf{C}$ . This means that the columns of  $\mathbf{B}$  form a generating set for the column space of  $\mathbf{A}$ . Therefore the dimension of this column space, or the column rank of  $\mathbf{A}$  should be less than the number of columns of  $\mathbf{B}$  which is  $r$ . Therefore we have  $\text{colrank}(\mathbf{A}) \leq r$ . But  $r = \text{rowrank}(\mathbf{A})$ , so we have  $\text{colrank}(\mathbf{A}) \leq \text{rowrank}(\mathbf{A})$ .
- ▶ But if we were using the matrix  $\mathbf{A}^T$ , instead of  $\mathbf{A}$ , we would have  $\text{colrank}(\mathbf{A}^T) \leq \text{rowrank}(\mathbf{A}^T)$ . Since the rows of  $\mathbf{A}$  are the columns of  $\mathbf{A}^T$  and vice-versa we have  $\text{rowrank}(\mathbf{A}) \leq \text{colrank}(\mathbf{A})$ . Combining the two inequalities, we have  $\text{rowrank}(\mathbf{A}) = \text{colrank}(\mathbf{A})$ .



## Lecture 3

Math Foundations Team



# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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.

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

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

- ▶ A positive-definite, symmetric bilinear mapping  $\Omega : V \times V \rightarrow \mathbb{R}$  is called an inner product. To denote an inner product on  $V$  we generally write  $\langle \mathbf{x}, \mathbf{y} \rangle$ .
- ▶ The pair  $(V, \langle \cdot, \cdot \rangle)$  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  $\mathbf{x}$  can be written as linear combination of the basis vectors. We use this to express an inner product in terms of a matrix.

---

**Theorem:** For a real-valued, finite-dimensional vector space  $V$  and an ordered basis  $B$  of  $V$ , it holds that  $\langle \cdot, \cdot \rangle : V \times V \rightarrow \mathbb{R}$  is an inner product if and only if there exists a symmetric, positive definite matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  with  $\langle \mathbf{x}, \mathbf{y} \rangle = \hat{\mathbf{x}}^T \mathbf{A} \hat{\mathbf{y}}$ .

**Proof:** One direction  $\rightarrow$ :  $\langle \cdot, \cdot \rangle$  is an inner product  $\implies \mathbf{A}$  is symmetric, positive-definite such that  $\langle \mathbf{x}, \mathbf{y} \rangle = \hat{\mathbf{x}}^T \mathbf{A} \hat{\mathbf{y}}$ .

Other direction  $\leftarrow$ :  $\mathbf{A}$  is symmetric, positive definite such that the operation  $\langle \mathbf{x}, \mathbf{y} \rangle$  is defined as  $\langle \mathbf{x}, \mathbf{y} \rangle = \hat{\mathbf{x}}^T \mathbf{A} \hat{\mathbf{y}}$   $\implies$  the operation defined is an inner product.

- ▶ We prove the  $\rightarrow$  direction.
- ▶ Let  $\langle \mathbf{x}, \mathbf{y} \rangle$  be the inner product between the vectors  $\mathbf{x}, \mathbf{y}$  in  $V$ . We can write  $\mathbf{x}$  in terms of say  $n$  basis vectors as

$$\mathbf{x} = \sum_{i=1}^{i=n} \psi_i \mathbf{b}_i. \text{ Similarly } \mathbf{y} = \sum_{i=1}^{i=n} \lambda_i \mathbf{b}_i.$$

- ▶ Since the inner product is bilinear we can write

$$\langle \mathbf{x}, \mathbf{y} \rangle = \left\langle \sum_{i=1}^{i=n} \psi_i \mathbf{b}_i, \sum_{i=1}^{i=n} \lambda_i \mathbf{b}_i \right\rangle = \sum_{i=1}^{i=n} \sum_{j=1}^{j=n} \psi_i \langle \mathbf{b}_i, \mathbf{b}_j \rangle \lambda_j = \hat{\mathbf{x}}^T \mathbf{A} \hat{\mathbf{y}}$$

where  $A_{ij} = \langle \mathbf{b}_i, \mathbf{b}_j \rangle$ .

- ▶ Here  $\hat{\mathbf{x}}, \hat{\mathbf{y}}$  are vectors which represent the coordinates of the original vectors  $\mathbf{x}, \mathbf{y}$  with respect to the basis vectors.

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

- ▶ Now let us consider an operation op such that  $x \text{op} y = \hat{x}^T A \hat{y}$  where  $A$  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  $x \text{op} y = \hat{x}^T A \hat{y}$  and  $y \text{op} x = \hat{y}^T A \hat{x} = \hat{y}^T (A \hat{x})$ . By a property of the dot product we can write  $\hat{y}^T (A \hat{x}) = (A \hat{x})^T \hat{y} = \hat{x}^T A^T \hat{y} = \hat{x}^T A \hat{y}$  where the last equality in the chain is possible since  $A$  is symmetric.
  - ▶ "op" also has bilinearity since we see that for  $r \in R$ ,  $(rx) \text{op} y = (r\hat{x})^T A \hat{y} = r \hat{x}^T A \hat{y} = r x \text{op} y$ .
  - ▶  $(x + y) \text{op} z = (\hat{x} + \hat{y})^T A \hat{z} = \hat{x}^T A \hat{z} + \hat{y}^T A \hat{z} = x \text{op} z + y \text{op} z$ .
  - ▶ Finally if  $x$  is a non-zero vector then  $\hat{x}$  is also a non-zero vector,  $x \text{op} x = \hat{x}^T A \hat{x} > 0$  since we are given that  $A$  is positive-definite.

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

- ▶ Inner products and norms are closely related in the sense that any inner product induces a norm:  $\|x\| = \sqrt{\langle x, x \rangle}$
- ▶ 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:  $\langle x, y \rangle \leq \|x\| \|y\|$ . Why is this true?

# Cauchy-Schwarz inequality



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

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

A metric  $d$  has the following properties:

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

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

- ▶ In addition to being able to capture the lengths of vectors and the distance between vectors, inner products can also capture the **angle**  $\omega$  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  $x$  and  $y$  are not the  $\mathbf{0}$  vector. Then the Cauchy-Schwarz inequality tells us that

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

- 
- ▶ 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

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

- ▶ The angle  $\omega$  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 the same direction and  $\omega \approx 0$ .

- ▶ Food for thought: Suppose we choose vectors  $x$  and  $y$  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 ...

- ▶ A key feature of the inner product is that we can use it to characterize vectors that are orthogonal.
- ▶ Two vectors  $x$  and  $y$  are orthogonal if and only if the inner product between them is 0. For an orthogonal pair of vectors  $x, y$  we can write  $x \perp y$ .
- ▶ By the above definition the **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  $\mathbf{x} = [1, 1]^T$  and  $\mathbf{y} = [-1, 1]^T$
- ▶ With respect to the inner product defined as a dot product we see that  $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = 1 * -1 + 1 * 1 = 0$ .
- ▶ With respect to the inner product  $\mathbf{x}^T \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{y}$ , the angle between the two vectors  $\mathbf{x}$  and  $\mathbf{y}$  becomes

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

## Example - angles and orthogonality



- ▶ Continuing with our example we have

$$\begin{aligned}\cos(\omega) &= \frac{\mathbf{x}^T \mathbf{A} \mathbf{y}}{\sqrt{\mathbf{x}^T \mathbf{A} \mathbf{x}} \sqrt{\mathbf{y}^T \mathbf{A} \mathbf{y}}} \\ &= \frac{2x_1y_1 + x_2y_2}{\sqrt{(2x_1^2 + x_2^2)(2y_1^2 + y_2^2)}} \\ &= \frac{-1}{3}\end{aligned}$$

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

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

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

$$\begin{aligned}\mathbf{A}^T \mathbf{A} &= \mathbf{I} = \mathbf{A} \mathbf{A}^T \\ \mathbf{A}^T &= \mathbf{A}^{-1}\end{aligned}$$

- ▶ 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  $\mathbf{A}$  be a square matrix with  $\mathbf{B}$  and  $\mathbf{C}$  the left and right inverses of  $\mathbf{A}$ :  $\mathbf{B}\mathbf{A} = \mathbf{I} = \mathbf{A}\mathbf{C} \implies \mathbf{B} = \mathbf{C}$ . Why is this true?

- ▶ Transformations by an orthonormal matrix preserve lengths.  
This can be seen as follows, using the dot product as the definition of the inner product:  
$$\|\mathbf{A}\mathbf{x}\|^2 = \mathbf{A}\mathbf{x}^T \mathbf{A}\mathbf{x} = \mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x} = \mathbf{x}^T \mathbf{I} \mathbf{x} = \mathbf{x}^T \mathbf{x}.$$
- ▶ 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  $\theta$  is the angle of rotation.

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

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

- ▶ 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  $n$ th 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.

- ▶ 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  $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$ . If it is true that  $\forall i, j = 1, \dots, n, i \neq j \langle \mathbf{b}_i, \mathbf{b}_j \rangle = 0$  and  $\langle \mathbf{b}_i, \mathbf{b}_i \rangle = 1$ , 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.

- ▶ 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  $v_1 = (3, 1)^T$  and  $v_2 = (2, 2)^T$ . Put these vectors into columns of a matrix  $A$  such that  $A = \begin{bmatrix} 3 & 2 \\ 1 & 2 \end{bmatrix}$ .
- ▶ The next step is to perform Gaussian elimination on the following augmented matrix:  $[A^T A | A^T] = \left[ \begin{array}{cc|cc} 10 & 8 & 3 & 1 \\ 8 & 8 & 2 & 2 \end{array} \right]$
- ▶ On performing Gaussian elimination of this augmented matrix we end up with  $\left[ \begin{array}{cc|cc} 1 & 0.8 & 0.3 & 0.1 \\ 0 & 1 & -0.25 & 0.75 \end{array} \right]$

- ▶ 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.
- ▶ What is the justification for this technique?
- ▶ First we see that when the  $m \times n$  matrix  $\mathbf{A}$  has full column rank, then the matrix  $\mathbf{A}^T \mathbf{A}$  is positive definite. To see this note that any solution  $x$  to  $\mathbf{Ax} = \mathbf{0}$  is also a solution to  $\mathbf{A}^T \mathbf{Ax} = \mathbf{0}$  and vice-versa. Why is this the case?
- ▶ When  $\mathbf{A}$  is full rank, there are no non-trivial solutions to  $\mathbf{Ax} = \mathbf{0}$ . Thus the fact that there are no non-trivial solutions to  $\mathbf{Ax} = \mathbf{0}$  means that  $\forall x \in \mathbb{R}^m, x \neq \mathbf{0}, x^T \mathbf{Ax} > 0$ .

- ▶ One of the steps of Gaussian elimination is 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.

This can be accomplished by the following elementary matrix

$$E = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ so that the product}$$

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

- ▶ A series of Gaussian elimination steps can be represented as a product of elementary transformations acting on  $\mathbf{A}$ :  
$$\mathbf{E}_m \mathbf{E}_{m-1} \dots \mathbf{E}_1 \mathbf{A}.$$
- ▶ 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  $\mathbf{L}^{-1} \mathbf{A} = \mathbf{U}$  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  $\mathbf{U}$  is an upper triangular matrix. Thus we have  
$$\mathbf{A} = \mathbf{L} \mathbf{U}.$$

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



## Lecture 4

Math Foundations Team



# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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.

A determinant of order  $n \times n$  is a scalar associated with a  $n \times n$

matrix and is denoted as follows:  $\det(\mathbf{A}) = \begin{vmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{vmatrix}$

We have a cofactor formula to calculate a determinant of order  $n$ :  
for  $n = 1$ , we have  $\det(\mathbf{A}) = a_{11}$ . For  $n \geq 2$  we have

$$\begin{aligned} D = \det(\mathbf{A}) &= a_{j1} C_{j1} + a_{j2} C_{j2} + \dots + a_{jn} C_{jn} \\ &= a_{1k} C_{1k} + a_{2k} C_{2k} + \dots + a_{nk} C_{nk} \end{aligned}$$

The first line above represents expansion along the  $j$ th row, while  
the second line represents expansion along the  $k$ th column.

- ▶ In the preceding slide, the  $C_{jk} = (-1)^{j+k} M_{jk}$  where  $M_{jk}$  represents the  $n - 1$  order determinant of the submatrix of  $\mathbf{A}$  obtained by removing the  $j$ th row and  $k$ th column.
- ▶  $M_{jk}$  is called the minor of  $a_{jk}$  in  $D$  and  $C_{jk}$  is called the cofactor of  $a_{jk}$  in  $D$ .
- ▶ Our definition for the determinant in the previous slide shows that the  $n \times n$  determinant is defined in terms of  $(n - 1) \times (n - 1)$  determinants which in turn are defined in terms of  $(n - 2) \times (n - 2)$  determinants recursively.
- ▶ Let us examine the computation for a simple  $3 \times 3$  determinant.

## Example

Let us compute  $D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$

For the entries in the second row the minors are  $M_{21} = \begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix}$ ,

$M_{22} = \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}$ ,  $M_{23} = \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix}$ . The cofactors are

$C_{21} = (-1)^{1+2} M_{21} = -M_{21}$ ,  $C_{22} = (-1)^{2+2} M_{22} = M_{22}$ ,

$C_{23} = (-1)^{2+3} M_{23} = -M_{23}$ .

Expanding along the second row we can write

$$D = a_{21} C_{21} + a_{22} C_{22} + a_{23} C_{23}.$$

**Theorem:** We can state the following for a  $n$ th order determinant under elementary row operations: (a) interchanging two rows multiplies the value of the determinant by  $-1$ , and (b) adding a multiple of a row to another row does not change the value of the determinant.

**Proof Sketch** Let us look at how to prove (a). The proof is by induction. The statement holds for  $n = 2$  since  $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$  whereas  $\begin{vmatrix} c & d \\ a & b \end{vmatrix} = bc - ad = -(ad - bc)$ . We make the induction hypothesis that the statement is true for all determinants of order  $(n - 1)$ . Let  $D$  represent the original determinant and  $E$  represent the determinant with rows interchanged.

Let us expand  $D$  and  $E$  along a **row that is not interchanged**.  
We have

$$D = \sum_{k=1}^{k=n} (-1)^{j+k} a_{jk} M_{jk}$$
$$E = \sum_{k=1}^{k=n} a_{jk} (-1)^{j+k} N_{jk}$$

where  $N_{jk}$  in  $E$  is obtained by exchanging two rows of the minor  $M_{jk}$  in  $D$ .  $M_{jk}$  and  $N_{jk}$  are determinants of order  $n - 1$  where one of the determinants has a pair of rows interchanged as compared to the other determinant. Therefore  $M_{jk} = -N_{jk}$ , and  $D = -E$ .

Let us now look at adding multiples of a row to another row.

- ▶ Add  $c$  times row  $i$  to row  $j$ . Then we get a new determinant  $D'$  whose entries in the  $j$ th row has  $a_{jk} + ca_{ik}$ . Expanding the determinant  $D'$  we have 
$$D' = \sum_{k=1}^{k=n} (a_{jk} + ca_{ik})(-1)^{j+k} M_{jk} = \sum_{k=1}^{k=n} (-1)^{j+k} a_{jk} M_{jk} + \sum_{k=1}^{k=n} (-1)^{j+k} ca_{ik} M_{jk}.$$
- ▶ The summation can be written as  $D' = D_1 + cD_2$  where  $D_1 = D$  and  $D_2$  represents the determinant of a matrix similar to the one we started out with except that rows  $j$  and  $i$  both have coefficients  $a_{ik}$  in them. Thus two rows of  $D_2$  are equal - this makes  $D_2 = 0 \rightarrow$  why is this?

## Proof sketch continued



- ▶ In part (a) we showed that interchanging two rows will negate the determinant. If we interchange rows  $i$  and  $j$  we will get the same determinant since the two rows are identical. But one of the determinants must be the negative of the other. This is only possible when both determinants are zero. Thus  $D_2 = 0$  and  $D' = D$ .
- ▶ **Bottom line** → Adding a multiple of one row to another row does not change the determinant.
- ▶ This will lead us to our next result.

---

**Theorem** An  $n \times n$  matrix  $A$  has rank  $n$  if and only if its determinant is not equal to zero.

**Proof sketch:**  $A$  has full rank  $\implies \det(A) \neq 0$ : Gaussian elimination reduces  $A$  to upper triangular matrix  $U = (U_{ij})$  whose determinant is the product of all the elements  $U_{ii}$ . But we know that  $\det(A) = (-1)^s \det(U)$  where  $s$  is the number of row interchanges performed during Gaussian elimination. Since  $A$  has full rank, the columns of  $A$  are linearly independent and the only solution to  $Ax = \mathbf{0}$  is  $x = \mathbf{0}$ . The system  $Ax = \mathbf{0}$  has the same set of solutions as  $Ux = \mathbf{0}$ , so this means  $U$  has only pivot columns. The pivots are all the elements  $U_{ii}$ . The product of all pivots is non-zero, and hence  $\det(U) = \det(A) \neq 0$ .

$\det(\mathbf{A}) \neq 0 \implies \mathbf{A}$  has full rank: If the determinant of  $\mathbf{A}$  is non-zero,  $\det(\mathbf{A}) = (-1)^s \det(\mathbf{U}) = \prod_{i=1}^{i=n} U_{ii}$  means that all the  $U_{ii}$  are non-zero. Therefore all the columns of  $\mathbf{U}$  are pivot columns with the pivots being the  $U_{ii}$ . The pivot columns are all linearly independent, so the only solution to  $\mathbf{Ux} = \mathbf{0}$  is  $x = \mathbf{0}$ . This is also the only solution to  $\mathbf{Ax} = \mathbf{0}$  which means  $\mathbf{A}$  has full rank.

- ▶ The trace of a  $n \times n$  square matrix  $\mathbf{A}$  is defined as  
 $tr(\mathbf{A}) = \sum_{i=1}^{i=n} a_{ii}$ , i.e the sum of the diagonal elements of  $A$ .
- ▶ The trace has the following properties:
  - ▶  $tr(\mathbf{A} + \mathbf{B}) = tr(\mathbf{A}) + tr(\mathbf{B})$ , for  $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$
  - ▶  $tr(\alpha \mathbf{A}) = \alpha tr(\mathbf{A})$ ,  $\alpha \in \mathbb{R}$
  - ▶  $tr(\mathbf{I}_n) = n$
  - ▶  $tr(\mathbf{AB}) = tr(\mathbf{BA})$  for  $\mathbf{A} \in \mathbb{R}^{n \times k}$ ,  $\mathbf{B} \in \mathbb{R}^{k \times n}$ .
- ▶ The proofs of these properties are not difficult.

# Characteristic polynomial



- ▶ For  $\lambda \in \mathbb{R}$  and  $\mathbf{A} \in \mathbb{R}^{n \times n}$  we can define  $p_{\mathbf{A}}(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I})$  and show that it can be written as  $c_0 + c_1 \lambda + \dots + c_{n-1} \lambda^n + (-1)^n \lambda^n$  where  $c_0, c_1 \dots c_{n-1} \in \mathbb{R}$ .
- ▶ We can show that  $c_0 = \det(\mathbf{A})$  and  $c_{n-1} = (-1)^{n-1} \text{tr}(\mathbf{A})$
- ▶ To see that  $c_0 = \det(\mathbf{A})$ , set  $\lambda = 0$  in  $\det(\mathbf{A} - \lambda \mathbf{I})$  to get  $p_{\mathbf{A}}(0) = \det(\mathbf{A}) = c_0$
- ▶ The formula for  $c_{n-1}$  takes a little bit of work - let us expand a

$$3 \times 3 \text{ determinant } \det(\mathbf{A} - \lambda \mathbf{I}) = \begin{vmatrix} a_{11} - \lambda & a_{12} & a_{13} \\ a_{21} & a_{22} - \lambda & a_{23} \\ a_{31} & a_{32} & a_{33} - \lambda \end{vmatrix}$$

- ▶ 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  $\lambda^3$  and  $\lambda^2$ . The other contributors to the determinant i.e  $a_{12}C_{12}$  and  $a_{13}C_{13}$  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  $\lambda^{n-2}$  as the minors for each such contributor will kill off a term containing  $\lambda$  in a given row and column.

# Characteristic polynomial



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

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

# Eigenvalues and eigenvectors - example



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

- ▶ 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  $\mathbf{A} - \lambda\mathbf{I}$  for the different roots  $\lambda$ .
- ▶ 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  $\mathbf{Ax} = \lambda x$  it seems that the action of  $\mathbf{A}$  on  $x$  is to preserve the direction of  $x$  but scale it up or down according to  $\lambda$ . Does this mean that a rotation matrix has no eigenvalues and eigenvectors?

- ▶  $\lambda$  is an eigenvalue of  $\mathbf{A}$  if and only if  $\lambda$  is a root of the characteristic polynomial  $p_{\mathbf{A}}(\lambda)$  of  $\mathbf{A}$ . This can be easily seen as a consequence of the definition of  $p_{\mathbf{A}}(\lambda)$ .
- ▶ For  $\mathbf{A} \in \mathbb{R}^{n \times n}$ , the set of eigenvectors corresponding to an eigenvalue  $\lambda$  spans a subspace of  $\mathbb{R}^n$  called the Eigenspace of  $\mathbf{A}$  with respect to  $\lambda$  and is denoted by  $E_{\lambda}$ .
- ▶ The set of all eigenvalues of  $\mathbf{A}$  is called the spectrum of  $\mathbf{A}$ .
- ▶ Look at the eigenvalues and eigenspace of the  $n \times n$  identity matrix  $\mathbf{I}_n$ . 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  $\det(\mathbf{A}) = \det(\mathbf{A}^T)$ . Then  $\det(\mathbf{A} - \lambda\mathbf{I}) = \det((\mathbf{A} - \lambda\mathbf{I})^T) = \det(\mathbf{A}^T - \lambda\mathbf{I}^T) = \det(\mathbf{A}^T - \lambda\mathbf{I})$ . The last expression in the chain of equalities is the characteristic polynomial for  $p_{\mathbf{A}^T}(\lambda)$ . Thus we have  $p_{\mathbf{A}}(\lambda) = p_{\mathbf{A}^T}(\lambda)$  which means the characteristic polynomials are equal and so the roots of the polynomials or the eigenvalues must be equal.
- ▶ The eigenspace  $E_\lambda$  is the nullspace of  $\mathbf{A} - \lambda\mathbf{I}$ .
- ▶ Symmetric, positive-definite matrices always have positive, real eigenvalues.

- ▶ The eigenvectors  $x_1, x_2 \dots x_n$  of a  $n \times n$  matrix  $\mathbf{A}$  with  $n$  **distinct** eigenvalues are linearly independent → why?
- ▶ Given a matrix  $\mathbf{A} \in \mathbb{R}^{m \times n}$  we can show that  $\mathbf{A}^T \mathbf{A} \in \mathbb{R}^{n \times n}$  is a symmetric, positive-definite matrix when the rank of  $\mathbf{A} = n$ . Why is this true? Clearly  $\mathbf{A}^T \mathbf{A}$  is a symmetric matrix and it is positive definite since  $x^T \mathbf{A}^T \mathbf{A} x = \|\mathbf{A}x\|^2 > 0 \quad \forall x \in \mathbb{R}^n \setminus 0$  since the nullspaces of  $\mathbf{A}^T \mathbf{A}$  and  $\mathbf{A}$  are the same, and  $\mathbf{A}$  is a full column rank matrix.
- ▶ The matrix  $\mathbf{A}^T \mathbf{A}$  is important in machine learning since it figures in the least-squares solution to a data matrix represented as  $\mathbf{A}$  where  $n$  represents the number of features and  $m$  is the number of data vectors.

---

**Theorem:** If  $A \in \mathbb{R}^{n \times n}$  is symmetric there exists an orthonormal basis of the corresponding vector space  $V$  consisting of the eigenvectors of  $A$ , 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.

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

- ▶ In the old formulation with real vectors, length-squared according to the Euclidean norm was  $x_1^2 + x_2^2 + \dots + x_n^2$ . If the  $x_i$  are complex we should take length-squared to be  $|x_1|^2 + |x_2|^2 + \dots + |x_n|^2$  where  $|\cdot|$  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  $\|x\|^2 = x^T x$ . 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  $x = \begin{bmatrix} 1+i \\ 2+i \end{bmatrix}$ . We have  
$$x^T x = (1+i)^2 + (2+i)^2 = 1 + 2i + i^2 + 4 + 4i + i^2 = 6i + 3.$$

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

We shall now show that all eigenvalues for a symmetric matrix are real. Let  $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ . Then premultiplying with  $\mathbf{x}^H$  on both sides we have  $\mathbf{x}^H\mathbf{A}\mathbf{x} = \lambda\mathbf{x}^H\mathbf{x}$

Now  $\mathbf{x}^H\mathbf{A}\mathbf{x}$  is a  $1 \times 1$  matrix. Taking the Hermitian of this matrix we have  $(\mathbf{x}^H\mathbf{A}\mathbf{x})^H = \mathbf{x}^H\mathbf{A}^H\mathbf{x} = \mathbf{x}^H\mathbf{A}\mathbf{x}$ , so the Hermitian of the matrix is itself which means that the matrix is real.

On the right hand side we note that  $\mathbf{x}^H\mathbf{x}$  is real, so this means that  $\lambda$  must be real.

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

$$\begin{aligned}\mathbf{y}^H \mathbf{A} \mathbf{x} &= \lambda \mathbf{y}^H \mathbf{x} \\ \mathbf{x}^H \mathbf{A} \mathbf{y} &= \mu \mathbf{x}^H \mathbf{y}\end{aligned}$$

But  $\mathbf{x}^H \mathbf{A} \mathbf{y} = (\mathbf{y}^H \mathbf{A}^H \mathbf{x})^H = (\mathbf{y}^H \mathbf{A} \mathbf{x})^H = \lambda \mathbf{x}^H \mathbf{y}$ . We already know that  $\mathbf{x}^H \mathbf{A} \mathbf{y} = \mu \mathbf{x}^H \mathbf{y}$ . This means  $\lambda \mathbf{x}^H \mathbf{y} = \mu \mathbf{x}^H \mathbf{y}$ . Since  $\lambda \neq \mu$ , this must mean  $\mathbf{x}^H \mathbf{y} = 0$ .

This shows that eigenvectors corresponding to different eigenvalues are orthogonal.

- ▶ 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 eigenvalues?

- ▶ 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  $\mathbf{A}$  as  $\mathbf{A} = \mathbf{Q}\Lambda\mathbf{Q}^T$  where  $\mathbf{Q}$  is an orthonormal basis (think orthonormal basis vectors for example), and  $\Lambda$  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  $\mathbf{A}$  and create a symmetric matrix out of it -  $\mathbf{A}^T\mathbf{A}$  and  $\mathbf{A}\mathbf{A}^T$  which are both used in Singular-Value Decomposition and PCA.

- ▶ 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  $p_A(\lambda)$  can be written as  $\prod_{i=1}^{i=n} (\lambda_i - \lambda)$ . The coefficient to  $\lambda^{n-1}$  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  $\lambda^{n-1}$  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  $\det(A) = \prod_{i=1}^{i=n} \lambda_i$ . To see why this is true, let us once again look at the factorisation of  $p_A(\lambda)$  as  $\det(A - \lambda I) = p_A(\lambda) = \prod_{i=1}^{i=n} (\lambda_i - \lambda)$ . Setting  $\lambda = 0$  in this equation gives the result.

**Theorem** A symmetric, positive-definite matrix  $A$  can be factorized into a product  $A = LL^T$  where  $L$  is a lower-triangular matrix with positive elements.

For an example  $3 \times 3$  matrix we can write

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{bmatrix}$$

We can solve for the elements of the lower triangular matrix to get

$$l_{11} = \sqrt{a_{11}}, \quad l_{22} = \sqrt{a_{22} - l_{21}^2}, \quad l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)}.$$

For the elements below the diagonal we have  $l_{21} = \frac{a_{21}}{l_{11}}$ ,  $l_{31} = \frac{a_{31}}{l_{11}}$  and  $l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}}$ .

# An application of Cholesky decomposition



- ▶ In the Data Science / Machine Learning context, distributions on data are frequently multivariate Gaussian.
- ▶ Multivariate Gaussian distributions are governed by a covariance matrix which is symmetric, positive-definite.
- ▶ We may need to draw samples from such distributions which is where the Cholesky decomposition finds an important application.
- ▶ To generate a sample from a multivariate Gaussian distribution, we factor the covariance matrix into its Cholesky factor  $L$ , generate a Gaussian random vector  $x$  on independent variables which is easy to do, and compute  $Lx$  which will be a sample according to the required distribution.



## Lecture 5

Math Foundations Team



# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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 decomposition. 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.

- ▶ 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  $\mathcal{D} = P^{-1}AP$
- ▶ In the definition of diagonalization, it is required that  $P$  is an invertible matrix. Assume  $p_1, p_2, \dots, p_n$  are the  $n$  columns of  $P$
- ▶ Rewriting we get  $AP = PD$ . By observing that  $\mathcal{D}$  is a diagonal matrix, we can simplify as

$$Ap_i = \lambda_i p_i$$

where  $\lambda_i$  is the  $i^{\text{th}}$  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  $\mathbf{P}^{-1}\mathbf{A}\mathbf{P}$  as follows

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

- ▶ 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

$$\mathbf{A} = \mathbf{P}\mathbf{D}\mathbf{P}^{-1}$$

- ▶ 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  $\det(\mathbf{A} - \lambda \mathbf{I}) = 0$
- ▶ 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 corresponding to  $\lambda_1 = 3.5$  is derived as

$$p_1 = \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}$$

- ▶ The eigenvector corresponding 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 **P** to diagonalize **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

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

- ▶ The eigendecomposition of the matrix  $A$  is given by

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

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

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

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

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

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

- ▶ It can be observed that

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

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

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

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

# Construction of U



- ▶ It can be observed that

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

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

$$\mathbf{A}\mathbf{A}^T = \mathbf{S}\mathbf{D}\mathbf{S}^T$$

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

## Construction of $U$ continued



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

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

- ▶ This is equivalent to the following

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

# 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

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

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

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

Now we construct the singular value matrix  $\Sigma$

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

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

- ▶ The last column is a column of zeros only

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

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

- ▶ The second vector

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

## Final Step : Combining $U$ , $\Sigma$ and $V$



We compile all the three matrices together to generate the SVD



$$A = \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{5}{\sqrt{30}} & 0 & \frac{-1}{\sqrt{6}} \\ \frac{-2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{6}} \end{bmatrix}^T$$

- ▶ The matrix  $\mathbf{U}$  is an  $2 \times 2$  matrix satisfying orthogonality property.
- ▶ The matrix  $\mathbf{V}$  is an  $3 \times 3$  matrix satisfying orthogonality property.

# Comparing SVD and EVD



- ▶ The left-singular vectors of  $\mathcal{A}$  are eigenvectors of  $\mathbf{A}\mathbf{A}^T$
- ▶ 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  $\Sigma$  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  $\mathcal{A}^T\mathcal{A}$  .

- ▶ We considered the SVD as a way to factorize  $\mathcal{A} = \mathbf{U}\Sigma\mathbf{V}^T$  into the product of three matrices, where  $\mathbf{U}$  and  $\mathbf{V}$  are orthogonal and  $\Sigma$  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.

- ▶ 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  $\Sigma$  multiplies only matching left and right singular vectors  $u_i v_i^T$  and scales them by the corresponding singular value  $\sigma_i$ .
- ▶ All terms  $\sigma_i u_i v_i^T$  vanish for  $i \neq j$  because  $\Sigma$  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  $A_i$ ,  $i = 1, \dots, r$  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 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  $\Sigma$  as follows

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

- ▶ 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

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.

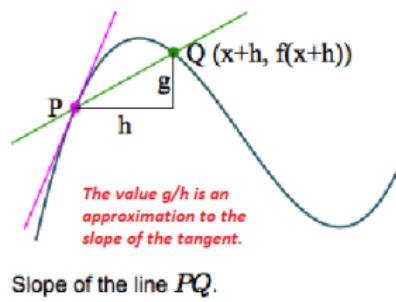
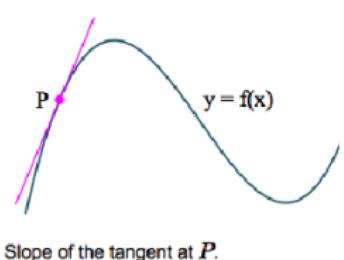
# Differentiation of Univariate Functions



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

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

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

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

# Derivative of a Polynomial



$$\begin{aligned}\frac{df}{dx} &= \lim_{h \rightarrow 0} \sum_{i=1}^n \binom{n}{i} x^{n-i} h^{i-1} \\ &= \lim_{h \rightarrow 0} \binom{n}{1} x^{n-1} + \lim_{h \rightarrow 0} \sum_{i=2}^n \binom{n}{i} x^{n-i} h^{i-1} \quad (3) \\ &= nx^{n-1}\end{aligned}$$

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  $f : \mathbb{R} \rightarrow \mathbb{R}$  at  $x_0$  is defined as

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

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

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

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

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  $k = 0, 1, 2, \dots, 6$

$f(1) = 1, f'(1) = 4, f''(1) = 12, f^{(3)}(1) = 24, f^{(4)}(1) = 24,$   
 $f^{(5)}(1) = 0, f^{(6)}(1) = 0$ . 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  $\pm 1$  (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

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

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:  $\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)g(x) - f(x)g'(x)}{(g(x))^2}$
- ▶ 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

$$f'(x) = 2$$

$$g'(f) = 4f^3$$

$$h'(x) = g'(f)f'(x) = (4f^3).2 = 8(2x + 1)^3$$

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.

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

$$\frac{\partial f}{\partial x_1} = \lim_{h \rightarrow 0} \frac{f(x_1 + h, x_2, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{h}$$

$$\frac{\partial f}{\partial x_2} = \lim_{h \rightarrow 0} \frac{f(x_1, x_2 + h, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{h}$$

⋮

$$\frac{\partial f}{\partial x_n} = \lim_{h \rightarrow 0} \frac{f(x_1, x_2, \dots, x_n + h) - f(x_1, x_2, \dots, x_n)}{h}$$

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

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

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

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

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

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

## Example 2



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

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

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

So the gradient is then

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

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.

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

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

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

Consider a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  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

$$\frac{df}{dt} = \left[ \frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \right] \begin{bmatrix} \frac{\partial x_1(t)}{\partial t} \\ \frac{\partial x_2(t)}{\partial t} \end{bmatrix} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \quad (17)$$

where  $d$  denotes the gradient and  $\partial$  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:

$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial s} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial s} \quad (18)$$

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial x_1} \frac{\partial x_1}{\partial t} + \frac{\partial f}{\partial x_2} \frac{\partial x_2}{\partial t} \quad (19)$$

and the gradient is obtained by the matrix multiplication

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

We have discussed partial derivatives and gradients of functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  mapping to the real numbers. Now we will generalize the concept of the gradient to vector-valued functions  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ , where  $n \geq 1$  and  $m > 1$ .

For a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  and a vector  $x = [x_1, \dots, x_n]^T$  corresponding vector of function values is given as

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

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

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

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

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  $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$  with respect to  $x \in \mathbb{R}^n$  by collecting these partial derivatives:

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

# 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  $f : \mathbb{R}^N \rightarrow \mathbb{R}^M$ , 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  $x_j$ :

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

We obtain the gradient using Jacobian

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

## Example 2: Gradients of Vector-Valued Functions



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

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

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

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

The desired gradient is computed by applying the chain rule:

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

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



## Lecture 7

Math Foundations Team



**BITS** Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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.

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  $mn \times pq$

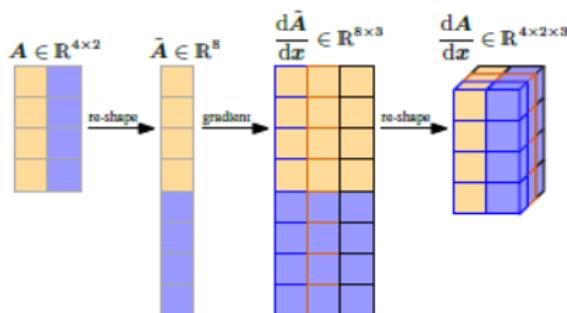
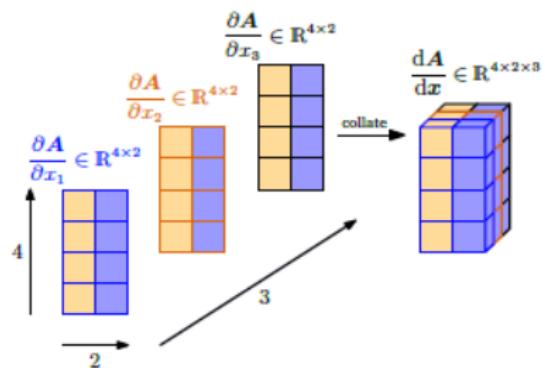
# Gradients of Matrices



$$A \in \mathbb{R}^{4 \times 2}$$
$$x \in \mathbb{R}^3$$

$$A \in \mathbb{R}^{4 \times 2}$$
$$x \in \mathbb{R}^3$$

Partial derivatives:



# 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

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

Now, we have

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

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

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

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

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

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

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

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

Clearly, we have

$$\partial_{pqij} = B_{iq} \quad \text{if } j = p, p \neq q$$

$$\partial_{pqij} = B_{ip} \quad \text{if } j = q, p \neq q$$

$$\partial_{pqij} = 2B_{iq} \quad \text{if } j = p, p = q$$

$$\partial_{pqij} = 0 \quad \text{otherwise}$$

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

# Useful Identities for Computing Gradients



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

# Useful Identities for Computing Gradients



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

for symmetric  $W$ .

Consider the function

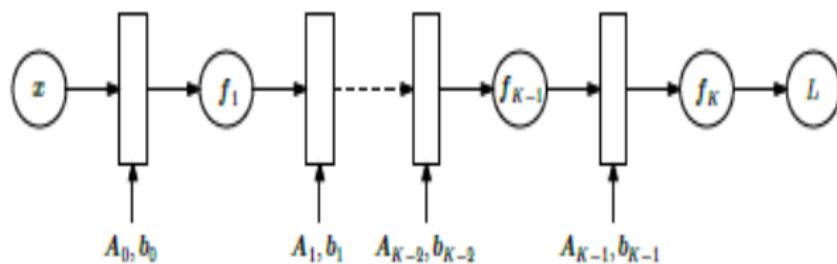
$$f(x) = \sqrt{x^2 + \exp(x^2)} + \cos(x^2 + \exp(x^2))$$

Taking derivatives

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

- ▶ 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.

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  $\sigma$  is an activation function.

---

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

$$f_0 := x$$

$$f_i := \sigma_i(A_{i-1}f_{i-1} + b_{i-1}), i = 1, \dots, K.$$

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

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

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

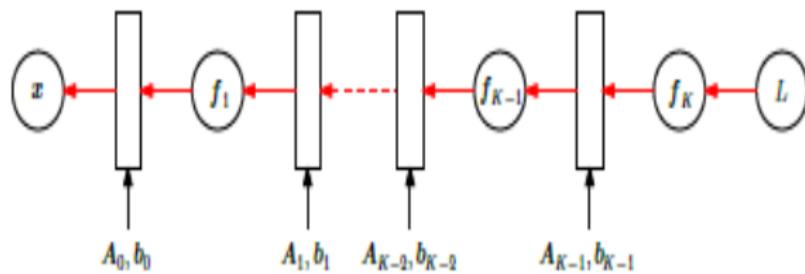
# Backpropagation



Using the chain rule, we get

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

# Backpropagation



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

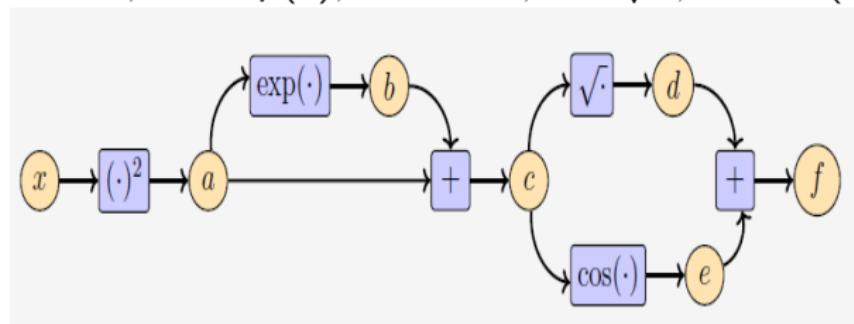
# Example

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

$$\frac{\partial f}{\partial c} = 1 \cdot \left( \frac{1}{2\sqrt{c}} + 1 \right) \cdot (-\sin(c))$$

$$\frac{\partial f}{\partial b} = \frac{\partial f}{\partial c} \cdot 1$$

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial b} \exp(a) + \frac{\partial f}{\partial c} \cdot 1$$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} 2x$$

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

Let  $x_1, \dots, x_d$  : input variables.

$x_{d+1}, \dots, x_{D-1}$  : intermediate variables.

$x_D$  : output variable, then we have,

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

Note that  $g_i$ s 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.

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.



## Lecture 8

Math Foundations Team



**BITS** Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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.

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

## 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.  $x$

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

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

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

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

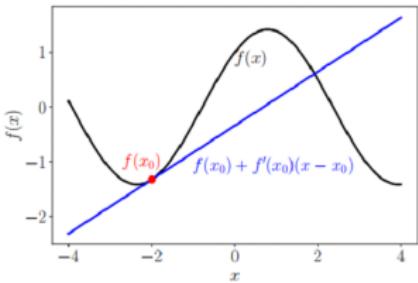
is symmetric. The Hessian is denoted as  $\nabla_{x,y}^2 f(x, y)$

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

$$f(x) \approx f(x_0) + (\nabla_x f)(x_0)(x - x_0) \quad (1)$$

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

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  $f : \mathbb{R}^D \rightarrow \mathbb{R}$ ,  $x \rightarrow f(x)$ ,

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

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

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

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

---

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

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

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

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

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

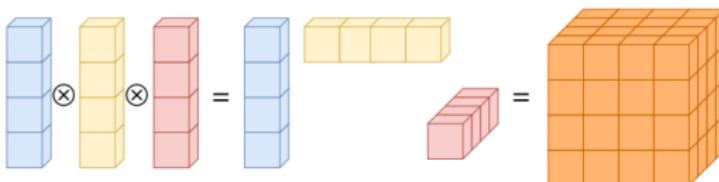
$k$ th-order tensor  $\delta^k \in \mathbb{R}^{\overbrace{D \times D \times \dots \times D}^{k \text{ times}}}$  is obtained as a  $k$ -fold outer product, denoted by  $\otimes$ , of the vector  $\delta \in \mathbb{R}^D$ . For example,

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

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



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



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

$$D_{\mathbf{x}}^k f(\mathbf{x}_0) \delta^k = \sum_{i_1=1}^D \cdots \sum_{i_k=1}^D D_{\mathbf{x}}^k f(\mathbf{x}_0)[i_1, \dots, i_k] \delta[i_1] \cdots \delta[i_k]$$

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

$k = 0, \dots, 3$  and  $\delta := \mathbf{x} - \mathbf{x}_0$ :

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

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

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

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

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

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

# Taylor Series Expansion of a Function with Two Variables

Innovate

achieve

lead

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...

Innovate

achieve

lead

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

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

Therefore, we obtain

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

such that

$$\frac{D_{x,y}^1 f(1, 2)}{1!} \delta = [6 \quad 14] \begin{bmatrix} x - 1 \\ y - 2 \end{bmatrix} = 6(x - 1) + 14(y - 2).$$

Therefore, we obtain

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

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

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

such that

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

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

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

The third-order derivatives are obtained as

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

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

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

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

Innovate

achieve

lead

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

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

and

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

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

Innovate

achieve

lead

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

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



## Lecture 9

Math Foundations Team



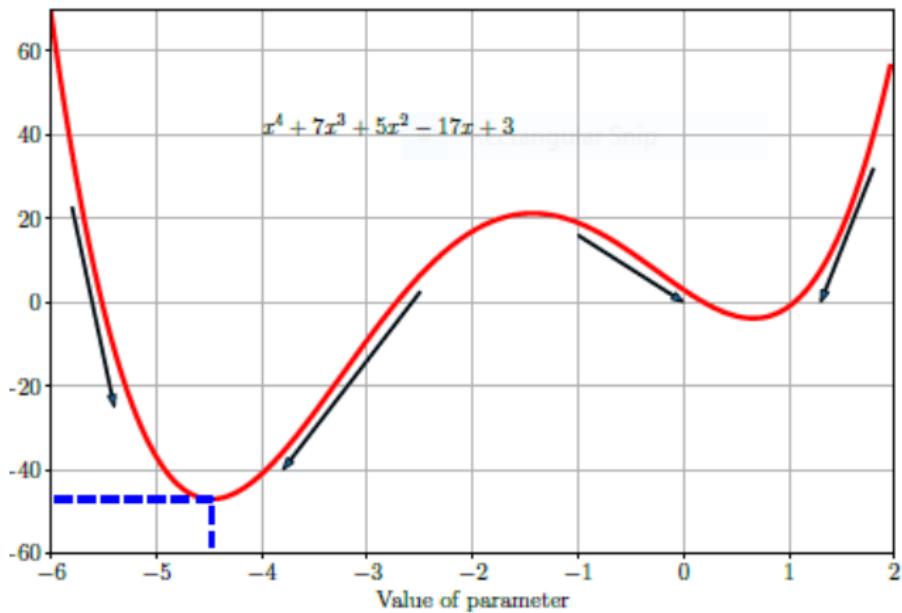
# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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

- ▶ 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.
- ▶ This constitutes a valley of the objective function, known as a local minimum
- ▶ For unconstrained optimization, this is enough for the big picture.

# Example



## Example



- ▶ Let  $I(x) = x^4 + 7x^3 + 5x^2 - 17x + 3$ .
- ▶ The gradient is  $\frac{dI(x)}{dx} = 4x^3 + 21x^2 + 10x - 17$ .
- ▶ 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.
- ▶ How do we check if we are dealing with a local minimum or local maximum? Look at the second derivative!
- ▶ If the second derivative at the point where the gradient vanishes is negative we are talking about a local maximum, otherwise it is a local minimum.
- ▶ The second derivative is  $12x^2 + 42x + 10$

- ▶ For low-order polynomials we can solve the equations analytically and find points at which the gradient is zero. Then we can do the second derivative test and identify whether these points are local minima/local maxima.
- ▶ In general we need to follow the negative gradient.
- ▶ Consider the problem of solving for the minimum of a real-valued function  $\min_x f(x)$  where  $f : R^d \rightarrow R$  is an objective loss function that captures the cost of using the current set of parameters.
- ▶ We assume our function  $f$  is differentiable but that the minimum cannot be found analytically in closed form.

- ▶ The main idea of gradient-descent, a first-order optimization algorithm, is to take a step from the current point of magnitude proportional to the negative gradient of the function at the current point.
- ▶ In mathematical terms if  $x_1 = x_0 - \gamma((\nabla f)(x_0))^T$  for a small step-size  $\gamma > 0$  then  $f(x_1) \leq f(x_0)$ .
- ▶ The above suggests that in order to find the optimum of  $f$ , say at  $f(x_*)$ , we can start at some initial point  $x_0$  and then iterate according to  $x_{i+1} = x_i - \gamma_i((\nabla f)(x_i))^T$
- ▶ For a suitable step-size  $\gamma_i$ , the sequence of points  $f(x_0) \geq f(x_1) \geq \dots$  converges to some local minimum.

# Example of gradient descent



- ▶ Let  $f = \frac{1}{2}x^T \mathbf{A}x + \mathbf{b}^T x$  where  $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ ,  $\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 20 \end{bmatrix}$  and  $\mathbf{b} = \begin{bmatrix} -5 \\ -3 \end{bmatrix}$ .
- ▶ From a previous lecture on Vector Calculus, we know that  $\nabla f = \mathbf{x}^T \mathbf{A} + \mathbf{b}^T$
- ▶ Starting at  $x_0 = \begin{bmatrix} -3 \\ -1 \end{bmatrix}$  and iterating according to the equation given on a previous slide, we will eventually encounter the minimum value.

# Choosing step-size



- ▶ If the step-size is too small, gradient descent will become too slow.
- ▶ If the step-size is too large, gradient descent might overshoot the target, fail to converge, or even diverge.
- ▶ One way to handle the issue of choosing step-sizes is to choose a different step-size at each step. If a given step-size leads to an increase in the value of the cost function, we have been too aggressive, so a better step-size would be half the current step-size. On the other hand, if the value of the function has decreased, the step-size could be larger. In each case we undo the current step and change the step-size.
- ▶ This heuristic guarantees monotonic convergence.

# 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.
  - ▶ In terms of equations we have
- $$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i ((\nabla f)(\mathbf{x}_i))^T + \alpha \Delta \mathbf{x}_i.$$

- ▶ The momentum term helps us deal with a noisy estimate of the current gradient as the memory term remembers the past direction and prevents the current gradient estimate from leading the optimization procedure in a completely wrong direction.
- ▶ Why do we need to work with noisy or approximate gradients? This is because computing the true gradient may be computationally very demanding.
- ▶ We can deal with an approximate gradient and then use the momentum term to smooth out the noise.
- ▶ Stochastic gradient descent is a stochastic way of approximating the gradient of an objective function which is expressed as a sum of differentiable functions.

- ▶ Let us consider a machine learning problem consisting of loss functions incurred at  $N$  data points. Let the loss function at the  $n$ th data point be  $L_n(\theta)$ , and let the total loss be  $L(\theta) = \sum_{n=1}^{n=N} L_n(\theta)$ . Here  $\theta$  is the parameter vector of interest and the goal of machine learning is to find the parameter vector  $\theta$  that minimizes the loss function.
- ▶ The standard gradient descent procedure is a batch optimization method in that the update step considers the gradient of the entire loss function  $L(\theta)$ ., i,e 
$$\theta_{i+1} = \theta_i - \gamma_i \nabla L(\theta_i)^T = \theta_i - \gamma_i \sum_{n=1}^{n=N} \nabla L_n(\theta_i)^T.$$
- ▶ This step is expensive since we may have many data points in the batch.

# Stochastic gradient descent



- ▶ We can perform a mini-batch gradient descent by taking only a small subset of the data points.
- ▶ In the extreme case we can choose only one  $L_n$  to estimate the gradient.
- ▶ Why does taking only a subset of data points work? It works because we get an unbiased estimate 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.
- ▶ Estimating the gradient out of large mini-batches will lead to lower variance in the estimate, in contrast to estimating the gradient from small batch sizes.

- ▶ Consider the following problem:  $\min_x f(x)$ ,  $f : \mathbb{R}^D \rightarrow \mathbb{R}$ , subject to additional constraints - so we are looking at a minimization problem except that the set of all  $x$  over which minimization is performed is not all of  $\mathbb{R}^D$ .
- ▶ The constrained problem becomes  $\min_x f(x)$  subject to  $g_i(x) \leq 0 \forall i 1, 2, \dots, m$ .
- ▶ Since we have a method of finding a solution to the unconstrained optimization problem, one way to proceed now is to convert the given constrained optimization problem into an unconstrained one.
- ▶ We construct  $J(x) = f(x) + \sum_{i=1}^{i=m} \mathbf{1}(g_i(x))$ , where  $\mathbf{1}(z) = 0$  for  $z \leq 0$  and  $\mathbf{1}(z) = \infty$  for  $z > 0$ .

- ▶ The formulation of  $J(\mathbf{x})$  in the previous slide ensures that its value is infinity if any one of the constraints  $g_i(\mathbf{x})$  is not satisfied. This ensures that the optimal solution to the unconstrained problem is the same as the constrained problem.
- ▶ The step-function is also difficult to optimize, and our solution is to replace the step-function by a linear function using Lagrange multipliers.
- ▶ We create the Lagrangian of the given constrained optimization problem as follows:  
$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{i=m} \lambda_i g_i(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}),$$
 where  $\lambda_i \geq 0$  for all  $i.$

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

# Minimax inequality



- ▶ Why is this inequality true?
- ▶ Assume that  $x, y: \max_y \min_x \phi(x, y) = \phi(x_A, y_A)$  and  $\min_x \max_y \phi(x, y) = \phi(x_B, y_B)$ .
- ▶ Fixing  $y$  at  $y_A$  we see that the inner operation on the left hand side of the minimax inequality is a min operation over  $x$  and returns  $x_A$ . Thus we have  $\phi(x_A, y_A) \leq \phi(x_B, y_A)$ .
- ▶ Fixing  $x$  at  $x_B$  we see that the inner operation on the right hand side of the minimax inequality is a max operation over  $y$  and returns  $y_B$ . Thus we have  $\phi(x_B, y_B) \geq \phi(x_B, y_A)$ .
- ▶ From the above we conclude that  $\phi(x_B, y_B) \geq \phi(x_A, y_A)$ .

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

# Minimax inequality



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

- ▶ In contrast to the original formulation  $\mathfrak{D}(\lambda) = \min_{x \in \mathbb{R}^d} \mathfrak{L}(x, \lambda)$  is an unconstrained optimization problem for a given value of  $\lambda$ .
- ▶ We observe that  $\mathfrak{D}(\lambda) = \min_{x \in \mathbb{R}^d} \mathfrak{L}(x, \lambda)$  is a point-wise minimum of affine functions and hence  $\mathfrak{D}(\lambda)$  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  $\min_x f(x)$  subject to  $g_i(x) \leq 0$  for all  $1 \leq i \leq m$  and  $h_j(x) = 0$  for  $1 \leq j \leq n$ .
- ▶ We model the equality constraint  $h_j(x) = 0$  with two inequality constraints  $h_j(x) \geq 0$  and  $h_j(x) \leq 0$ .
- ▶ 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.

- ▶ 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?

- ▶ 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  $f : \mathbb{R}^d \rightarrow \mathbb{R}$  be a function whose domain is a convex set  $C$ .
- ▶ The function is a convex function if for any  $\mathbf{x}, \mathbf{y} \in C$ ,  
$$f(\theta\mathbf{x} + (1 - \theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1 - \theta)f(\mathbf{y})$$
- ▶ Another way of looking at a convex function is to use the gradient: for any two points  $\mathbf{x}$  and  $\mathbf{y}$ , we have  
$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla_{\mathbf{x}}f(\mathbf{x})(\mathbf{y} - \mathbf{x}).$$

## Example



- ▶ The negative entropy, a useful function in Machine Learning, is convex:  $f(x) = x\log_2x$  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  
 $f(0.5 * 2 + 0.5 * 4) = f(3) = 3\log_23 \approx 4.75$ . Then  
 $\theta f(2) + (1 - \theta)f(4) = 0.5 * 2\log_22 + 0.5 * 4\log_24 = \log_232 = 5$ . Therefore the convexity criterion is satisfied for these two points.
- ▶ Let us now use the gradient criterion. We have  
 $\nabla f(x) = \log_2x + x \frac{1}{x\log_e2}$ . Calculating  $f(2) + \nabla f(2) * (4 - 2)$  gives  $2\log_22 + (\log_22 + \frac{1}{\log_e2}) * 2 \approx 6.9$ . We see that  $f(4) = 4\log_24 = 8$  which shows that the gradient criterion is also satisfied.

- ▶ 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  $\min_x \mathbf{c}^T \mathbf{x}$  subject to  $\mathbf{A}\mathbf{x} \leq \mathbf{b}$  where  $\mathbf{A} \in \mathbb{R}^{m \times d}$  and  $\mathbf{b} \in \mathbb{R}^{m \times 1}$ .
- ▶ The Lagrangian  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$  is given by  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{c}^T \mathbf{x} + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{x} - \mathbf{b})$  where  $\boldsymbol{\lambda} \in \mathbb{R}^m$  is the vector of non-negative Lagrangian multipliers.
- ▶ We can rewrite the Lagrangian as  $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = (\mathbf{c} + \mathbf{A}^T \boldsymbol{\lambda})^T \mathbf{x} - \boldsymbol{\lambda}^T \mathbf{b}$ .

- ▶ Taking the derivative of the Lagrangian with respect to  $x$  and setting it to zero we get  $\mathbf{c} + \mathbf{A}^T \boldsymbol{\lambda} = 0$ .
- ▶ Since  $\mathfrak{D}(\boldsymbol{\lambda}) = \min_{x \in \mathbb{R}^d} \mathcal{L}(x, \boldsymbol{\lambda})$ , plugging in the above equation gives  $\mathfrak{D}(\boldsymbol{\lambda}) = -\boldsymbol{\lambda}^T \mathbf{b}$ .
- ▶ 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:

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

- ▶ 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:

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

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

# Quadratic programming



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



## Lecture 10

Math Foundations Team



**BITS** Pilani

Pilani | Dubai | Goa | Hyderabad

- ▶ 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.
- ▶ How do we ensure correctness of the gradient computation?

- ▶ Let  $J(\mathbf{w}) = J(w_1, w_2, \dots, w_d)$ .
- ▶ We can compute the gradient of the function via a finite-difference approximation.
- ▶ We sample a few of the parameters  $w_1, w_2, \dots, w_d$  and compute the partial derivative  $\frac{\partial J}{\partial w_i}$  using the finite-difference approximation.
- ▶ We compute the partial derivative approximately as follows:

$$\frac{\partial J}{\partial w_i} = \frac{J(w_1, w_2, \dots, w_i + \Delta, \dots, w_d) - J(w_1, w_2, \dots, w_i, \dots, w_d)}{\Delta}$$

# Learning rate 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
- ▶ In this case, it will take a very long time for the algorithm to converge to the optimal point.

- ▶ The solution to the learning rate problem is to 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  $\mathbf{w}_{n+1} = \mathbf{w}_n - \alpha_t \nabla J$ .
- ▶ The  $t$  in the update step is usually measured in terms of the number of cycles over all the training points, so the learning rate remains a constant during a particular cycle.

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

exponential decay:  $\alpha_t = \alpha_0 e^{-kt}$

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.

- ▶ In the bold-driver algorithm, the learning rate changes based on whether the objective function is improving or worsening.
- ▶ The learning rate is increased by a factor of around 5% in each iteration as long as the steps improve the objective function.
- ▶ As soon as the objective function worsens in a given step, the step is undone and the learning rate is reduced by a factor of 50%, and the new learning rate is used in a fresh update step.
- ▶ This process is continued to convergence.
- ▶ When the objective function is estimated using samples of data, we can only get a noisy estimate of the gradient, and the bold driver algorithm may not work properly.
- ▶ In such cases we can test the objective function and adjust the learning rate after  $m$  steps, rather than a single step.

- ▶ 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  $J(\mathbf{w})$  be the function being optimized, and let  $\mathbf{g}_t$  be the descent direction at the beginning of the  $t$ th step with the parameter vector being  $\mathbf{w}_t$ .
- ▶ In the steepest-descent method, the direction  $\mathbf{g}_t$  is the same as  $-\nabla J(\mathbf{w}_t)$ .
- ▶ In light of the above the update step is  $\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t \mathbf{g}_t$ .

- ▶ In line search the learning rate  $\alpha_t$  is chosen at the  $t$ th step so as to minimize the value of the objective function at  $\mathbf{w}_{t+1}$ .
- ▶ Therefore the step-size  $\alpha_t$  is computed as  
$$\alpha_t = \min_{\alpha} J(\mathbf{w}_t + \alpha \mathbf{g}_t).$$
- ▶ After performing this step the gradient is computed at  $\mathbf{w}_{t+1}$ .
- ▶ The gradient at  $\mathbf{w}_{t+1}$  will be perpendicular to the search direction at  $\mathbf{g}_t$ . Otherwise  $\alpha_t$  will not be optimal and there will exist another point at which the objective function is minimized.

- ▶ To see this result, note that if the gradient at  $\mathbf{w}_{t+1}$  is not perpendicular to the search direction  $\mathbf{g}_t$ , we can improve the objective function by moving a further  $+\delta$  or  $-\delta$  along  $\mathbf{g}_t$  at  $\mathbf{w}_{t+1}$ .
- ▶ Using Taylor's series expansion we have
$$J(\mathbf{w}_t + \alpha_t \mathbf{g}_t \pm \delta \mathbf{g}_t) \approx J(\mathbf{w}_t + \alpha_t \mathbf{g}_t) \pm \delta \mathbf{g}_t^T \nabla J(\mathbf{w}_t + \alpha_t \mathbf{g}_t).$$
- ▶ If  $\mathbf{g}_t^T \nabla J(\mathbf{w}_t + \alpha_t \mathbf{g}_t) \neq 0$  then depending on its sign we can choose  $\delta$  to be positive or negative to ensure that
$$J(\mathbf{w}_t + \alpha_t \mathbf{g}_t + \delta \mathbf{g}_t) < J(\mathbf{w}_t + \alpha_t \mathbf{g}_t).$$

- ▶ One question remains - how do we perform the optimization  $\min_{\alpha} J(\mathbf{w}_t + \alpha \mathbf{g}_t)$ ?
- ▶ An important property that we exploit of typical line-search settings is that the objective function is a unimodal function of  $\alpha$ .
- ▶ 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  $[0, \alpha_{\max}]$  in which to perform the search for the optimum  $\alpha$ .

- ▶ We can sweep evaluate the objective function values at geometrically increasing values of  $\alpha$ .
- ▶ 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  $\alpha$ , and the last of the methods is an inexact method that does not rely on unimodality.
- ▶ The Armijo rule therefore has broader applicability than either the binary search or golden-section search methods.

- ▶ We start by initialising the search interval  $\alpha$  to  $[a, b] = [0, \alpha_{\max}]$ .
- ▶ In binary search over  $[a, b]$ , we evaluate the objective function at two points close to the midpoint of the interval, ie at  $\frac{a+b}{2}$  and  $\frac{a+b+\epsilon}{2}$  and find out whether the function is increasing or decreasing at  $\frac{a+b}{2}$ . Here  $\epsilon$  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.

- ▶ We initialise the search interval to  $[a, b] = [0, \alpha_{\max}]$ .
- ▶ The process of narrowing the search interval is different from the one adopted for binary search - here 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  $[a, m_1)$ .

- ▶ 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.

- ▶ The basic idea of the Armijo rule is that as one goes down the direction of improvement  $\mathbf{g}_t$  at the starting point  $\mathbf{w}_t$ , the rate of improvement of the objective function comes down.
- ▶ The rate of improvement of the objective function along the search direction at the starting point is  $\mathbf{g}_t^T \nabla F(\mathbf{w})$ .
- ▶ The typical improvement of the objective function for a particular  $\alpha$  can be expected to be  $\alpha \mathbf{g}_t^T \nabla F(\mathbf{w})$  for most real-world objective functions.
- ▶ The Armijo rule is satisfied for a fraction  $\mu \in (0, 0.5)$  of this improvement. A typical value of  $\mu$  is 0.25.
- ▶ We want to find the largest step-size  $\alpha$  such that  $F(\mathbf{w}_t) - F(\mathbf{w}_t + \alpha \mathbf{g}_t) \geq \mu \alpha \mathbf{g}_t^T \nabla F(\mathbf{w}_t)$ .

- ▶ At first sight, it seems that the Armijo rule is not saying much since for small enough values of  $\alpha$ , we can see that the finite-difference approximation ensures that the rule is true for  $\mu = 1.0$ .
- ▶ However we want a larger step-size to get faster convergence.
- ▶ What is the largest step-size we can use?
- ▶ We test successively decreasing values of  $\alpha$  and stop the first time the condition  $F(\mathbf{w}_t) - F(\mathbf{w}_t + \alpha \mathbf{g}_t) \geq \mu \alpha \mathbf{g}_t^T \nabla F(\mathbf{w}_t)$  is satisfied.
- ▶ In backtracking line search we start by checking  $H(\alpha_{\max})$ , then  $H(\beta \alpha_{\max}) \dots H(\beta^r \alpha_{\max})$ . We then use  $\alpha = \beta^r \alpha_{\max}$ . Note that a typical value of  $\beta$  is 0.25.

# 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.

- ▶ Gradient descent must start at some value of the parameters.  
How is an initial point chosen?
- ▶ For many applications, the vector components of the initial point can be chosen from  $[-1, +1]$ .
- ▶ In case the parameters are constrained to be non-negative, the components can be chosen in the range  $[0, 1]$ .
- ▶ What about more complex applications where the individual components are correlated with each other? In this case the choice of the initial point can be critical.
- ▶ It may also happen that choosing improper magnitudes of some components may cause overflow or underflow errors during updates.

- ▶ Most objective functions in machine learning penalize the deviation of a predicted value from an observed value in one form or the other.
- ▶ A common loss function is  $J(\mathbf{w}) = \sum_{i=1}^{i=n} \|\mathbf{w}^T \mathbf{X}_i - y_i\|^2$ . Here  $\mathbf{X}_i^T$  is the  $i$ th row vector in a matrix consisting of  $n$  row vectors where each row vector represents a training point, and  $y_i$  contains the real-valued observation of the  $i$ th training point.
- ▶ The above loss function occurs in least squared regression, and represents the sum of squared differences between the observed values  $y_i$  in the data and the predicted values  $\hat{y}_i = \mathbf{w}^T \mathbf{X}_i$ .

- ▶ Another form of penalization is the log-likelihood objective function.
- ▶ This form of objective function uses the probability that the model's prediction of a dependent variable matches the observed value in the data.
- ▶ Higher values of these probabilities are desirable and the model should learn parameters to maximize these probabilities.
- ▶ The model might output the probability of a particular class in a binary classification setting, and a good goal for the model would be to maximize the probability of predicting the correct class.

- ▶ Consider the pair  $(\mathbf{X}_i, y_i)$  where  $\mathbf{X}_i$  is the  $i$ th training point and  $y_i$  is the observed class. Let the probability of predicting class  $y_i$  for the training point  $\mathbf{X}_i^T$  given model parameters  $\mathbf{w}$  be  $P(\mathbf{X}_i, y_i, \mathbf{w})$ .
- ▶ The probability of correct prediction over all training pairs  $\mathbf{X}_i, y_i$  is the product of probabilities  $P(\mathbf{X}_i, y_i, \mathbf{w})$  over all  $\mathbf{X}_i, y_i$ .
- ▶ The goal of the machine learning algorithm is to find model parameters  $\mathbf{w}$  that maximizes this product of probabilities.
- ▶ We convert this maximization problem into a minimization one by taking the negative logarithm of the product of all the probabilities.

# Log-likelihood objective function



- ▶ One benefit of working with the logarithm of the product of probabilities rather than the product itself is that we can avoid underflow issues owing to the product of a large number of numbers that lie between 0 and 1.

$$\begin{aligned} J(\mathbf{w}) &= -\log_e \prod_{i=1}^{i=n} P(\mathbf{X}_i, y_i, \mathbf{w}) \\ &= -\sum_{i=1}^{i=n} \log_e P(\mathbf{X}_i, y_i, \mathbf{w}) \end{aligned}$$

- ▶ The objective function is now an additionally separable sum over the training examples.

## Additionally separable sum



- ▶ We can write the total objective function  $J(\mathbf{w}) = \sum_{i=1}^n J_i(\mathbf{w})$ .
- ▶ 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.
- ▶ The advantage of using stochastic gradient descent is that optimization can be speeded up.

- ▶ Stochastic gradient descent is especially useful when the data set is very large and one can get good descent directions using modest samples of the data.
- ▶ Let there be  $n$  datapoints  $\mathbf{X}_1, \mathbf{X}_2 \dots \mathbf{X}_n$  and let  $S$  be a subset of the indices  $\{1, 2, \dots n\}$ .
- ▶ The set  $S$  of data points can be treated as a sample and a sample-centric objective function can be constructed as follows:  $J(S) = \sum_{i \in S} (\mathbf{w}^T \mathbf{X}_i - y_i)^2$
- ▶ The key idea in stochastic gradient descent is that the gradient of the sample-specific objective function,  $J(S)$  with respect to the parameter vector  $\mathbf{w}$  is an excellent approximation of the true gradient.

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

$$[w_1, w_2, \dots, w_d]^T \leftarrow [w_1, w_2, \dots, w_d]^T - \alpha \left[ \frac{\partial J(S)}{\partial w_1}, \frac{\partial J(S)}{\partial w_2}, \dots, \frac{\partial J(S)}{\partial w_d} \right]^T$$

- ▶ This approach is referred to as mini-batch stochastic gradient.
- ▶ In the extreme case  $S$  can contain only one index chosen at random, and the approach is then called as stochastic gradient descent.

# Stochastic gradient descent



- ▶ Mini-batch stochastic gradient descent uses batch sizes that are powers of 2 to fit inside the memory units of GPUs.
- ▶ Instead of choosing data points at random, one can permute the training examples randomly and choose blocks of points from the ordering.
- ▶ The points are selected by cycling through the entire dataset, i.e the first 256 points followed by the next 256 points and so on.
- ▶ Therefore all points in the dataset are processed before arriving at a data point again.
- ▶ Each cycle of the mini-batch gradient descent procedure is called an epoch.

# Stochastic gradient descent



- ▶ When there are  $n$  data points in the training set and the mini-batch size is 1, there are  $n$  updates in an epoch. When the size of the mini-batch is  $k$ , an epoch will contain  $\lceil \frac{n}{k} \rceil$  updates.
- ▶ Why does stochastic gradient work so well?
- ▶ In the beginning the parameter vector  $w$  is grossly incorrect and it suffices to know the gradient only approximately in order to steer the parameter vector in a good direction.
- ▶ As we get to convergence, the noisy gradient actually helps in regularization and is beneficial to machine learning algorithms.

- ▶ There are some subtle differences between traditional optimization and optimization in the machine learning context.
- ▶ 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 training data which is unseen.
- ▶ Optimizing the objective function too much on the training data may cause the model to perform poorly on the test data!
- ▶ We need to avoid the problem of overfitting the training data.

- ▶ 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  $\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5$  where  $y$  is the observed value for the given values of the inputs.
- ▶ We would like to minimize the squared error between prediction and actual output over all the given training examples.

- ▶ Assuming that we have fewer than 5 training examples, we can actually find values of  $w_1, w_2, w_3, w_4, w_5$  to get the squared error to be zero since we can solve for values of  $w_1, w_2, w_3, w_4, w_5$  by solving fewer than five linear equations. Such a solution is possible because we have fewer equations than variables.
- ▶ Our model parameters will perform very well on the training data but might perform poorly on the test data.

## 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	59
40	0	4	0.5	40
68	0	10	1.0	70

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.

## Example of overfitting



- ▶ Consider the solution

$w_1 = 0, w_2 = 7, w_3 = 5, w_4 = 0, w_5 = 20$ . 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.

- ▶ On the other hand, the previous 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  $x_1$ . 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.

- ▶ The term "hyperparameter" is used to refer to parameters used to regulate the design of the model like learning rate. The learning rate is not to be confused with actual parameters like the weights of a linear regression model.
- ▶ Machine learning algorithms operate in a two-tiered way - first the values of hyperparameters are fixed, and these values are then used in the model to learn the actual model parameters.
- ▶ The hyperparameters should not be tuned using the same data set used for gradient descent in order to avoid overfitting the training data.

- ▶ A portion of the training data is held out as validation data, and model performance is recorded for various choices of the hyperparameters on the validation data.
- ▶ The main challenge in hyperparameter tuning is that different combinations of parameters need to be tested for their performance.
- ▶ We can perform a grid-search to test all possible combinations of values for the hyperparameters, but the number of grid points to be considered will grow exponentially in the number of hyperparameters.
- ▶ One way to get around this is to use coarse grids initially to narrow down the range of the hyperparameters and then use a finer grid.

- ▶ 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 age variable has a range of 100, ie  $[0, 100]$  and the number of years in the college education is in the range  $[0, 10]$ .
- ▶ The formula for wealth  $y$  is  $y = w_1x_1^2 + w_2x_2^2$ .
- ▶ 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  $w_1$ .

- ▶ 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.
- ▶ 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  $j$ th feature of the  $i$ th datapoint as follows:  $x_{ij} = \frac{x_{ij} - \min_j}{\max_j - \min_j}$ .



## Lecture 11

Math Foundations Team



**BITS** Pilani

Pilani | Dubai | Goa | Hyderabad

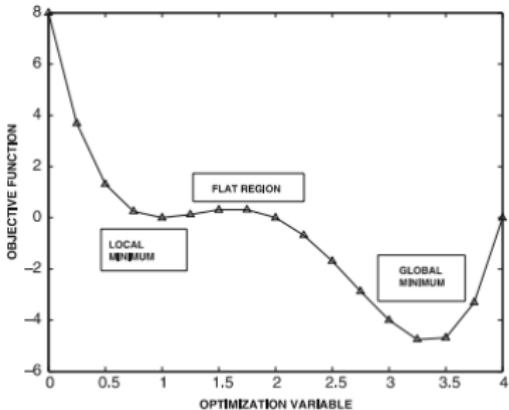
- ▶ We will take a deeper look at some challenges in non-linear optimization, continuing on from the previous lecture.
- ▶ First we will address the problem of flat regions and local optima.
- ▶ Second we will look at the problem of different levels of curvature in different directions.
- ▶ We will need to figure out strategies to deal with the above two situations to design a good optimization algorithm.

- ▶ Consider the following one-dimensional function:  
 $F(x) = (x - 1)^2((x - 3)^2 - 1).$
- ▶ Taking the derivative and setting it to zero, we have  
 $F'(x) = 2(x - 1)((x - 1)(x - 3) + (x - 3)^2 - 1) = 0.$
- ▶ The solutions to this equation are  
 $x = 1, x = \frac{5-\sqrt{3}}{2} = 1.634, x = \frac{5+\sqrt{3}}{2} = 3.366.$
- ▶ 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  $F''(x) < 0$ .
- ▶ Evaluating the function values at these points, we find  
 $F(1) = 0, F\left(\frac{5-\sqrt{3}}{2}\right) = 0.348, F\left(\frac{5+\sqrt{3}}{2}\right) = -4.848.$

# Local optima and flat regions



If we start gradient descent from any point less than 1.634, we will arrive only at a local minimum.

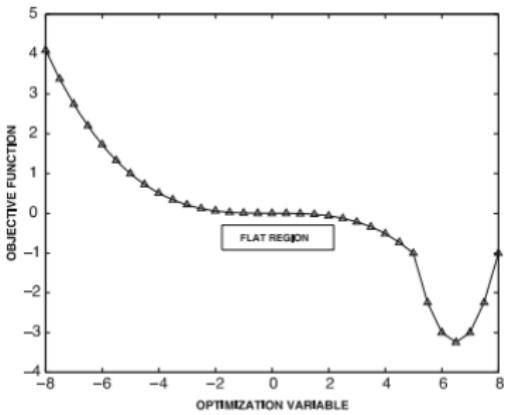


(a) Local optima with flat regions

- ▶ We might never arrive at a global minimum if we keep choosing wrong starting points. Thus we will get stuck at a local minimum, not knowing that a better solution exists.
- ▶ The problem becomes worse with high-dimensionality.
- ▶ Consider an objective function consisting of the sum of  $d$ -univariate functions in the variables  $x_1, x_2, \dots, x_d$ , each a function of a different variable, say
$$F(x_1, x_2, \dots, x_d) = A_1(x_1) + A_2(x_2) + \dots + A_n(x_n).$$
- ▶ 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  $(x_1^*, x_2^*, \dots, x_d^*)$ , where  $x_i^*$  is a local minima of the function  $A_i(x_i)$ , is a solution to  $\frac{\partial F}{\partial x_i} = 0$ .
- ▶ This is because  $\frac{\partial F}{\partial x_i}|_{(x_1^*, x_2^*, \dots, x_d^*)} = A'_i(x_i^*) = 0$  since  $x_i^*$  is a local minimum of  $A_i(x_i)$ .

- ▶ There are therefore  $\prod_{i=1}^{i=d} k_i$  local minima for the function  $F(x_1, x_2, \dots, x_d)$ , 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.
- ▶ Another problem to contend with is the presence of flat regions where the gradient is close to zero. An example of this situation is shown in the next slide.
- ▶ 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.

# Local optima and flat regions



(b) Only global optimum with flat region

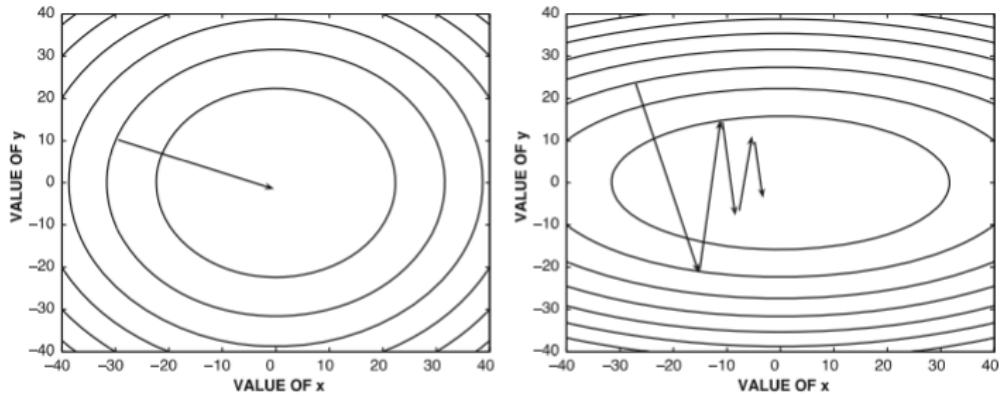
- ▶ In multi-dimensional settings, the components of the gradient with respect to different parameters can vary widely. As has already been mentioned in a previous slide, this will cause convergence problems since there is oscillation in the update step with respect to some components and a steady movement 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.

- ▶ 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.

- ▶ The second-prder 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.

- ▶ In the next slide we show contour plots of the perfect and elliptical bowls discussed previously. We see that 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.
- ▶ Note that the gradient at any point is orthogonal to the contour line at that point. This because the dot product of the gradient  $\nabla F$  and a small displacement  $\delta x$  along the contour line gives the change in the value of the function along the displacement  $x$ . Since the function remains constant along the contour line,  $\nabla F \cdot x = 0$

# Contour plots



(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

- ▶ 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 Guns and Butter respectively, and the output attribute is Happiness.

Guns (number per capita)	Butter (ounces per capita)	Happiness (index)
0.1	25	7
0.8	10	1
0.4	10	4

We intend to find a relationship of the form  $y = w_1x_1 + w_2x_2$  from the data. The coefficients  $w_1$  and  $w_2$  are found using gradient descent on the loss function computed from the above data.

## Revisiting feature normalization



- ▶ From the given three examples we can set up the loss function as follows:

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

- ▶ We note that this objective function is much more sensitive to  $w_2$  than  $w_1$  since the coefficients for  $w_2$  in the expression above are much larger than those for  $w_1$ .
- ▶ 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.

# Difficult topologies - cliffs



- ▶ Two examples of high curvature surfaces are cliffs and valleys.
- ▶ An example of a cliff is shown below:

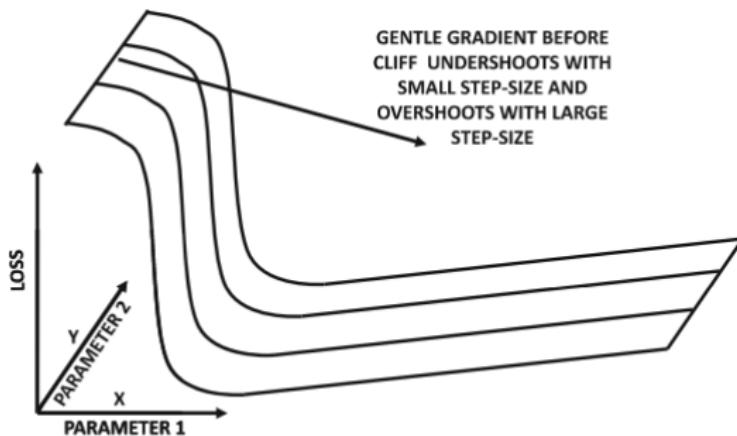


Figure 5.3: An example of a cliff in the loss surface

# Difficult topologies - cliffs



- ▶ Considering the previous figure, we see that 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.
- ▶ The problem is caused by the nature of curvature - the first order gradient does not have any information that will help control the size of the update.
- ▶ We need to look at second-order derivatives in this case.

## Difficult topologies - valleys



- ▶ Consider the figure below depicting a valley where 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.

# Difficult topologies - valleys

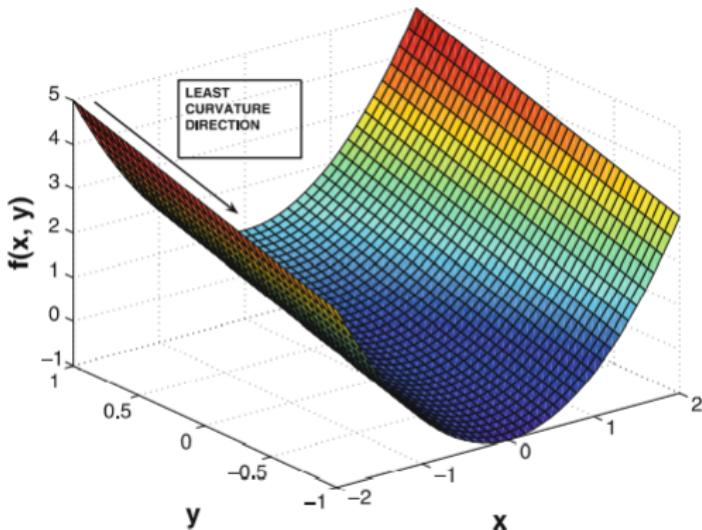


Figure 5.4: The curvature effect in valleys

- ▶ We need to find ways of magnifying movement along consistent directions of the gradient to avoid bouncing around the optimal solution.
- ▶ We could use second-order information to modify the first-order derivatives by taking curvature into account as we modify components of the gradient. Obtaining second-order information is computationally expensive.
- ▶ A computationally less expensive way to handle the problem is to use different learning rates for different parameters because parameters with large partial derivatives show oscillatory behaviour whereas those with small partial derivatives show consistent behaviour.

- ▶ Momentum-based methods attack the issues of flat-regions, cliffs and valleys by emphasizing medium-term to long-term directions of consistent movement.
- ▶ An aggregated measure of feedback is used to reinforce movement along certain directions and speed up gradient descent.
- ▶ 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.
- ▶ The normal update procedure for gradient descent can be written as  $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{v}$  where  $\mathbf{v} \leftarrow -\alpha \frac{\partial J}{\partial \mathbf{w}}$ .

- ▶ The gradient-descent procedure is modified in case of momentum-based learning in the sense that the update vector  $\mathbf{v}$  inherits a fraction  $\beta \in (0, 1)$  of its velocity from the previous step:  $\mathbf{v} \leftarrow \beta \mathbf{v} - \alpha \frac{\partial J}{\partial \mathbf{w}}$ .
- ▶ When  $\beta = 0$ , we have the standard gradient descent approach.
- ▶ When  $\beta$  is close to 1, we end up reinforcing a consistent velocity in the correct direction.
- ▶  $\beta$  is referred to as the momentum parameter or the friction parameter.

# Momentum-based learning



- ▶ The figure below how momentum-based learning compares to gradient descent, and how momentum allows it to navigate potholes.

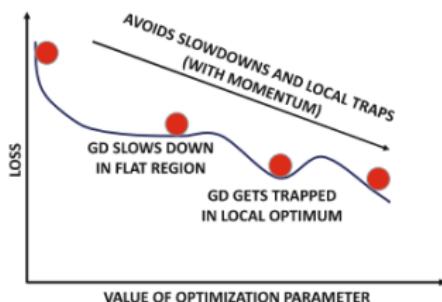


Figure 5.5: Effect of momentum in navigating complex loss surfaces. The annotation “GD” indicates pure gradient descent without momentum. Momentum helps the optimization process retain speed in flat regions of the loss surface and avoid local optima

- ▶ Momentum-based learning 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.
- ▶ In the figure on the next slide, it should be clear that momentum increases the relative component of the gradient in the correct direction.

# Momentum-based learning

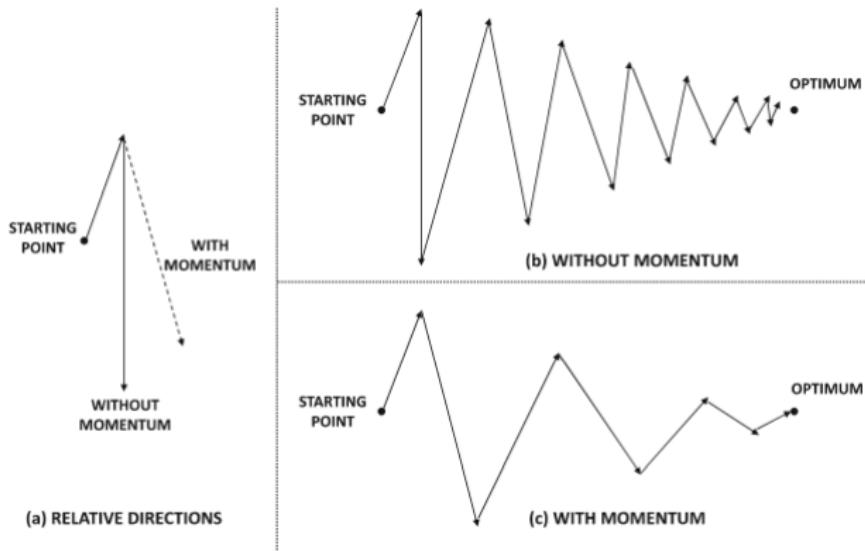


Figure 5.6: Effect of momentum in smoothing zigzag updates

- ▶ Here we keep track of the aggregated squared magnitude of the partial derivative with respect to each parameter over the course of the algorithm.
- ▶ Mathematically the aggregated squared partial derivative is stored in  $A_i$  for the  $i$ th variable and we can write
$$A_i \leftarrow A_i + \left( \frac{\partial J}{\partial w_i} \right)^2, \forall i.$$
- ▶ The actual update step becomes  $w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \frac{\partial J}{\partial w_i}, \forall i.$
- ▶ Sometimes we might need to avoid ill-conditioning; this is done by adding a small  $\epsilon = 10^{-8}$  to  $A_i$  in the expression above.

- ▶  $A_i$  measures only the historical magnitude of the gradient rather than the sign.
- ▶ If the gradient takes values +100 and -100 alternatively,  $A_i$  will be pretty large and the update step along the parameter in question will be pretty small. On the other hand, 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 than in the oscillating case.
- ▶ The update step along the consistent gradient will be emphasized relative to the update step along the oscillatory component.
- ▶ With the passage of time, however, 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, and the fact that  $A_i$  is aggregated over the entire history of partial derivatives which may make the method stale.
- ▶ Instead of simply adding squared gradients to estimate  $A_i$ , it uses exponential averaging. Therefore the scaling factor  $A_i$  does not constantly increase.  $A_i$  is updated according to  $A_i \leftarrow \rho A_i + (1 - \rho)(\frac{\partial J}{\partial w_i})^2$  where  $\rho \in (0, 1)$ .
- ▶ The update step is  $w_i \leftarrow w_i - \frac{\alpha}{\sqrt{A_i}} \frac{\partial J}{\partial w_i}$ ,  $\forall i$ .
- ▶ The key idea that differentiates RMSProp from AdaGrad is that the importance of ancient gradients decays exponentially with time as the gradient from  $t$  steps before is weighted by  $\rho^t$ .

- ▶ The Adam method uses similar normalization as AdaGrad and RMSProp.
- ▶ It also incorporates momentum into the update and addresses the initialization bias present in RMSProp.
- ▶  $A_i$  is exponentially averaged the same way as in RMSProp, i.e  $A_i \leftarrow \rho A_i + (1 - \rho)(\frac{\partial J}{\partial w_i})^2$  where  $\rho \in (0, 1)$ .
- ▶ An exponentially smoothed gradient for which the  $i$ th component is  $F_i$  is maintained. Smoothing is performed with a decay parameter  $\rho_f$ :  $F_i \leftarrow \rho_f F_i + (1 - \rho_f) \frac{\partial J}{\partial w_i}$
- ▶ The following update is used at the  $t$ th iteration:  
 $w_i \leftarrow w_i - \frac{\alpha_t F_i}{\sqrt{A_i}}$ .

- ▶ There are two key differences with the RMSProp algorithm - the first difference is the gradient is replaced with the exponentially smoothed value in order to incorporate momentum.
- ▶ The second difference is that the learning rate depends on the iteration index  $t$ , and is defined as follows:  $\alpha_t = \alpha \frac{\sqrt{1-\rho^t}}{1-\rho_f^t}$ .
- ▶ 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  $\alpha_t$ .
- ▶ As  $t \rightarrow \infty$ ,  $\rho^t \rightarrow 0$ ,  $\rho_f^t \rightarrow 0$  and  $\alpha_t \rightarrow \alpha$  since  $\rho, \rho_f \in (0, 1)$ . The default suggested value for  $\rho_f$  and  $\rho$  are 0.9 and 0.999 respectively.



## Lecture 12

Math Foundations Team



**BITS** Pilani

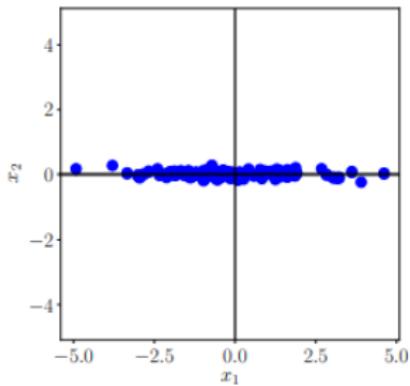
Pilani | Dubai | Goa | Hyderabad

- ▶ We will look at principle components analysis and dimension reduction in this lecture.
- ▶ High-dimensional data is hard to visualize 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

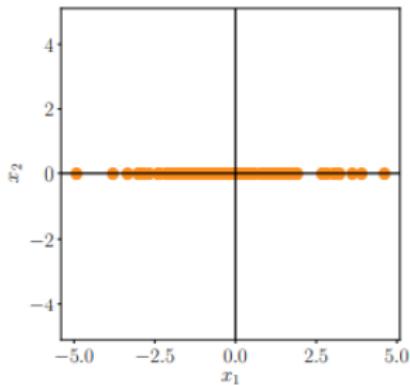
# Principle components analysis



- 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

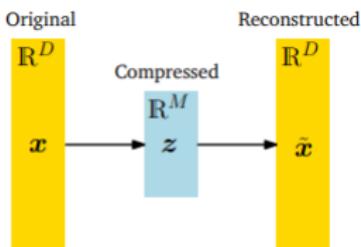


- ▶ We are interested in finding projections  $\tilde{x}_n$  of datapoints  $x_n$  which are as similar as possible to the original datapoints but have lower dimensionality.
- ▶ Consider an independent, identically distributed dataset  $\{x_1, x_2, \dots, x_n\}, x_n \in \mathbb{R}^D$  with mean  $\mathbf{0}$  which possesses the data covariance matrix  $S = \frac{1}{N} \sum_{n=1}^{n=N} x_n x_n^T$ .
- ▶ We assume there exists a lower-dimensional compressed representation  $z_n$  of  $x_n$  such that  $z_n = B^T x_n$  where the projection matrix  $B = [b_1, \dots, b_m] \in \mathbb{R}^{D \times M}$ .
- ▶ The columns of  $B$  are orthonormal which means  $b_i^T b_j = 0$  when  $i \neq j$  and  $b_i^T b_i = 1$ .
- ▶ We seek an  $M$ -dimensional subspace  $U \subseteq \mathbb{R}^D$ , where  $\dim(U) < D$  onto which we can project the given data.

# Problem setting



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



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

- ▶ 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  $B$  so that we can retain as much information as possible by projecting the data on the columns  $b_1, b_2, \dots, b_M$  of the matrix.

- ▶ For the data covariance matrix  $\mathbf{S} = \frac{1}{N} \sum_{n=1}^{n=N} \mathbf{x}_n \mathbf{x}_n^T$  we assume centred data, and can make this assumption without loss of generality.
- ▶ Let us assume that  $\mu$  is the mean of the data. Centred data means that we work with data columns  $\mathbf{x} - \mu$ , rather than the original columns  $\mathbf{x}$  but this does not change the variance.
- ▶ To see this note that
$$\mathbb{V}_z(z) = \mathbb{V}_x(\mathbf{B}^T(\mathbf{x} - \mu)) = \mathbb{V}_x(\mathbf{B}^T\mathbf{x} - \mathbf{B}^T\mu) = \mathbb{V}_x(\mathbf{B}^T\mathbf{x}).$$
- ▶ Therefore we assume that the data has a mean of  $\mathbf{0}$  for this lecture.
- ▶ Letting the mean be  $\mathbb{E}_x(x) = \mathbf{0}$  means
$$\mathbb{E}_z(z) = \mathbb{E}_x(\mathbf{B}^T\mathbf{x}) = \mathbf{B}^T\mathbb{E}_x(\mathbf{x}) = \mathbf{0}$$

- ▶ We maximize the variance of the low-dimensional code by following a sequential approach.
- ▶ First we aim to maximize the variance of the first coordinate  $z_{1n}$  of  $\mathbf{z} \in R^M$ , so that  $V_1 = \mathbb{V}(z_1) = \frac{1}{N} \sum_{n=1}^{n=N} z_{1n}^2$ .
- ▶ In the above expression for the variance we have used the fact that the data  $\mathbf{x}$  is independent.
- ▶ We can rewrite  $z_{1n} = \mathbf{b}_1^T \mathbf{x}_n$ , and can be viewed as the orthogonal projection of  $\mathbf{x}_n$  onto the one-dimensional subspace spanned by  $\mathbf{b}_1$ .
- ▶ Then we have  $V_1 = \frac{1}{N} \sum_{n=1}^{n=N} (\mathbf{b}_1^T \mathbf{x}_n)^2 = \frac{1}{N} \sum_{n=1}^{n=N} \mathbf{b}_1^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{b}_1 = \mathbf{b}_1^T \left( \sum_{n=1}^{n=N} \mathbf{x}_n \mathbf{x}_n^T \right) \mathbf{b}_1 = \mathbf{b}_1^T \mathbf{S} \mathbf{b}_1$ .

- ▶ Arbitrarily increasing the magnitude of the vector  $\mathbf{b}_1$  in the previous slide will increase the variance - so we seek to maximize the variance subject to  $\|\mathbf{b}_1\| = 1$ .
- ▶ Finding the direction  $\mathbf{b}_1$  that maximizes variance can be set up as a constrained optimization problem

$$\max \mathbf{b}_1^T \mathbf{S} \mathbf{b}_1 \text{ subject to}$$

$$\|\mathbf{b}_1\| = 1$$

- ▶ To solve this problem we set up the Lagrangian  $\mathcal{L}(\mathbf{x}, \lambda) = \mathbf{b}_1^T \mathbf{S} \mathbf{b}_1 + \lambda(1 - \mathbf{b}_1^T \mathbf{b}_1)$ .
- ▶ How do we solve this Lagrangian?

# Solving the Lagrangian



- ▶ To solve the Lagrangian, we set the partial derivatives with respect to  $\lambda$  and  $\mathbf{b}_1$  to zero, ie  $\frac{\partial \mathbb{L}}{\partial \lambda} = 0$  and  $\frac{\partial \mathbb{L}}{\partial \mathbf{b}_1} = \mathbf{0}$ .
- ▶ The partial derivatives can be calculated as follows:

$$\frac{\partial \mathbb{L}}{\partial \lambda} = 1 - \mathbf{b}_1^T \mathbf{b}_1$$

$$\frac{\partial \mathbb{L}}{\partial \mathbf{b}_1} = 2\mathbf{b}_1^T \mathbf{S} - 2\lambda \mathbf{b}_1^T$$

- ▶ Setting these partial derivatives to zero we get the following two equations  $\mathbf{S}\mathbf{b}_1 = \lambda\mathbf{b}_1$  and  $\mathbf{b}_1^T \mathbf{b}_1 = 1$ .
- ▶ Thus we find that the direction  $\mathbf{b}_1$  we seek is an eigenvector of the covariance matrix  $\mathbf{S}$  and  $\lambda$  is its corresponding eigenvalue.

# First principal component



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

# Why does this method work?



- ▶ Suppose we have the following constrained optimization problem:

$$\begin{aligned} & \max f(x, y) \text{ subject to} \\ & g(x, y) = c \end{aligned}$$

- ▶ We note that at the optimal solution  $(x_0, y_0)$ , if we move a small distance  $(\delta x, \delta y)$ , we must continue to remain on the surface  $g(x, y) = c$ . This means  
$$g(x_0 + \delta x, y_0 + \delta y) = c = g(x, y).$$
- ▶ But  $dg = g(x_0 + \delta x, y_0 + \delta y) - g(x, y) = \nabla g \cdot (\delta x, \delta y) = 0$ .
- ▶ Therefore  $\nabla g$  is orthogonal to the displacement vector  $(\delta x, \delta y)$ .

# Why does this method work?



- ▶ As we move along the displacement vector  $(\delta x, \delta y)$ , the value of the objective function also cannot change because otherwise we can get a better solution by moving along the displacement vector or its negative direction.
- ▶ Thus we have  $\nabla f \cdot (\delta x, \delta y) = 0$ , so  $\nabla f$  is orthogonal to the displacement vector  $(\delta x, \delta y)$ .
- ▶ Keeping in mind the result from the previous slide, we note that both  $\nabla f$  and  $\nabla g$  are orthogonal to the displacement vector  $(\delta x, \delta y)$  and hence must be parallel.
- ▶ This leads us to write the equation  $\nabla f = \lambda \nabla g$  which is what we get when we set  $\frac{\partial \mathbb{L}}{\partial b_1} = 0$ .
- ▶ The other partial derivative  $\frac{\partial \mathbb{L}}{\partial \lambda} = 0$  merely enforces the constraint in the original constrained optimization problem.

- ▶ 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  $m$ th principal component can be found by subtracting from the data the contribution of the first  $m - 1$  components  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{m-1}$ . Essentially we are trying to find principal components that compress the remainder of the information.

- ▶ We then arrive at a new data matrix  $\hat{\mathbf{X}} = \mathbf{X} - \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^T \mathbf{X}$  where  $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N] \in \mathbb{R}^{D \times N}$  contains the data points as column vectors and  $\mathbf{B}_{m-1} = \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^T$  is a projection matrix that projects  $\mathbf{X}$  onto the subspace spanned by  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{m-1}$ .
- ▶ Note that we are collecting the data vectors  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$  as column vectors in the data matrix rather than as row vectors as is done conventionally.
- ▶ To find the  $m$ th principal component we maximize  $V_m = \mathbb{V}[z_m] = \frac{1}{N} \sum_{n=1}^{n=N} z_{mn}^2 = \frac{1}{N} \sum_{n=1}^{n=N} (\mathbf{b}_m^T \hat{\mathbf{x}}_n)^2 = \mathbf{b}_m^T \hat{\mathbf{S}} \mathbf{b}_m$ .
- ▶ What is  $\hat{\mathbf{S}}$  in the above equation?

- ▶  $\hat{\mathbf{S}}$  is the data covariance matrix of the transformed data set represented by  $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_N]$ .
- ▶ As before we set up a constrained optimization problem to find the first principal component, and establish that the optimal solution  $\mathbf{b}_m$  is the eigenvector of  $\hat{\mathbf{S}}$  that corresponds to the largest eigenvalue.
- ▶ We now establish that  $\mathbf{b}_m$  is an eigenvector of the original data matrix  $\mathbf{X}$ .
- ▶ More generally the sets of eigenvectors for  $\hat{\mathbf{S}}$  and  $\mathbf{S}$  are the same.

# Eigenvectors of $\mathbf{S}$ and $\hat{\mathbf{S}}$



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

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

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

# Eigenvectors of $\mathbf{S}$ and $\hat{\mathbf{S}}$



- ▶ We have two cases:  $i \geq m$  and  $i \leq m - 1$ .
- ▶ When  $i \geq m$ ,  $\mathbf{b}_i$  is an eigenvector not among the first  $m - 1$  components.
- ▶ Since  $\mathbf{B}_{m-1} = \sum_{i=1}^{m-1} \mathbf{b}_i \mathbf{b}_i^T$  and  $\mathbf{b}_m$  is orthogonal to the  $\mathbf{b}_i, 1 \leq i \leq m - 1$ , we have  $\mathbf{B}_{m-1} \mathbf{b}_i = 0$ .
- ▶ Plugging this into the last equation on the previous slide, we see that  $\hat{\mathbf{S}}\mathbf{b}_i = (\mathbf{S} - \mathbf{B}_{m-1}\mathbf{S})\mathbf{b}_i = \mathbf{S}\mathbf{b}_i = \lambda_i \mathbf{b}_i$ .
- ▶ Thus  $\mathbf{S}\mathbf{b}_m = \lambda_m \mathbf{b}_m$ .  $\lambda_m$  is the  $m$ th largest eigenvalue of  $\mathbf{S}$  and is also the largest eigenvalue of  $\hat{\mathbf{S}}$  because of the way the constrained optimization problem is set up.
- ▶ On the other hand, when  $i \leq m - 1$ , we can see that  $\mathbf{B}_{m-1} \mathbf{b}_i = \mathbf{b}_i$ .

# Eigenvectors of $\mathbf{S}$ and $\hat{\mathbf{S}}$



- ▶ From the previous slide, when  $i \leq m - 1$  we have  $\mathbf{B}_{m-1} \mathbf{b}_i = \mathbf{b}_i$ .
- ▶ Plugging this into  $\hat{\mathbf{S}}\mathbf{b}_i = (\mathbf{S} - \mathbf{S}\mathbf{B}_{m-1} - \mathbf{B}_{m-1}\mathbf{S} + \mathbf{B}_{m-1}\mathbf{S}\mathbf{B}_{m-1})\mathbf{b}_i$ , we get  $\mathbf{S}\mathbf{b}_i = \mathbf{0}$ .
- ▶ Thus the vectors  $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_{m-1}$  are eigenvectors for  $\hat{\mathbf{S}}$  which are associated with the eigenvalue 0.
- ▶ Since  $V_m = \mathbf{b}_m \mathbf{S} \mathbf{b}_m = \lambda_m$ , we see that the variance of the data projected onto the  $m$ th principal component is  $\lambda_m$ .
- ▶ To find an  $M$ -dimensional subspace that retains as much information as possible, PCA tells us to choose the columns of the matrix  $\mathbf{B}$  as the  $M$  eigenvectors of the data covariance matrix  $\mathbf{S}$  that have the largest eigenvalues.

- ▶ We derived the PCA as an algorithm that maximizes the variance in the projected space to retain as much information as possible.
- ▶ Now we shall derive the PCA using a projection perspective to minimize the average reconstruction error. The original data is modeled as  $x_n$  and the reconstruction is modeled as  $\tilde{x}_n$ . We seek to minimize the distance between  $x_n$  and  $\tilde{x}_n$ .
- ▶ Assume that we have an orthonormal basis  $B = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_D)$  of  $\mathbb{R}^D$ .
- ▶ We can write any  $x \in \mathbb{R}^D$  as  $x = \sum_{d=1}^D \xi_d \mathbf{b}_d = \sum_{m=1}^M \xi_m \mathbf{b}_m + \sum_{j=M+1}^D \xi_j \mathbf{b}_j$  for suitable coordinates  $\xi_d \in \mathbb{R}$ .

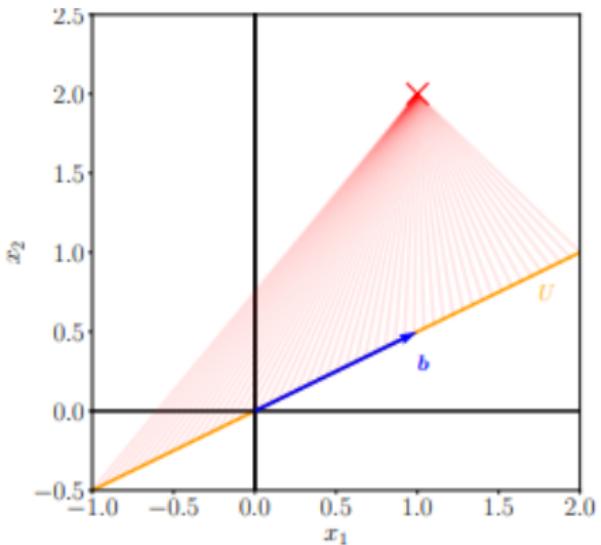
- ▶ We are interested in finding vectors  $\tilde{\mathbf{x}} \in \mathbb{R}^D$  which live in a lower dimensional subspace  $U \subset \mathbb{R}^D$  where  $\dim(U) = M$  so that  $\tilde{\mathbf{x}} = \sum_{m=1}^M z_m \mathbf{b}_m \in U \subset \mathbb{R}^D$  is as similar to  $\mathbf{x}$  as possible, by which we mean that the distance  $\|\mathbf{x} - \tilde{\mathbf{x}}\|$  is as small as possible.
- ▶ We assume that the dataset  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$  is centred at  $\mathbf{0}$  or  $E(X) = \mathbf{0}$ . This is not a restrictive assumption, in that we get the same results with or without the assumption, but it considerably simplifies the mathematical development.
- ▶ We call the subspace  $U \subset \mathbb{R}^D$ , on which we are projecting the vectors  $\mathbf{x}$ , the principal subspace. We can write  $\tilde{\mathbf{x}}_n = \sum_{m=1}^M z_{mn} \mathbf{b}_m = \mathbf{B}\mathbf{z}_n \in \mathbb{R}^D$ . Here  $\mathbf{z} = [z_{1n}, \dots, z_{Mn}]$  is the coordinate vector of  $\mathbf{x}_n$  with respect to the basis  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$ .

# Finding optimal coordinates



- ▶ The reconstruction error can be written as  $J_M = \frac{1}{N} \sum_{n=1}^N \|x_n - \tilde{x}_n\|^2$  where we use the subscript  $M$  to denote the dimension of the subspace on which to project data.
- ▶ We would like to find optimal coordinates  $z_{1n}, z_{2n}, \dots, z_{Mn}$  with respect to the basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$  for  $\tilde{x}_n, n = 1, \dots, N$ .
- ▶ In geometrical terms, finding the optimal coordinates boils to finding the representation with respect to  $\mathbf{b}$  that minimizes the distance between  $x$  and  $\tilde{x}$ .
- ▶ To do this, we need to find the orthogonal projection of  $x$  onto  $\mathbf{b}$ . The concept is illustrated on the next slide.

# Finding optimal coordinates



(b) Differences  $\mathbf{x} - \tilde{\mathbf{x}}_i$  for 50 different  $\tilde{\mathbf{x}}_i$  are shown by the red lines.

# Finding optimal coordinates



- ▶ Assume an orthonormal basis (ONB)  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$  of the subspace  $U \subset \mathbb{R}^D$ .
- ▶ To find the optimal coordinates, we take the derivative of the reconstruction error  $J_M = \frac{1}{N} \sum_{n=1}^N \|\mathbf{x}_n - \tilde{\mathbf{x}}_n\|^2$  with respect to the coordinates  $z_{in}$  and set the derivative to zero, i.e  $\frac{\partial J_M}{\partial z_{in}} = 0$ .
- ▶ We can write  $\frac{\partial J_M}{\partial z_{in}} = \frac{\partial J_M}{\partial \tilde{\mathbf{x}}_n} \frac{\partial \tilde{\mathbf{x}}_n}{\partial z_{in}}$  using the chain rule.
- ▶ Note that only  $\tilde{\mathbf{x}}_n$  is a function of the  $z_{in}$ s and the given  $\mathbf{x}_n$  is a fixed vector independent of the coordinates  $z_{in}$ .
- ▶ We shall now find expressions for each of the partial derivatives on the right hand side.

# Finding optimal coordinates



$$\frac{\partial J_M}{\partial \tilde{\mathbf{x}}_n} = -\frac{2}{N}(\mathbf{x} - \tilde{\mathbf{x}}_n)^T$$

$$\frac{\partial \tilde{\mathbf{x}}_n}{\partial z_{in}} = \frac{\partial}{\partial z_{in}} \left( \sum_{m=1}^M z_{mn} \mathbf{b}_m \right) = \mathbf{b}_i$$

$$\frac{\partial J_M}{\partial z_{in}} = -\frac{2}{N}(\mathbf{x}_n - \tilde{\mathbf{x}}_n)^T \mathbf{b}_i$$

$$\frac{\partial J_M}{\partial z_{in}} = -\frac{2}{N}(\mathbf{x}_n - \sum_{m=1}^M z_{mn} \mathbf{b}_m)^T \mathbf{b}_i$$

$$\frac{\partial J_M}{\partial z_{in}} = -\frac{2}{N}(\mathbf{x}_n^T \mathbf{b}_i - z_{in})$$

# Finding optimal coordinates



- ▶ Note that in the preceding sequence of equations we used the orthonormality property of the basis vectors  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$  to write  $(\sum_{m=1}^M z_{mn} \mathbf{b}_m)^T \mathbf{b}_i = z_{in} \mathbf{b}_i^T \mathbf{b}_i = z_{in}$ .
- ▶ Setting the right-hand side of the last equation in the previous slide to zero yields  $z_{in} = \mathbf{x}_n^T \mathbf{b}_i = \mathbf{b}_i^T \mathbf{x}_n$ .
- ▶ The coordinates of  $\tilde{\mathbf{x}}_n$  with respect to the basis  $(\mathbf{b}_1, \dots, \mathbf{b}_M)$   $\mathbf{x}_n$  are the coordinates of the orthogonal projection of  $\mathbf{x}_n$  onto the principal subspace.

# Finding a basis of the principal subspace



- ▶ To find the basis vectors  $\mathbf{b}_1, \dots, \mathbf{b}_M$  of the principal subspace we have to reformulate the loss function.
- ▶ We can write  $\tilde{\mathbf{x}}_n = \sum_{m=1}^M z_{mn} \mathbf{b}_m = \sum_{m=1}^M (\mathbf{x}_n^T \mathbf{b}_m) \mathbf{b}_m$ .
- ▶ We can rewrite this as  $\tilde{\mathbf{x}}_n = (\sum_{m=1}^M \mathbf{b}_m \mathbf{b}_m^T) \mathbf{x}_n$ .
- ▶ The original data point  $\mathbf{x}_n$  can also be written as a linear combination of all the basis vectors as follows:  
$$\mathbf{x}_n = \sum_{d=1}^D (\mathbf{x}_n^T \mathbf{b}_d) \mathbf{b}_d.$$
- ▶ How can we do the above? First write  $\mathbf{x}_n = \sum_{d=1}^D z_{dn} \mathbf{b}_d$ , and take the inner product with  $\mathbf{b}_d$  on both sides to obtain  $z_{dn}$ . Orthonormality of the basis  $\mathbf{b}_1, \dots, \mathbf{b}_D$  gives the result.
- ▶ By rearranging the expression for  $\mathbf{x}_n$ , we can write  
$$\mathbf{x}_n = \sum_{d=1}^D (\mathbf{b}_d \mathbf{b}_d^T) \mathbf{x}_n.$$

# Finding a basis of the principal subspace



- ▶ We can split the expression for  $x_n$  as follows:  
$$x_n = (\sum_{m=1}^M b_m \mathbf{b}_m^T) x_n + (\sum_{j=m+1}^D \mathbf{b}_j \mathbf{b}_j^T) x_n.$$
- ▶ Then we find that the displacement vector  $x - \tilde{x}_n$  can be written as  $x - \tilde{x}_n = \sum_{j=m+1}^D \mathbf{b}_j \mathbf{b}_j^T x_n = \sum_{j=m+1}^D (x_n^T \mathbf{b}_j) \mathbf{b}_j$ .
- ▶ The last expression shows that the displacement vector is exactly the projection of the original data point  $x_n$  onto the orthogonal complement of the principal subspace.
- ▶ We can now express the loss function as follows:  
$$J_M = \frac{1}{N} \sum_{n=1}^N \|x_n - \tilde{x}_n\|^2 = \frac{1}{N} \sum_{n=1}^N \| \sum_{j=M+1}^D (\mathbf{b}_j^T x_n) \mathbf{b}_j \|^2.$$
- ▶ We now expand the squared-norm and exploit the fact that the  $\mathbf{b}_j$  form an orthonormal basis to rewrite the loss function as in the next slide.

# Finding a basis of the principal subspace



$$\begin{aligned} J_M &= \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D (\mathbf{b}_j^T \mathbf{x}_n)^2 \\ &= \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D \mathbf{b}_j^T \mathbf{x}_n \mathbf{b}_j^T \mathbf{x}_n \\ &= \frac{1}{N} \sum_{n=1}^N \sum_{j=M+1}^D \mathbf{b}_j^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{b}_j \\ &= \sum_{j=M+1}^D \mathbf{b}_j^T \left( \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n \mathbf{x}_n^T \right) \mathbf{b}_j \\ &= \sum_{j=M+1}^D \mathbf{b}_j^T S \mathbf{b}_j \end{aligned}$$

# Finding a basis of the principal subspace



- ▶ We note that  $J_M = \sum_{j=M+1}^D \mathbf{b}_j^T S \mathbf{b}_j = \text{tr}(\sum_{j=M+1}^D \mathbf{b}_j^T S \mathbf{b}_j) = \sum_{j=M+1}^D S \mathbf{b}_j \mathbf{b}_j^T) = \text{tr}(\sum_{j=M+1}^D \mathbf{b}_j \mathbf{b}_j^T S)$
- ▶ In the above we exploit the fact that the trace operator is invariant with respect to the cyclic permutation of its arguments.
- ▶  $S$  in the above equations is the data covariance matrix since we assume that  $E(X) = 0$ .
- ▶ We can formulate the average reconstruction error as the covariance matrix of the data projected onto the orthogonal complement of the principal subspace.

# Finding a basis of the principal subspace



- ▶ To minimize the average reconstruction error we need to minimize the variance of the data when projected onto the space we ignore which is the orthogonal complement of the principal subspace.
- ▶ This is equivalent to maximizing the variance of the data when projected onto the principal subspace which therefore leads us to the same kind of solution as in the maximum variance perspective.
- ▶ The smallest value of the average squared reconstruction error turns out to be  $J_M = \sum_{j=M+1}^D \lambda_j$  where the  $\lambda_j$ s are the last  $D - M$  eigenvalues of the data covariance matrix.



## Lecture 13

MFDS Team



# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

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

- ▶ 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.

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

- ▶ Equivalently we get

$$S = \frac{1}{N} X X^T \quad (2)$$

$$X = [x_1, \dots, x_N] \in \mathbb{R}^{D \times N} \quad (3)$$

- ▶ Note that  $X$  is a  $D \times N$  matrix,

- ▶ To get the eigenvalues and the corresponding eigenvectors of  $S$ , we can follow two approaches
- ▶ We can perform an eigendecomposition 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

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

- ▶ The columns of  $U$  are the eigenvectors of  $S$ .
- ▶ The eigenvalues  $\lambda_d$  of  $S$  are related to the singular values of  $X$  via

$$\lambda_d = \frac{\sigma_d^2}{N} \quad (4)$$

- ▶ 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.

- ▶ 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  $\tilde{X}_M$

$$\tilde{X}_M = \operatorname{argmin}_{\operatorname{rank}(A) \leq M} \|X - A\|_2 \quad (5)$$

- ▶ 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
- ▶  $\Sigma_M$  has  $M$  largest singular values of  $X$  as diagonal entries

- ▶ 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.

- ▶ 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.

- ▶ 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 eigendecomposition

- ▶ 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  $k = 0, 1, \dots$

$$x_{k+1} = \frac{Sx_k}{\|Sx_k\|} \quad (6)$$

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

- 
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.,  $N \ll D$

- ▶ Assume we have a centered dataset  $x_1, \dots, x_N, x_n \in \mathbb{R}^D$ . Then the data covariance matrix is given as  $S = \frac{1}{N}XX^T$
- ▶

$$Sb_m = \frac{1}{N}XX^T b_m = \lambda_m b_m \quad (7)$$

$$\frac{1}{N}X^T XX^T b_m = \lambda_m X^T b_m \quad (8)$$

$$\frac{1}{N}X^T X c_m = \lambda_m c_m \quad (9)$$

- ▶ Here  $c_m = X^T b_m$ .
- ▶ The nonzero eigenvalues of  $XX^T$  is same as the nonzero eigenvalues of  $X^T X$ .

- 
- ▶ Now that we have the eigenvectors of  $\frac{1}{N}X^T X$ ,
  - ▶ We need to derive the eigenvectors of  $XX^T$ , which we still need for PCA

$$\frac{1}{N}XX^T Xc_m = \lambda_m Xc_m \quad (10)$$

- ▶ Here, we recover the data covariance matrix again.
- ▶ This now also 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 and we want to use PCA to project it onto a one-dimensional subspace. The key steps are given below
  - ▶ Mean subtraction
  - ▶ Standardization
  - ▶ Eigendecomposition of the covariance matrix
  - ▶ Projection

## 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

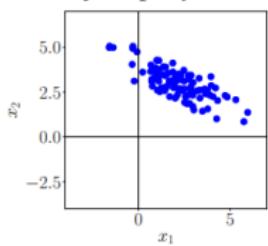
## Standardization

1. Divide the data points by the standard deviation  $\sigma_d$  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.

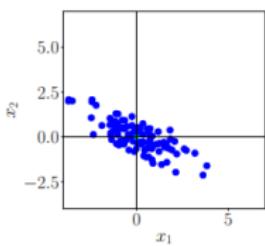
# Key Steps of PCA in Practice



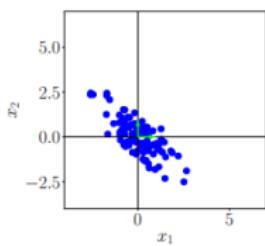
Figure: PCA



(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.

## Eigendecomposition of the covariance matrix

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

## 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

$$x_*^{(d)} = \frac{x_*^{(d)} - \mu_d}{\sigma_d}, \quad d = 1, \dots, D \quad (11)$$

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

# Key Steps of PCA in Practice



1. We obtain the projection as

$$\tilde{x} = BB^T x_* \quad (12)$$

2. The coordinates are

$$z_* = B^T x_* \quad (13)$$

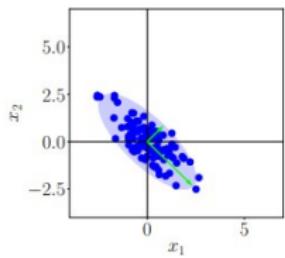
with respect to the basis of the principal subspace.

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

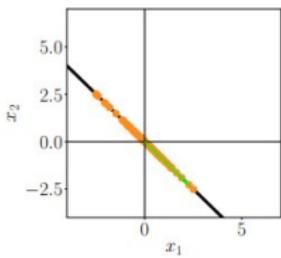
# Key Steps of PCA in Practice



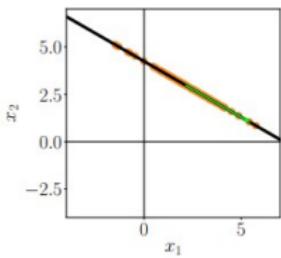
Figure: PCA



(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 from two perspectives: (a) maximizing the variance in the projected space; (b) minimizing the average reconstruction error.
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  $Bz$ .



## Lecture 14

Math Foundations Team



# BITS Pilani

Pilani | Dubai | Goa | Hyderabad

# Minimax Optimization(Recap)



Consider the minimization problem of the form:

$$P = \min F(w), \quad w \text{ is a vector}$$

$$\text{s.t. } f_i(w) \leq 0, \forall i \in \{1, 2, \dots, m\}$$

This problem is called *Primal problem* in optimization parlance.

We denote the minimax optimization function of Lagrangian relaxation as  $H(w, \alpha)$ :

$$H(w, \alpha) = F(w) + \sum_{i=1}^m \alpha_i f_i(w)$$

Here,  $w$  contains the minimization variables and  $\alpha$  contains the maximization variables. Minimax inequality states that :

$$\max_{\alpha \geq 0} \min_w H(w, \alpha) \leq \min_w \max_{\alpha \geq 0} H(w, \alpha)$$

Consider the primal and dual minimax optimization problems:

$$P = \min_w \max_{\alpha \geq 0} H(w, \alpha) \dots (OP1) Primal$$

$$D = \max_{\alpha \geq 0} \min_w H(w, \alpha) \dots (OP2) Dual$$

For a solution  $(w, \alpha)$  to be optimal to the primal minimax problem (OP1),  $w$  must be a feasible solution satisfying  $f_i(w) \leq 0 \forall i$ . If any constraint  $f_i(w) \leq 0$  is satisfied with strict inequality, then setting  $\alpha_i = 0$  ensures maximization of (OP1) with respect to  $\alpha$ . This ensures that we have  $\alpha_i f_i(w) = 0$  for each  $i$  for any optimal solution to (OP1). The condition  $\alpha_i f_i(w) = 0$  is called the complementary slackness condition.

# Kuhn-Tucker Optimality Conditions



The minimax theorem tells us that the optimal pairs  $(w, \alpha)$  are the same in the two cases of the primal (OP1) and the dual (OP2) when the function  $H(w, \alpha)$  is convex in  $w$  and concave in  $\alpha$ . Another important condition that needs to be satisfied is that the gradient of  $H(w, \alpha)$  with respect to the primal variables  $w$  need to be set to 0 because we are minimizing this objective function at each fixed value of  $\alpha$ . This leads to the stationarity conditions:

$$\nabla_w H(w, \alpha) = \nabla F(w) + \sum_{i=1}^m \alpha_i \nabla f_i(w) = 0$$

The Kuhn-Tucker conditions are obtained by combining the primal feasibility conditions, dual feasibility conditions, complementary slackness conditions, and stationarity conditions.

Consider an optimization problem in which we wish to minimize the convex objective function  $F(w)$ , subject to convex constraints of the form  $f_i(w) \leq 0$  for  $i \in \{1 \dots m\}$ . Then, a solution  $w$  is optimal for the primal and a solution  $\alpha$  is optimal for the dual, if and only if:

1. **Feasibility:**  $w$  is feasible for the primal by satisfying each  $f_i(w) \leq 0$  and  $\alpha$  is feasible for the dual by being nonnegative.
2. **Complementary slackness:** We have
$$\alpha_i f_i(w) = 0 \forall i \in \{1 \dots m\}$$
3. **Stationarity:** The primal and dual variables are related as follows:

$$\nabla F(w) + \sum_{i=1}^m \alpha_i \nabla f_i(w) = 0$$

---

For a convex optimization problem, any pair  $(w, \alpha)$  that satisfies primal feasibility, dual feasibility, Complementary slackness, and stationarity conditions is an optimal solution to the original optimization problem.

The stationarity conditions relate the primal and dual variables, and therefore they are often useful for eliminating primal variables from the Lagrangian. We will also refer to them as primal-dual (PD) constraints, because they relate primal and dual variables at optimality. The stationarity conditions are often used to formulate the minimax dual purely in terms of the dual variable (and therefore create a pure maximization problem).

# General Procedure for Using Duality



Here we formulate  $L(\alpha)$  which is the objective function of the dual problem (OP2), after eliminating primal variables:

$$L(\alpha) = \min_w H(w, \alpha)$$

The primal variables  $w$  can often be eliminated from  $L(\alpha)$  by setting the gradients of  $H(w, \alpha)$  with respect to the primal variables  $w$  to zero and it will result in exactly as many conditions as the number of primal variables. These are exactly the stationarity conditions. These are also referred as primal-dual (PD) constraints, as they relate the primal and dual variables. The (PD) constraints can be used to substitute for (and eliminate) the primal variables  $w$ , and obtain a pure maximization objective function  $L(\alpha)$ , which is expressed in terms of  $\alpha$ .

# Inferring the Optimal Primal Solution from Optimal Dual Solution

---



We need to find the optimal primal variables in order to have an interpretable solution. So how one can infer an optimal primal solution  $w$  from the optimal dual solution  $\alpha$ . In this context, the stationarity conditions are very helpful, because they can be used to substitute in the values of the optimal dual variables and solve for the primal variables.

## Application: Formulating the SVM Dual



We will illustrate how duality is being used in the Support vector Machine. The objective function of Hinge loss SVM is given by:

$$J = \frac{1}{\lambda} \sum_{i=1}^n \max \{0, (1 - y_i [W \cdot X_i^T])\} + \frac{1}{2} \|W\|^2 \quad (1)$$

To construct the dual, we need to reformulate the problem as a constrained optimization problem without the maximization operator. This is achieved with the use of slack variables  $\xi_1, \xi_2, \dots, \xi_n$  as follows:

Minimize  $J = \frac{1}{2} \|W\|^2 + C \sum_{i=1}^n \xi_i$   
subject to

$\xi_i \geq 1 - y_i [W \cdot X_i^T], \forall i \in \{1, 2, \dots, n\}$  [Margin Constraints]  
 $\xi_i \geq 0, \forall i \in \{1, 2, \dots, n\}$  [Nonnegativity Constraints]

We want  $\xi = \max \{0, (1 - y_i[W.X_i^T])\}$ . Note that the constraints do allow values of  $\xi$  larger than  $\max \{0, (1 - y_i[W.X_i^T])\}$ , but such values can never be optimal.

The first set of constraints is referred to as the set of *margin* constraints, because they define the margins for the predicted values of  $y_i$  beyond which points are not penalized.

The Lagrangian multiplier  $\alpha_i$  for the  $i^{th}$  of  $n$  margin constraints and the multiplier  $\gamma_i$  for the  $i^{th}$  nonnegativity constraint on  $\xi_i$ .

With these notations, the Lagrangian relaxation is expressed as:

$$\begin{aligned} L_D(\alpha, \gamma) = \min J_r = & \frac{1}{2} \|W\|^2 + C \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (\xi_i - 1 + y_i[W.X_i^T]) \\ & - \sum_{i=1}^n \gamma_i \xi_i \end{aligned}$$

$J_r$  is the relaxed objective function. Since the relaxed constraints are inequalities, it follows that both  $\alpha_i$  and  $\gamma_i$  must be nonnegative for the relaxation to make sense. Therefore, we optimize over the dual variables such as  $\alpha_i$  and  $\gamma_i$ .

Here, one first minimizes over primal variables (with dual variables fixed) to obtain  $L_D(\alpha, \gamma)$  and then maximizes  $L_D(\alpha, \gamma)$  over the dual variables, while imposing box constraints on them. This can be expressed as minimax optimization problem:

$$L_D^* = \max_{\alpha, \gamma \geq 0} L_D(\alpha, \gamma) = \max_{\alpha, \gamma \geq 0} \min_{W, \xi_i} J_r$$

The general approach to solving the dual is to use the (PD) constraints to eliminate the primal variables in order to create a pure maximization problem in terms of the dual variables. The (PD) constraints are obtained by setting the gradient of the minimax objective with respect to the primal variables to 0.

$$\frac{\partial J_r}{\partial W} = W - \sum_{i=1}^n \alpha_i y_i X_i^T = 0 \quad (2)$$

$$\frac{\partial J_r}{\partial \xi_i} = C - \alpha_i - \gamma_i = 0, \forall i \in \{1, 2, \dots, n\} \quad (3)$$

Based on Equations in previous slide, we can substitute

$W = \sum_{i=1}^n \alpha_i y_i X_i^T$  everywhere it occurs in  $J_r$ . By dropping the terms involving  $\xi_i$  and substituting for  $W$ ,  $J_r$  is simplified as follows:

$$\begin{aligned} J_r &= \frac{1}{2} \|W\|^2 + \sum_{i=1}^n \alpha_i (1 - y_i [W \cdot X_i^T]) \\ &= \frac{1}{2} \left\| \sum_{i=1}^n \alpha_i y_i X_i^T \right\|^2 + \sum_{i=1}^n \alpha_i \left( 1 - y_i \left[ \sum_{j=1}^n \alpha_j y_j X_i \cdot X_j \right] \right) \quad (4) \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j X_i \cdot X_j \end{aligned}$$

The objective function is expressed in the terms of the dual variables.

Furthermore, the variable  $\gamma_i$  has dropped out of the optimization formulation. The constraint  $\gamma_i \geq 0$  also needs to be modified by substituting  $\gamma_i$  as:

$$\gamma_i = C - \alpha_i \geq 0 \quad (5)$$

Therefore, the variables  $\alpha_i$  satisfy the box constraints  $0 \leq \alpha_i \leq C$ . We can multiply the objective function by  $-1$  in order to turn the maximization problem into a minimization problem:

$$\min_{0 \leq \alpha_i \leq C} \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j X_i \cdot X_j - \sum_{i=1}^n \alpha_i$$

The dual problem (in minimization form) is always convex, one can show that the leading term in the quadratic is of the form  $\alpha^T H \alpha$ , where  $H$  is a positive semidefinite matrix of similarities between points. This makes the dual problem convex.

The quadratic term  $\sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j X_i \cdot X_j$  in the dual can be expressed in the form  $\alpha^T B B^T \alpha$ , where  $B$  is  $n \times d$  matrix in which the  $i^{th}$  row of  $B$  contains  $y_i X_i$ . In other words, the  $i$ th row of  $B$  simply contains the  $i$ th data instance, after multiplying it with the class label  $y_i \in \{-1, 1\}$ .

*Proof:* This result can be shown by simply expanding the  $(i,j)^{th}$  term of  $\alpha^T B B^T \alpha$ . The matrices of the form  $B B^T$  are always positive semidefinite. Therefore, this is a convex optimization problem

# Inferring the Optimal Primal Solution from Optimal Dual Solution

---



The (PD) constraints can be used to infer the primal variables from the dual variables. In the particular case of the SVM, the constraints correspond to Eqns. (2-3). Among these constraints, Eqn. (2) is in a particularly useful form, because it directly yields all the primal variables in terms of the dual variables:

$$W = \sum_{i=1}^n \alpha_i y_i X_i^T \quad (6)$$

One can obtain the slack variables  $\xi_i$  by using the constraints among the primal variables and substituting the inferred value of  $W$ .

---

The dual is a constrained optimization problem, albeit a simple one because of the use of box constraints. The dual can be solved using almost all the primal optimization techniques. Therefore, we still need the primal algorithms for constrained optimization, even though we are working with the dual. In the following, we provide some examples of computational algorithms.

We state the dual problem in minimization form with box constraints:

$$\begin{aligned} \text{Minimize } L_D &= \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j X_i \cdot X_j - \sum_{i=1}^n \alpha_i \\ \text{subject to } 0 \leq \alpha_i \leq C, \forall i &\in \{1, 2, \dots, n\} \end{aligned}$$

The partial derivative of  $L_D$  with respect to  $\alpha_k$  is as follows:

$$\frac{\partial L_D}{\partial \alpha_k} = y_k \sum_{s=1}^n y_s \alpha_s X_k \cdot X_s - 1, \forall k \in \{1, 2, \dots, n\} \quad (7)$$

One can use the standard gradient procedure:

$$\alpha \leftarrow \alpha - \eta \frac{\partial L_D}{\partial \alpha} \quad (8)$$

One problem is that an update might lead to some of the values of  $\alpha_k$  violating the feasibility constraints. The value of each  $\alpha_k$  is reset to 0 if it becomes negative, and it is reset to C if it exceeds C. Therefore, one starts by setting the vector of Lagrangian parameters  $\alpha = [\alpha_1 \dots \alpha_n]$  to an n-dimensional vector of 0s and uses the following update steps with learning rate  $\eta$ :

**repeat**

Update  $\alpha_k \leftarrow \alpha_k + \eta[1 - y_k \sum_{s=1}^n y_s \alpha_s X_k \cdot X_s]$  for each  $k \in \{1, 2, \dots, n\}$

**for** each  $k \in \{1, 2, \dots, n\}$  **do begin**

$\alpha_k = \min\{\alpha_k, C\}$

$\alpha_k = \max\{\alpha_k, 0\}$

**end for**

**until** convergence

# Exercise Problems



Please attempt Problems 6.4.2, 6.4.3 and 6.4.4 from Charu Agarwal book page no. 281



## Lecture 15

Math Foundations Team



**BITS** Pilani

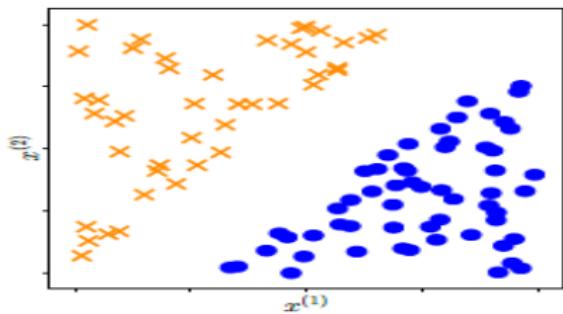
Pilani | Dubai | Goa | Hyderabad

- ▶ In last lecture, we discussed about KKT conditions for strong duality
- ▶ Using KKT conditions, we discussed SVM primal problem and formulated its dual and algorithm to solve it.
- ▶ In this lecture, we will study in detail about SVM primal and dual problem.

# Binary Classification Problem



In binary classification, the space is split into two parts corresponding to the positive and negative classes.



Example 2D data, illustrating the intuition of data where we can find a linear classifier that separates orange crosses from blue discs.

Similar to plane, in a D- dimensional vector space, the space is split into two parts using a hyperplane, an affine subspace of dimension D-1.

# Separating Hyperplanes



---

Consider  $x \in \mathbb{R}^D$  be an element of the data space. Consider a function

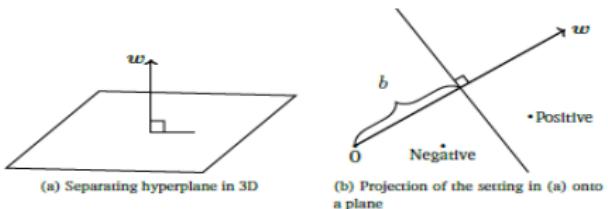
$$f : \mathbb{R}^D \rightarrow \mathbb{R}$$

$$f(x) := \langle w, x \rangle + b, w \in \mathbb{R}^D, b \in \mathbb{R}$$

The hyperplane that separates the two classes in the binary classification problem is given by

$$\{x \in \mathbb{R}^D | f(x) = 0\}$$

It can be shown that  $w$  is a normal vector to the hyperplane.



Proof: Let  $x_a$  and  $x_b$  be on the hyperplane.

$$\Rightarrow f(x_a) = f(x_b) = 0$$

$$\Rightarrow f(x_a) - f(x_b) = (\langle w, x_a \rangle + b) - (\langle w, x_b \rangle + b)$$

$$\Rightarrow f(x_a) - f(x_b) = \langle w, x_a - x_b \rangle = 0$$

Thus  $w$  is orthogonal to  $x_a - x_b$  and hence is a normal to the hyperplane.

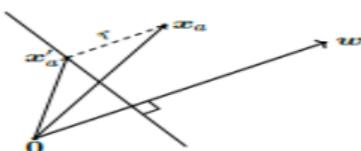
---

If  $f(x) \geq 0$  means above the hyperplane and can classify as 1. If  $f(x) \leq 0$  means below the hyperplane and can classify as -1.  
Thus when training the classifier, we want

$$\begin{aligned}\langle w, x_n \rangle + b &\geq 0 \text{ when } y_n = 1 \\ \langle w, x_n \rangle + b &\leq 0 \text{ when } y_n = -1, \\ \Rightarrow y_n \langle w, x_n \rangle + b &\geq 0\end{aligned}$$

- ▶ For a dataset  $\{(x_1, y_1), \dots, (x_n, y_n)\}$  that is linearly separable, there are infinitely many hyperplanes.
- ▶ To find a unique solution, choose the separating hyperplane that maximizes the margin between the positive and negative examples.
- ▶ Margin is the distance of the separating hyperplane to the closest example  $x_a$  in the dataset that is linearly separable.

# Concept of Margin



WLOG let  $x_a$  be on the positive side of the hyperplane,  $r$  be the distance of  $x_a$  from hyperplane and  $x_a'$  be the orthogonal projection of  $x_a$  onto the hyperplane Then

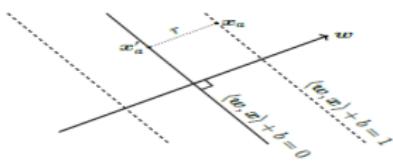
$$x_a - x_a' = r \frac{w}{\|w\|}$$
$$\Rightarrow y_n(\langle w, x_n \rangle + b) \geq r$$

We can assume  $\|w\| = 1$  as our interest is in the direction of  $w$

Together, we get

$$\max_{w, b, r} r$$

subject to  $y_n(\langle w, x_n \rangle + b) \geq r$ ,  $\|w\| = 1$ ,  $r > 0$ .



Another way is to rescale the axes such that the closest point  $x_a$  lies exactly on the margin

$$\langle w, x_n \rangle + b = 1$$

# Hard Margin SVM



As,  $x_a'$  is the orthogonal projection of  $x_a$  on the hyperplane ,

$$\langle w, x_a - r \frac{w}{\|w\|} \rangle + b = 0, \text{ as } x_a' = x_a - r \frac{w}{\|w\|} \Rightarrow r = \frac{1}{\|w\|}$$

$$\text{Thus } \max_{w,b} \frac{1}{\|w\|}$$

$$\text{subject to } y_n(\langle w, x_n \rangle + b) \geq 1, \forall n = 1, \dots, N$$

It can be written as, where no violations of margins are allowed,

$$\min_{w,b} \frac{1}{2} \|w\|^2 \text{ (hard margin SVM)}$$

$$\text{subject to } y_n(\langle w, x_n \rangle + b) \geq 1, \forall n = 1, \dots, N$$

# Equivalence of 2 derivations



We have  $\max_{w,b,r} r$

subject to  $y_n(\langle w, x_n \rangle + b) \geq r, \|w\| = 1, r > 0.$

Replacing  $r$  by  $r^2$  and  $w = \frac{w'}{\|w'\|}$  and dividing the constraint by  $r$ .

We get  $\max_{w',b,r} r^2$

subject to  $y_n\left(\langle \frac{w'}{\|w'\|r}, x_n \rangle + \frac{b}{r}\right) \geq 1, r > 0.$

If  $w'' = \frac{w'}{\|w'\|_r}$ ,  $b'' = \frac{b}{r}$  then we get

$$\max_{w'', b''} \frac{1}{\|w''\|^2}$$

subject to  $y_n(\langle w'', x_n \rangle + b'') \geq 1$

$\max_{w'', b''} \frac{1}{\|w''\|^2}$  can be replaced by  $\min_{w'', b''} \|w''\|^2$

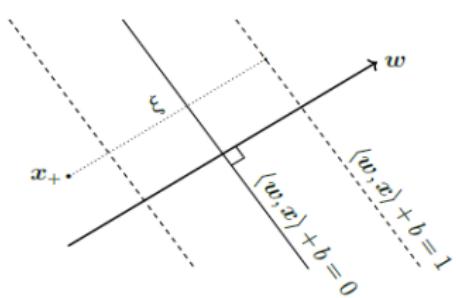
$$\min_{w'', b''} \frac{1}{2} \|w''\|^2$$

subject to  $y_n(\langle w'', x_n \rangle + b'') \geq 1$

# Soft Margin SVM



When data is not linearly separable, we may wish to allow some points to fall within the region or even to be on the wrong side of hyperplane.



The model that allows some classification errors is called the soft margin SVM.

The key geometric idea is to introduce a slack variable  $\xi_n$  corresponding to each example - lable pair  $(x_n, y_n)$  that allows a particular example to be within the margin or even on the wrong side of the hyperplane.

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n$$

$$\text{subject to } y_n(\langle w, x_n \rangle + b) \geq 1 - \xi_n$$

$$\xi_n \geq 0, \quad n = 1, \dots, N$$

# Soft Margin SVM: Loss Function View



Consider the error between the output of a predictor  $f(x_n)$  and the label  $y_n$ .

$$l(t) := \max\{0, 1 - t\} \text{ where } t = yf(x) = y(\langle w, x \rangle + b)$$

If  $f(x)$  is on the correct side and  $|f(x)| \geq 1$  then  $t \geq 1 \Rightarrow l(t) = 0$  and if  $|f(x)| \leq 1 \Rightarrow l(t) > 0$  ( $x$  is close to margin). If  $f(x)$  is on the wrong side  $l(t)$  will be more.

---

Alternate way to express the hinge loss is by defining

$$\begin{aligned}l(t) &= 0 \text{ if } t \geq 1 \\l(t) &= 1 - t \text{ if } t < 1\end{aligned}$$

The loss function corresponding to the hard margin SVM is

$$\begin{aligned}l(t) &= 0 \text{ if } t \geq 1 \\l(t) &= \infty \text{ if } t < 1\end{aligned}$$

To minimize the total loss, using the hinge loss, we get the unconstrained optimization problem

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \max\{0, 1 - y_n(\langle w, x_n \rangle + b)\}$$

We can equivalently replace minimization of hinge loss over  $t$  by minimization of slack variable  $\xi$  with 2 constraints. In equation form

$$\min_t \max\{0, 1 - t\}$$

is equivalent to  $\min_{\xi,t} \xi$  subject to  $\xi \geq 0, \xi \geq 1 - t$

This is same as soft margin SVM

Consider the primal soft margin SVM. The Lagrangian is then given by

$$L(w, b, \xi, \alpha) = \frac{1}{2} \|w\|^2 + C \sum_{n=1}^N \xi_n - \sum_{n=1}^N \alpha_n (y_n (\langle w, x_n \rangle + b) - 1 + \xi_n) - \sum_{n=1}^N \gamma_n \xi_n$$

Then

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{n=1}^N \alpha_n y_n x_n$$

$$\frac{\partial L}{\partial b} = - \sum_{n=1}^N \alpha_n y_n = 0, \quad \frac{\partial L}{\partial \xi_n} = C - \alpha_n - \gamma_n = 0$$

Using the PD constraints and simplifying, we get the dual problem

$$\max_{\alpha} -\frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

i.e.

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle$$

subject to

$$\sum_{i=1}^N \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad \forall i = 1, \dots, N$$

From dual parameters, we can find  $w$  and  $b$ .

- ▶ We would like to build convex hull that is the smallest possible convex set containing all  $x_n$  with same label
- ▶ This can be done by introducing non negative weights  $\alpha_n \geq 0$  for each  $x_n$ .

Then

$$conv(X) = \left\{ \sum_{i=1}^N \alpha_i x_i \right\} \text{ with } \sum_{i=1}^N \alpha_i = 1, \alpha_i \geq 0, \forall i = 1, \dots, N$$

Let  $c$  from positive class convex hull be closest to the negative class and  $d$  be vice versa, then

$$w := c - d, \text{ where } c = \sum_{i:y_i=1} \alpha_i^+ x_i, \quad d = \sum_{i:y_i=-1} \alpha_i^- x_i$$

For  $c, d$  to be the closest is same as

$$\begin{aligned} \arg \min_w \|w\| &= \arg \min_w \frac{1}{2} \|w\|^2 \\ \Rightarrow \min_{\alpha \geq 0} \frac{1}{n} \left\| \sum_{i:y_i=1} \alpha_i^+ x_i - \sum_{i:y_i=-1} \alpha_i^- x_i \right\|^2 \end{aligned}$$

Now

$$\sum_{i:y_i=1} \alpha_i^+ = 1, \quad \sum_{i:y_i=-1} \alpha_i^- = 1 \Rightarrow \sum_{i=1}^N y_i \alpha_i = 0$$

Consider

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i=1}^N \alpha_i$$

Let set of features  $\phi(x_i)$  replaces  $x_i$ . Then if  $\phi(x)$  is non linear, we can construct classifiers that are non linear. Instead of explicitly defining nonlinear  $\phi(.)$  and computing the inner product, we can define similarity function  $k(x_i, x_j)$  between  $x_i, x_j$  called as kernels.

The certain class of similarity functions called kernels, the similarity function implicitly defines a non linear map  $\phi(\cdot)$ .  
Kernels are by definition, functions  $k$  such that

$$k : \chi \times \chi \rightarrow \mathbb{R}$$

$$k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_H$$

where  $\phi : \chi \rightarrow H$  such that  $\phi(x) = k(., x)$

The matrix  $K \in \mathbb{R}^{N \times N}$  resulting from the inner product, called as Gram matrix or kernel matrix, is symmetric and positive semidefinite.



**BITS** Pilani  
Pilani Campus

# Lecture 16

MFDS/MFML Team

# Topics to be covered

- 
- Nonlinear SVM
  - Kernel Trick
  - SVM Kernels
  - Multi-Class Problem
  - SVM vs Logistic Regression
  - SVM Applications
-

# Solving the Optimization Problem

1. Maximize margin  $2/\|\mathbf{w}\|$
2. Correctly classify all training data points:

$$\mathbf{x}_i \text{ positive } (y_i = 1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \geq 1$$

$$\mathbf{x}_i \text{ negative } (y_i = -1) : \quad \mathbf{x}_i \cdot \mathbf{w} + b \leq -1$$

*Quadratic optimization problem:*

Find  $\mathbf{w}$  and  $b$  such that

$$\Phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \text{ is minimized;}$$

$$\text{and for all } \{(\mathbf{x}_i, y_i)\}: y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$$

# Solving the Optimization Problem

- The solution involves constructing a *dual problem* where a *Lagrange multiplier*  $\alpha_i$  is associated with every constraint in the primary problem:

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

- Taking partial derivative with respect to  $w$ ,  $\frac{\partial L}{\partial w} = 0$ 
  - $w - \sum \alpha_i y_i x_i = 0$
  - $w = \sum \alpha_i y_i x_i$
- Taking partial derivative with respect to  $b$ ,  $\frac{\partial L}{\partial b} = 0$ 
  - $-\sum \alpha_i y_i = 0$
  - $\sum \alpha_i y_i = 0$

# Solving the Optimization Problem

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

- Expanding above equation:

$$L(w, b, \alpha_i) = \frac{1}{2} w^T w - \sum \alpha_i y_i w^T x_i + \sum \alpha_i y_i b + \sum \alpha_i$$

- Substituting  $w = \sum \alpha_i y_i x_i$  and  $\sum \alpha_i y_i = 0$  in above equation

$$L(w, b, \alpha_i) = \frac{1}{2} (\sum_i \alpha_i y_i x_i) (\sum_j \alpha_j y_j x_j) - (\sum_i \alpha_i y_i x_i) (\sum_j \alpha_j y_j x_j) + \sum \alpha_i$$

$$L(w, b, \alpha_i) = \sum \alpha_i - \frac{1}{2} (\sum_i \alpha_i y_i x_i) (\sum_j \alpha_j y_j x_j)$$

$$L(w, b, \alpha_i) = \sum \alpha_i - \frac{1}{2} (\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i \cdot x_j)$$

# The Dual Problem

- The new objective function is in terms of  $\alpha_i$  only
- It is known as the dual problem: if we know  $\mathbf{w}$ , we know all  $\alpha_i$ ; if we know all  $\alpha_i$ , we know  $\mathbf{w}$
- The original problem is known as the primal problem
- The objective function of the dual problem needs to be maximized (comes out from the KKT theory)
- The dual problem is therefore:

$$\text{max. } W(\boldsymbol{\alpha}) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1, j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to  $\alpha_i \geq 0$ ,

$$\sum_{i=1}^n \alpha_i y_i = 0$$

Properties of  $\alpha_i$  when we introduce the Lagrange multipliers

The result when we differentiate the original Lagrangian w.r.t. b

# 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  $\{(\mathbf{x}_i, y_i)\}$ :  $y_i (\mathbf{w}^T \mathbf{x}_i + b) \geq 1$

$$L(w, b, \alpha_i) = \frac{1}{2} \|w\|^2 - \sum \alpha_i [y_i (w^T x_i + b) - 1]$$

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \left( \sum_i \sum_j \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \right)$  is

maximized and

$$(1) \sum \alpha_i y_i = 0$$

$$(2) \alpha_i \geq 0 \text{ for all } \alpha_i$$

# 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

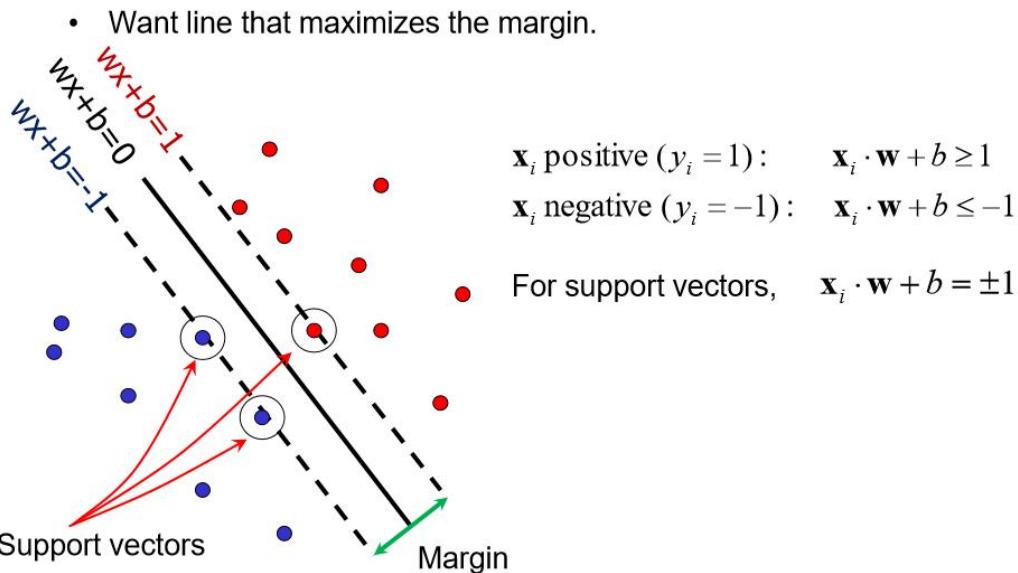
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0 \text{ and } \alpha_i > 0$$

For support vectors:

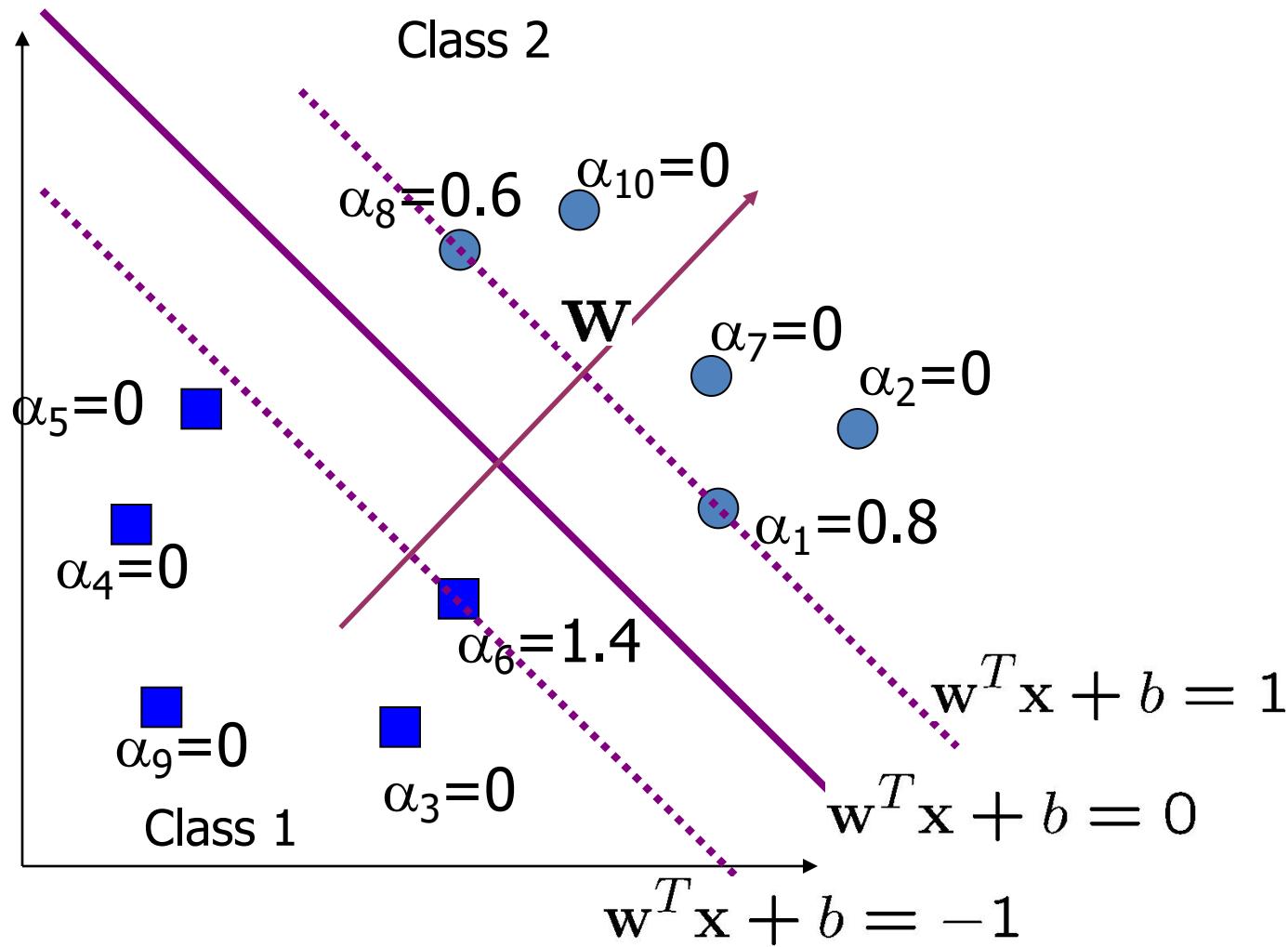
$$y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 = 0$$

For all points other than  
support vectors:

$$\alpha_i = 0$$



# A Geometrical Interpretation



# Solving the Optimization Problem

- 
- Solution:  $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$



# 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:

$$\begin{aligned}f(x) &= \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) \\&= \text{sign}\left(\sum_i \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right)\end{aligned}$$

If  $f(x) < 0$ , classify as negative, otherwise classify as positive.

- Notice that it relies on an *inner product* between the test point  $\mathbf{x}$  and the support vectors  $\mathbf{x}_i$
- (Solving the optimization problem also involves computing the inner products  $\mathbf{x}_i \cdot \mathbf{x}_j$  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  $\alpha_i$ .
- Both in the dual formulation of the problem and in the solution training points appear only inside dot products:

Find  $\alpha_1 \dots \alpha_N$  such that

$Q(\alpha) = \sum \alpha_i - \frac{1}{2} \sum \sum \alpha_i \alpha_j y_i y_j x_i^T x_j$  is maximized and

$$(1) \sum \alpha_i y_i = 0$$

$$(2) 0 \leq \alpha_i \leq C \text{ for all } \alpha_i$$

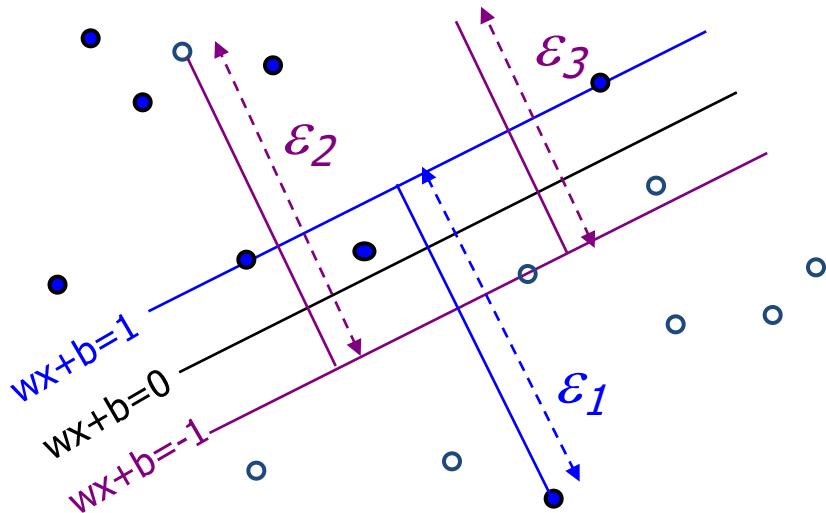
$$f(x) = \sum \alpha_i y_i x_i^T x + b$$

# Soft Margin Classification

***Slack variables*  $\xi_i$  can be added to allow misclassification of difficult or noisy examples.**

What should our quadratic optimization criterion be?

Minimize



$$\frac{1}{2} \mathbf{w} \cdot \mathbf{w} + C \sum_{k=1}^R \varepsilon_k$$

# Soft Margin

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^N \xi_i$$

The  $\mathbf{w}$  that minimizes...

Maximize margin      Minimize misclassification

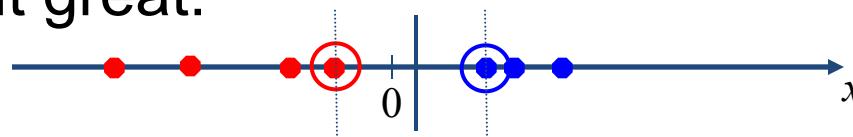
Misclassification cost      # data samples      Slack variable

subject to     $y_i \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i,$

$\xi_i \geq 0, \quad \forall i = 1, \dots, N$

# Non-linear SVMs

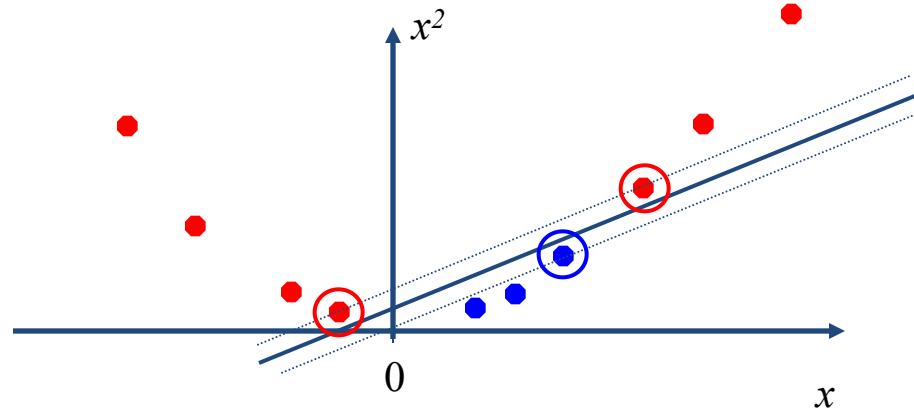
- Datasets that are linearly separable with some noise soft margin work out great:



- But what are we going to do if the dataset is just too hard?



- How about... mapping data to a higher-dimensional space:



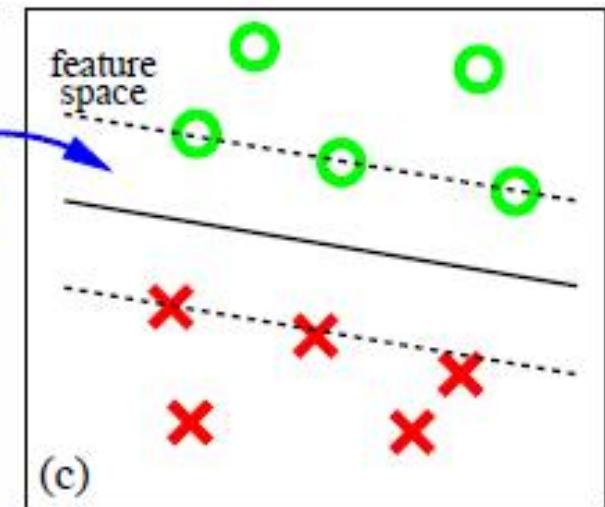
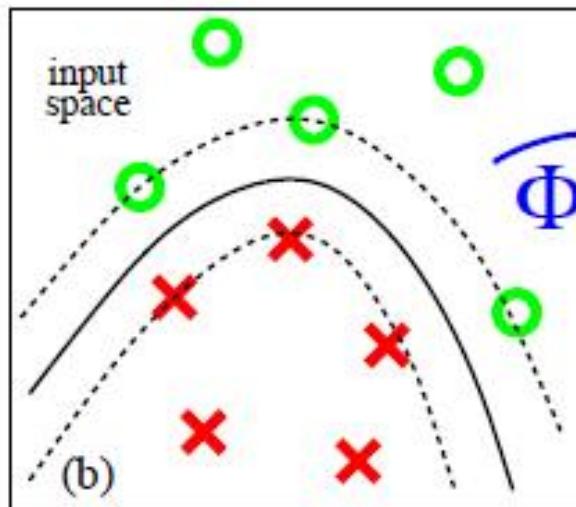
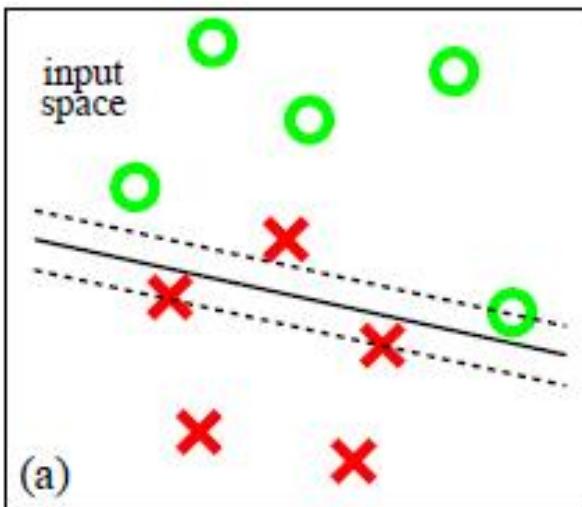
# The “Kernel Trick”

- The linear classifier relies on dot product between vectors
  - $x_i^T \cdot x_j$
- If every data point is mapped into high-dimensional space via some transformation  $\Phi: x \rightarrow \phi(x)$ , the dot product becomes:
$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$
- A *kernel function* is some function that corresponds to an inner product in some expanded feature space.

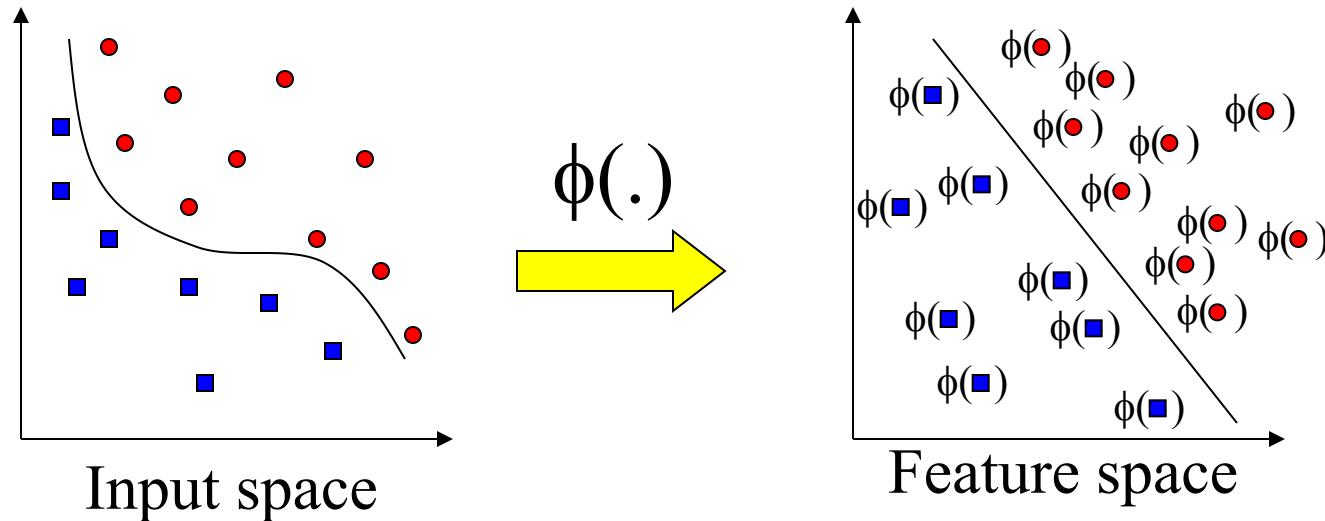
# SVM Kernels

- 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.

# Find a feature space



# Transforming the Data

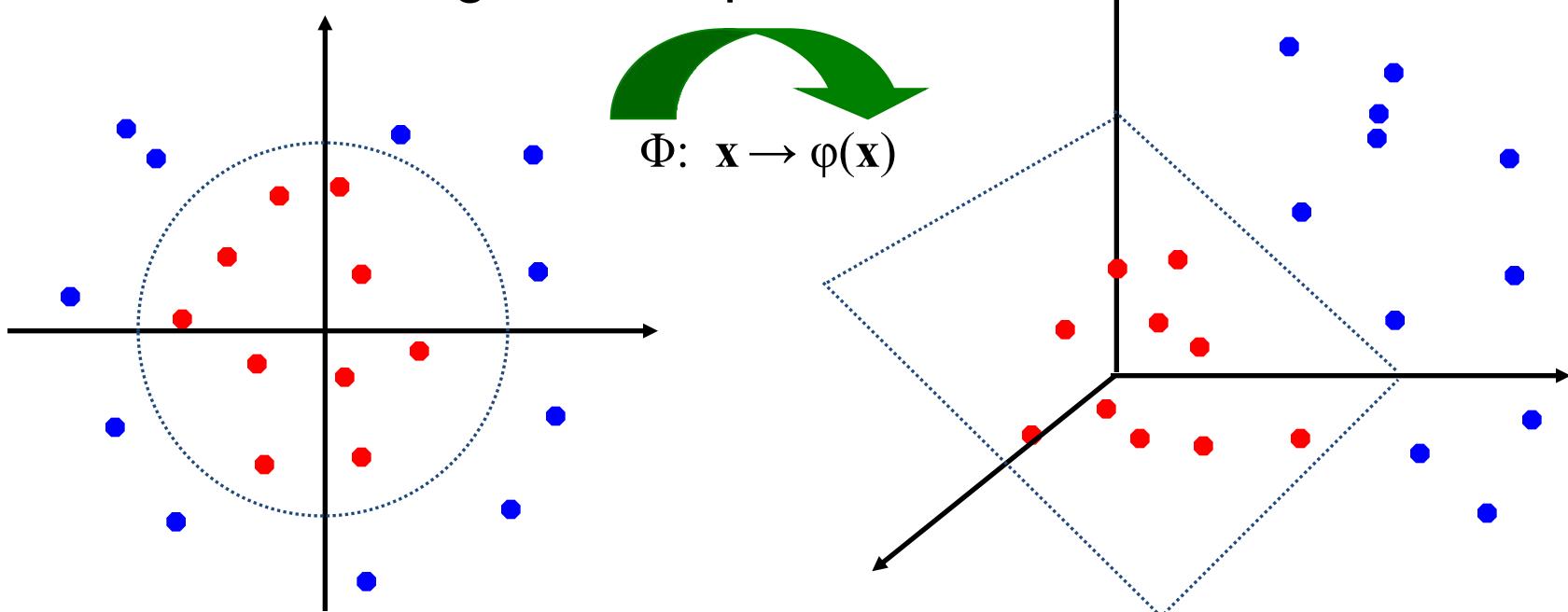


Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

# 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

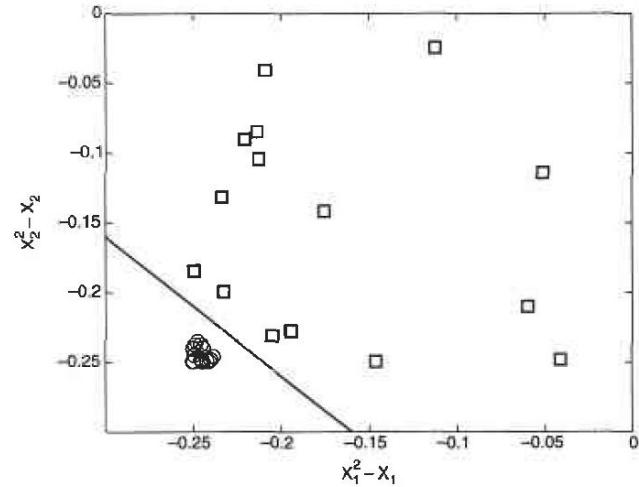
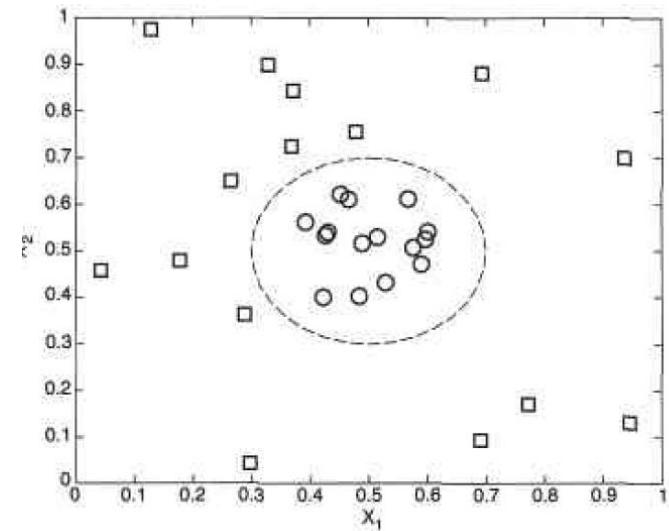
- Data is not separable linearly
- Margin will become inefficient
- Data needs to be transformed from original coordinate space  $\mathbf{x}$  to a new space  $\Phi(\mathbf{x})$ , so that linear decision boundary can be applied
- A non-linear transformation function is needed, like, ex:

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

- In the transformed space we can choose  $w = (w_0, w_1, \dots, w_4)$  such that

$$w_4x_1^2 + w_3x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

- The linear decision boundary in the transformed space has the following form:  $w \cdot \Phi(\mathbf{x}) + b = 0$



# What Functions are Kernels?

- Kernel is a continuous function  $k(x,y)$  that takes two arguments  $x$  and  $y$  (real numbers, functions, vectors, etc.) and maps them to a real value independent of the order of the arguments, i.e.,  $k(x,y)=k(y,x)$  .
- For some functions  $K(x_i, x_j)$  checking that  $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$  can be cumbersome.
- Mercer's theorem:

*Every positive-semidefinite symmetric function is a kernel*

# What Functions are Kernels?

---

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.

2) We can *construct kernels from other kernels*:

- if  $k$  is a kernel and  $\alpha > 0$ , then  $\alpha k$  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 Kernel Functions

---

- Linear:  $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
- Polynomial of power  $p$ :  $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^\top \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

- Sigmoid:  $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^\top \mathbf{x}_j + \beta_1)$

Name	Function	Type problem
Polynomial Kernel	$(x_i^T x_j + 1)^q$ q is degree of polynomial	Best for Image processing
Sigmoid Kernel	$\tanh(ax_i^T x_j + k)$ k is offset value	Very similar to neural network
Gaussian Kernel	$\exp(-\ x_i - x_j\ ^2 / 2\sigma^2)$	No prior knowledge on data
Linear Kernel	$(1 + x_i^T x_j) \min(x_i, x_j) - \frac{(x_i + x_j)}{2} \min(x_i, x_j)^2 + \frac{\min(x_i, x_j)^3}{3}$	Text Classification
Laplace Radial Basis Function (RBF)	$(e^{-\lambda \ x_i - x_j\ }), \lambda >= 0$	No prior knowledge on data

There are many more kernel functions.

# Non-linear SVMs Mathematically

- The solution is:

$$f(\mathbf{x}) = \sum \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b$$

- Optimization techniques for finding  $\alpha_i$ '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.

# Multi-Class Problem

---

Instead of just two classes, we now have C classes

- E.g. predict which movie genre a viewer likes best
- Possible answers: action, drama, indie, thriller, etc.

Two approaches:

- One-vs-all
- One-vs-one

# Multi-Class Problem

---

Instead of just two classes, we now have C classes

- E.g. predict which movie genre a viewer likes best
- Possible answers: action, drama, indie, thriller, etc.

Two approaches:

- One-vs-all
- One-vs-one

# Multi-Class Problem

---

## One-vs-all (a.k.a. one-vs-others)

- Train C classifiers
- In each, pos = data from class  $i$ , neg = data from classes other than  $i$
- The class with the most confident prediction wins
- Example:
  - You have 4 classes, train 4 classifiers
  - 1 vs others: score 3.5
  - 2 vs others: score 6.2
  - 3 vs others: score 1.4
  - 4 vs other: score 5.5
  - Final prediction: class 2
- Issues?

# Multi-Class Problem

---

## One-vs-one (a.k.a. all-vs-all)

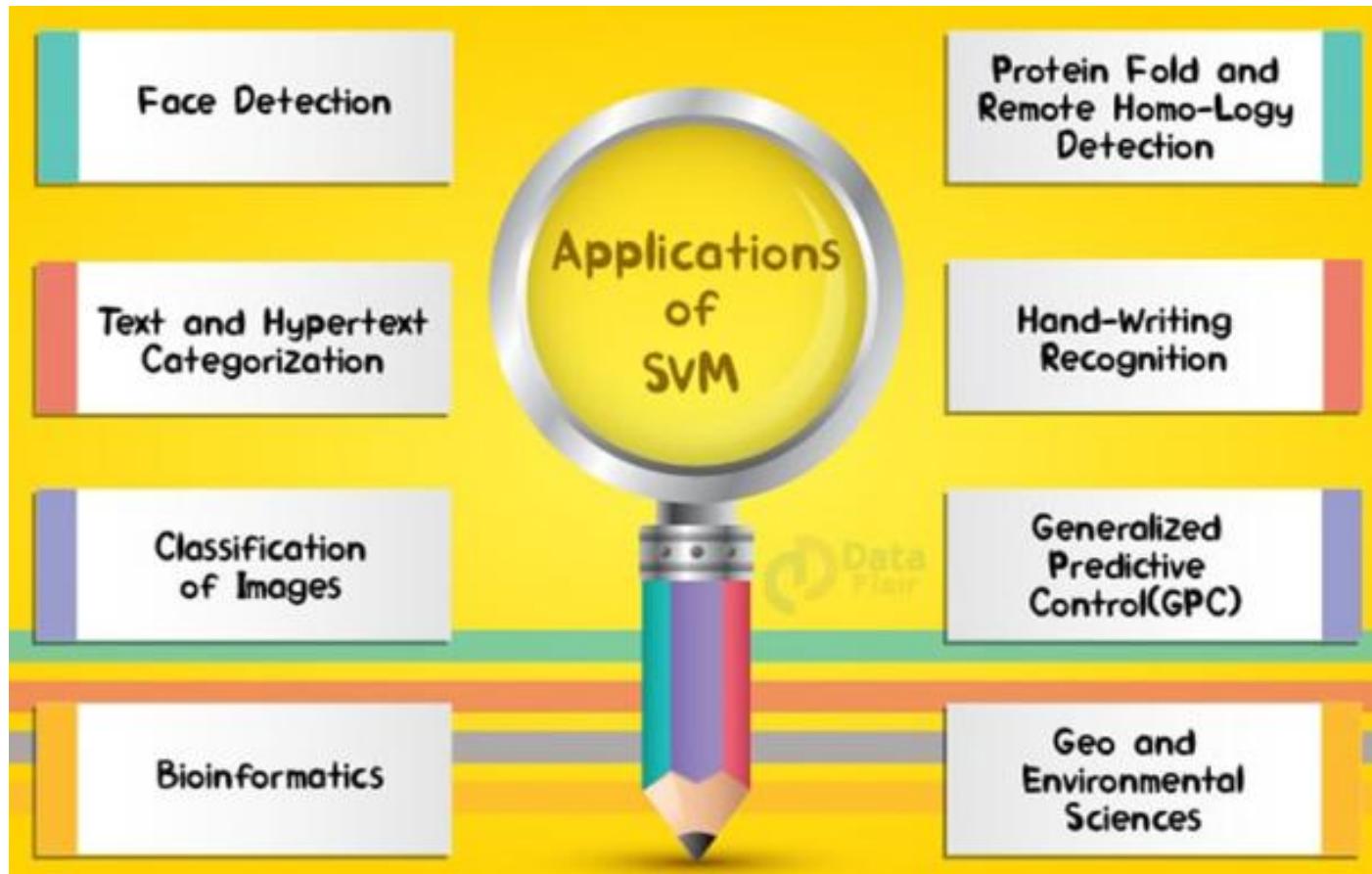
- Train  $C(C-1)/2$  binary classifiers (all pairs of classes)
- They all vote for the label
- Example:
  - You have 4 classes, then train 6 classifiers
  - 1 vs 2, 1 vs 3, 1 vs 4, 2 vs 3, 2 vs 4, 3 vs 4
  - Votes: 1, 1, 4, 2, 4, 4
  - Final prediction is class 4

# Properties of SVM

- **Flexibility in choosing a similarity function**
- **Sparseness of solution when dealing with large data sets**
  - Only support vectors are used to specify the separating hyperplane
  - Therefore SVM also called sparse kernel machine.
- **Ability to handle large feature spaces**
  - complexity does not depend on the dimensionality of the feature space
- **Overfitting can be controlled by soft margin approach**
- **Nice math property: a simple convex optimization problem which is guaranteed to converge to a single global solution**
- **Feature Selection**

# SVM Applications

SVM has been used successfully in many real-world problems



# Application : Text Categorization

---

- Task: The classification of natural text (or hypertext) documents into a fixed number of predefined categories based on their content. A document can be assigned to more than one category, so this can be viewed as a series of binary classification problems, one for each category

# Text Categorization using SVM

---

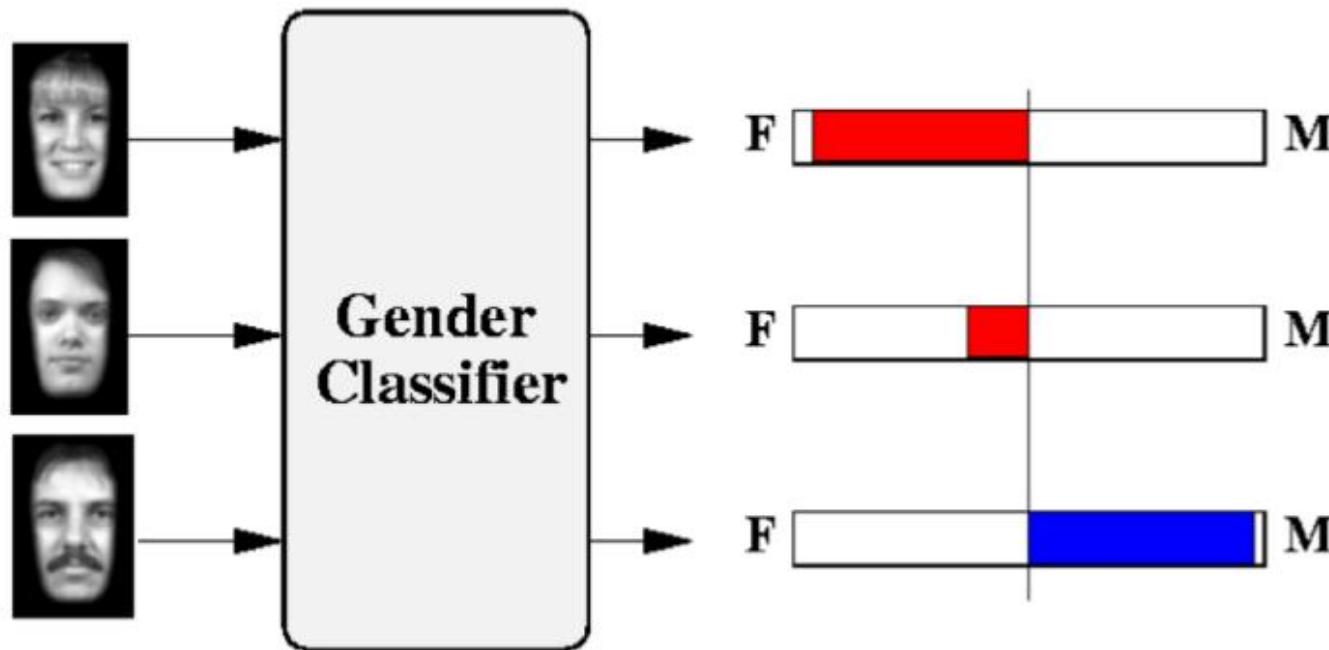
- The distance between two documents is  $\phi(x) \cdot \phi(z)$
- $K(x,z) = \phi(x) \cdot \phi(z)$  is a valid kernel, SVM can be used with  $K(x,z)$  for discrimination.
- Why SVM?
  - High dimensional input space
  - Few irrelevant features (dense concept)
  - Sparse document vectors (sparse instances)
  - Text categorization problems are linearly separable

# Using SVM

---

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.
-

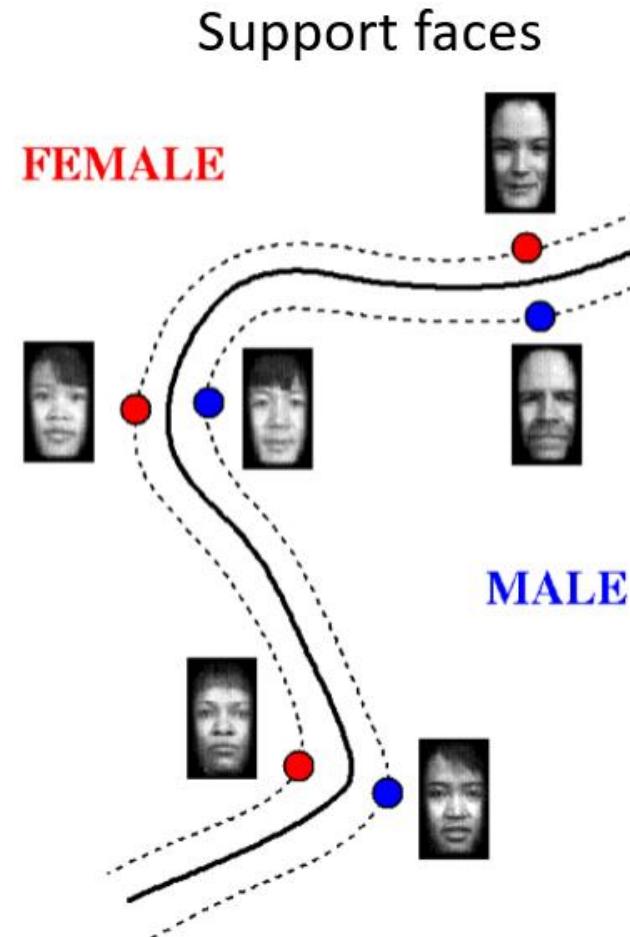
# Learning Gender from image with SVM



Moghaddam and Yang, Learning Gender with Support Faces, TPAMI 2002

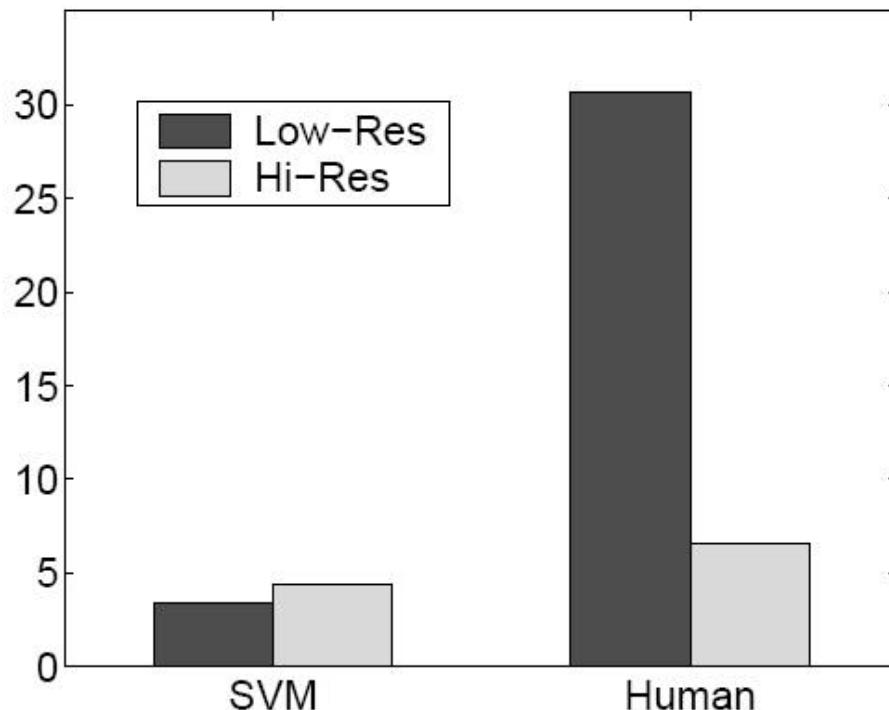
Moghaddam and Yang, Face & Gesture 2000

# Support faces



# Accuracy of SVM Classifier

% Error Rates



- SVMs performed better than humans, at either resolution

**Figure 6. SVM vs. Human performance**



---

# Thank You